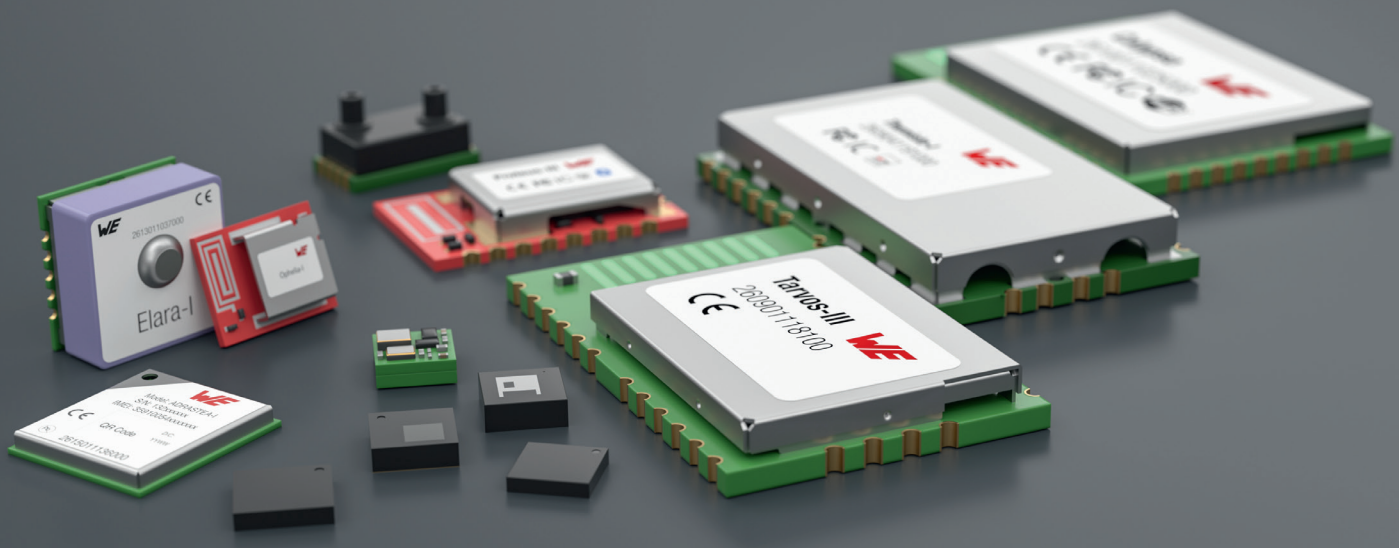


User manual

Proteus-IV

2621011024000

Version 1.1

February 17, 2026

WÜRTH ELEKTRONIK MORE THAN YOU EXPECT

****

MUST READ

Check for firmware updates

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

We strongly recommend including the possibility of a firmware update in the customer system design.

Contents

Overview of helpful application notes

Application note ANR008 - Wireless Connectivity Software Development Kit

http://www.we-online.com/ANR008
To ease the integration of the Würth Elektronik eiSos radio modules into an application, Würth Elektronik eiSos offers the corresponding Software Development Kit (SDK) for most commonly used host processors. This SDK contains drivers and examples in C-code to communicate with the corresponding radio module. This application note shows which SDKs are available and describes how to download and use them.

Application note ANR010 - Range estimation

http://www.we-online.com/ANR010
This application note presents the two most used mathematical range estimation models, Friis and two ray ground reflection, and its implementation in the range estimation tool of the RED-EXPERT.

Application note ANR027 - Bluetooth qualification guide

http://www.we-online.com/ANR027
Every product containing Bluetooth® technology needs to be qualified at the Bluetooth® SIG (special interest group). This application note explains the steps to be done to gain a Bluetooth® qualification for the end product using a Würth Elektronik eiSos Bluetooth® LE radio module.

Application note ANR030 - nRF Connect

http://www.we-online.com/ANR030
This application note gives a short overview about the options to create a custom firmware for Würth Elektronik eiSos radio modules by using the hardware platform and the embedded nRF5x system on chip. It presents options on firmware development environments and accessories (like SDKs) for the use within the nRF5 ecosystem. The reader is informed on how to access to a multitude of radio standards (like Bluetooth® LE, Bluetooth® MESH, Bluetooth® LE Audio, Matter, Zigbee, Thread, Wirepas) for custom firmware developments whilst the hardware platform can stay the same.

Application note ANR031 - Certification of custom modules

http://www.we-online.com/ANR031
This application note explains how certifications of a standard product can be used to gain the certification of a customized product. This is done for firmware, which has been adapted by Würth Elektronik eiSos, as well as for firmware written by customer.

Ground plane effects on radio module antennas

http://www.we-online.com/ANR033

The ground plane plays a critical role in the performance of radio module antennas, affecting parameters such as radiation pattern, gain, and efficiency. This application note provides practical insights into how ground plane size, shape, and placement influence antenna behavior, offering guidance for optimal integration in real-world designs. Simulation results and measurement data are included to illustrate key effects and support design decisions.

Application note ANR036 - Build your own firmware - getting started with Zephyr®

http://www.we-online.com/ANR036
This application note describes how to use Zephyr® to develop firmware for the supported Würth Elektronik eiSos BYOF (build your own firmware) radio modules. This includes the installation of the Zephyr® build system, as well as the selection of an appropriate example for the underlying radio module hardware.

1. Revision history

Manual version	FW version	HW version	Notes	Date
1.0	1.1	2.0	• Initial release	February 2026

2. Abbreviations

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.
EV (Board)	Evaluation (Board)	Proteus-IV populated on motherboard with USB interface for test and evaluation purpose.
GAP	Generic Access Profile	The GAP provides a basic level of functionality that all Bluetooth® devices must implement.
I/O	Input/output	Pinout description.
LESC	Low energy secure connection	Elliptic curve encryption method for Bluetooth® LE encryption and authentication
LPM	Low power mode	Mode for efficient power consumption.
LRM	Long range mode	Radio mode with higher range and lower throughput.
MAC		MAC address of the module.
MITM	Man In The Middle	It's a type of cyberattack where a malicious actor secretly intercepts and possibly alters the communication between two parties who believe they are directly communicating with each other.
MTU	Maximum transmission unit	Maximum packet size of the Bluetooth® connection.
Payload		The intended message in a frame / package.
RRAM	Resistive random-access memory	Type of non-volatile memory.
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.
SPI	Serial Peripheral Interface	Allows the serial communication with the module.
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.

3. Introduction

The Proteus-IV module is a radio submodule for wireless communication between devices such as control systems, remote controls, sensors, and similar applications.
Based on Bluetooth® LE 6.0 [1], it offers fast and secure transmission of data packets between two or more parties. A serial interface (UART) is available for communication with the host system.

Even with its small dimensions of 8 x 12 mm, the Proteus-IV module provides a highly miniaturized integrated PCB antenna. Additionally, it is possible to connect an external antenna for applications requiring extended radio range.

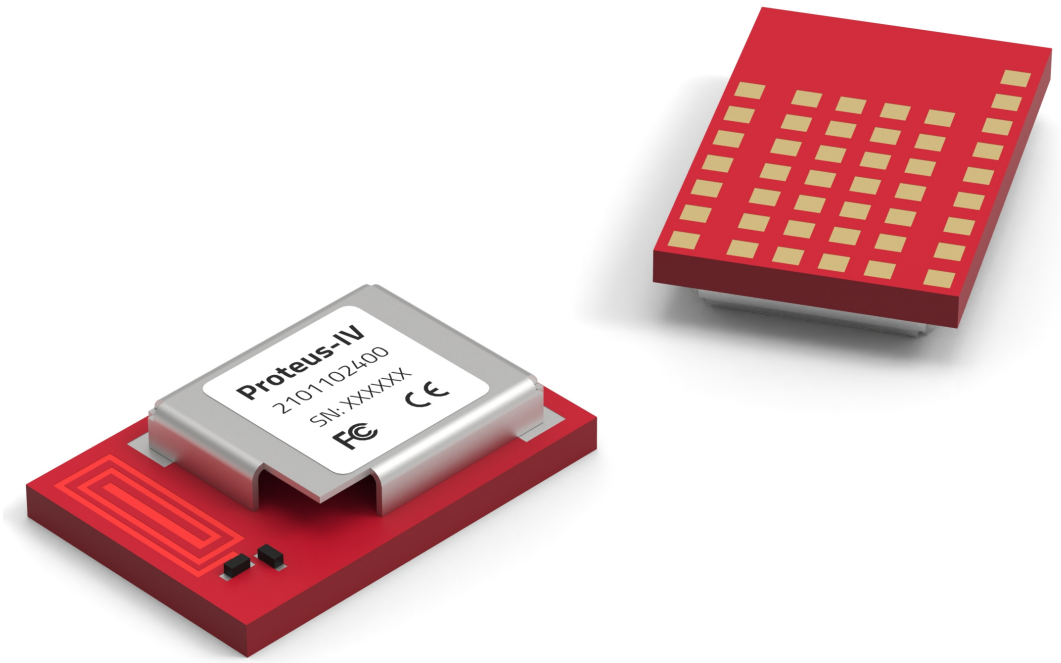


Figure 1: Proteus-IV

3.1. Block diagram

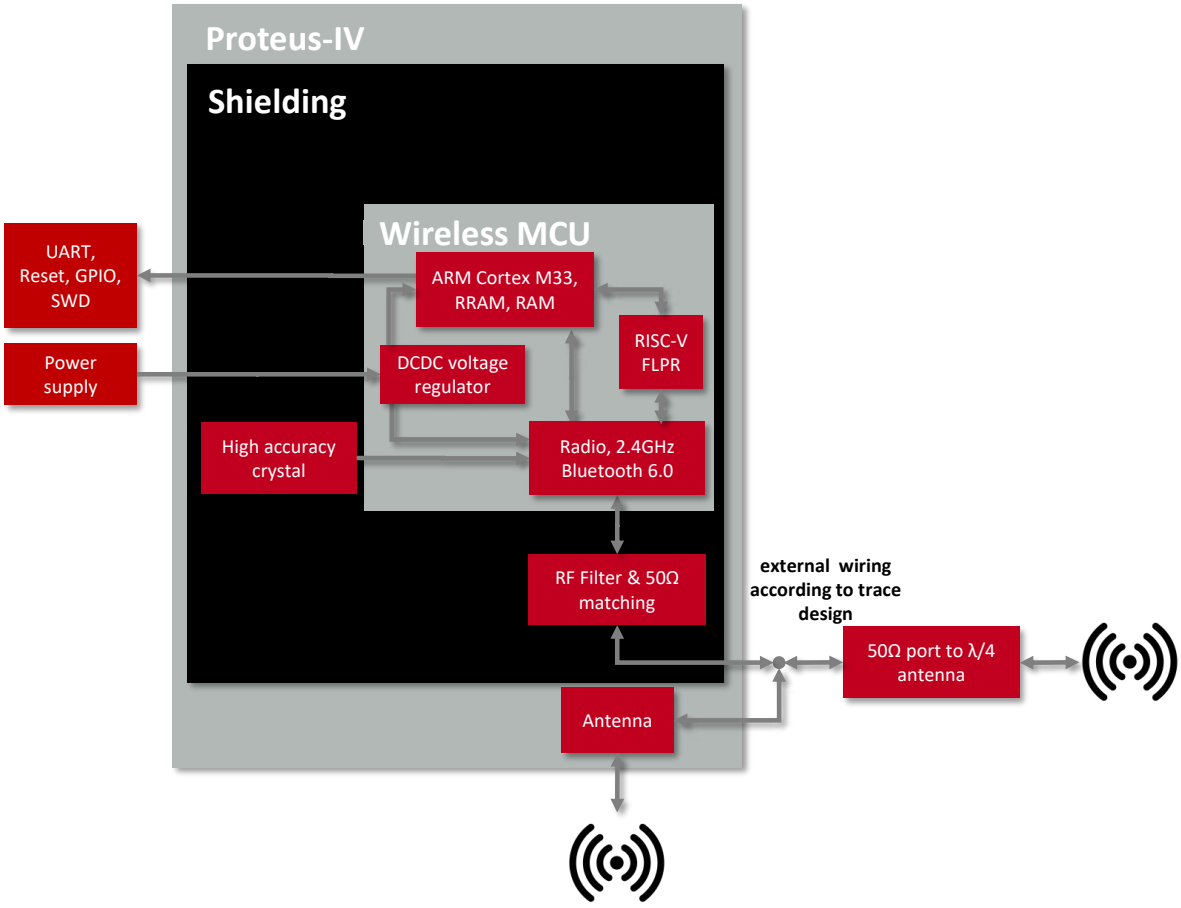


Figure 2: Block diagram of the module

3.2. Key features

The Proteus-IV offers the following key features that are described in this document in more detail:

Connection-based secure data transmission: The Proteus-IV firmware implements the **SPPlikeV2** Bluetooth® LE profile that allows bidirectional data transmission between several Proteus-IV modules and/or other Bluetooth® LE devices supporting this profile. Any module in the network can initiate a connection. Secure connections allow the transmission of authenticated and encrypted data.

Multi-connect: The Proteus-IV supports up to 6 parallel Bluetooth® LE connections (1 as a central plus up to 5 as a peripheral).

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

Latest microprocessor generation provided by Nordic Semiconductor nRF54L15 series: The heart of the Proteus-IV is a Bluetooth® LE chip of the nRF54L15 [2] series offering high performance values combined with low power consumption. It is a 128 MHz Arm Cortex-M33 processor with 1.5 MB RRAM and 256 KB RAM and up to 8 dBm output power.

Bluetooth® stack: The Bluetooth® 6 stack enables fast and energy-efficient data transmission using state-of-the-art technology from 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-IV automatically switches its role and can act as peripheral and central at the same time.

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

Transparent mode: The Proteus-IV firmware provides the "transparent mode" that allows the exchange of payload data without the need for software integrating the radio module.

Config mode: The so-called "config mode" allows safe configuration or recovery of the device settings.

Additional Bluetooth® radio modes: The Proteus-IV provides advanced radio modes including 2 MBit mode for faster data transmission and the LE coded mode that allows long-range data transmissions.

Flexible wired interfacing: The Proteus-IV is equipped with extra pins suited for custom device/sensor connection. With the help of these, a tailored firmware can be developed that 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.

3.3. Ordering information

WE order code	Description
2621011024000	Proteus-IV Bluetooth® LE Module, Tape & Reel
2621129024001	Bluetooth® LE EV-Kit with Proteus-IV EV-Board

Table 3: Ordering information

4. Electrical specifications

Unless otherwise stated, the following values have been measured on a Proteus-IV EV-Board with T = 25 ℃, VDDS = 3 V, f = 2.44 GHz, internal DC-DC converter in use.

4.1. Operating conditions

Description	Min.	Typ.	Max.	Unit
Temperature	-40	25	105	° C
Supply voltage (VDD) at -40 - 85 ° C	1.7	3	3.6	V
Supply voltage (VDD) at 85 - 105 ° C	1.7	3	3.4	V

Table 4: Operating conditions

4.2. Absolute maximum ratings

Description	Min.	Typ.	Max.	Unit
Supply voltage (VDD)	-0.3		+3.9	V
Voltage on any digital pin, VDD $\leq$ 3.6 V)	-0.3		VDD+0.3	V
Voltage on any digital pin, VDD $>$ 3.6 V)			3.9	V

Table 5: At -40 - 85 ° C

Description	Min.	Typ.	Max.	Unit
Supply voltage (VDD)	-0.3		+3.7	V
Voltage on any digital pin, VDD $\leq$ 3.4 V)	-0.3		VDD+0.3	V
Voltage on any digital pin, VDD $>$ 3.4 V)			3.7	V

Table 6: At 85 - 105 ° C

Description	Min.	Typ.	Max.	Unit
Input RF level			10	dBm
RRAM (non-volatil memory) endurance	10 000			Write/erase cycles

Table 7: General

4.3. Power consumption

4.3.1. Static consumption

Parameter	Test conditions	Value	Unit
TX current consumption	Transmitter only, DC/DC converter enabled, 1 Mbps Bluetooth® LE, CPU current not included, nRF54L15 product specification, maximum output power (`RF_TXPower` = 8)	9.8	mA
	Radio module module current consumption DC/DC converter enabled, Bluetooth® LE DTM firmware[1], maximum output power (`RF_TXPower` = 8)	12	mA

Table 8: Current consumption - transmitting

Parameter	Test conditions	Value	Unit
RX current consumption	Receiver only, DC/DC converter enabled, 1 Mbps Bluetooth® LE, CPU current not included, nRF54L15 product specification	3.4	mA
	Radio module module current consumption DC/DC converter enabled, Bluetooth® LE DTM firmware[1]	3.6	mA

Table 9: Current consumption - receiving

Parameter	Test conditions	Value	Unit
Current consumption	Sleep (system off mode)	0.6	µA
Current consumption	CPU running Coremark at 128 MHz from NVM, Cache enabled	2.6	mA

Table 10: Current consumption - low power

[1] The current measurement is done using the DTM firmware, including the radio peripheral plus CPU and UART in energy efficient state. Reference *nRFConnect DTM 3.1.0 example*.

4.4. Radio characteristics

Parameter	Min.	Max.	Unit
Frequency	2402	2480	MHz

Table 11: Frequency range

Parameter	Min.	Max.	Unit
RSSI accuracy valid range (± 2 dB)	-90	-30	dBm

Table 12: RSSI accuracy

Parameter	Test conditions	Value	Unit
Output power	Conducted (50 Ω), `RF_TXPower` = 8	8	dBm
	Radiated, `RF_TXPower` = 8	4.5	dBm
Input sensitivity	Conducted (50 Ω), BER = 1E-3	-96	dBm
	Radiated, BER = 1E-3	-92	dBm

Table 13: Measured transmit and receive power, 1 Mbit Bluetooth® LE physical layer

All transmit and receive power levels are measured on the EV-Board. The values already include losses of transitions from module to motherboard to SMA or module's PCB antenna. They are realistic values for the end application.
Radiated TX power and RX sensitivity were measured using the radio module's integrated PCB antenna.

4.5. Pin characteristics

When configured as a digital pin output, "standard drive" is used in the Proteus-IV firmware.

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 ≥1.7V	1	3	4	mA
Current at VSS+0.4 V, output set low, high drive, VDD ≥ 1.7 V	3			mA
Current at VSS+0.4 V, output set low, extra drive, VDD ≥ 1.7 V	16			mA
Current at VDD-0.4 V, output set high, **standard drive**, VCC ≥1.7V	1	3	4	mA
Current at VDD-0.4 V, output set high, high drive, VDD ≥ 1.7 V	4			mA
Current at VDD-0.4 V, output set high, extra drive, VDD ≥ 1.7 V	14			mA
Internal pull-up resistance	12	14	16	kΩ
Internal pull-down resistance	12	14	18	kΩ
Rise/Fall time, high drive mode, 20-80%		4		ns
Rise/Fall time, extra drive mode, 20-80%		0.9		ns

Table 14: Pin characteristics

5. Pinout

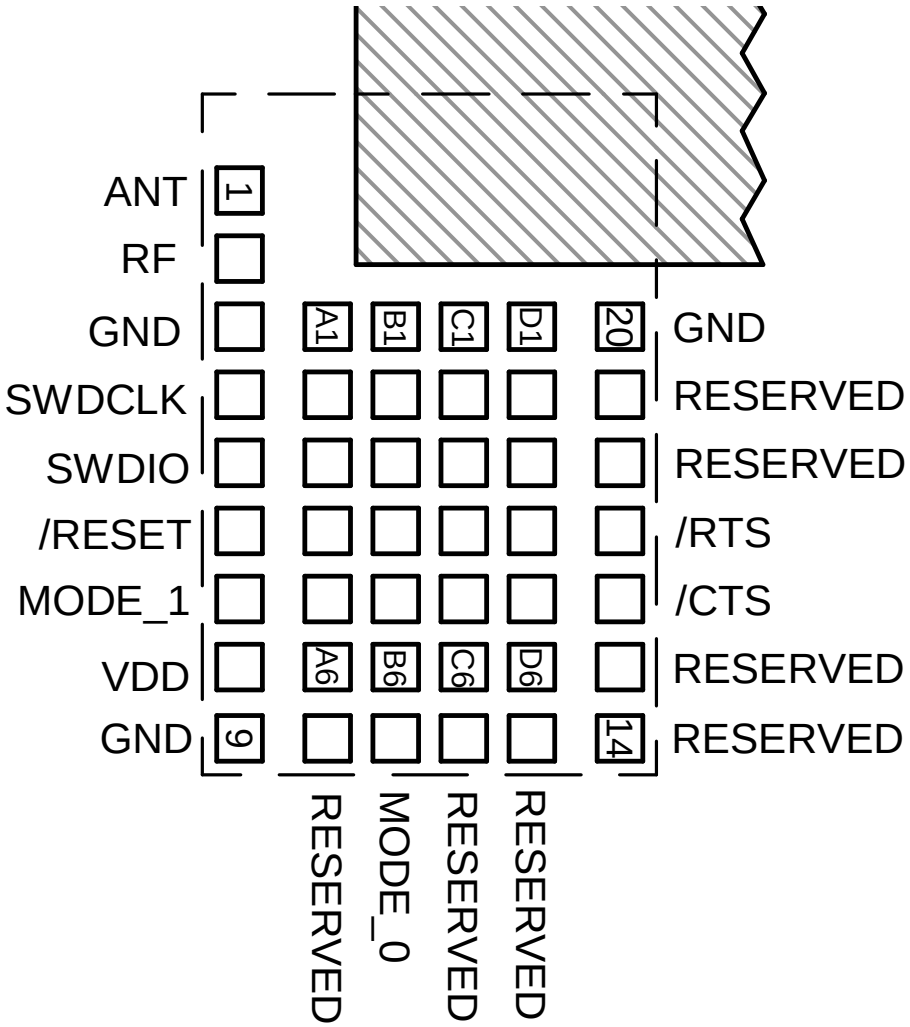


Figure 3: Pinout (top view)

No	µC Pin	Designation	I/O	Description
1		*ANT*	I/O	RF connection to PCB antenna. (see section 18)
2	*RF*	*RF*	RF	50 Ω RF connection through radio front end to transceiver part of chipset. (see section 18)
3	*GND*	*GND*	Supply	Ground
4	*SWDCLK*	*SWDCLK*	Input	Serial wire clock (SWD interface). Uses internal pull down resistor. Do not connect if not needed.

5	*SWDIO*	*SWDIO*	I/O	Serial wire input/output (SWD interface). Uses internal pull up resistor. Do not connect if not needed.
6	*/RESET*	*/RESET*	Input	Reset pin with internal pull-up. Triggers reset, when released to HIGH.
7	P2.05	*MODE_1*	Input	Operation mode pin with internal pull-down resistor during start-up. Otherwise floating.
8	*VDD*	*VDD*	Supply	Supply voltage
9	*GND*	*GND*	Supply	Supply voltage
10	P2.00	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
11	P1.05/AIN1	*MODE_0*	Input	Operation mode pin with internal pull-down resistor during start-up. Otherwise floating.
12	P1.00/XL1[1]	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
13	P1.01/XL2[1]	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
14	P2.01	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
15	P1.08	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
16	P1.07/AIN3	*/CTS*	Input	/CTS signal with internal pull-down resistor. Do not connect if not needed.
17	P1.06/AIN2	*/RTS*	Output	/RTS signal, if UART is enabled. High otherwise. Do not connect if not needed.
18	P1.02/NFC1[2]	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
19	P1.12/AIN5	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
20	*GND*	*GND*	Supply	Ground
A1	P0.00	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
A2	P2.10	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
A3	P2.09	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
A4	P2.06	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
A5	P2.04	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.

A6	P2.02	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
B1	P0.01	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
B2	P2.08	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
B3	*GND*	*GND*	Supply	Ground
B4	*GND*	*GND*	Supply	Ground
B5	P2.03	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
B6	P2.07	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
C1	*GND*	*GND*	Supply	Ground
C2	P1.10	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
C3	*GND*	*GND*	Supply	Ground
C4	*GND*	*GND*	Supply	Ground
C5	P1.03/NFC2[2]	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
C6	P1.04/AIN0	*UTXD*	Output	UART (Transmission), if UART is enabled. High otherwise.
D1	*GND*	*GND*	Supply	Ground
D2	P1.09	*LED_0*	Output	Indicates the module state (active high). Do not connect if not needed.
D3	P1.11/AIN4	*LED_1*	Output	Indicates the module state (active high). Do not connect if not needed
D4	P1.13/AIN6	*UART_ENABLE*	Input	Enable UART from sleep, see chapter 8.2.5. Uses internal pull-up. Active LOW pin. Do not connect, if not needed.
D5	P1.14/AIN7	RESERVED	–	Reserved pin for future use. High impedance, no pull. Do not connect.
D6	P1.15	*URXD*	Input	UART (Reception) with internal pull up resistor.

Table 15: Pinout

[1] Pins available to connect an external crystal in custom firmware. The standard firmware of Proteus-IV does not implement this function.
[2] NFC pins available for NFC function in custom firmware. The standard firmware of Proteus-IV does not implement this function.

6. Quick start

6.1. Minimal pin configuration

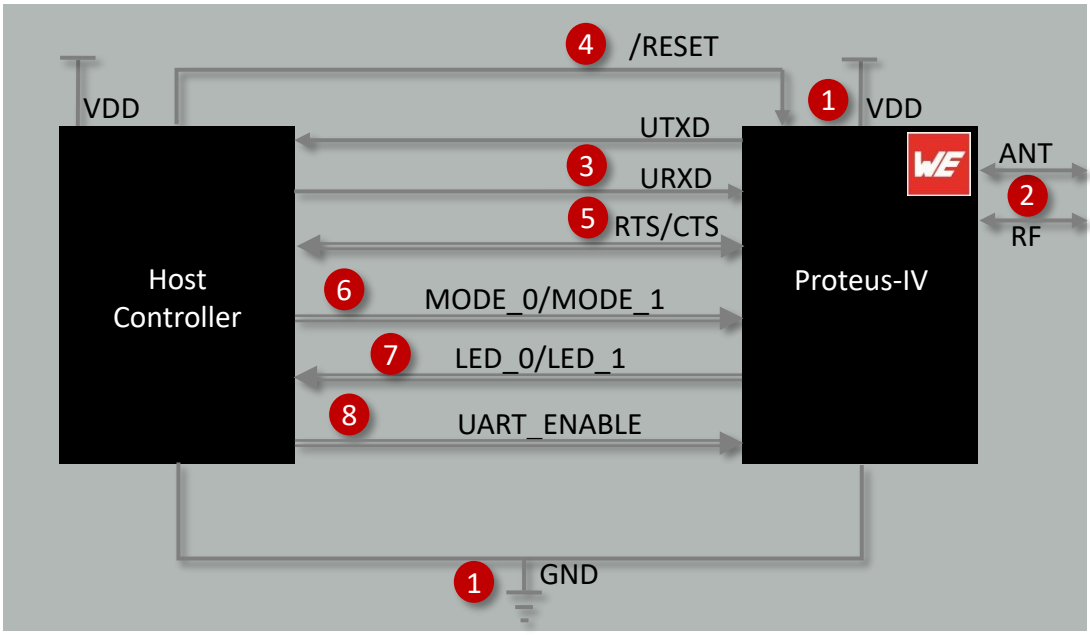


Figure 4: Minimal pin connections

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

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

2. Antenna connection
   Then the antenna configuration must be performed. See chapter 18.

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

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

5. UART flow control
   Connect the */RTS* and */CTS* pins to the host controller if UART flow control is to be used. This is mandatory for fast UART baud rates and highly recommended.

6. (Optional) Operation mode selection
   Connect the *MODE_0* and *MODE_1* pins to the host controller to switch between the operation modes.

7. (Optional) Status indication
   Connect the *LED_0* and *LED_1* pins to the host controller to allow easy indication of the status.

8. (Optional) UART enable
   For some application scenarios, it is beneficial to power down the UART temporarily to save power. To re-enable it, the *UART_ENABLE* pin is used.

6.2. Power up

After a stable power supply has been applied to the module, set the mode pins *MODE_0* and *MODE_1* depending on the desired operation mode (see chapter `Operation modes`). Then the */RESET* pin can be released to a high state to boot up the Proteus-IV.
If the module is not in transparent mode, the */RTS* line goes low and a start-up message is sent via UART to the connected host MCU as soon as the module has booted. From this point on, the Proteus-IV is ready to be controlled via commands by the host MCU.
In transparent mode, the host has to wait for the t_{boot} time, as there is no indication. The */RTS* will go low after the first Bluetooth® peer has connected and opened a Bluetooth® link.

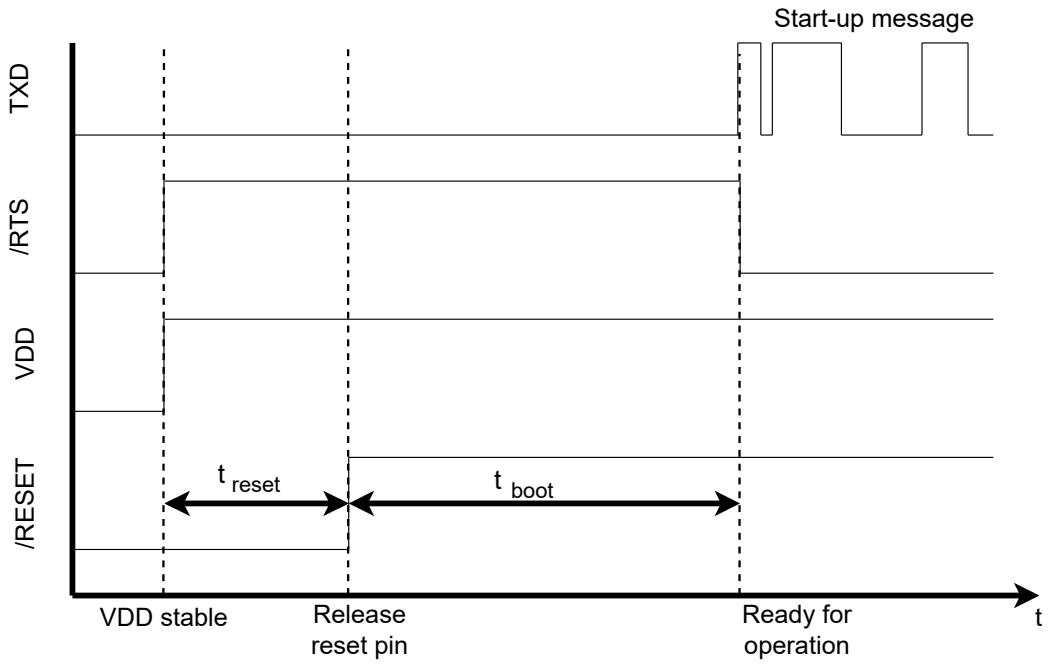


Figure 5: Power up

Variable	Value	Unit
t_{reset}	5	ms
t_{boot}	70	ms

Table 16: Start-up timings

6.3. Host connection

The factory configuration of the UART is 115200 Baud with flow control and a data format of 8 data bits, no parity, and 1 stop bit ("8n1"). The baud rate, flow control, and parity of the UART can be configured by means of the user setting `UART_ConfigIndex`.
The transmission of characters on the serial interface runs with secondary priority. For this reason, short interruptions may occur between the outputs of individual successive bytes.
On the other hand, the radio module starts interpreting the received UART bytes after a pause that must be larger than 10 byte durations and 1 ms.

6.4. Quick start example

This guide will walk you through your first wireless data transmission with the Proteus-IV. We'll explore both transparent mode and command mode, showing you how easy it is to get started. By the end of this guide, you'll have successfully sent data wirelessly from your smartphone to your PC!
As described in chapter `Operation modes`, the Proteus-IV supports several operation modes. In this chapter, we will focus on the two main operation modes: transparent mode and command mode.

What you'll need:

- Proteus-IV Evaluation Board (EV-Board)
- USB cable (provided with the EV-Board)
- PC with a serial terminal program installed
- Smartphone with WE Bluetooth LE Terminal app [3, 4] installed

6.4.1. Transparent mode

Transparent mode is the simplest way to get started. In this mode, data you send via Bluetooth® appears directly on the UART - no commands needed!

Step 1: Connect the EV-Board Take your Proteus-IV EV-Board and connect it to your PC using the provided USB cable. To ensure the module boots in transparent mode, configure the jumpers on JP1 as follows:

- *MODE_0*: Set to LOW (unset the jumper)
- *MODE_1*: Set to HIGH (set the jumper)

Make sure the UART jumpers on JP2 are installed for *RXD*, *TXD*, */RTS*, and */CTS*.

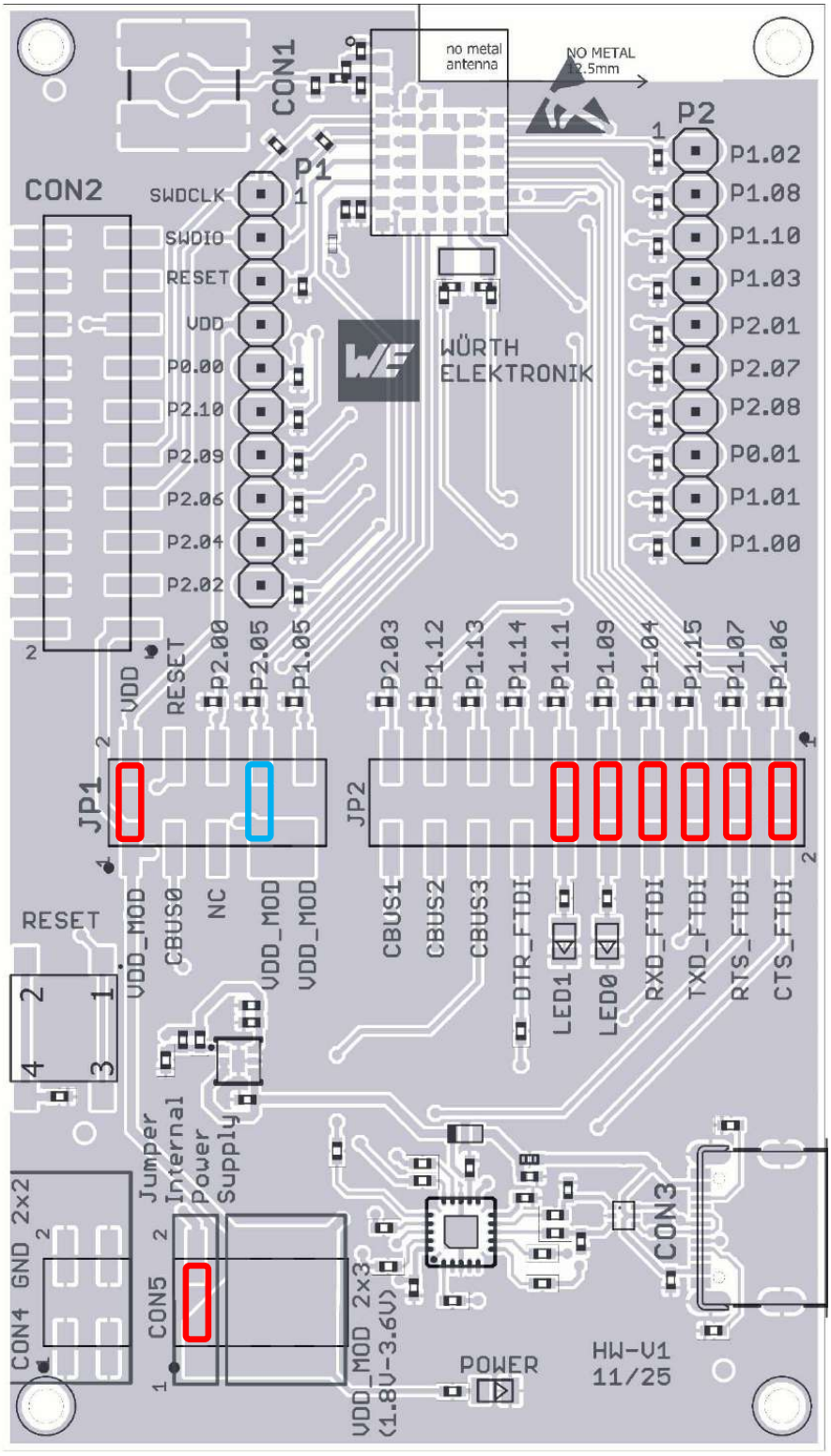


Figure 6: Proteus-IV EV-Board configured for transparent mode

The module will boot automatically and *LED_0* will start blinking, indicating it's advertising and ready for connections.

Step 2: Open the WE Bluetooth LE Terminal app Open the WE Bluetooth LE Terminal app on your smartphone. You should see the home screen with a "Scan" button.

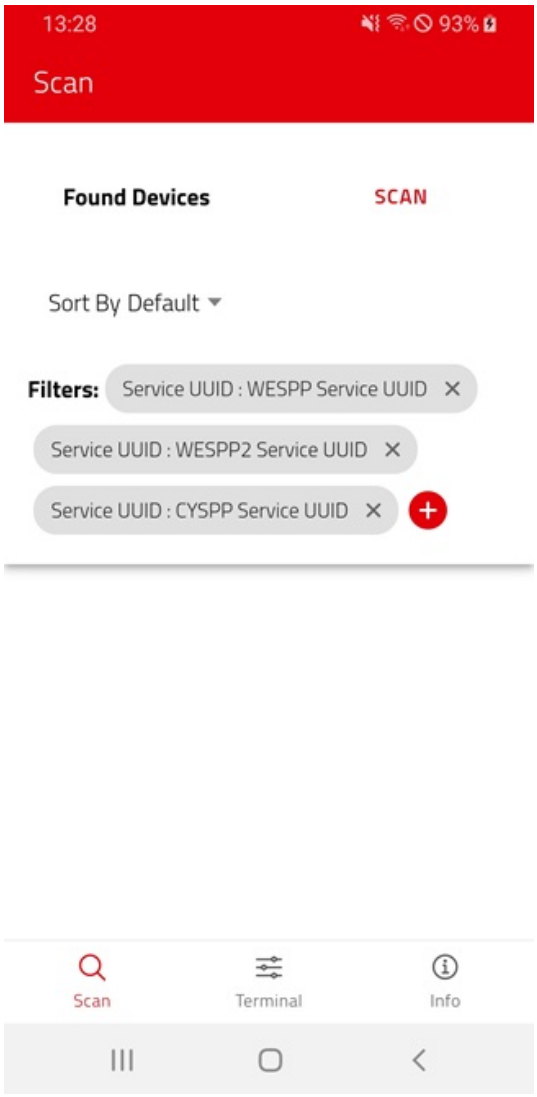


Figure 7: WE Bluetooth LE Terminal app home screen

Step 3: Scan for devices Tap the "Scan" button on the home screen. After a short moment, your device named "Proteus-IV" (or the custom name you configured) should appear in the device list. Tap on the device name to initiate the connection.

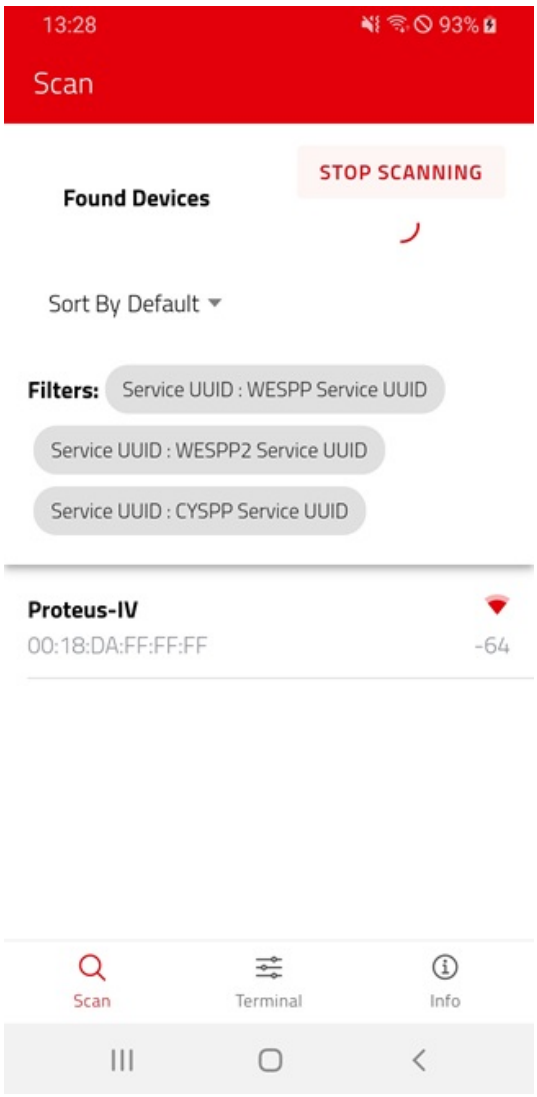


Figure 8: Scanning for Bluetooth® devices

Step 4: Select the module type A "Select Module" popup will appear. Choose "Proteus-IV" as the module type, then press the "Select" button. This tells the app which protocol to use for communication.

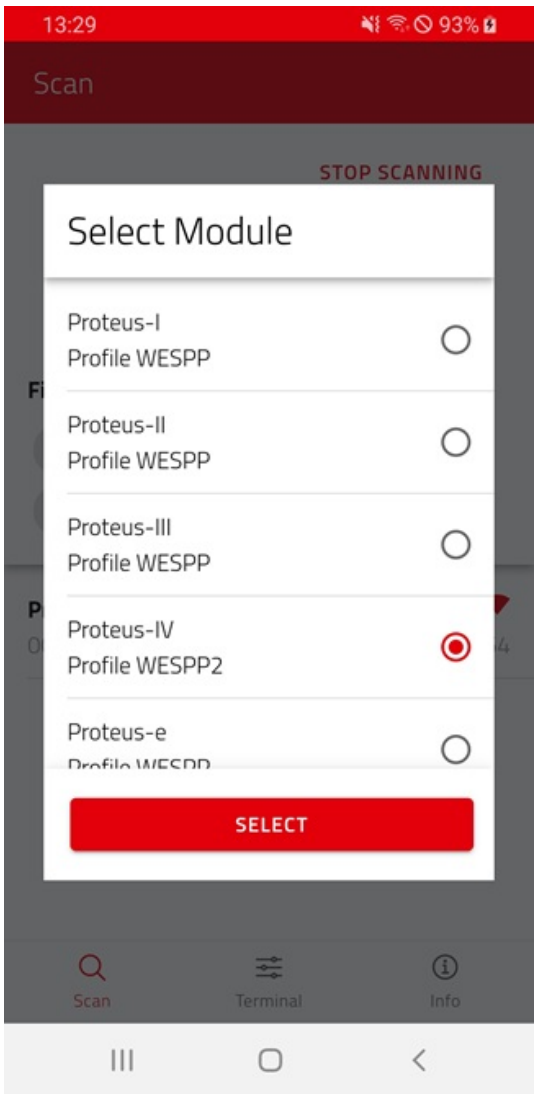


Figure 9: Selecting the Proteus-IV module type

Step 5: Connection established The app will now connect to your module and automatically set up the communication channel. Watch the EV-Board - *LED_1* should now be solid ON, indicating an active connection.

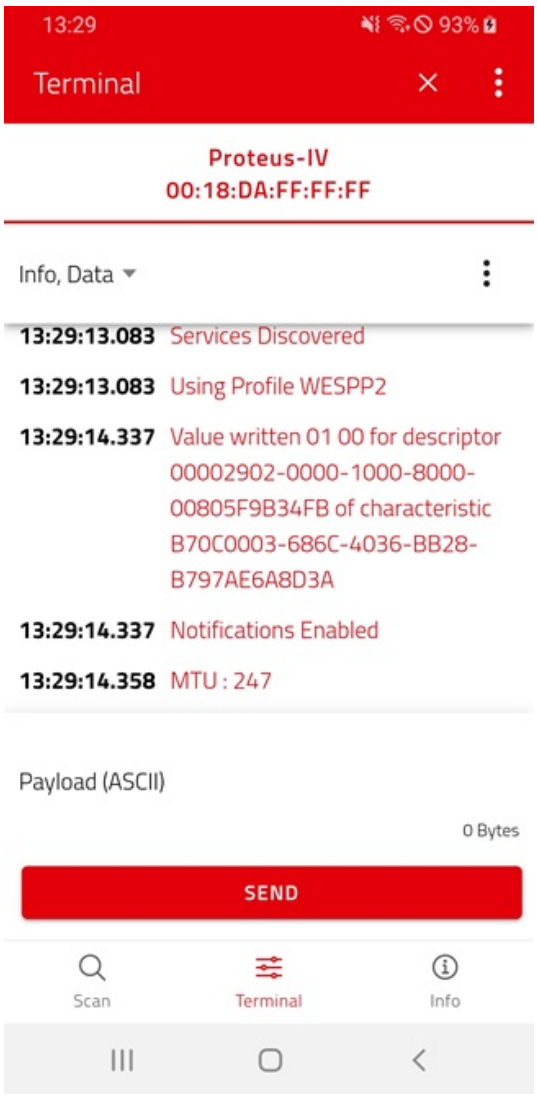


Figure 10: Successfully connected to Proteus-IV

Step 6: Connect your PC terminal On your computer, open a serial terminal program. We recommend using HTerm, PuTTY, or the WE UART Terminal. If you connected your EV-Board in Step 1, a COM port should now be available. Connect to this COM port using the following settings:

- Baud rate: 115200
- Data format: 8N1 (8 data bits, no parity, 1 stop bit)
- Flow control: RTS and CTS (hardware flow control enabled)
- \r\n as ETX

Leave the COM port connection open - you're now ready to receive data!

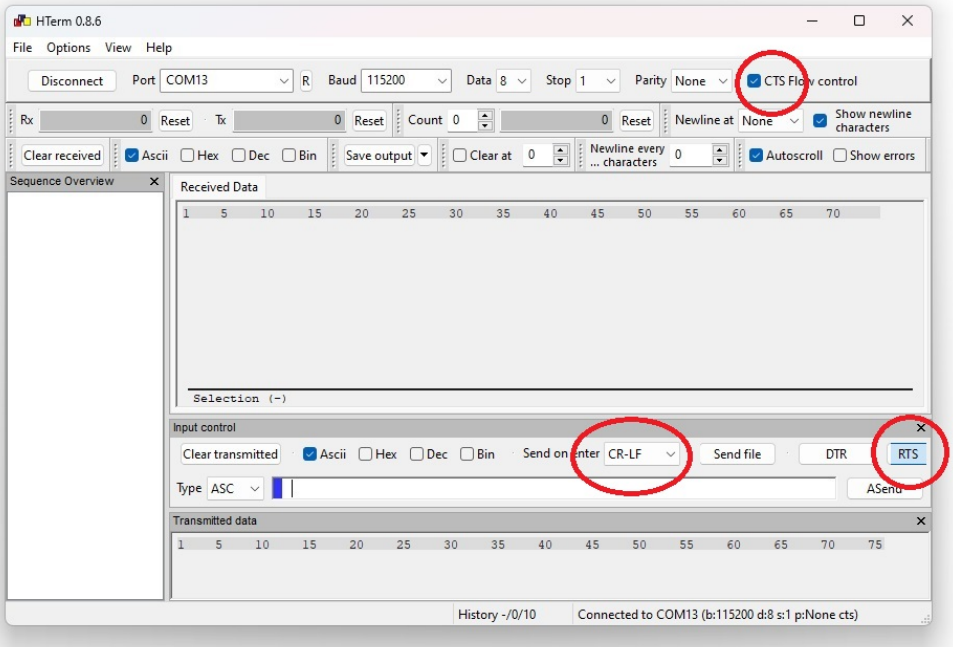


Figure 11: Serial terminal configured and connected

Step 7: Send data from your smartphone Go back to your smartphone and the WE Bluetooth LE Terminal app. Type a message in the text field at the bottom of the screen. For this example, let's type "Transparent Hello!". When you're ready, press the "Send" button.

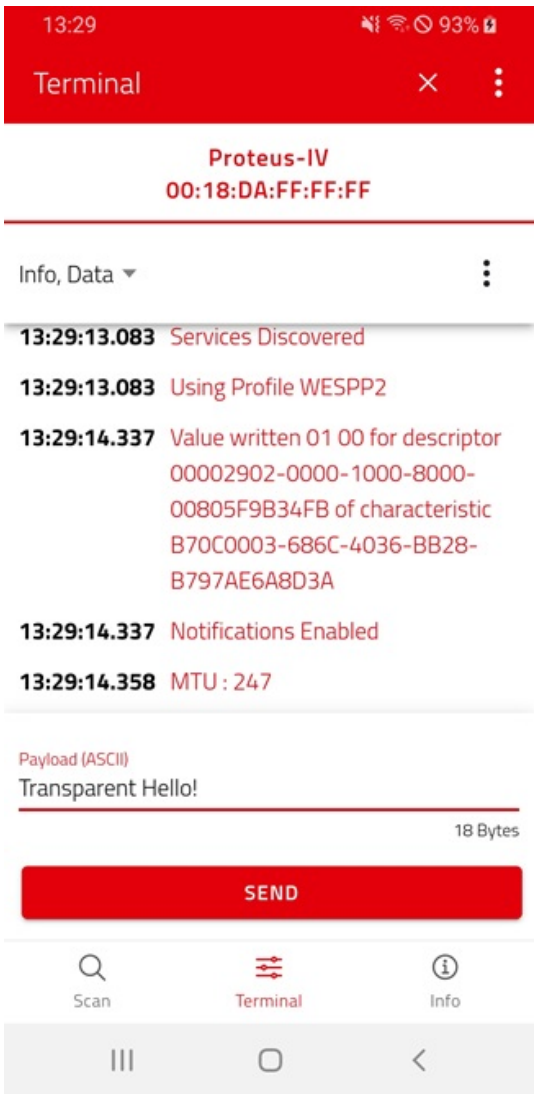


Figure 12: Typing a message to transmit

Step 8: Transmission confirmation The message "Transparent Hello!" is transmitted wirelessly. The app confirms the successful transmission with a "Data sent" notification.

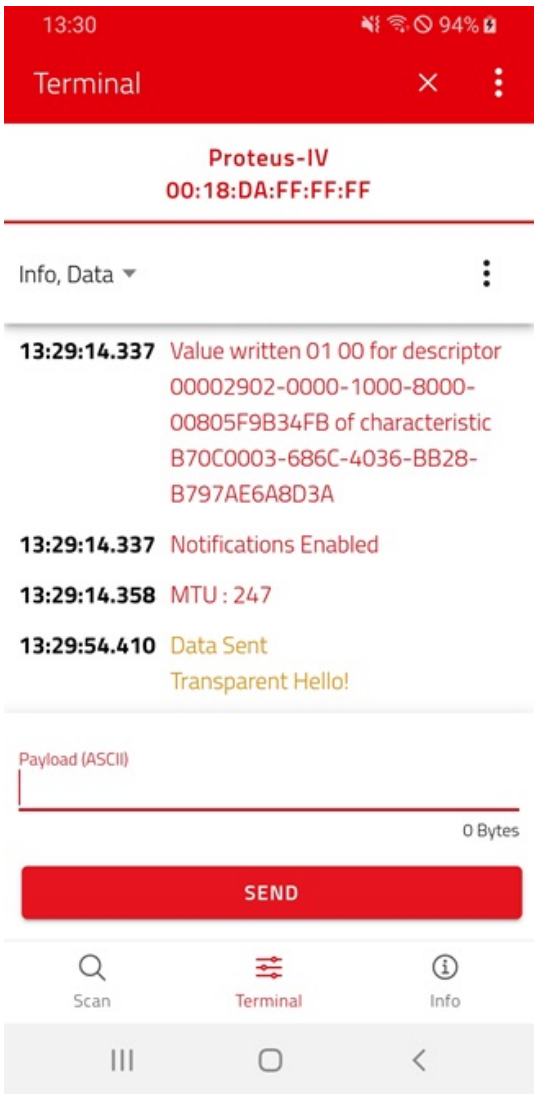


Figure 13: Data transmission confirmed

Step 9: Receive data on your PC Switch to your PC and look at the serial terminal. The message appears immediately! You can see "Transparent Hello!" printed on the screen - the exact text you sent from your smartphone.

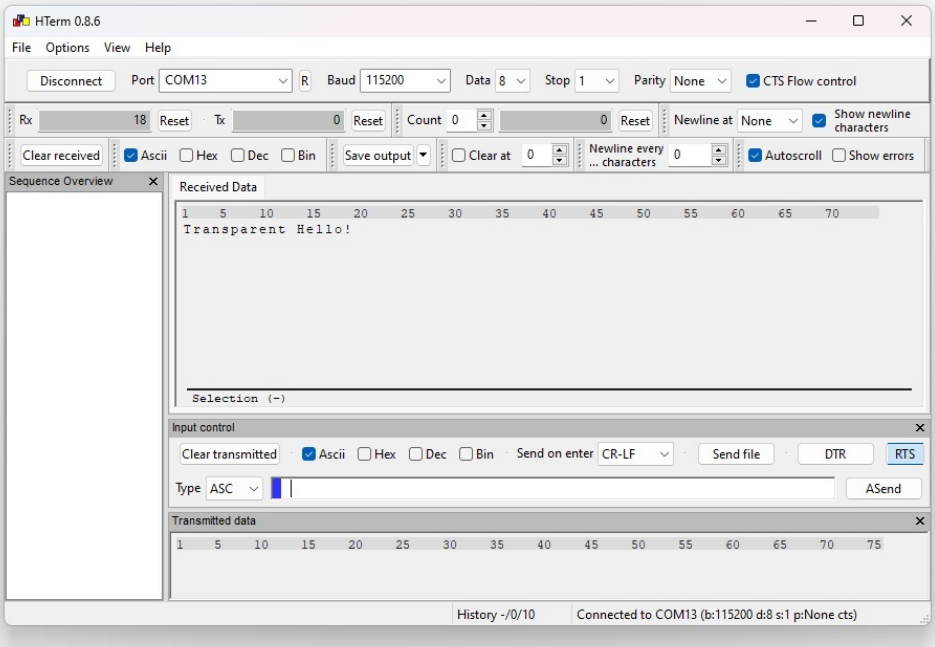


Figure 14: Message received on PC

Step 10: Transmit data from your PC Now let's transmit data from the PC to your smartphone. Go to your PC terminal software, for example HTerm. In the text box where you would type strings to transmit, enter `"Hello from computer!"`. After you finish typing, simply press Enter. You should see the string sent out at the bottom of the screen in the "Transmitted data" window.

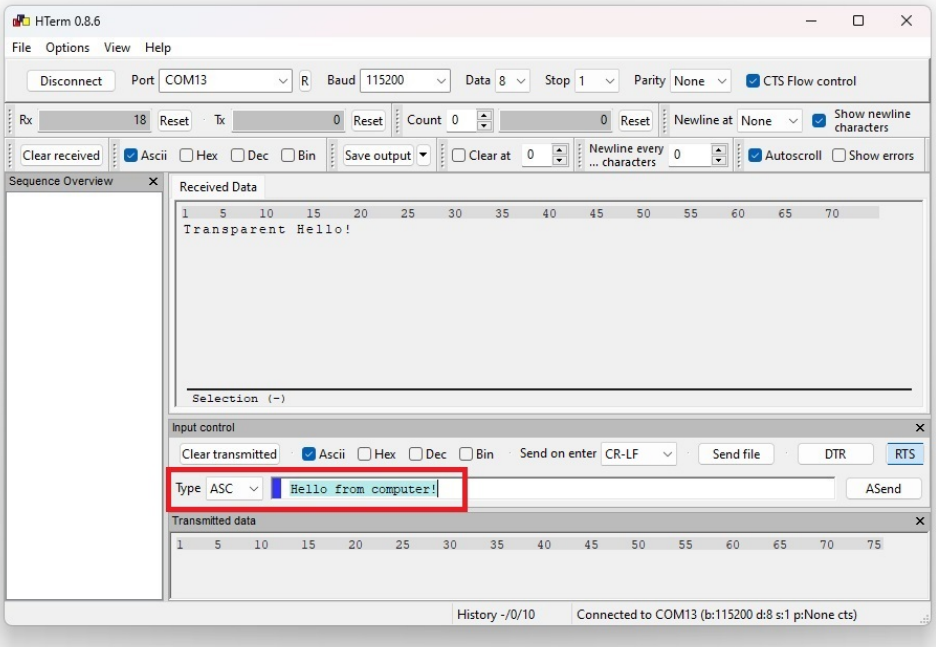


Figure 15: Transmitting data from PC in transparent mode

Step 11: Receive data on your smartphone Now take a look at your smartphone in the WE Bluetooth LE Terminal app. The message `"Hello from computer!"` appeared there! This confirms successful transmission from PC to smartphone.

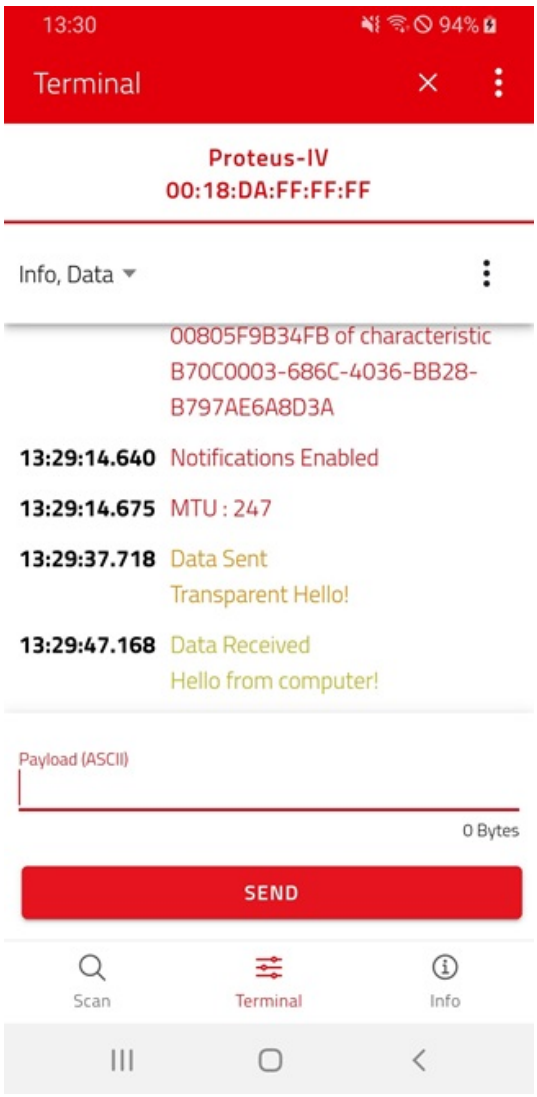


Figure 16: Message received on smartphone in transparent mode

What you've accomplished: Congratulations! You've just completed your first wireless data transmission using the Proteus-IV in transparent mode! The data traveled from your smartphone via Bluetooth® to the Proteus-IV module, then through the UART to your PC - all transparently without any command protocols.
You can use this simple approach to:

- Control devices wirelessly by sending commands like "ON" or "OFF"
- Transmit sensor data from remote locations
- Create wireless serial port replacements
- Send any kind of data - text, numbers, or binary data
- Automate processes with simple string commands

Transparent mode is perfect for applications where you want the simplest possible wireless serial communication!

6.4.2. Command mode

Command mode gives you full control over the module's functions. In this mode, all data is wrapped in command frames, giving you access to connection information, status feedback, and advanced features.

Step 1: Connect the EV-Board Take your Proteus-IV EV-Board and connect it to your PC using the provided USB cable. To ensure the module boots in command mode, configure the jumpers on JP1 as follows:

- *MODE_0*: Set to LOW (unset the jumper)
- *MODE_1*: Set to LOW (unset the jumper)

Make sure the UART jumpers on JP2 are installed for *RXD*, *TXD*, */RTS*, and */CTS*.

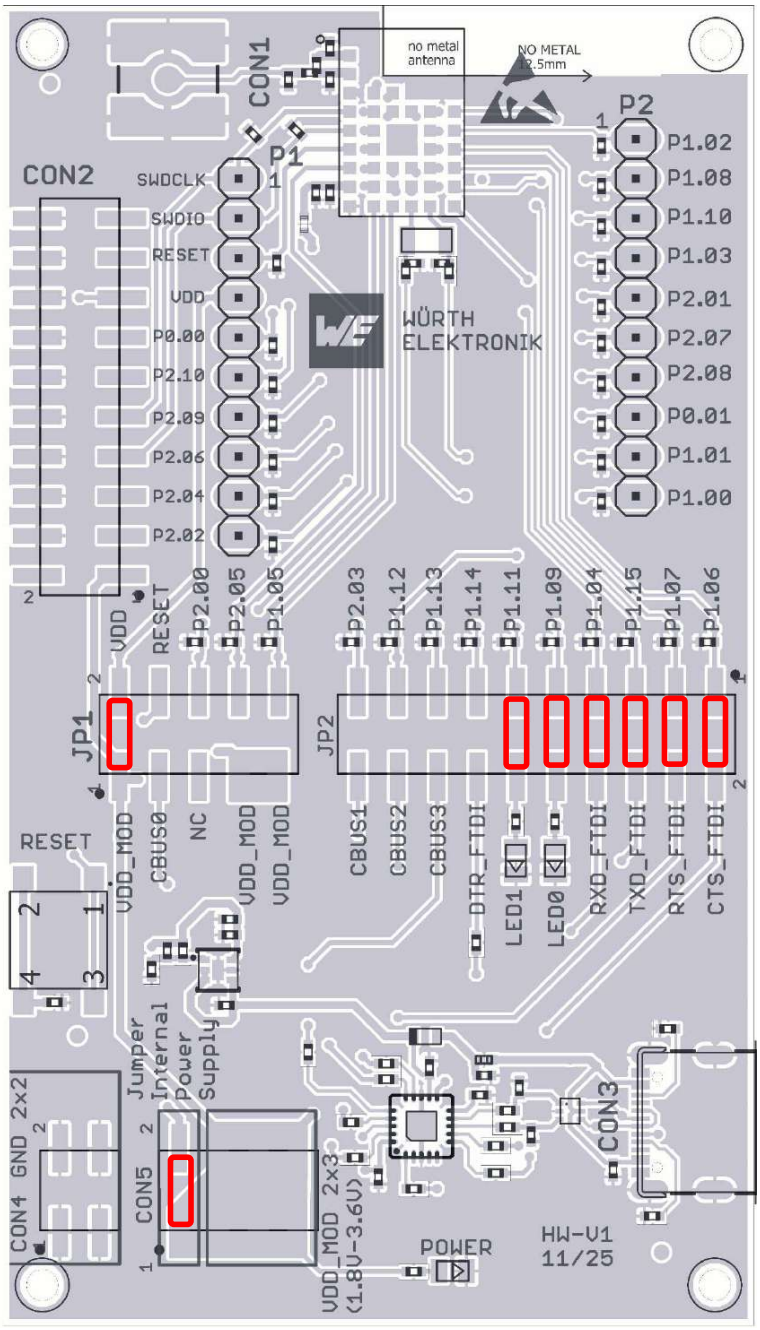


Figure 17: Proteus-IV EV-Board configured for command mode

The module will boot automatically and *LED_0* will start blinking, indicating it's advertising and ready for connections.

Step 2: Open the WE Bluetooth LE Terminal app Open the WE Bluetooth LE Terminal app on your smartphone. You should see the home screen with a "Scan" button.

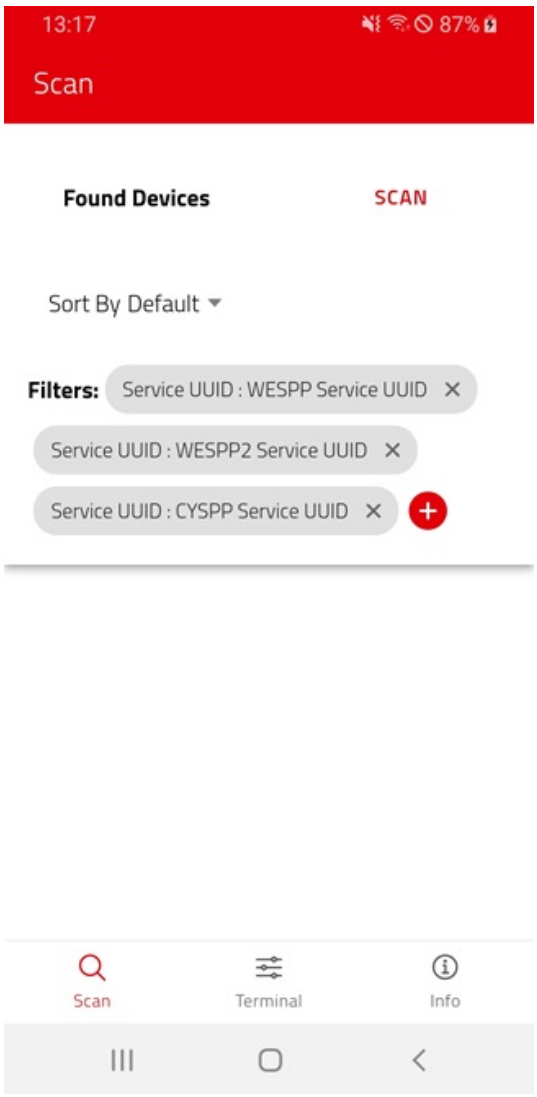


Figure 18: WE Bluetooth LE Terminal app home screen

Step 3: Scan for devices Tap the "Scan" button on the home screen. After a short moment, your device named "Proteus-IV" (or the custom name you configured) should appear in the device list. Tap on the device name to initiate the connection.

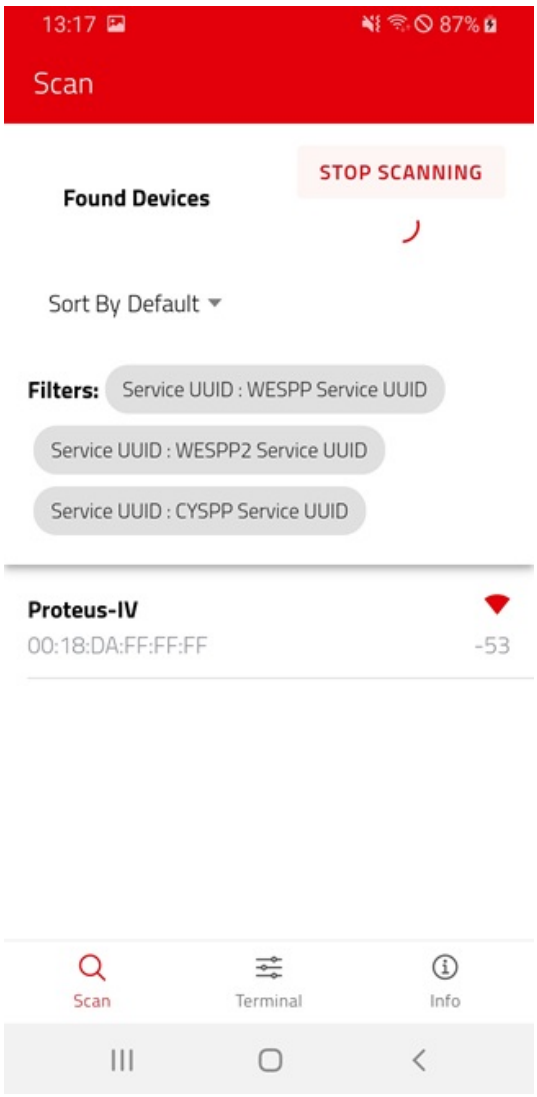


Figure 19: Scanning for Bluetooth® devices

Step 4: Select the module type A "Select Module" popup will appear. Choose "Proteus-IV" as the module type, then press the "Select" button. This ensures the app uses the correct protocol for communication.

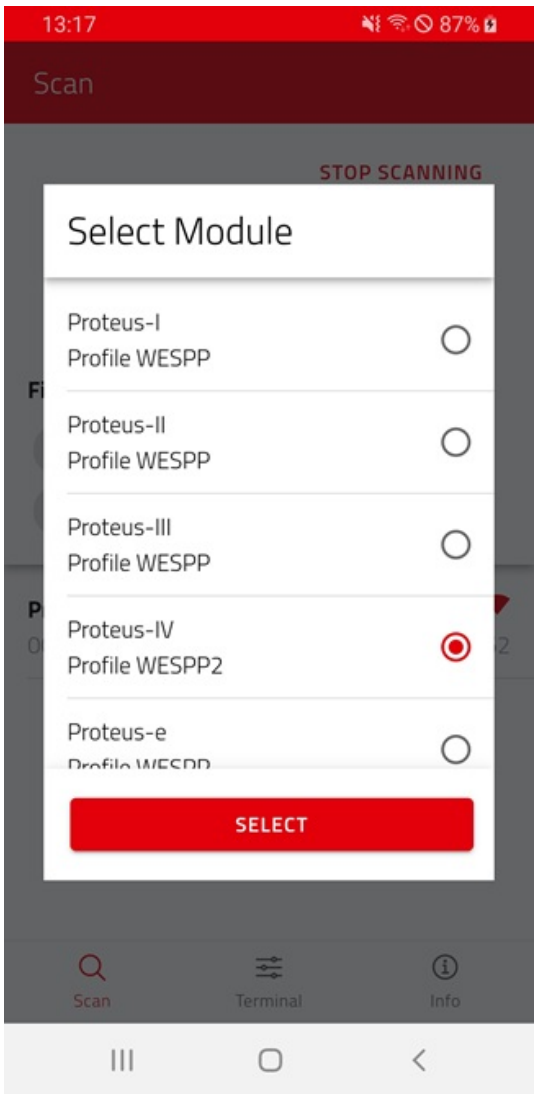


Figure 20: Selecting the Proteus-IV module type

Step 5: Connection established The app will now connect to your module and automatically set up the communication channel. Watch the EV-Board - *LED_1* should now be solid ON, indicating an active connection.

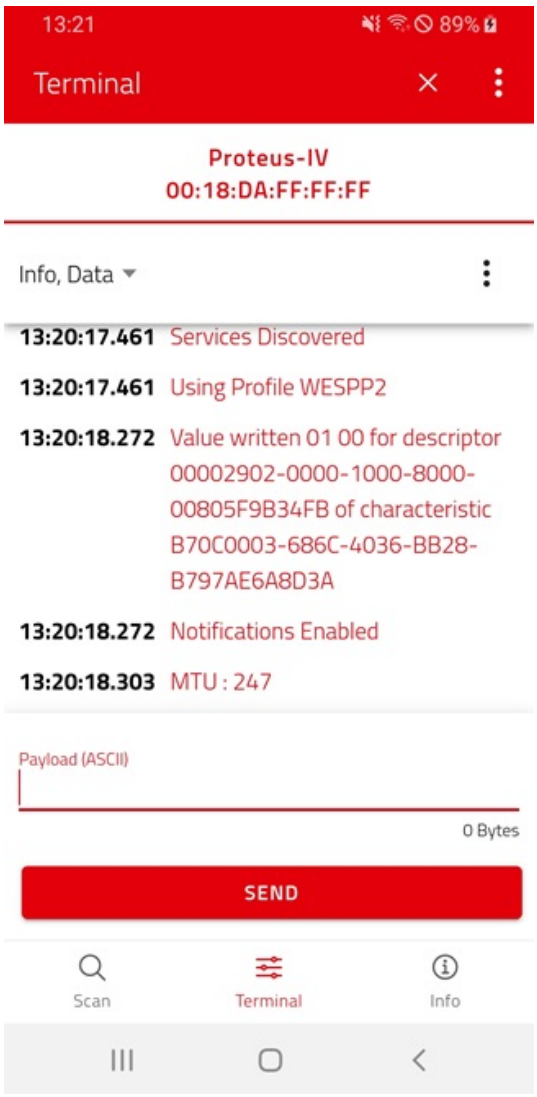


Figure 21: Successfully connected to Proteus-IV

Step 6: Connect your PC terminal On your computer, open a serial terminal program. While you can use any terminal software like HTerm or PuTTY, we highly recommend using the official WE UART Terminal software for command mode. This specialized tool automatically decodes incoming command messages, making it much easier to understand what's happening.
If you connected your EV-Board in Step 1, a COM port should now be available. Open the WE UART Terminal software and select "Proteus-IV" as the module type - this automatically configures all the correct settings (115200 baud, 8N1, CTS flow control). If using another terminal, configure these settings manually:

- Baud rate: 115200
- Data format: 8N1 (8 data bits, no parity, 1 stop bit)
- Flow control: CTS (hardware flow control enabled)

Leave the COM port connection open - you're now ready to receive command messages!

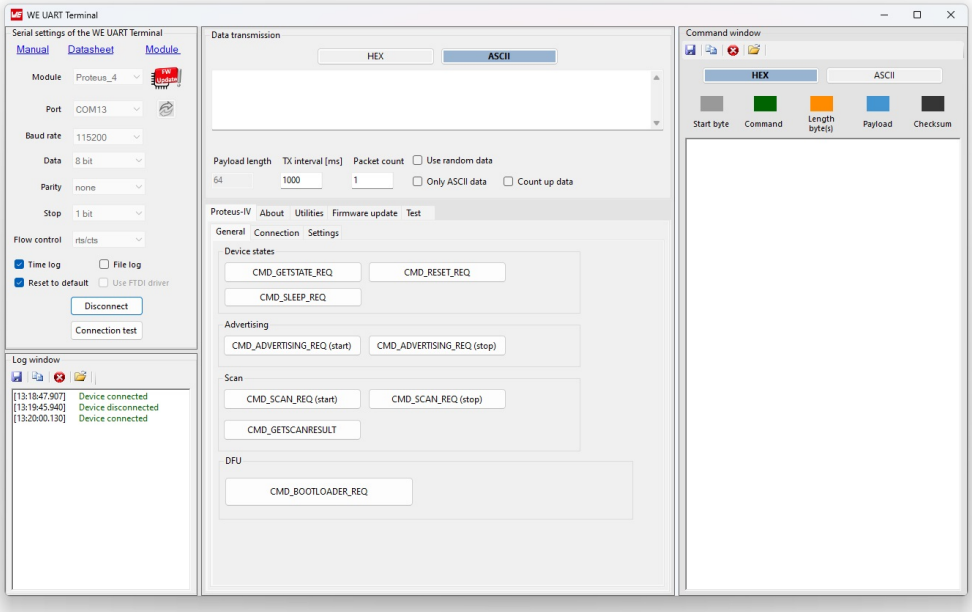


Figure 22: WE UART Terminal configured and connected

Step 7: Send data from your smartphone Go back to your smartphone and the WE Bluetooth LE Terminal app. Type a message in the text field at the bottom of the screen. For this example, let's type "Hello Proteus!". When you're ready, press the "Send" button.

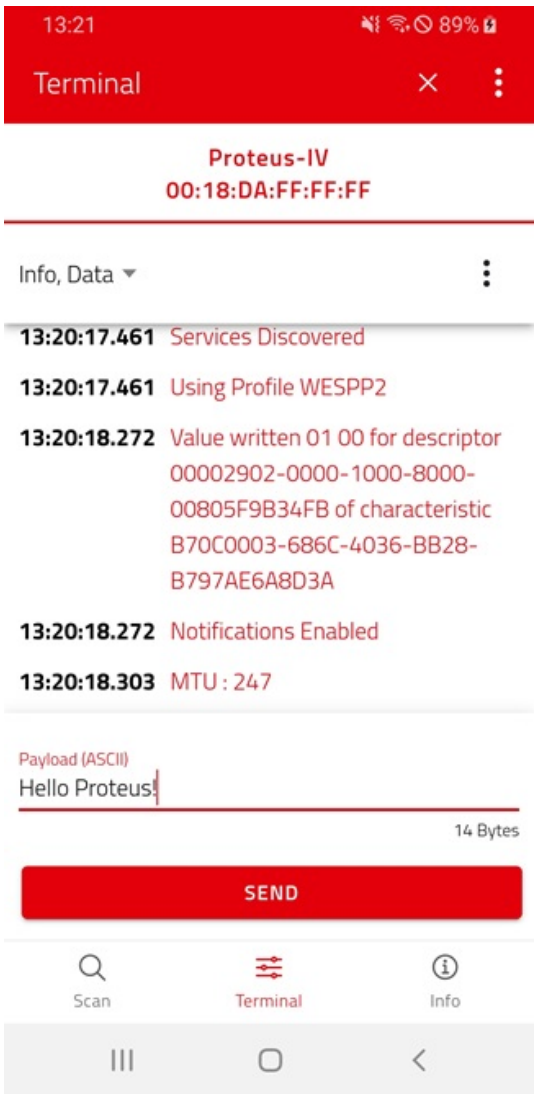


Figure 23: Typing a message to transmit

Step 8: Transmission confirmation The message "Hello Proteus!" is transmitted wirelessly. The app confirms the successful transmission with a "Data sent" notification.

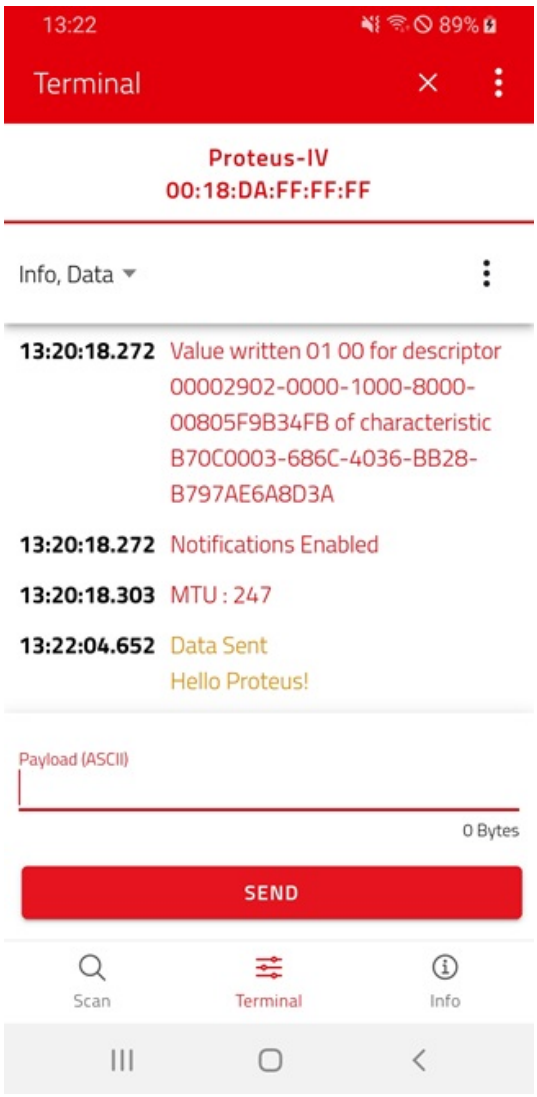


Figure 24: Data transmission confirmed

Step 9: Receive data on your PC Switch to your PC and look at the WE UART Terminal. The message appears immediately - but notice it's wrapped in a command frame! You can see the complete `CMD_DATA_IND` message with the payload containing "Hello Proteus!" (48 65 6C 6C 6F 20 50 72 6F 74 65 75 73 21 in hexadecimal).

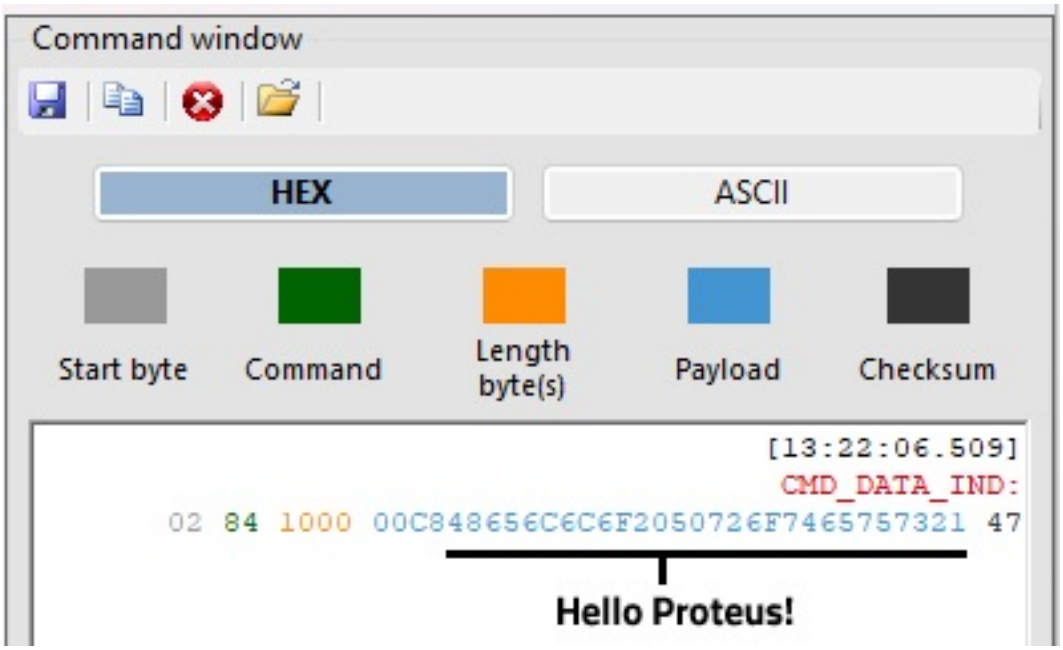


Figure 25: Command message received and decoded on PC

The WE UART Terminal software helpfully decodes this for you, showing:

- Command type: `CMD_DATA_IND` (data received indication)
- Connection ID: 0x00 (the first connection)
- RSSI: Signal strength of the received packet
- Payload: Your message "Hello Proteus!" in both hex and ASCII

Step 10: Transmit data from your PC Now let's transmit data from the PC to your smartphone. On your PC, open the WE UART Terminal software and select the tab "Connection". Type the string you want to transmit into the upper large text box at the top of the window (marked in red in the screenshot). For this example, type `"Hello from computer!"`. After typing, click the button `CMD_DATA_REQ` (also marked in red). You should see the `CMD_DATA_REQ` command sent out on the right side of the screen in the command window.

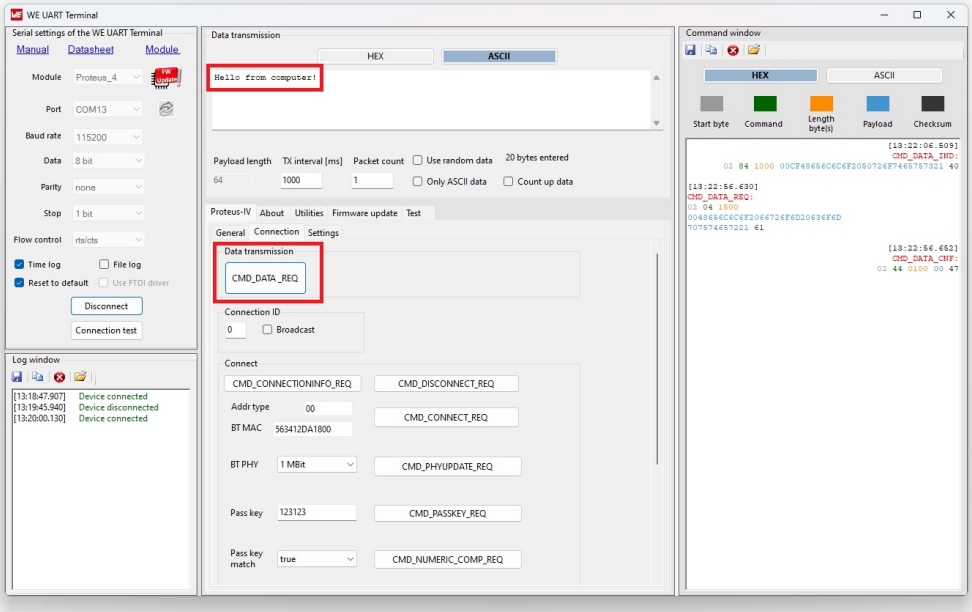


Figure 26: Sending a string from PC in command mode

Step 11: Receive data on your smartphone Now take a look at your smartphone in the WE Bluetooth LE Terminal app. The message `"Hello from computer!"` appeared there! This confirms successful transmission from PC to smartphone in command mode. Unlike transparent mode, here the data was not sent as a raw string but wrapped inside a `CMD_DATA_REQ` command, which allows you to control many parameters such as connection ID, payload size, and transmission confirmation options.

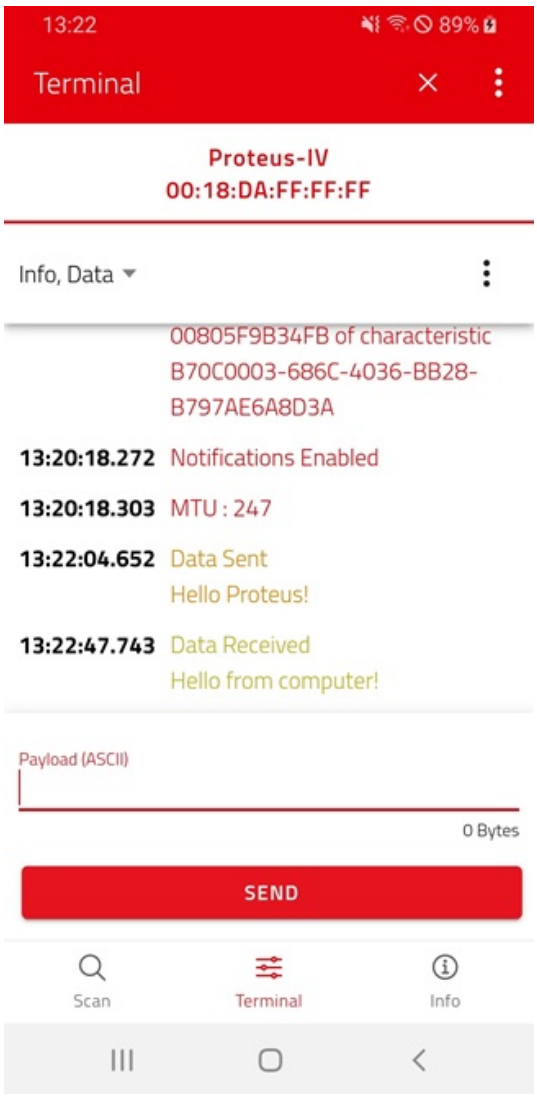


Figure 27: Message received on smartphone in command mode

What you've accomplished: Congratulations! You've successfully completed your first wireless data transmission using the Proteus-IV in command mode! Unlike transparent mode, command mode gives you detailed information about every communication event.
Command mode enables you to:

- Track which device sent data using connection IDs
- Monitor signal strength (RSSI) for each received packet
- Manage multiple simultaneous connections
- Implement security features like passkey authentication
- Get confirmation for every operation
- Control all aspects of Bluetooth® communication
- Send control commands like "ON" or "OFF" to automate devices

- Transmit sensor readings, status information, or any data type
- Build sophisticated IoT applications with full protocol control

Command mode is perfect for professional applications where you need full visibility and control over your wireless communication!

Next steps: Now that you've mastered the basics in both modes, explore the detailed tutorials in chapter `Use cases and examples` to learn advanced features like:

- Configuring module settings (device name, TX power, security)
- Acting as a central device to connect to other peripherals
- Implementing custom beacon broadcasting
- Using sleep mode for ultra-low power operation
- Updating firmware over-the-air (FOTA)

The Proteus-IV is a powerful and flexible Bluetooth® LE module - you're just getting started!

7. Functional description

7.1. Operation modes

The Proteus-IV acts as a slave and can be fully controlled by an external host via UART. It supports the following operating modes:

- The **command mode**, where the Proteus-IV can be controlled by the host controller via UART commands. The command mode allows the use of all functions of the radio module. Functions like data transmission via Bluetooth® LE or configuration tasks can be triggered by predefined commands (see chapter `Command mode`) that are sent as telegrams over the UART interface.
- The **config mode** is a reduced version of the command mode, where only commands for restoring and configuration are enabled. Radio-related commands are not supported, and the radio is turned off.
  In config mode, a fixed UART configuration (9600 Baud, no parity, no flow control) is used, which allows restoration or configuration of the radio module with known UART settings.
  For command description, please see chapter `Command mode`.

While configuring the radio module using `CMD_SET_REQ` or other commands, the module reboots several times. It is highly recommended to constantly hold the pins *MODE_0* and *MODE_1* at the corresponding level until all configuration tasks have been completed.

- The **transparent mode** (see chapter `Transparent mode`) provides a transparent UART interface. This mode supports only data transmission to all peer devices that are connected via Bluetooth® LE. The data transmission can be performed by the host without using any commands.
- The **FOTA mode** (see chapter `15.2`) enables the SMP service, which provides the interface to update the firmware of the radio module via the Bluetooth® LE interface.

On boot-up of the radio module, the host can select the operation mode using the pins *MODE_0* and *MODE_1*. A module-internal pull-down resistor is applied temporarily to these pins during boot-up.

MODE_0	*MODE_1*	Operation mode
LOW	LOW	Command mode (default)
LOW	HIGH	Transparent mode
HIGH	LOW	Config mode
HIGH	HIGH	FOTA mode

Table 17: Operation modes

Switching between the modes during runtime is not possible. To boot into the desired operation mode, apply the corresponding voltage level to the pins and perform a pin reset by pulling the */RESET* pin LOW for 10 ms and releasing it to HIGH again.

7.2. State indication using the LED pins

The pins *LED_0* and *LED_1* of the Proteus-IV can be used to determine the module state.

State	*LED_0*
Advertising	Blinking (On for 1000 ms, Off for 1000 ms)
Other	Off

Table 18: *LED_0* behavior of the Proteus-IV

State	*LED_1*
Radio link is open	On
Other	Off

Table 19: *LED_1* behavior of the Proteus-IV

7.3. Low power operation

Depending on the end application, 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 current consumption are listed below:

- `CMD_SLEEP_REQ`: This command puts the module into sleep mode, where it consumes the lowest current. Both the UART and the Bluetooth® LE interface are shut down. See `Sleep mode` for more information.
- `CMD_UARTDISABLE_REQ`: This command disables the UART interface. It is enabled again as soon as the module is reset/woken or when the module outputs a message, e.g., when a connection request has been received or the *UART_ENABLE* pin of the module is used.
- `RF_TXPower`: This setting can be used to configure the output power of the module in radio transmission mode at the cost of reduced radio range.
- `RF_AdvertisingInterval` and `RF_ConnectionInterval`: These parameters define the timing behavior of the Proteus-IV when advertising or during an open connection. The larger these intervals are, the less often data is transmitted via radio and thus power consumption decreases.
- Advertising: The `RF_MaxPeripheralConnections` parameter defines the maximum number of peripheral connections. If the maximum number of connections is not reached, the radio module is still advertising, which consumes extra power. If the end device does not need additional Bluetooth® connections, reduce the value of `RF_MaxPeripheralConnections` or turn advertising off using the `CMD_ADVERTISING_REQ` command.
- Scanning: Scanning is the most power-consuming radio action. To reduce power consumption, keep the scanning times as short as possible (see `CMD_SCAN_REQ`).
- The 2 MBit radio mode transmits data packets faster and thus reduces power consumption slightly.

7.3.1. Sleep mode

Especially for battery-powered devices, the sleep (system-off) mode supports very low power consumption. It can be entered by sending the command `CMD_SLEEP_REQ` to the module. As a response, the module will send a `CMD_SLEEP_CNF` and then enter sleep mode. Sleep mode can be started only if no peer device is connected via radio.
To leave sleep mode, the module has to be woken up by a pin reset: apply a LOW signal to the *RESET* pin for at least 5 ms before releasing the signal back to HIGH. The module then restarts completely, so that all volatile settings are set to default. A `CMD_STARTUP_IND` will be sent when the module is ready for operation again.

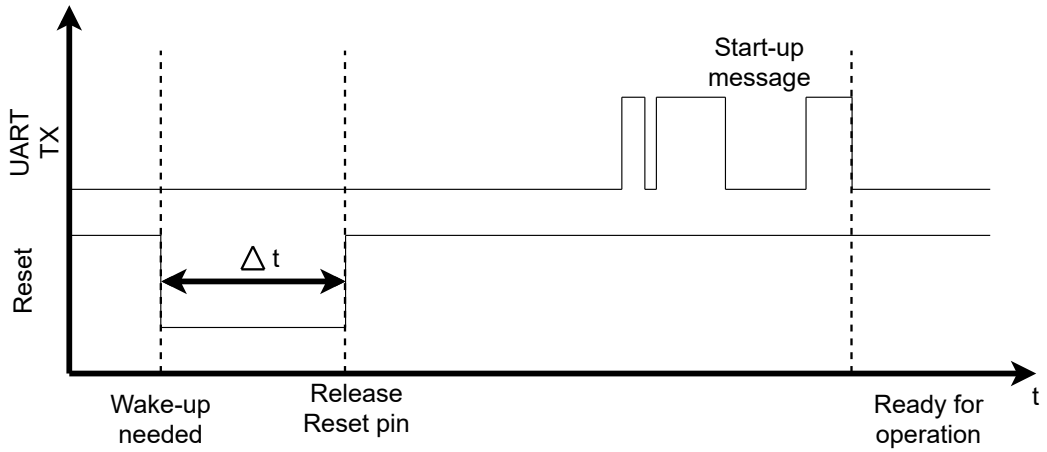


Figure 28: Power up

In sleep mode, the UART is disabled. Thus the module will not receive or transmit any data.

7.4. Bluetooth® radio behavior

7.4.1. Identification of a Proteus-IV device on air

The Proteus-IV can be identified on the radio interface by its 6-byte-long Bluetooth®-compliant MAC address `FS_BTMAC`, which is part of the data package sent during advertising mode. By default, the `FS_BTMAC` consists of the Würth Elektronik eiSos MAC ID 0x0018DA followed by the module's serial number `FS_SerialNumber`. Nevertheless, the `FS_BTMAC` can be modified by a `CMD_SET_REQ`.
To simplify the identification of Proteus-IV devices on the Bluetooth® LE interface, a short user-defined name (see user setting `RF_DeviceName`) can be given to the module, which is also part of the advertising packet.

7.4.2. Multi-connect - Parallel Bluetooth® connections to multiple peers

The Proteus-IV provides the so-called multi-connect feature, which allows holding several parallel connections at the same time.
It supports one central connection and up to 5 peripheral connections in parallel. The maximum number of peripheral connections must be configured using the user setting

RF_MaxPeripheralConnections. The number of (parallel) central connections is limited to one.

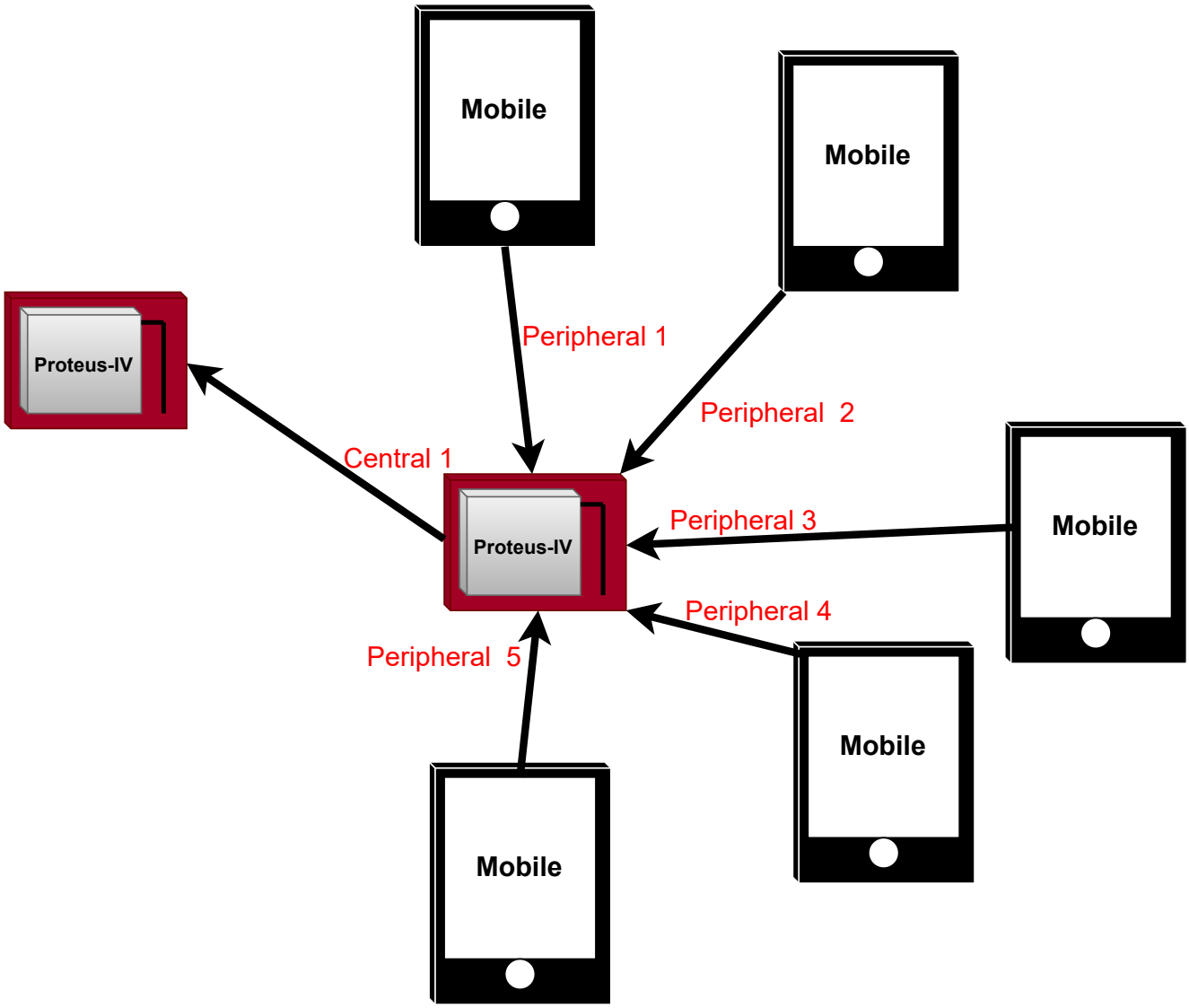


Figure 29: Multiple connections in parallel

A central connection to another peripheral can be set up by the CMD_CONNECT_REQ command. If an active central connection already exists, the module will return a CMD_CONNECT_CNF with status 0x10 (no free connection) on any new CMD_CONNECT_REQ.

When acting as a peripheral, the radio module will run advertising as long as the number of active peripheral connections is lower than the configured maximum value RF_MaxPeripheralConnections. This signals that it is ready to accept additional peripheral connections. If the maximum number is reached, the Proteus-IV will stop advertising and thus block any additional connection setup.

For each active connection, a unique Conn_ID identifies the connection. It will be created by the CMD_CONNECT_IND command, which is the first of several commands sent by the radio module during connection setup. This Conn_ID must be used in all connection-related com-

mands (i.e., connection handling and data transfer). For data transfer using the CMD_DATA_REQ command, a broadcast Conn_ID can be used in addition, which allows transmitting data to all connected Bluetooth® LE peer devices.

7.4.3. Connection setup

The intended way of data exchange using Bluetooth® LE is connection-based. To establish a Bluetooth® LE connection, several steps need to be run subsequently. Figure 30 shows the steps performed during connection setup:

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

2. Pairing process
   The authentication and exchange of encryption information is part of the pairing process. The central device must request at least the same security level to access the characteristics of the Proteus-IV. The security level and method depend on the configuration of both connection partners; see chapter Bluetooth® security.

If the pairing is not initiated by the central device, it cannot access the characteristics of the peripheral.

3. Exchange of the maximum transmission unit (MTU) (optional)
   The maximum transmission unit can be increased to allow the transmission of larger data packets. The Proteus-IV allows an MTU of up to 247 bytes, which results in a maximum payload size (MPS) of 244 bytes. Not selecting a higher MTU will use the Bluetooth® LE 4.0 default MTU, which results in an MPS of 20 bytes, but will be compatible with pre-Bluetooth® LE 4.2 devices.

4. Discover the characteristics of the Proteus-IV **SPPlikeV2** profile
   The characteristics offered by the Proteus-IV have to be discovered by the central.

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

If a Proteus-IV is used as a central device, all described steps will be run by the radio module in the background once the connected host sends a CMD_CONNECT_REQ command to the Proteus-IV.

If a smartphone is used as a central device, a smartphone app needs to be implemented that runs the described steps. As a reference, the WE Bluetooth LE Terminal app [5, 3, 4] is available in the app stores and as multiplatform source code on GitHub. It can be used for evaluating the radio module as well as a reference for custom app development.

Please find connection setup examples in chapter `Connection setup and data exchange` and `Quick start example`.

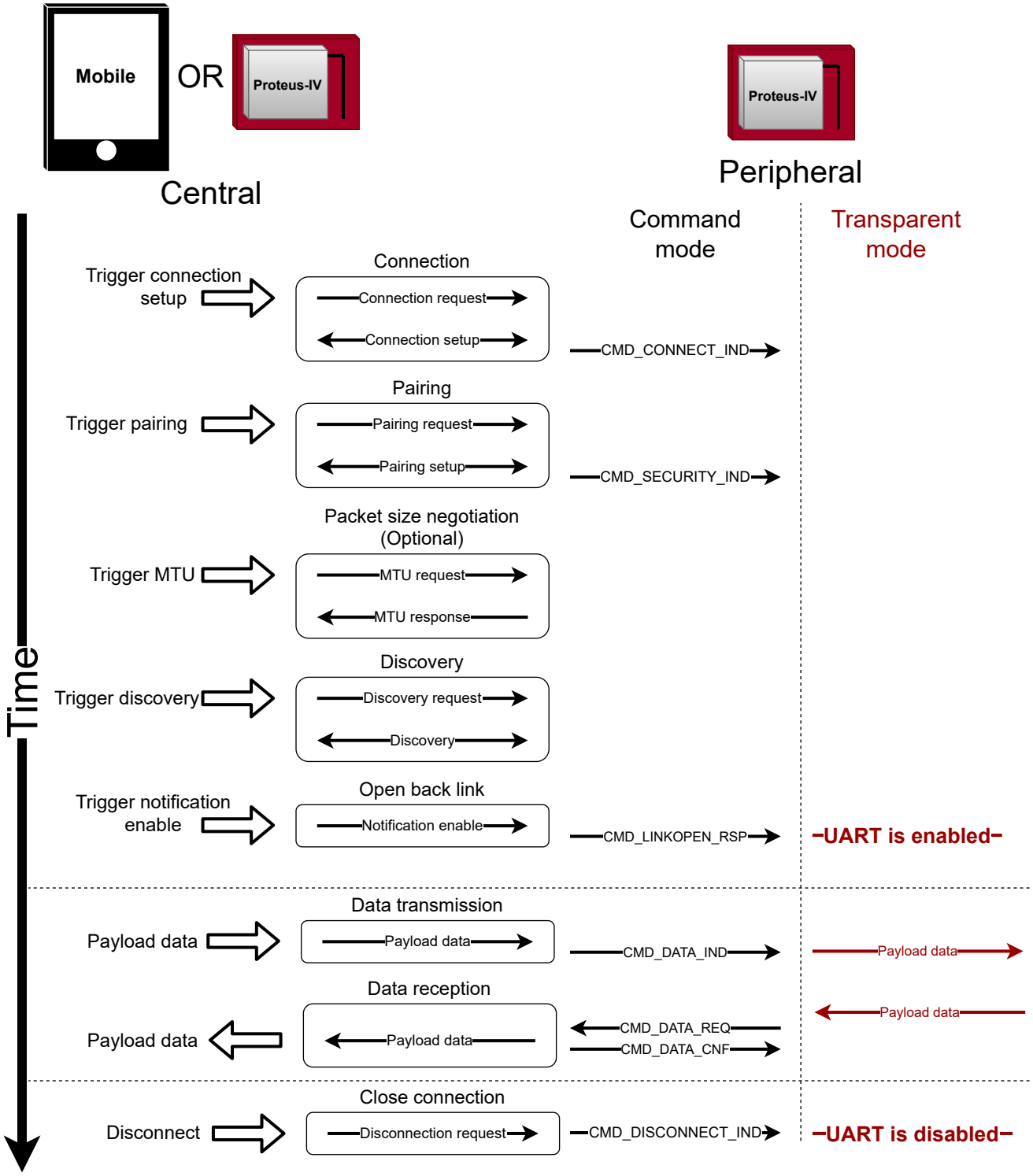


Figure 30: Steps for the connection setup

7.4.4. Bluetooth® security

The Bluetooth® LE interface has built-in security modes for Bluetooth® link authentication and encryption. To make use of them, the user setting `RF_SecFlags` needs to be used. Both the radio module and the peer device define what input/output capabilities they provide. Input/output capabilities are the ability of the device to enter a passkey (e.g., via keyboard) or to display a passkey (e.g., via display).

Based on that, the security method to establish a Bluetooth® LE link is negotiated during connection setup. Please find the negotiated security method and the negotiated security level in Figure 31.

		Initiator			
		No input no output	**Keyboard only**	**Display only**	**Display and keyboard**
Responder	**No input no output**	Just works Negotiated level: L2	Just works Negotiated level: L2	Just works Negotiated level: L2	Just works Negotiated level: L2
	Keyboard only	Just works Negotiated level: L2	Passkey entry Initiator: Enter Responder: Enter Negotiated level: L4	Passkey entry Initiator: Display Responder: Enter Negotiated level: L4	Passkey entry Initiator: Display Responder: Enter Negotiated level: L4
	Display only	Just works Negotiated level: L2	Passkey entry Initiator: Enter Responder: Display Negotiated level: L4	Just works Negotiated level: L2	Passkey entry Initiator: Enter Responder: Display Negotiated level: L4
	Display and keyboard	Just works Negotiated level: L2	Passkey entry Initiator: Enter Responder: Display Negotiated level: L4	Passkey entry Initiator: Display Responder: Enter Negotiated level: L4	Numeric comparison Initiator: Confirm Responder: Confirm Negotiated level: L4

Figure 31: Bluetooth® LE pairing table

Level	Description
L1	No encryption and no authentication
L2	Encryption and no authentication (no protection against MITM[1])
L3	Encryption and authentication (protection against MITM[1])
L4	Authenticated Secure Connections and 128-bit key

Table 20: Security levels

If the negotiated security level of the connection is lower than the minimum security level of the peripheral's profile (see Table `Security configuration flags`), the Bluetooth® LE link cannot be opened and payload data cannot be exchanged. In case of failure, the `CMD_SECURITY_IND` message with a status code not equal to 0 is printed.

[1]MITM stands for Man-in-the-Middle. It's a type of cyberattack where a malicious actor secretly intercepts and possibly alters the communication between two parties who believe they are directly communicating with each other.

Example 1 - Success: The central (initiator) device uses RF_SecFlags=0x01 (No input, no output, minimum peripheral level L2) and the peripheral (responder) uses RF_SecFlags=0x02 (Keyboard only, minimum peripheral level L2). The negotiated security level will be L2, as shown in Figure 31. As L2 is equal to the minimum level of the peripheral (L2 has been configured as RF_SecFlags=0x02), the Bluetooth® link will be secured and established with the "Just Works" method.

Example 2 - Fail: The central (initiator) device uses RF_SecFlags=0x01 (No input, no output, minimum peripheral level L2) and the peripheral (responder) uses RF_SecFlags=0x03 (Keyboard only, minimum peripheral level L4). The negotiated security level will be L2, as shown in Figure 31. As L2 is lower than the minimum level of the peripheral (L4 has been configured as RF_SecFlags=0x03), the Bluetooth® link cannot be secured and thus will not be set up. In that case, a CMD_SECURITY_IND message is printed with status 4 ("The requested security level could not be reached").

Depending on the negotiated pairing method, an action must be performed, i.e., a passkey must be entered or a passkey must be compared. If this is not done within 30 s, the pairing procedure is canceled and the connection is dropped.

If both parties have "keyboard only" IO capabilities, both hosts are requested to enter a key. In that scenario, the user must choose a key and enter it on both sides.

Bonding is enabled, such that on each reconnection, the negotiated keys are reused. The keys of up to 16 devices are stored in the radio module's memory.

Note that unauthenticated overwriting of bonding information is blocked to avoid vulnerability.
Explanation: If two devices are bonded, one of both looses its bonding information, but the second one keeps its bonding information, re-bonding is blocked until the second device deletes its bonding information as well. On Proteus-IV deleting the bonding information be done by the CMD_DELETEBONDS_REQ command.
This ensures that existing bonded connections cannot be attacked by externals.

7.4.4.1. Just works

The connection is established without host interaction.

7.4.4.2. Passkey entry

To establish the connection, a passkey is shown on one device (using CMD_DISPLAY_PASSKEY_IND). On the other device, a CMD_PASSKEY_IND is sent to the host. The host must respond with a CMD_PASSKEY_REQ message including the passkey shown on the other device.

7.4.4.3. Numeric comparison

To establish the connection, a passkey is shown on both devices (using `CMD_DISPLAY_PASSKEY_IND`). Both hosts must reply with a `CMD_NUMERIC_COMP_REQ` message to confirm that both keys are equal.

7.4.5. Advertising data

The content of standard advertising data of the Proteus-IV is automatically defined. It contains the

- Bluetooth® flags, the UUID of the Bluetooth® LE profile, and the TX power of the Proteus-IV in the advertising packet.
- the device name `RF_DeviceName` in the scan response packet.

In a few cases, like the transmission of Bluetooth® beacons, the standard content of the advertising and scan response packet is no longer suitable.

Bluetooth® beacons are advertising and scan response data of a specific format, like Eddystone beacon [6] or iBeacon [7].

7.4.5.1. Custom advertising and scan response data

To place custom data (like beacons) in the advertising and scan response packet, the Proteus-IV implements the user settings `RF_AdvertisingData` and `RF_ScanResponseData`. Both settings contain the raw data that is to be placed in the advertising packet and scan response packet respectively, with a maximum of 31 bytes per packet.

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

If these user settings contain data, the standard content is no longer used in the advertising and scan response packet during advertising.
The content of the standard advertising and scan response packet is only cleared if the user setting `RF_AdvertisingData` contains at least one byte. In this case, the content of `RF_AdvertisingData` is placed in the advertising packet.
If the user setting `RF_ScanResponseData` contains at least one byte in addition, this content is placed in the scan response packet.
Thus, to place custom data in the scan response packet, custom data must be placed in the advertising packet first.

There are two options to place custom data in the settings `RF_AdvertisingData` and `RF_ScanResponseData`:

1. Use the command CMD_SET_REQ to write the custom data to the setting placed in non-volatile RRAM. The new data is only applied after a device reset. As this command uses one write cycle, it is suited to define the default content of the advertising and scan response packet.
2. Use the command CMD_SETRAM_REQ to write the custom data to the local copy of the setting placed in RAM, which is set to default after device reset. This setting is suited to apply new data in the advertising and scan response packet during runtime.

7.4.5.1.1. Limitation

If custom data is to be placed in the advertising and scan response packet, the following restrictions must be respected:

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

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

7.4.6. 2 MBit and LE Coded PHY

In addition to the legacy 1 MBit PHY, Bluetooth® LE allows data transmission at 2 MBit data rate as well as in LE Coded mode. The LE Coded mode is the so-called "Long range mode" that was introduced with Bluetooth® 5.0. It uses the Direct Sequence Spread Spectrum (DSSS) technique that spreads the signal and thus generates redundant information. On the receiver side, it uses the Forward Error Correction (FEC) technique to use the redundancy to correct a received perturbed signal. The combination of both DSSS and FEC enables higher ranges in data transmission.
To be backward compatible with Bluetooth® LE 4.x devices, Bluetooth® LE connections must still be set up using the 1 MBit PHY. As soon as a connection has been set up, the connection can be updated to 2 MBit or LE Coded mode. To switch the PHY after the connection has been set up, the Proteus-IV offers the command CMD_PHYUPDATE_REQ. In response to this request, a CMD_PHYUPDATE_IND is returned from the Proteus-IV, which gives feedback on whether the connection was switched to the new PHY or if the connection partner rejected the request.

7.4.7. Bluetooth® LE profiles

The Bluetooth® specification uses so-called "Profiles" to specify the general behavior of a Bluetooth®-enabled device to communicate with other Bluetooth® devices. Profiles are built on the Bluetooth® standard to clearly define what kind of data is transmitted. The device's application determines which profiles it must support, from hands-free capabilities to heart rate sensors to alerts and more. A device may support more than one profile. For two devices to be compatible, they must support the same Bluetooth® LE profile.

The Proteus-IV implements the second version of the SPP-like profile, called **SPPlikeV2**. As with version 1.0 of the SPP-like profile used in Proteus-I, -II, -III, and -e, the **SPPlikeV2** is a

custom Bluetooth® LE profile as well. It allows transmitting and receiving data on two different characteristics (RX_CHARACTERISTIC and TX_CHARACTERISTIC).

The Bluetooth® profiles of the Proteus-IV cannot be removed or added during runtime. This needs to be done in a custom firmware tailored to the specific needs of the custom application (refer to chapter 14).

7.4.7.1. SPPlikeV2

The GATT structure of the SPPlikeV2 looks as follows:

```
Primary service (UUID b70c0001-686c-4036-bb28-b797ae6a8d3a)
 ├─RX_CHARACTERISTIC (UUID b70c0002-686c-4036-bb28-b797ae6a8d3a)
 │    write permission to "write without response"
 ├─TX_CHARACTERISTIC (UUID b70c0003-686c-4036-bb28-b797ae6a8d3a)
 │   sends notifications
 │   └─Descriptor
 │        read/write permissions to "enable notifications"
```

Figure 32: SPPlikeV2 Bluetooth® LE profile

By means of the user settings RF_SPPServiceUUID, RF_SPPRXUUID, and RF_SPPTXUUID, the UUIDs can be adapted to generate a custom profile.

The first characteristic of the Proteus-IV primary service is RX_CHARACTERISTIC:

- The data is sent from central/client to peripheral/server using a write-without-response command.
- Server:
  - Must allow a write-without-response command.
- Client:
  - Use write-without-response command to send data to the server.

The second characteristic of the Proteus-IV primary service is TX_CHARACTERISTIC:

- The data is sent from peripheral/server to central/client using a notification.
- Server:
  - Must allow/enable notifications. Notify client/central when sending data.
  - When the notification enable bit is written in the CCCD (Client Characteristic Configuration Descriptor) by the central, the peripheral prints a CMD_LINKOPEN_RSP on the UART to signal that the peripheral can now send data to the central. The central can only write this bit when the configured security level of the peripheral has been met.

- Client:
  - Must enable notifications during connection setup.

7.4.8. Radio compatibility to other Proteus devices

To be able to communicate with Proteus-I, -II, or -III ("legacy Proteus"), the following points must be considered:

1. The connection interval of the legacy Proteus device must overlap with the connection interval used by the Proteus-IV (see `RF_ConnectionInterval`). In that case, both Bluetooth® devices are able to negotiate a common connection interval.
2. The security mode of the legacy Proteus device must be equal to the security mode used by the Proteus-IV (see `RF_SecFlags`). In that case, the central device has permission to access the peripheral's services.
3. As the legacy Proteus device uses the SPPlike profile in version 1, the Bluetooth® LE protocol of the legacy Proteus must be simulated. See also the example below.
   - UUIDs of the Proteus-IV must match the UUIDs of the legacy Proteus (or vice versa). Thus either the UUIDs of the legacy device or the Proteus-IV must be modified.
   - When sending data from the Proteus-IV to the legacy Proteus, a 0x01 byte must be prepended to the payload.
   - If the Proteus-IV receives data from a legacy Proteus, the header byte 0x01 must be removed from the payload.

   To do so, the use of command mode is required.

Example: Connect with a Proteus-III to a Proteus-IV

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

Info	Proteus-III	Proteus-IV
⇐ Response `CMD_GETSTATE_CNF`: Proteus-III started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⇐ Response `CMD_STARTUP_IND`: Proteus-IV started in command mode.		02 81 01 00 01 83

2. Set the UUIDs of the Proteus-IV to match the UUIDs of the legacy Proteus device:

Info	Proteus-III	Proteus-IV
⇒ Set the `RF_SPPServiceUUID`		02 11 11 00 20 1B C5 D5 A5 02 00 3D 95 E5 11 52 C3 01 00 40 6E 6C
⇐ Response `CMD_SET_CNF` and `CMD_STARTUP_IND`		02 51 01 00 00 52 02 81 01 00 01 83

⇒ Set the RF_SPPRXUUID		02 11 11 00 21 1B C5 D5 A5 02 00 3D 95 E5 11 52 C3 02 00 40 6E 6E
⇐ Response CMD_SET_CNF and CMD_STARTUP_IND		02 51 01 00 00 52 02 81 01 00 01 83
⇒ Set the RF_SPPTXUUID		02 11 11 00 22 1B C5 D5 A5 02 00 3D 95 E5 11 52 C3 03 00 40 6E 6C
⇐ Response CMD_SET_CNF and CMD_STARTUP_IND		02 51 01 00 00 52 02 81 01 00 01 83

3. Disable the Bluetooth® security of the Proteus-IV as Proteus-III has "no security" set by default:

Info	Proteus-III	Proteus-IV
⇒ Set the RF_SecFlags		02 11 02 00 0C 00 1D
⇐ Response CMD_SET_CNF and CMD_STARTUP_IND		02 51 01 00 00 52 02 81 01 00 01 83

4. Scan for compatible Bluetooth® LE devices

Info	Proteus-III	Proteus-IV
⇒ Start the scan CMD_SCANSTART_REQ	02 09 00 00 0B	
⇐ Response CMD_SCANSTART_CNF	02 49 01 00 00 4A	
⇒ Stop the scan CMD_SCANSTOP_REQ	02 0A 00 00 08	
⇐ Response CMD_SCANSTOP_CNF	02 4A 01 00 00 49	
⇒ Request the scanned devices	02 0B 00 00 09	
⇐ Response CMD_GETDEVICES_CNF: Found one device with BTMAC **0xFF 0xFF 0xFF 0xDA 0x18 0x00**	02 4B 0B 00 00 01 **FF FF FF DA 18 00** CA 08 00 BC	

5. Connect Proteus-III to Proteus-IV via Bluetooth®.

Info	Proteus-III	Proteus-IV
⇒ Request CMD_CONNECT_REQ with FS_BTMAC of Proteus-IV	02 06 06 00 **FF FF FF DA 18 00** 3F	

⇐ 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 **0xFF 0xFF 0xFF 0xDA 0x18 0x00**	02 86 07 00 00 **FF FF FF DA 18 00** BE	
⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC **0x11 0x22 0x00 0xDA 0x18 0x00** and Conn_ID 0x00		02 86 09 00 00 00 00 **11 22 00 DA 18 00** 7C

Info	Proteus-III	Proteus-IV
⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC **0xFF 0xFF 0xFF 0xDA 0x18 0x00** and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 **FF FF FF DA 18 00** F3 02	
⇐ Indication CMD_MAXPAYLOAD_IND: Updated max payload to 244 successfully		02 93 03 00 00 F4 00 66
⇐ Indication CMD_LINKOPEN_RSP: Channel opened successfully		02 C6 02 00 00 01 C7

6. Once the connection is active, data can be sent in each direction. Let us send a string "ABCD" from Proteus-IV to Proteus-III.

Info	Proteus-III	Proteus-IV
⇒ Request CMD_DATA_REQ: Send "ABCD" with prepended 0x01 to Proteus-III using Conn_ID **0x00**		02 04 06 00 **00** 01 41 42 43 44 05
⇐ 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 **0xFF 0xFF 0xFF 0xDA 0x18 0x00** with RSSI of 0xC5 (-59 dBm)	02 84 0B 00 **FF FF FF DA 18 00 C5** 41 42 43 44 71	

7. Reply with "EFGH" to Proteus-IV.

Info	Proteus-III	Proteus-IV
⇒ Request CMD_DATA_REQ: Send "EFGH" to Proteus-IV	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 data from Conn_ID **0x00** with RSSI of 0xC2 (-62 dBm). The header 0x01 must be removed to get the payload EFGH		02 84 07 00 **00** C2 01 45 46 47 48 4E
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully	02 C4 01 00 00 C7	

8. Now Proteus-III closes the connection, so both modules will get a disconnect indication.

Info	Proteus-III	Proteus-IV
⇒ 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 with Conn_ID 0x00 closed		02 87 02 00 00 13 94

7.5. Best practices

When designing an end device using the Proteus-IV, the following points should be considered to create a stable and reliable system:

- When sending data from the host via UART to the radio module to the connected Bluetooth® peer, the overall throughput depends on both the radio throughput and the UART throughput. The radio throughput is mainly defined by the payload size, the connection interval, and the PHY used, while the UART throughput is defined by the UART baud rate.
  The Proteus-IV buffers the UART data and forwards it via radio as fast as possible. The UART's */RTS* line tells the host when the receive buffers of the radio module are full and the host must wait before it can continue sending data. Thus, it is highly recommended to enable and respect the UART's flow control.

- Conversely, the Bluetooth® interface does not provide flow control. Thus, radio data received by the Proteus-IV must be sent to the host via UART as soon as possible. If the UART transmission buffers are full, data received via radio may be discarded. Thus, it is highly recommended to increase the UART baud rate to a level that supports the desired radio throughput/speed.

- When high throughput is needed on a Bluetooth® link, all other radio activities should be disabled. This means that if the radio module is advertising in parallel, it should be stopped. Unneeded Bluetooth® connections should be dropped to allow the radio to deliver the highest performance on the high-throughput link.

8. Command mode

In command mode, the configuration and 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 the 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 feedback on the requested operation status. In the 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.

Note that the command interface is already implemented in the Wireless Connectivity SDK. This SDK allows you to copy the radio module drivers to your host microcontroller for quick design-in. Furthermore, it brings examples that demonstrate the use of the radio module. You can find it on GitHub [8]. The application note ANR008 [9] describes the content and process of design-in.

If the transmission of the UART command has not finished within the packet transmission duration, depending on the currently selected UART baud rate, the module discards the received bytes and waits for a new command. This means that the delay between 2 successive bytes in a frame must be kept as low as possible.

Please note that the different commands are only valid in specific module states (see Figure 33). If a command is not permitted in the current state, the command confirmation returns "Operation not permitted" (0xFF) as a response.

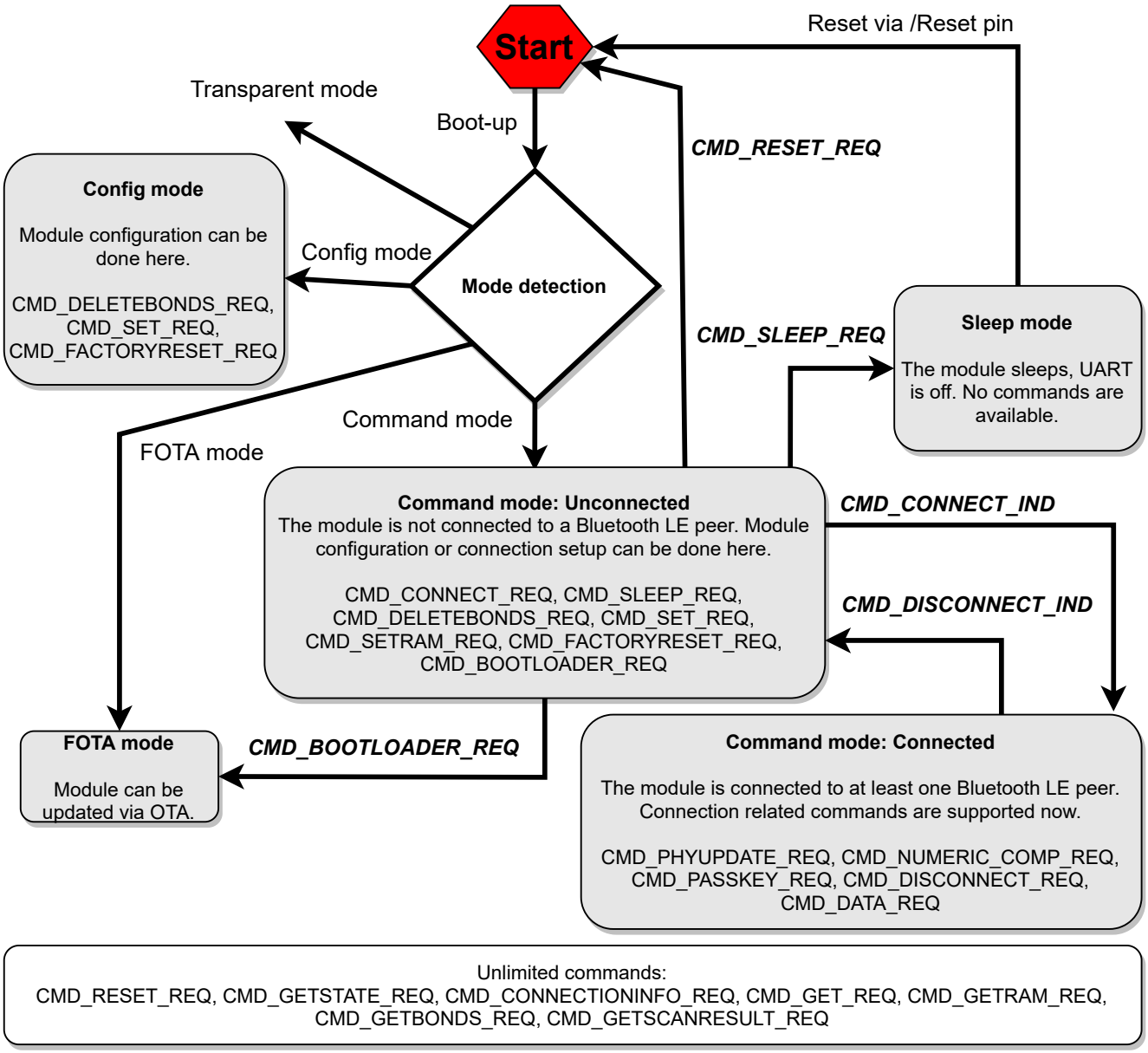


Figure 33: State overview

8.1. Command structure

The commands follow the following structure:

Start signal	Command	Length	Payload	CS
0x02	1 Byte	2 Bytes, 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 Payload field of the command. Length is a 16-bit field with LSB first.

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

Checksum: Bitwise XOR combination of all preceding bytes including the start signal, i.e., 0x02 ^ Command ^ Length ^ Payload = CS

Host integration example codes for checksum calculation and command frame structure can be found in annex A and B, as well as in the *Wireless Connectivity SDK* [8, 9].

8.2. Manage the device state

8.2.1. CMD_STARTUP_IND

This command is sent by the radio module after it has been booted up.

Start signal	Command	Length	Module mode	CS
0x02	0x81	0x01 0x00	1 Byte	1 Byte

Module mode:

0x00: Error

0x01: Command

0x02: Config

0x03: Bootloader

0x04: Sleep

Other: Reserved

8.2.1.1. Example 1

The module has booted in command mode.

Start signal	Command	Length	Module mode	CS
0x02	0x81	0x01 0x00	0x01	0x83

8.2.2. CMD_GETSTATE_REQ

This command returns the current state of the module. Format:

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

Response (`CMD_GETSTATE_CNF`):

Start signal	Command | 0x40	Length	Status	Module mode	More info	CS
0x02	0x41	0x03 0x00	1 Byte	1 Byte	1 Byte	1 Byte

Status:

0x00: Success

0x01: Failed, due to invalid command

0xFF: Operation not permitted

Module mode:

0x00: Error

0x01: Command

0x02: Config

0x03: Bootloader

0x04: Sleep

Other: Reserved

More info:

- Bit 0: Advertising
- Bit 1: Scanning
- Bit 2: Connected
- Bit 3: Open peripheral link
- Bit 4: Open central link

8.2.2.1. 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	Status	Module mode	More info	CS
0x02	0x41	0x03 0x00	0x00	0x01	0x01	0x40

The module is advertising.

8.2.3. 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, due to invalid command

0xFF: Operation not permitted

8.2.4. CMD_SLEEP_REQ

This command is used to put the module into sleep mode, enabling low power consumption. For more details, see chapter `7.3.1`.
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, due to invalid command

0xFF: Operation not permitted

8.2.5. 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 *UART_ENABLE* pin is used (apply a falling edge, hold low for at least 500 µs before applying a rising edge and hold high for at least 10 ms). In this case, either the received data or a `CMD_UARTENABLE_IND` is transmitted by the module. Afterwards, the UART will stay active until another `CMD_UARTDISABLE_REQ` has been received.
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 due to invalid command

0xFF: Operation not permitted

It is strongly recommended to disable the UART only if it is foreseeable that there will be no UART communication for several seconds. Use cases could be during the advertising phase to wait for connecting Bluetooth® LE devices.

8.2.6. CMD_UARTENABLE_IND

This indication is sent when the module's UART interface is re-enabled (after executing a `CMD_UARTDISABLE_REQ`). Once this message is received, the UART can be used for all operations again.
Format:

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

Status:

0x00: UART has been successfully re-enabled

8.2.7. CMD_BOOTLOADER_REQ

This command enables the FOTA function of the Proteus-IV. It unlocks the SMP Bluetooth® LE service and allows to update the firmware of the radio module via Bluetooth® LE interface.
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.

0xFF: Operation not permitted

Other: Operation failed, due to internal error

8.3. Advertise to be visible on the radio for other devices

8.3.1. CMD_ADVERTISING_REQ

This command allows to turn the advertising on or off.
Format:

Start signal	Command	Length	Enable	CS
0x02	0x12	0x01 0x00	1 Byte	1 Byte

Enable:

0x00: Disable advertising

0x01: Enable advertising

Response (`CMD_ADVERTISING_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x52	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request successfully run

0x01: Operation failed, due to invalid command

0x02: Operation failed, as no more free peripheral connections are available. The module will start advertising later, once a free connection is available.

0xFF: Operation not permitted

8.3.1.1. Example 1

Enable advertising:

Start signal	Command	Length	Enable	CS
0x02	0x12	0x01 0x00	0x01	0x10

Response:

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

Advertising enabled successfully.

8.4. Scan for other devices in range

8.4.1. CMD_SCAN_REQ

This command enables or disables scanning.
All detected devices that meet the Proteus-IV specification are reported over UART in a `CMD_SCAN_IND` message. Depending on whether the duplicate device filter is enabled, each device will generate either a single or multiple `CMD_SCAN_IND` messages. For filter configuration, see the user setting `CFG_Flags`.
Format:

Start signal	Command	Length	Enable	CS
0x02	0x09	0x01 0x00	1 Byte	0x0B

Enable:

0x00: Disable scanning

0x01: Enable scanning

Response (`CMD_SCAN_CNF`):

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

Status:

0x00: Request received; scanning will start or stop now

0x01: Operation failed due to invalid command

0xFF: Operation not permitted

8.4.1.1. Example 1

Enable scanning:

Start signal	Command	Length	Enable	CS
0x02	0x09	0x01 0x00	0x01	0B

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x49	0x01 0x00	0x00	0x4A

Scanning enabled successfully.

8.4.2. CMD_SCAN_IND

This command is sent by the radio module after it has found a peer device while scanning.

Start signal	Command	Length	Addr. type	BTMAC	RSSI	Device name length	Device name	CS
0x02	0x89	0x13 0x00	1 Byte	6 Bytes	1 Byte	1 Byte	Device name length bytes	1 Byte

Addr. type:

0x00: Public

BTMAC:

BTMAC of the connected peer device

RSSI:

RSSI value is printed in two's complement notation [dBm]

8.4.2.1. Example 1

Start signal	Command	Length	Addr. Type	BTMAC	RSSI	Device name length	Device name	CS
0x02	0x89	0x13 0x00	0x00	**0x12** **0x34** **0x56** **0xDA** **0x18** **0x00**	0xC3	0x0A	0x50 0x72 0x6F 0x74 0x65 0x75 0x73 0x2D 0x49 0x56	0x8B

Found a device with:

- BTMAC: **0x0018DA563412**
- RSSI: -61 dBm (0xC3)
- Device name: Proteus-IV (0x50726F746575732D4956)

8.5. Manage Bluetooth® LE connections

8.5.1. CMD_CONNECT_REQ

This command attempts to establish a connection to the Proteus-IV, identified by the FS_BTMAC specified in the command. After the module sends a CMD_CONNECT_CNF to confirm that the request was received, an indication message CMD_CONNECT_IND follows, which indicates whether the connection request was accepted by the peer device.
If security features are enabled (see the setting RF_SecFlags), a CMD_SECURITY_IND is also sent.
Once the connection setup is complete and all services have been successfully discovered, a CMD_LINKOPEN_RSP is sent to the host. At this point, data can be transmitted using CMD_DATA_REQ.
Format:

Start signal	Command	Length	Addr. type	BTMAC	CS
0x02	0x06	0x07 0x00	1 Byte	6 Bytes	1 Byte

Addr. type:

0x00: Public

BTMAC:

BTMAC of the peer device to connect to

Response (CMD_CONNECT_CNF):

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

Status:

0x00: Request received; attempting to connect to the device with FS_BTMAC

0x01: Operation failed due to invalid command

0x10: Operation failed; radio module has no free connections

0x16: Operation failed due to invalid Addr. type / BTMAC pair (e.g., 0x00/0x000000000000 or 0x00/0xFFFFFFFFFFFF)

0xFF: Operation not permitted

Other: Operation failed due to internal error

8.5.1.1. Example 1

Connect to a Bluetooth® device with public (0x00) BT_MAC 0x0018DA123456:

Start signal	Command	Length	Addr. type	BTMAC	CS
0x02	0x06	0x07 0x00	0x00	0x56 0x34 0x12 0xDA 0x18 0x00	0xB1

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x46	0x01 0x00	0x00	0x45

Request accepted; searching for Bluetooth® device and connecting now.

8.5.2. CMD_CONNECT_IND

This telegram indicates that a Bluetooth® LE connection has been estabilished.
Format:

Start signal	Command	Length	Conn_ID	Status	Addr. type	BTMAC	CS
0x02	0x86	0x09 0x00	1 Byte	1 Byte	1 Byte	6 Bytes	1 Byte

Conn_ID:

ID of the connection

Status:

0x00: Success

0x02: Error, cannot find device on radio with the requested BTMAC

Addr. type:

0x00: Public

BTMAC:

BTMAC of the connected peer device

8.5.2.1. Example 1

Start signal	Command	Length	Conn_ID	Status	Addr. type	BTMAC	CS
0x02	0x86	0x09 0x00	0x00	0x00	0x00	0x84 0x9C 0x60 0x30 0xB2 0x4C	0x35

Successfully connected to device with BTMAC 0x849C6030B24C. Conn_ID 0x00 has been chosen.

8.5.3. CMD_SECURITY_IND

This telegram indicates the security status of the Bluetooth® LE connection.
Format:

Start signal	Command	Length	Conn_ID	Status	Level	CS
0x02	0x88	0x03 0x00	1 Byte	1 Byte	1 Byte	1 Byte

Conn_ID:

ID of the connection

Status:

0x00: Success, encrypted link established

0x01: Authentication failed

0x02: Pin or key is missing

0x04: The requested security level could not be reached.

0x05: Pairing is not supported

0x06: Pairing is not allowed

0x07: Invalid parameters

0x08: Distributed Key Rejected

0x09: Pairing failed but the exact reason could not be specified

Other: Error

Level:

0x01: Security level L1: No encryption and no authentication

0x02: Security level L2: Encryption and no authentication (no MITM)

0x03: Security level L3: Encryption and authentication (MITM)

0x04: Security level L4: Authenticated Secure Connections and 128-bit key

The "Level" contains only a meaningful value, if "Status" equals 0 (success).

8.5.3.1. Example 1

Start signal	Command	Length	Conn_ID	Status	Level	CS
0x02	0x88	0x03 0x00	0x00	**0x00**	0x02	0x8B

Security of connection with Conn_ID 0x00 **successfully** set to level L2.

8.5.4. CMD_LINKOPEN_RSP

This command is sent to the host as soon as connection setup has been completed successfully and Bluetooth® LE link has been opened or closed. If the link is open, data can be transmitted using the command `CMD_DATA_REQ`.
Format:

Start signal	Command	Length	Conn_ID	Status	CS
0x02	0xC6	0x02 0x00	1 Byte	1 Byte	1 Byte

Conn_ID:

ID of the connection

Status:

0x00: Link closed

0x01: Link opened

8.5.4.1. Example 1

Start signal	Command	Length	Conn_ID	Status	CS
0x02	0xC6	0x02 0x00	0x00	0x01	0xC7

Link has been opened on connection with Conn_ID 0x00.

8.5.5. CMD_DISCONNECT_REQ

This command terminates an existing connection. Afterward, the module sends a `CMD_DISCONNECT_CNF` to confirm that the request has been received. Then, the indication message `CMD_DISCONNECT_IND` follows, indicating whether the disconnection was successful.
Format:

Start signal	Command	Length	Conn_ID	CS
0x02	0x07	0x01 0x00	1 Byte	1 Byte

Conn_ID:

0xFE: Broadcast - disconnect all connections

Other: ID of the connection

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, due to invalid command

0x13: Operation failed, due to invalid Conn_ID

0xFF: Operation not permitted

Other: Operation failed, due to internal error

8.5.5.1. Example 1

Disconnect all connections using broadcast connection ID:

Start signal	Command	Length	Conn_ID	CS
0x02	0x07	0x01 0x00	0xFE	FA

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x47	0x01 0x00	0x00	0x44

Request understood, will disconnect now.

8.5.6. CMD_DISCONNECT_IND

This indication signals that the connection has been successfully terminated. It is sent as a result of a `CMD_DISCONNECT_REQ`.
Format:

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

Conn_ID:

ID of the connection

Reason:

0x08: Connection timeout

0x13: User terminated connection

0x16: Host terminated connection

0x3B: Connection interval unacceptable

0x3D: Connection terminated due to MIC failure (Unable to connect because of poor link quality or ignored due to incorrect key)

0x3E: Connection setup failed

8.5.6.1. Example 1

Start signal	Command	Length	Conn_ID	Reason	CS
0x02	0x87	0x02 0x00	0x00	0x13	0x94

Bluetooth® LE connection with Conn_ID 0x00 has been closed by the user.

8.5.7. CMD_CONNECTIONINFO_REQ

This command returns all information related to an open connection. The **Conn_ID** parameter determines whether the information of a single or all connection is requested.
Format:

Start signal	Command	Length	Conn_ID	CS
0x02	0x22	0x01 0x00	1 Byte	1 Byte

Conn_ID:

0xFE: Broadcast - all connections

Other: ID of the connection

Response (`CMD_CONNECTIONINFO_CNF`):

Start signal	Command | 0x40	Length	Status	`#Devices`	Payload	CS
0x02	0x62	2 Bytes	1 Byte	1 Byte	(Length - 2) Bytes	1 Byte

Status:

0x00: Request ok, `#Devices` and Payload fields are filled with data

0x01: Operation failed, due to invalid command

0xFF: Operation not permitted

Payload:
The Payload sequentially lists the data of the `#Devices` connections. It consists of `#Devices` times the following telegram.

Conn_ID	Addr. type	BTMAC	Link status	Max payload	Connection interval	PHY Rx	PHY Tx
1 Byte	1 Byte	6 Bytes	1 Byte	2 Bytes	2 Bytes	1 Byte	1 Byte

Conn_ID:

ID of the connection

Addr. type:

0x00: Public

BTMAC:

BTMAC of the peer device

Link status:

0x00: Link closed

0x01: Link opened

Max payload:

Maximum payload of the connection

Connection interval:

Interval of the connection (Unit: 1.25 ms)

PHY Rx/PHY Tx:

0x01: Using 1 MBit PHY

0x02: Using 2 MBit PHY

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

8.5.7.1. Example 1

Request for the connection info of all connected devices using the broadcast Conn_ID.

Start signal	Command	Length	Conn_ID	CS
0x02	0x22	0x01 0x00	0xFE	DF

Response:

Start signal	Command | 0x40	Length	Status	#Devices	Payload	CS
0x02	0x62	0x11 0x00	0x00	0x01	0x00 0x01 0x84 0x9C 0x60 0x30 0xB2 0x4C **0x01** 0x14 0x00 0x24 0x00 0x01 0x01	0xF6

One device is connected:

- Conn_ID 0x00
- Private (0x01) BTMAC 0x84 0x9C 0x60 0x30 0xB2 0x4C
- Bluetooth® link open (**0x01**)
- Max payload 20 (0x0014)
- 45 ms (0x0024) connection interval
- 1 MBit phy Rx/Tx

8.5.8. CMD_PHYUPDATE_REQ

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

Format:

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

Conn_ID:

ID of the connection

PHY:

0x01: 1 MBit PHY

0x02: 2 MBit PHY

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

Response (`CMD_PHYUPDATE_CNF`):

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

Status:

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

0x01: Operation failed, e.g. due to invalid command or Conn_ID

0x13: Operation failed, due to invalid Conn_ID

0xFF: Operation not permitted

Other: Operation failed, due to internal error

8.5.8.1. Example 1

Set the connection with Conn_ID 0x00 to 2 MBit mode:

Start signal	Command	Length	Conn_ID	PHY	CS
0x02	0x1A	0x02 0x00	0x00	0x02	0x18

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x5A	0x01 0x00	0x00	0x58

Request understood, will change PHY now.

8.5.9. CMD_PHYUPDATE_IND

This command indicates that there was an attempt to update the PHY of the existing connection. This command is the result of the `CMD_PHYUPDATE_REQ`.
Format:

Start signal	Command	Length	Conn_ID	PHY TX	PHY RX	CS
0x02	0x9A	0x03 0x00	0x00	1 Byte	1 Byte	1 Byte

Conn_ID:

ID of the connection

PHY Rx/PHY Tx:

0x01: Using 1 MBit PHY now

0x02: Using 2 MBit PHY now

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

8.5.9.1. Example 1

Start signal	Command	Length	Conn_ID	PHY Rx	PHY Tx	CS
0x02	0x9A	0x03 0x00	0x00	0x02	0x02	0x9B

Bluetooth® LE connection with Conn_ID 0x00 has been set to 2 MBit mode.

8.5.10. CMD_PASSKEY_REQ

When a `CMD_PASSKEY_IND` is received during connection setup, the peer device requests a passkey to authenticate the connecting device. To respond to this request, the `CMD_PASSKEY_REQ` message must be sent to the Proteus-IV including the peer's passkey. The permissible characters for the passkey range from 0x30 to 0x39 (inclusive), which correspond to ASCII digits (0-9).
Format:

Start signal	Command	Length	Conn_ID	Pass key	CS
0x02	0x0D	0x07 0x00	1 Byte	6 Bytes	1 Byte

Conn_ID:

ID of the connection

Passkey:

Passkey for authentication

Response (CMD_PASSKEY_CNF):

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

Status:

0x00: Passkey accepted and request answered

0x01: Operation failed due to invalid characters in passkey

0x03: Operation failed; no request pending

0x13: Operation failed due to invalid Conn_ID

0xFF: Operation not permitted

Other: Operation failed due to internal error

8.5.10.1. Example 1

Reply with the passkey "123123" to the connection with Conn_ID 0x00:

Start signal	Command	Length	Conn_ID	Pass key	CS
0x02	0x0D	0x07 0x00	0x00	0x31 0x32 0x33 0x31 0x32 0x33	0x08

9 Response:

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

Passkey accepted.

8.5.11. CMD_PASSKEY_IND

Depending on the security settings, a passkey may be required to authenticate the device. When such a passkey authentication request is received, the CMD_PASSKEY_IND message is sent to the host. In this case, the passkey must be entered using CMD_PASSKEY_REQ to successfully complete the connection procedure.
Format:

Start signal	Command	Length	Conn_ID	CS
0x02	0x8D	0x01 0x00	1 Byte	1 Byte

Conn_ID:

ID of the connection

8.5.11.1. Example 1

Start signal	Command	Length	Conn_ID	CS
0x02	0x8D	0x01 0x00	0x00	0x8E

Passkey requested for connection with Conn_ID 0x00.

8.5.12. CMD_DISPLAY_PASSKEY_IND

Depending on the security settings, a passkey may be displayed either to show it to the connection partner or to confirm/reject it.
In "Passkey" mode, the peripheral outputs the passkey using this message so it can be entered on the central device for authentication.
In "NumComp" mode, both the central and peripheral output the passkey to their respective hosts. Each host must reply with a `CMD_NUMERIC_COMP_REQ` message to confirm that both keys match.
Format:

Start signal	Command	Length	Conn_ID	Action	Pass key	CS
0x02	0xA4	0x08 0x00	1 Byte	1 Byte	6 Bytes	1 Byte

Conn_ID:

ID of the connection

Action:

0x00: Key is displayed for entry on the peer device; no action required on this device

0x01: Key is displayed; confirm or reject it using `CMD_NUMERIC_COMP_REQ`

Pass key:

Passkey to use or confirm/reject

8.5.12.1. Example 1

Start signal	Command	Length	Conn_ID	Action	Pass key	CS
0x02	0xA4	0x08 0x00	0x00	0x00	0x31 0x32 0x33 0x31 0x32 0x33	0xAE

Passkey "123123" for connection with Conn_ID 0x00 displayed. Please enter it on the peer device.

8.5.13. CMD_NUMERIC_COMP_REQ

Depending on the device's security settings, a passkey may be displayed for numeric comparison to confirm or reject it.
Format:

Start signal	Command	Length	Conn_ID	Answer	CS
0x02	0x24	0x02 0x00	1 Byte	1 Byte	1 Byte

Conn_ID:

ID of the connection

Answer:

0x00: The keys displayed on the central and peripheral devices match; connection setup can continue

0x01: The keys displayed on the central and peripheral devices do not match; connection setup will be canceled

Response (`CMD_NUMERIC_COMP_CNF`):

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

Status:

0x00: Answer accepted

0x01: Operation failed due to invalid command

0x03: Operation failed; no request pending

0x13: Operation failed due to invalid Conn_ID

0xFF: Operation not permitted

Other: Operation failed due to internal error

8.5.13.1. Example 1

Reply with "OK" to a numeric comparison request for the connection with Conn_ID 0x00:

Start signal	Command	Length	Conn_ID	Answer	CS
0x02	0x24	0x02 0x00	0x00	0x00	0x24

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x64	0x01 0x00	0x00	0x67

Answer accepted.

8.5.14. CMD_GETBONDS_REQ

This command requests the information of all bonded devices. It will respond with a `CMD_GETBONDS_CNF` message. The corresponding information will be output one after the other in the field behind `#Devices` in the `CMD_GETBONDS_CNF` response.
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

Status:

0x00: Request successfully processed

0x01: Operation failed, due to invalid command

0xFF: Operation not permitted

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

Addr. Type	BTMAC
1 Byte	6 Bytes

8.5.14.1. Example 1

Request list of bonded devices:

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

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x4F	0x09 0x00	0x00 0x01 **0x00** 0xB5 0xA9 0x05 0x06 0x63 0x64	0x5D

1 device is currently bonded with type **0x00** and BTMAC 0xB5 0xA9 0x05 0x06 0x63 0x64.

8.5.15. CMD_DELETEBONDS_REQ

This command removes the bonding information for all devices or a single bonded device. To delete all bonding information, set Length to 0x0000 and leave the fields for Addr. type and BTMAC empty. Otherwise, specify the address type and BTMAC of the peer device.
Format:

Start signal	Command	Length	Addr. type	BTMAC	CS
0x02	0x0E	2 Bytes	0 or 1 Byte	0 or 6 Bytes	1 Byte

Addr. type:

0x00: Public

BTMAC:

BTMAC of the bonded peer device

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 due to invalid command

0xFF: Operation not permitted

Other: Operation failed due to internal error (e.g., device not found)

8.5.15.1. 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.

8.5.15.2. Example 2

Request to remove the bonding of the device with address type 0x00 and BTMAC 0xC976BEED9CA8:

Start signal	Command	Length	Addr. type	BTMAC	CS
0x02	0x0E	0x07 0x00	0x00	0xC9 0x76 0xBE 0xED 0x9C 0xA8	0xD3

Response:

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

Successfully removed the bonding information.

8.5.16. CMD_MAXPAYLOAD_IND

This command indicates that there was an attempt to update the max payload of the existing connection.
Format:

Start signal	Command	Length	Conn_ID	Max payload	CS
0x02	0x93	0x03 0x00	0x00	2 Bytes	1 Byte

Conn_ID:

ID of the connection

Max payload:

Maximum payload of the connection

8.5.16.1. Example 1

Start signal	Command	Length	Conn_ID	Max payload	CS
0x02	0x93	0x03 0x00	0x00	0x1400	0x86

The maximum payload of the Bluetooth® LE connection with Conn_ID 0x00 has been set to 20 (0x0014) Bytes.

8.6. Transmit and receive data

8.6.1. CMD_DATA_REQ

This command provides simple data transfer between connected Bluetooth® LE devices. This command is suitable for transmission via broadcast or a point-to-point connection. The number of payload data bytes is negotiated during the connection phase. It can be a maximum of 243 bytes, but at least 19 bytes.
When the data is processed by the module, a `CMD_DATA_CNF` is returned to the host.

In case of broadcast transmission, the status byte in the `CMD_DATA_CNF` message summarizes the results of all transmissions. Thus, if the transmission to a single Bluetooth® peer does not succeed, the status will be "error".

Additionally, if `CMD_TXCOMPLETE_RSP` messages are enabled, a `CMD_TXCOMPLETE_RSP` will follow for each single connection transmission as soon as the data has been sent. To enable this option, refer to the user setting `CFG_Flags`.
The receiving Proteus-IV will get a `CMD_DATA_IND` message containing the transmitted payload data.
Format:

Start signal	Command	Length	Conn_ID	Payload	CS
0x02	0x04	2 Bytes	1 Byte	(Length - 1) Bytes	1 Byte

Conn_ID:

0xFE: Broadcast - transmit to all open connections

Other: ID of the connection

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: Operation failed due to invalid command or Conn_ID

0x02 + 0xXXXX: Operation failed + 0xXXXX maximum payload size (if it has been exceeded)

0x13: Operation failed due to invalid Conn_ID

0xFF: Operation not permitted

Other: Operation failed due to internal error

8.6.1.1. Example 1

Transmit 0x11223344 to peer device using the connection of Conn_ID 0x00

Start signal	Command	Length	Conn_ID	Payload	CS
0x02	0x04	0x05 0x00	0x00	0x11 0x22 0x33 0x44	0x47

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x44	0x01 0x00	0x00	0x47

Request understood, will transmit the data now.

8.6.2. CMD_TXCOMPLETE_RSP

This command is sent to the host as soon as the data, which was requested by a `CMD_DATA_REQ`, has been transmitted successfully via radio. This message is only printed, if it has been enabled in the user setting `CFG_Flags`.
Format:

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

Conn_ID:

ID of the connection

Status:

0x00: Data transmitted successfully

0x01: Data transmission failed

8.6.2.1. Example 1

Start signal	Command	Length	Conn_ID	Status	CS
0x02	0xC4	0x02 0x00	0x00	0x00	0xC4

Data has been transmitted successfully to peer device connected using Conn_ID 0x00.

8.6.3. 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.
Format:

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

Conn_ID:

ID of the connection

RSSI:

RSSI value is printed in two's complement notation [dBm]

In case RSSI recording is disabled in `CFG_Flags`, this value has no meaning.

8.6.3.1. Example 1

Start signal	Command	Length	Conn_ID	RSSI	Payload	CS
0x02	0x84	0x06 0x00	0x00	0xC6	0xAA 0xBB 0xCC 0xDD	0x46

Received data (0xAABBCCDD) from peer device connected using Conn_ID 0x00 with RSSI of 0xC6 (-58 dBm).

8.7. Configuring the module and modifying the device settings

The radio module's parameters are stored in non-volatile RRAM, but have a local copy in RAM. These parameters can be modified by the `CMD_SET_REQ`, read by the `CMD_GET_REQ` and retain their content even when resetting the module.
Single parameters located in RAM can be modified by the `CMD_SETRAM_REQ`, read by the `CMD_GETRAM_REQ`.

8.7.1. CMD_FACTORYRESET_REQ

This command triggers a factory reset of the module. First, the default user settings are restored, then bonding data is removed, and 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 due to invalid command

0xFF: Operation not permitted

Other: Operation failed due to internal error

To save the parameters in the RRAM 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.

8.7.2. CMD_SET_REQ

This command enables direct manipulation of the parameters in the module's settings in non-volatile RRAM. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 35.
Parameters of 2 or more bytes have to be transferred with the LSB first unless noted differently in the corresponding description.

The RRAM memory used to store these settings has a limited count of write cycles of at least 10,000. Try to avoid performing periodic CMD_SET_REQ as each command will use one write cycle.

STOP

To save the parameters in the RRAM 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 write 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 parameter value

0x02: Operation failed due to invalid setting

0x03: New and old value of the setting are identical. Module reset will not be applied.

0xFF: Operation not permitted

Other: Operation failed due to internal error

8.7.2.1. Example 1

Set the RF_TXPower (settings index 17) to 4 dBm.

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

Response:

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

Successfully updated the user setting.

8.7.3. CMD_GET_REQ

This command can be used to query individual setting parameters in RRAM. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 35.
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 settings 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 due to invalid command

0x02: Operation failed due to invalid setting

0xFF: Operation not permitted

8.7.3.1. Example 1

Read the `RF_TXPower` (settings index 17).

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

Response:

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x08	0x58

Successfully read out the value 0x08 (8 dBm).

8.7.4. CMD_SETRAM_REQ

This command enables direct manipulation of the parameters in the module's settings in the local RAM copy. Changes to the parameters are applied immediately. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 36. Parameters of two or more Bytes have to be transferred with the LSB first unless noted differently in the corresponding description.
Format:

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x21	2 Bytes	1 Byte	(Length - 1) Bytes	1 Byte

Response (`CMD_SETRAM_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x61	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, settings set successfully

0x01: Operation failed, due to invalid command

0x02: Operation failed, due to invalid parameter

0xFF: Operation not permitted

8.7.4.1. Example 1

Setting the scan response packet `RF_ScanResponseData` to "TX power 4 dBm" (0x02 0x0A 0x04) seconds.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x21	0x04 0x00	0x0E	0x02 0x0A 0x04	0x25

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x61	0x01 0x00	0x00	0x62

Setting was set successfully.

8.7.5. CMD_GETRAM_REQ

This command can be used to query individual setting parameters in RAM. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 36.
Parameters of two or more bytes have to be transferred with the LSB first unless noted differently in the corresponding description.
Format:

Start signal	Command	Length	Settings index	CS
0x02	0x20	0x01 0x00	1 Byte	1 Byte

Response (CMD_GET_CNF):

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x60	2 Bytes	1 Byte	(Length - 1) Bytes	1 Byte

Status:

0x00: Request received, read out of setting successful

0x01: Operation failed, due to invalid command

0x02: Operation failed, due to invalid parameter

0xFF: Operation not permitted

8.7.5.1. Example 1

Request the current scan response packet RF_ScanResponseData.

Start signal	Command	Length	Settings index	CS
0x02	0x20	0x01 0x00	0x0E	0x2D

Response: The current RF_ScanResponseData in RAM is "TX power 4 dBm" (0x02 0x0A 0x04).

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x60	0x04 0x00	0x00	0x02 0x0A 0x04	0x6A

Setting was read successfully.

8.8. Message overview

Start signal	CMD	Message name	Short description	Chapter
0x02	0x00	CMD_RESET_REQ	Reset the module	8.2.3
0x02	0x01	CMD_GETSTATE_REQ	Request the current module state	8.2.2
0x02	0x02	CMD_SLEEP_REQ	Go to sleep	8.2.4
0x02	0x04	CMD_DATA_REQ	Send data to the connected device	8.6.1
0x02	0x06	CMD_CONNECT_REQ	Setup a connection with another device	8.5.1
0x02	0x07	CMD_DISCONNECT_REQ	Close the connection	8.5.5
0x02	0x09	CMD_SCAN_REQ	Start/stop scan	8.4.1
0x02	0x0D	CMD_PASSKEY_REQ	Respond to a pass key request	8.5.10
0x02	0x0E	CMD_DELETEBONDS_REQ	Delete bonding information	8.5.15
0x02	0x0F	CMD_GETBONDS_REQ	Read the MACs of bonded devices	8.5.14
0x02	0x10	CMD_GET_REQ	Read the module settings in RRAM	8.7.3
0x02	0x11	CMD_SET_REQ	Modify the module settings in RRAM	8.7.2
0x02	0x1A	CMD_PHYUPDATE_REQ	Update the PHY	8.5.8
0x02	0x1B	CMD_UARTDISABLE_REQ	Disable the UART	8.2.5
0x02	0x1C	CMD_FACTORYRESET_REQ	Perform a factory reset	8.7.1
0x02	0x1F	CMD_BOOTLOADER_REQ	Enable FOTA	8.2.7
0x02	0x24	CMD_NUMERIC_COMP_REQ	Confirm/reject the displayed pass key	8.5.13

Table 30: Message overview: Requests

Start signal	CMD	Message name	Short description	Chapter
0x02	0x40	CMD_RESET_CNF	Reset request received	8.2.3
0x02	0x41	CMD_GETSTATE_CNF	Return the current module state	8.2.2
0x02	0x42	CMD_SLEEP_CNF	Sleep request received	8.2.4
0x02	0x44	CMD_DATA_CNF	Data transmission request received	8.6.1
0x02	0x46	CMD_CONNECT_CNF	Connection setup request received	8.5.1

0x02	0x47	CMD_DISCONNECT_CNF	Disconnection request received	8.5.5
0x02	0x49	CMD_SCAN_CNF	Scan started/Stopped	8.4.1
0x02	0x4D	CMD_PASSKEY_CNF	Received the pass key	8.5.10
0x02	0x4E	CMD_DELETEBONDS_CNF	Deleted bonding information	8.5.15
0x02	0x4F	CMD_GETBONDS_CNF	Return the MAC of all bonded devices	8.5.14
0x02	0x50	CMD_GET_CNF	Return the requested module RRAM settings	8.7.3
0x02	0x51	CMD_SET_CNF	Module RRAM settings have been modified	8.7.2
0x02	0x5A	CMD_PHYUPDATE_CNF	Update Phy request received	8.5.8
0x02	0x5B	CMD_UARTDISABLE_CNF	Disable UART request received	8.2.5
0x02	0x5C	CMD_FACTORYRESET_CNF	Factory reset request received	8.7.1
0x02	0x5F	CMD_BOOTLOADER_CNF	Enabled the FOTA	8.2.7
0x02	0x64	CMD_NUMERIC_COMP_CNF	CMD_NUMERIC_COMP_REQ accepted	8.5.13

Table 31: Message overview: Confirmations

Start signal	CMD	Message name	Short description	Chapter
0x02	0x81	CMD_STARTUP_IND	Module has booted up	8.2.1
0x02	0x84	CMD_DATA_IND	Data has been received	8.6.3
0x02	0x86	CMD_CONNECT_IND	Connection established	8.5.2
0x02	0x87	CMD_DISCONNECT_IND	Disconnected	8.5.6
0x02	0x88	CMD_SECURITY_IND	Secured connection established	8.5.3
0x02	0x89	CMD_SCAN_IND	Peer device has been scanned	8.4.2
0x02	0x8D	CMD_PASSKEY_IND	Received a pass key request	8.5.11
0x02	0x93	CMD_MAXPAYLOAD_IND	Maximum payload has been updated	8.5.16
0x02	0x9A	CMD_PHYUPDATE_IND	PHY has been updated	8.5.9
0x02	0x9B	CMD_UARTENABLE_IND	UART was re-enabled	8.2.6
0x02	0xA4	CMD_DISPLAY_PASSKEY_IND	Display pass key	8.5.12
0x02	0xC4	CMD_TXCOMPLETE_RSP	Data has been sent	8.6.2
0x02	0xC6	CMD_LINKOPEN_RSP	Channel open, data transmission possible	8.5.4

Table 32: Message overview: Indications

9. User settings - Module configuration values

The settings described in this chapter are stored permanently in the module's RRAM 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.

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

9.1. 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.

9.1.1. 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

9.2. FS_DeviceInfo: Read the chip type and OS version

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
15	FS_DeviceInfo	-	-	read	13

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	Part ID
3 Bytes	4 Bytes	2 Bytes	4 Bytes

OS version:

0x030002: NCS 3.0.2

Build code:

0x41414242: AABB

Package variant:

0x5146: QFN48

0x4341: CSP300

PART ID:

0x00054B15: nRF54L15

9.2.1. 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 = 0x030002 (NCS 3.0.2)
Build code = 0x41414242 (AABB)
Package variant = 0x5146 (QFN48)
Chip ID = 0x00054B15
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	0x0E 0x00	0x00	0x02 0x00 0x03 0x42 0x42 0x41 0x41 0x46 0x51 **0x15 0x4B 0x05 0x00**	0x11

9.3. FS_MAC: Read the MAC address

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

This setting contains the unique MAC address of the module.

9.3.1. Example 1

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

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

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

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

9.4. FS_BTMAC: Modify the Bluetooth® conform MAC address

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
4	FS_BTMAC	See description	See description	read/write	6

This setting contains the Bluetooth® LE conform **public** MAC address of the module. It is used to identify the device on the radio interface.

By default, this user setting contains 0xFFFFFFFFFFFF, which means that the MAC address bought by Würth Elektronik eiSos is used. It consists of the Würth Elektronik eiSos MAC ID 0x0018DA followed by the FS_SerialNumber of the module.

The FS_BTMAC can be set to any public MAC address, other than 0x0000000000 and 0xFFFFFFFFFFFF. For purchasing public MAC addresses, please refer to the IEEE standards associaton [10]. If set to 0xFFFFFFFFFFFF, the Proteus-IV uses the default address mentioned above.

Please note that LSB is transmitted first in all commands.

9.4.1. Example 1

Set the Bluetooth®-conform MAC address of the module to 0x31 0x32 0x33 0x34 0x35 0x36 using CMD_SET_REQ with settings index 4

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x07 0x00	0x04	0x31 0x32 0x33 0x34 0x35 0x36	0x17

Response CMD_SET_CNF: Successfully modified the setting.

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

9.4.2. Example 2

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

9.5. 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.

9.5.1. 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	0x47

9.6. RF_DeviceName: Modify the device name

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
2	RF_DeviceName	See description	"Prot4"	read/write	1-28

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.

9.6.1. 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

9.6.2. 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 name as 0x41 0x32 0x37 0x32 0x31 = "A2721".

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

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

9.7.1. Example 1

Set the static passkey 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

9.7.2. Example 2

Request the static passkey 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

9.8. RF_SecFlags: Modify the security settings

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

This setting configures the input/output (IO) capabilities of the radio module and the minimum security level that needs to be reached to access the Bluetooth® characteristics for data transmission.
Both the radio module and the peer device define what input/output capabilities they provide. Based on that, the security level of the Bluetooth® LE link is negotiated during connection setup. For more information, see chapter Bluetooth® security.

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.

Value	IO caps	Minimum peripheral level	Description
0x01	No IO caps	Level L2	The end device cannot enter or show a key.
0x02	Keyboard only	Level L2	The end device can enter but not display a key.
0x03	Keyboard only	Level L4	The end device can enter but not display a key.
0x04	Display only Static passkey	Level L2	A static key RF_StaticPasskey is used by the end device. It cannot enter a key.
0x05	Display only Static passkey	Level L4	A static key RF_StaticPasskey is used by the end device. It cannot enter a key.
0x06	Display only	Level L2	The end device can display but not enter a key.
0x07	Display only	Level L4	The end device can display but not enter a key.
0x08	Keyboard and display	Level L2	The end device can display and enter a key.
0x09	Keyboard and display	Level L4	The end device can display and enter a key.

Table 33: Security configuration flags

9.8.1. Example 1

Set the security flags to 0x03 using CMD_SET_REQ with settings index 12

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x0C	0x03	0x1E

Response `CMD_SET_CNF`: Successfully modified the setting.

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

9.8.2. 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 1.

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

9.9. RF_AdvertisingData: Modify the content of the advertising packet

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
13	RF_AdvertisingData	See description	None	read/write	0-31

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

Please check chapter 7.4.5 for details and restrictions before modifying this user setting.

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

To use the standard content of the advertising packet again, please set this user setting RF_AdvertisingData to a value with zero length.

9.9.1. Example 1

Set the custom data of the advertising packet to:

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

using CMD_SET_REQ with settings index 13

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

Response CMD_SET_CNF: Successfully modified the setting.

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

9.9.2. Example 2

Request the custom advertising data of the module using `CMD_GET_REQ` with settings index 13

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

Response `CMD_GET_CNF`:Successfully read out the content as:

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

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

9.10. RF_ScanResponseData: Modify the content of the scan response packet

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
14	RF_ScanResponseData	See description	None	read/write	0-31

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

Please check chapter 7.4.5 for details and restrictions before modifying this user setting.

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

To use the standard content of the scan response packet again, please set this user setting RF_ScanResponseData to a value with zero length.

9.10.1. Example 1

Set the custom data of the scan response packet to:

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

using CMD_SET_REQ with settings index 14

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

Response CMD_SET_CNF: Successfully modified the setting.

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

9.10.2. Example 2

Request the custom scan response data of the module using `CMD_GET_REQ` with settings index 14

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

Response `CMD_GET_CNF`:Successfully read out the content as:

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

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

9.11. RF_AdvertisingInterval: Modify the advertising interval

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
7	RF_AdvertisingInterval	See description	80-160	read/write	2 + 2

The RF_AdvertisingInterval defines how often advertising packets are transmitted. The value is the minimum and maximum advertising interval in 0.625 ms steps. This means a value of 10 is 6.25 ms.

Example:
A value of 0x00A00050 means 0x0050 (80 = 50 ms) - 0x00A0 (160 = 100 ms).

The value for the minimum advertising interval and the value for the maximum advertising interval must be within 32 (20 ms) and 16384 (10240 ms). Only settings are accepted where the minimum advertising interval is lower than or equal to the maximum advertising interval.

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

9.11.1. Example 1

Set the advertising interval to 20-30 ms (32/0x0020 - 48/0x0030) using CMD_SET_REQ with settings index 7.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x05 0x00	0x07	0x20 0x00 0x30 0x00	0x01

Response CMD_SET_CNF: Successfully modified the setting.

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

9.11.2. Example 2

Request the advertising interval 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 of 0x00A00050, which means 0x0050 (80 = 50 ms) - 0x00A0 (160 = 100 ms).

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x05 0x00	0x00	0x50 0x00 0xA0 0x00	0xA7

9.12. RF_ConnectionInterval: Modify the connection interval

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
8	RF_ConnectionInterval	See description	12-36	read/write	2 + 2

The user setting RF_ConnectionInterval defines the minimum and maximum connection interval, which is used to negotiate the connection interval during connection setup. The value is the minimum and maximum connection interval in 1.25 ms steps. This means a value of 10 is 12.5 ms.

The 4-byte value of the user setting RF_ConnectionInterval consists of the 2-byte value of the minimum connection interval, followed by the 2-byte value of the maximum connection interval.

Example:
A value of 0x0024000C means 0x000C (12 = 15 ms) - 0x0024 (36 = 45 ms).

The value for the minimum connection interval and the value for the maximum connection interval must be within 6 (7.5 ms) and 3200 (4000 ms). Only settings are accepted where the minimum connection interval is lower than or equal to the maximum connection interval.

Further information:

- The minimum and maximum connection interval parameters specify the boundaries of the connection interval as determined in the negotiation procedure between the central and the peripheral during connection setup. The connection interval defines the frequency of communication during connection setup and data transmission. The lower the connection interval is, the more frequently the connected devices communicate with each other and thus the more power is consumed.
  If a Bluetooth® LE device (e.g., a smartphone) connects as a central to a Proteus-IV module (peripheral) and the connection interval settings do not coincide, the Proteus-IV requests the smartphone to accept its settings after 5 s. If the smartphone does not accept the settings, it will be requested a further 3 times with a delay of 10 s. If the peripheral's settings requests have been rejected in all cases, the connection will be shut down.

Please ensure that all members (Proteus-IV, smartphones, 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 open connection.

Please note that the minimum connection interval supported by iOS is 15 ms. Furthermore, the minimum connection interval for Apple devices shall be a multiple of 15 ms! Please refer to Apple's Bluetooth® LE developer guide for best practices.

9.12.1. Example 1

Set the `RF_ConnectionInterval` to 30-45 ms (24/0x0018 - 36/0x0024) using `CMD_SET_REQ` with settings index 8.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x05 0x00	0x08	0x18 0x00 0x24 0x00	0x22

Response `CMD_SET_CNF`: Successfully modified the setting.

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

9.12.2. Example 2

Request the minimum and maximum connection interval of the module using `CMD_GET_REQ` with settings index 8

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

Response `CMD_GET_CNF`: Successfully read out the value 15-45 ms (12/0x000C - 36/0x0024).

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x05 0x00	0x00	0x0C 0x00 0x24 0x00	0x7F

9.13. RF_TXPower: Modify the output power

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

This setting determines the output power in dBm of the module. The value must 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
Two's complement, hexadecimal	0xD8	0xEC	0xF0	0xF4	0xF8	0xFC	0x00

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

9.13.1. 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

9.13.2. 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 = 4 dBm

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

9.14. RF_SPPServiceUUID: Configure the SPP service UUID

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
32	RF_SPPServiceUUID	See description	0xb70c0001686c4 036bb28b797ae6 a8d3a	read/write	16

Set the service UUID of the SPP-like profile. For more information about the UUID definition, please refer to chapter 10.2.

Please note that the UUID 0xFB349B5F8000008000100000xxxx0000 (0000xxxx-0000-1000-8000-00805f9b34fb) is reserved for 16-bit UUIDs and must not be used here. Furthermore, the service UUID must be different from RF_SPPTXUUID and RF_SPPRXUUID.

9.14.1. Example 1

Set the service UUID to 0xEFEEEDEC-EBEA-E9E8-E7E6-E5E4E3E2E1E0 using CMD_SET_REQ with settings index 32

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x11 0x00	0x20	0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF	0x22

Response CMD_SET_CNF: Successfully modified the setting.

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

9.14.2. Example 2

Request the service UUID of the module using CMD_GET_REQ:

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

Response CMD_GET_CNF: Successfully read out the value 0xb70c0001-686c-4036-bb28-b797ae6a8d3a.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x11 0x00	0x00	0x3A 0x8D 0x6A 0xAE 0x97 0xB7 0x28 0xBB 0x36 0x40 0x6C 0x68 0x01 0x00 0x0C 0xB7	0x4B

9.15. RF_SPPRXUUID: Configure the SPP RX UUID

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
33	RF_SPPRXUUID	See description	0xb70c0002686c4 036bb28b797ae6a 8d3a	read/write	16

Set the RX UUID of the SPP-like profile. This characteristic has the function to transmit data from the connected remote peer to the radio module via write command. For more information about the UUID definition, please refer to chapter 10.2.

Please note that the UUID 0xFB349B5F8000008000100000xxxx0000 (0000xxxx-0000-1000-8000-00805f9b34fb) is reserved for 16-bit UUIDs and must not be used here. Furthermore, the RX UUID must be different from RF_SPPServiceUUID and RF_SPPTXUUID.

9.15.1. Example 1

Set the RX UUID to 0xEFEEEDEC-EBEA-E9E8-E7E6-E5E4E3E2E1E0 using CMD_SET_REQ with settings index 33

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x11 0x00	0x21	0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF	0x23

Response CMD_SET_CNF: Successfully modified the setting.

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

9.15.2. Example 2

Request the RX UUID of the module using CMD_GET_REQ:

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

Response CMD_GET_CNF: Successfully read out the value 0xb70c0002-686c-4036-bb28-b797ae6a8d3a.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x11 0x00	0x00	0x3A 0x8D 0x6A 0xAE 0x97 0xB7 0x28 0xBB 0x36 0x40 0x6C 0x68 0x02 0x00 0x0C 0xB7	0x48

9.16. RF_SPPTXUUID: Configure the SPP TX UUID

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
34	RF_SPPTXUUID	See description	0xb70c0003686c4 036bb28b797ae6a 8d3a	read/write	16

Set the TX UUID of the SPP-like profile. This characteristic has the function to transmit data from the radio module to the connected remote peer via notification. For more information about the UUID definition, please refer to chapter 10.2.

Please note that the UUID 0xFB349B5F8000008000100000xxxx0000 (0000xxxx-0000-1000-8000-00805f9b34fb) is reserved for 16-bit UUIDs and must not be used here. Furthermore, the TX UUID must be different from RF_SPPServiceUUID and RF_SPPRXUUID.

9.16.1. Example 1

Set the TX UUID to 0xEFEEEDEC-EBEA-E9E8-E7E6-E5E4E3E2E1E0 using CMD_SET_REQ with settings index 34

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x11 0x00	0x22	0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF	0x20

Response CMD_SET_CNF: Successfully modified the setting.

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

9.16.2. Example 2

Request the TX UUID of the module using CMD_GET_REQ:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x22	0x21

Response CMD_GET_CNF: Successfully read out the value 0xb70c0003-686c-4036-bb28-b797ae6a8d3a.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x11 0x00	0x00	0x3A 0x8D 0x6A 0xAE 0x97 0xB7 0x28 0xBB 0x36 0x40 0x6C 0x68 0x03 0x00 0x0C 0xB7	0x49

9.17. 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® device. It is a 2-byte field defined by the Bluetooth® SIG. Please check the Bluetooth® assigned numbers [11].

9.17.1. 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

9.17.2. 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

9.18. RF_MaxPeripheralConnections: Define the maximum number of peripheral connections

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
30	RF_MaxPeripheralConnections	1 - 5	1	read/write	1

This setting determines how many peripheral connections the radio module can hold in parallel. If the number of active peripheral connections equals RF_MaxPeripheralConnections, the module stops advertising to block additional connections. In case the number of active peripheral connections is lower than RF_MaxPeripheralConnections, the module continues advertising to allow further connections to be established.

9.18.1. Example 1

Set the maximum peripheral connections to 5, using CMD_SET_REQ with settings index 30

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x1E	0x05	0x0A

Response CMD_SET_CNF: Successfully modified the setting.

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

9.18.2. Example 2

Request the value of RF_MaxPeripheralConnections using CMD_GET_REQ with settings index 30

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

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

9.19. UART_ConfigIndex: Modify the UART speed

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
11	UART_ConfigIndex	See description	23	read/write	1

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

UART_ConfigIndex	Rate [Baud]	Real rate [Baud]	Flow control	Parity
0	1200	1205	no	none
1	1200	1205	yes	none
2	2400	2396	no	none
3	2400	2396	yes	none
4	4800	4808	no	none
5	4800	4808	yes	none
6	9600	9598	no	none
7	9600	9598	yes	none
8	14400	14401	no	none
9	14400	14401	yes	none
10	19200	19208	no	none
11	19200	19208	yes	none
12	28800	28777	no	none
13	28800	28777	yes	none
14	38400	38369	no	none
15	38400	38369	yes	none
16	56000	55944	no	none
17	56000	55944	yes	none
18	57600	57762	no	none
19	57600	57762	yes	none
20	76800	76923	no	none
21	76800	76923	yes	none
22	115200	115108	no	none
23	115200	115108	yes	none
25	230400	231884	yes	none
27	250000	250000	yes	none
29	460800	457143	yes	none
31	921600	941176	yes	none

33	1000000	1000000	yes	none
64	1200	1205	no	even
65	1200	1205	yes	even
66	2400	2396	no	even
67	2400	2396	yes	even
68	4800	4808	no	even
69	4800	4808	yes	even
70	9600	9598	no	even
71	9600	9598	yes	even
72	14400	14401	no	even
73	14400	14401	yes	even
74	19200	19208	no	even
75	19200	19208	yes	even
76	28800	28777	no	even
77	28800	28777	yes	even
78	38400	38369	no	even
79	38400	38369	yes	even
80	56000	55944	no	even
81	56000	55944	yes	even
82	57600	57762	no	even
83	57600	57762	yes	even
84	76800	76923	no	even
85	76800	76923	yes	even
86	115200	115108	no	even
87	115200	115108	yes	even
89	230400	231884	yes	even
91	250000	250000	yes	even
93	460800	457143	yes	even
95	921600	941176	yes	even
97	1000000	1000000	yes	even

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

Please note that due to the HF-activity of the chip, single Bytes on the UART can get lost, when using a very fast UART data rate. To avoid loosing single bytes, please enable the UART flow control.

9.19.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

9.19.2. 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 0x16, which equals 115200 Baud without flow control and parity none.

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

9.20. UART_TransparentETXConfig: Configure the usage of the ETX in transparent mode

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
6	UART_TransparentETXConfig	See description	0	read/write	1

The user setting UART_TransparentETXConfig specifies the behavior of the ETX triggers in transparent mode.

Bit no.	Description
1 : 0	Reserved
2	TRANSP_REMOVE_RECEIVE_ETX: If this bit is set, the ETX characters will be removed from the UART data stream before transmitting it via radio.
3	TRANSP_ADD_TRANSMIT_ETX: If this bit is set, the ETX characters will be added to the radio data stream before transmitting it via UART.

9.20.1. Example 1

Remove the ETX characters using CMD_SET_REQ with settings index 6

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x06	0x04	0x13

Response CMD_SET_CNF: Successfully modified the setting.

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

9.20.2. Example 2

Request the UART_TransparentETXConfig using CMD_GET_REQ:

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

Response CMD_GET_CNF: Successfully read out the value 0x04, meaning that the ETX characters are removed from the byte stream.

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

9.21. UART_TransparentReceiveETX: Modify the transparent receive ETX characters

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
9	UART_TransparentReceiveETX	-	0x0D0A	read/write	2

This parameter defines the ETX characters used to trigger radio transmission in transparent mode.

9.21.1. Example 1

Set the ETX to 0x1122 using CMD_SET_REQ with settings index 9.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x09	0x11 0x22	0x2A

Response CMD_SET_CNF: Successfully modified the setting.

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

9.21.2. Example 2

Request the ETX characters 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 0x0D 0x0A.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x0D 0x0A	0x56

9.22. UART_TransparentTransmitETX: Modify the transparent transmit ETX characters

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
10	UART_TransparentTransmitETX	-	0x0D0A	read/write	2

This parameter defines the ETX characters that are added to the received radio data before sending it to the host via UART.

9.22.1. Example 1

Set the ETX to 0x1122 using CMD_SET_REQ with settings index 10.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x0A	0x11 0x22	0x29

Response CMD_SET_CNF: Successfully modified the setting.

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

9.22.2. Example 2

Request the ETX characters of the module using CMD_GET_REQ with settings index 9.

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

Response CMD_GET_CNF: Successfully read out the value 0x0D 0x0A.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x0D 0x0A	0x56

9.23. CFG_Flags: Configure the module

Settings index	Designation	Permissible values	Default value	Permissions	Number of bytes
5	CFG_Flags	See description	11 (0b0000000000001011)	read/write	2

The user setting CFG_Flags specifies various module features.

Bit no.	Name	Description
0	Autostart advertising	Set this bit to 1 to autostart advertising after boot-up.
1	Record RSSI	Set this bit to 1 to record the RSSI value of each received radio packet.
2	Auto start FOTA	When this bit is 1, the FOTA functionality is always available and does not need to be started by the user.
3	Double device scan filter	When this bit is 1, the scanner prints only one CMD_SCAN_IND message per device found during scan.
4	Not connectable advertising	When this bit is 1, the module advertises in non-connectable mode.
5	Enable CMD_TXCOMPLETE_RSP	When this bit is 1, on each CMD_DATA_REQ, one or several CMD_TXCOMPLETE_RSP messages are returned once the data has been sent out to the peer devices. Enabling this message will result in more UART traffic, reducing the throughput.
6-15	Reserved	Do not modify.

9.23.1. Example 1

Switch all features off using CMD_SET_REQ with settings index 5

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x05	0x00 0x00	0x15

Response CMD_SET_CNF: Successfully modified the setting.

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

9.23.2. Example 2

Request the CFG_Flags using CMD_GET_REQ:

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

Response `CMD_GET_CNF`: Successfully read out the value 0x0003, meaning that advertising autostart and RSSI recording are enabled.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x03 0x00	0x52

Settings index	Designation	Summary	Permissible values	Default value	Permissions	Number of bytes
1	FS_FWVersion	Version of the firmware	-	-	read	3
2	RF_DeviceName	Name of the module	See description	"Prot4"	read/write	28
3	FS_MAC	MAC address of the module	-	-	read	8
4	FS_BTMAC	Bluetooth® LE conform MAC address of the module	-	-	read/write	6
5	CFG_Flags	CFG Flags	See description	11	read/write	2
6	UART_ TransparentETXConfig	ETX configuration	See description	0	read/write	1
7	RF_ AdvertisingInterval	Advertising interval	See description	80-160	read/write	2 + 2
8	RF_ConnectionInterval	See description	Connection interval	12-36	read/write	2 + 2
9	UART_ TransparentReceiveETX	ETX characters	See description	0x0D0A	read/write	2
10	UART_ TransparentTransmitETX	ETX characters	See description	0x0D0A	read/write	2
11	UART_ConfigIndex	Baud rate of the UART	See description	23	read/write	1
12	RF_SecFlags	Security settings of the module	See description	1	read/write	1
13	RF_AdvertisingData	Custom advertising packet data	See description	-	read/write	0-31
14	RF_ScanResponseData	Custom scan response packet data	See description	-	read/write	0-31
15	FS_DeviceInfo	Information about the chip	-	-	read	13
16	FS_SerialNumber	Serial number of the module	-	-	read	3
17	RF_TXPower	Output power [dBm] Two's complement	See description	8	read/write	1
18	RF_StaticPasskey	6 digit pass key	See description	"123123"	read/write	6
25	RF_Appearance	Appearance	0-65535	0	read/write	2
30	RF_ MaxPeripheralConnections	Number of peripheral connections	1 - 5	1	read/write	1
32	RF_SPPServiceUUID	Service UUID	See description	See description	read/write	16
33	RF_SPPRXUUID	RX characteristic UUID	See description	See description	read/write	16
34	RF_SPPTXUUID	TX characteristic UUID	See description	See description	read/write	16

Table 35: Table of user settings

Settings index	Designation	Summary	Permissible values	Default value	Permissions	Number of bytes
13	RF_AdvertisingData	Custom advertising packet data	See description	-	read/write	0-31
14	RF_ScanResponseData	Custom scan response packet data	See description	-	read/write	0-31

Table 36: Table of runtime settings

10. Customizing the Proteus-IV module

10.1. Device name and appearance

The name of the device appearing on the Bluetooth® LE interface is defined by the parameter `RF_DeviceName`. Furthermore, its appearance can be configured by the parameter `RF_Appearance`. It is a 16-bit value defined by the Bluetooth® SIG to show the device type.

10.2. UUID

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

The SPP-like profile consists of the 128-bit UUIDs of the underlying characteristics and services:

Characteristic	UUID
128-bit `RF_SPPServiceUUID`	0x6E400001-C352-11E5-953D-0002A5D5C51B
128-bit `RF_SPPRXUUID`	0x6E400002-C352-11E5-953D-0002A5D5C51B
128-bit `RF_SPPTXUUID`	0x6E400003-C352-11E5-953D-0002A5D5C51B

Table 37: UUID default values

To generate custom UUIDs, the Bluetooth® SIG recommends using the tool:
http://www.uuidgenerator.net/

11. Transparent mode

The Proteus-IV implements a feature that allows easy integration of the Proteus-IV Bluetooth® LE module to an already existing host. The transparent mode offers plug-and-play installation without previous configuration of the Proteus-IV. It is tailored for easy communication with mobile Bluetooth® LE devices like smartphones. Please refer to chapter 7.1 to see how to enable transparent mode.

The transparent mode contains the following key features:

- Automatic UART enabling: The UART of the Proteus-IV is automatically enabled when at least one link to the Bluetooth® LE peer is open. If no link is open, the UART is switched off.
- Transparent UART interface: The serial interface of the Proteus-IV is no longer driven by commands. This means when the UART of the module is enabled, all data sent to the UART is forwarded to all open Bluetooth® LE links. On the other hand, all data received by radio is sent from the Proteus-IV to the connected host without additional protocol data.
- The data received via UART is broadcast to all open Bluetooth® LE links.
- The complete configuration of the Proteus-IV in transparent mode, like UART baud rate or the Bluetooth® device name, is shared with the settings defined in command mode. Since the commands of the command interface are no longer valid, a Proteus-IV cannot be configured when running in transparent mode. To configure the Proteus-IV, it must be booted in command mode once to use the CMD_SET_REQ command.

The data sent to the module's UART is buffered in the Proteus-IV. If UART flow control is enabled in the Proteus-IV, the */RTS* pin indicates when the buffer is ready to receive more data. The radio transmission of the buffered data is done when

- the two UART_TransparentReceiveETX characters are received by the radio module's UART, followed by a pause (see screenshot in Figure 34). The required pause must be larger than 10 byte durations and 1 ms.
- the currently used UART buffer is full (see screenshot below). The Proteus-IV uses several UART buffers. If all of them are full, the */RTS* line stays high until the first buffer has been emptied again.

In case of the trigger caused by a full UART buffer, the */RTS* goes high 11 µs after the last valid byte has been received. Hence, when using baud rates faster than 460800 Baud, bytes get lost if the host transmits the bytes too quickly to the module.

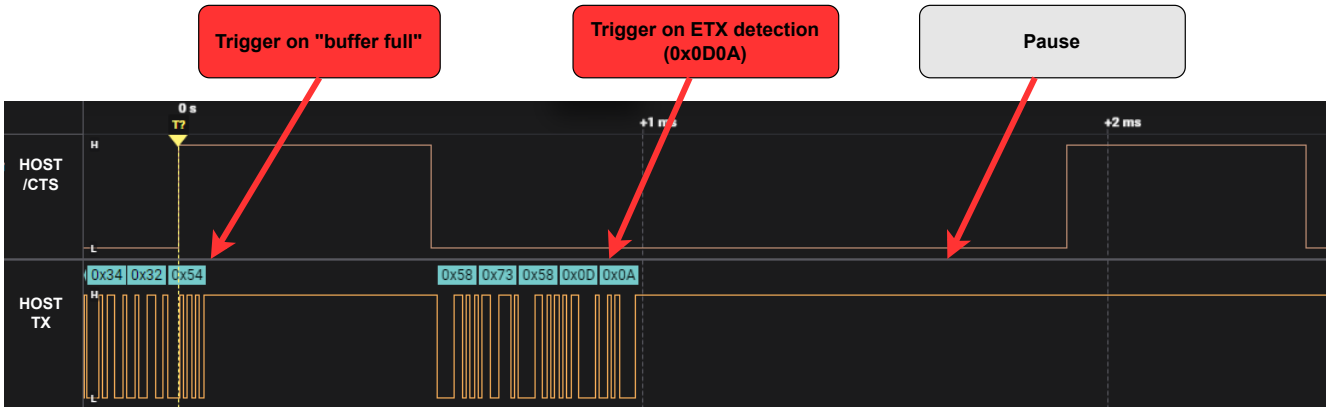


Figure 34: Trigger Bluetooth® LE transmission

The data collected in the module's UART buffer is split into several Bluetooth® LE packets and transmitted sequentially.
As the packets are sent via broadcast messages to all open Bluetooth® LE links, the link with the lowest maximum transmission unit (MTU) determines the size of the Bluetooth® LE packet. The Proteus-IV supports up to 244 bytes payload per Bluetooth® LE packet (corresponding to an MTU of 247 bytes), which may be negotiated by the central device using an MTU request. If no MTU request is sent by the connecting central device, a payload packet size of 20 bytes is used per connection interval as given by the Bluetooth® standard (compatibility mode to Bluetooth® LE 4.0 devices).

11.1. Best practices

- As in transparent mode no data integrity check can be performed by the radio module, it is recommended to check for invalid or missing bytes on the host.
- For baud rates faster than 460800 Baud, use the trigger on ETX detection method to avoid data loss.

12. Timing parameters

12.1. Reset and sleep

In command mode, after power-up, module reset, or wake-up, a `CMD_STARTUP_IND` is sent via the serial interface as soon as the module is ready for operation.

Description	Typ.	Unit
Startup message is sent after reset/sleep	70	ms

12.2. Bluetooth® LE timing parameters

The timing behavior of the Bluetooth® interface is determined by the following settings:

- The `RF_AdvertisingInterval` defines the advertising interval, which is the period during which advertising packets are sent. The larger this interval, the lower the current consumption in idle phase.
- The `RF_ConnectionInterval` defines the connection interval, which is the period during which data is exchanged on the Bluetooth® LE interface after connection setup. The larger this interval, the lower the current consumption in connection phase, but the lower the throughput.

12.3. Connection establishment

The time needed to establish a connection is the sum of the time needed to detect the selected peripheral on air and the time needed for connection parameter negotiation and service discovery.

Peripheral detection To establish a connection, the initiating device (central) waits for an advertising packet that was sent by the peripheral to which it wants to connect. 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 (see `RF_AdvertisingInterval`).

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

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

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

12.4. Connection-based data transmission in command mode

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

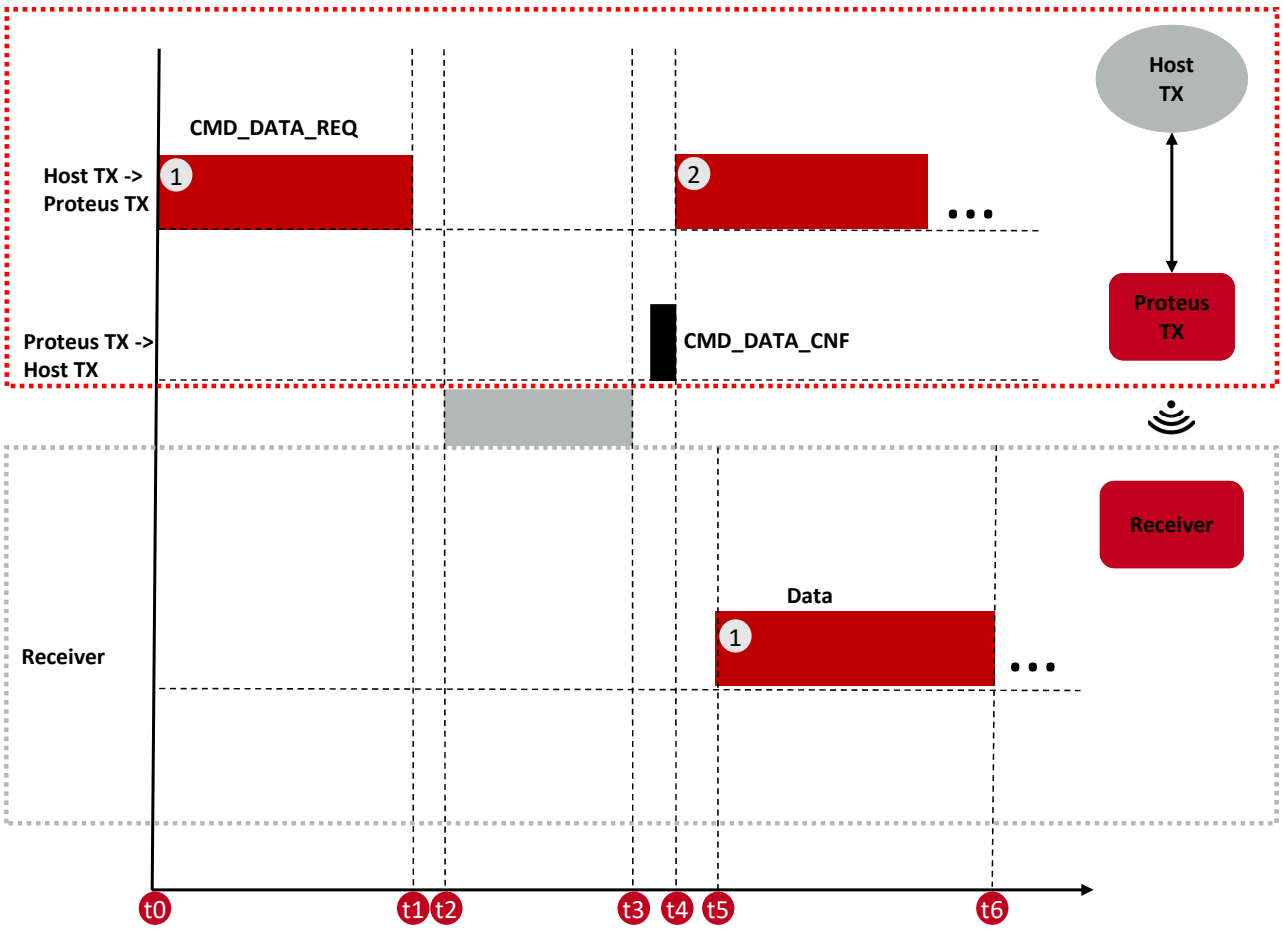


Figure 35: Command sequence when transmitting data

12.4.1. Maximum command mode data throughput

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

- A Proteus-IV radio module in command mode connected to a fast microcontroller (STM32 on NUCLEO-F401RE).

- A Bluetooth®-enabled peer device acting as a receiver. One active connection[1], no parallel advertising or scanning.
- Transmit 244 bytes payload per packet.

Receiver	Radio mode [kBit/s]	Connection interval [ms]	Transmitter UART Baudrate [kBaud]	244×8/(t6-t0) [kBit/s] (Throughput)
Another Proteus-IV using 1 MBaud UART	125	7.5	115.2	20.796
Another Proteus-IV using 1 MBaud UART	125	7.5	1000	20.375
Another Proteus-IV using 1 MBaud UART	1000	7.5	115.2	72.995
Another Proteus-IV using 1 MBaud UART	1000	7.5	1000	241.274
Another Proteus-IV using 1 MBaud UART	2000	7.5	1000	245.534
Android phone	1000	11.25	1000	239.921
Android phone	2000	11.25	1000	242.251
iOS phone	2000	15	1000	202.504

Table 38: Maximum throughput timings, packet error rate = 0%

[1] In case of multiple active connections, the data throughput is reduced.

12.5. Connection-based data transmission in transparent mode

After the connection has been set up, data can be transmitted to the radio module as described in chapter `Transparent mode`. It buffers the data in the module and sends it with the next connection interval event. Here the */RTS* indicates whether there is memory left to buffer another UART packet or not.
The following image shows the sequence when sending data:

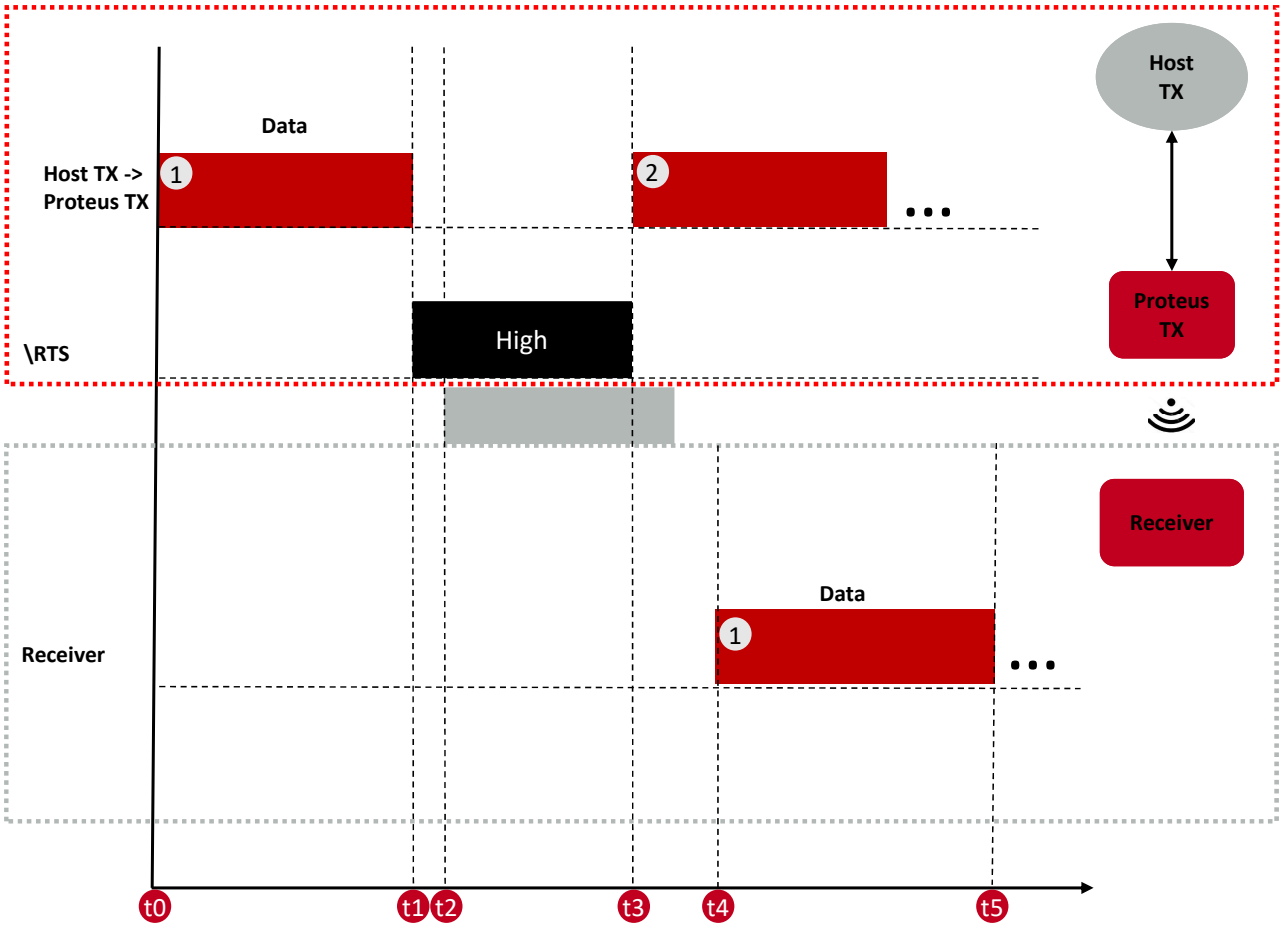


Figure 36: Sequence when transmitting data

12.5.1. Maximum transparent data throughput

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

- A Proteus-IV radio module in transparent mode connected to a fast microcontroller (STM32 on NUCLEO-F401RE).
- A Bluetooth®-enabled peer device acting as a receiver. One active connection[2], no par-

[2] In case of multiple active connections, the data throughput is reduced.

allel advertising or scanning.

- Transmit 244 bytes payload chunks[3] while considering the UART's */RTS* line.

Receiver	Radio mode [kBit/s]	Connection interval [ms]	Transmitter UART Baudrate [kBaud]	242×8/(t5-t0) [kBit/s] (Throughput)
Another Proteus-IV using 1 MBaud UART	1000	7.5	115.2	81.807
Another Proteus-IV using 1 MBaud UART	1000	7.5	1000	267.084
Another Proteus-IV using 1 MBaud UART	2000	7.5	1000	472.904
Android phone	1000	11.25	1000	265.086
Android phone	2000	11.25	1000	392.697
iOS phone	2000	15	1000	343.931

Table 39: Maximum throughput timings, packet error rate = 0%

[3] Radio transmission trigger is not done by ETX, but by radio buffer full.

13. Use cases and examples

This chapter guides you through practical, real-world applications of the Proteus-IV module. Each tutorial is designed as a complete hands-on example that you can follow step-by-step, from initial setup through to successful operation. Whether you're new to Bluetooth® LE development or an experienced engineer, these tutorials will help you quickly understand and implement the module's capabilities.
The examples are structured to build your knowledge progressively:

- **Module configuration** tutorials show you how to customize the module's settings in both command and config mode.
- **Connection setup and data exchange** tutorials demonstrate the three primary operating modes: transparent mode for simple applications, and peripheral/central command mode for full function control.
- **Advanced features** cover power optimization through sleep mode, firmware updates, and custom beacon broadcasting.

By combining the techniques from these tutorials, you can create virtually any Bluetooth® LE application. For additional code examples and ready-to-use implementations, please refer to the *Wireless Connectivity SDK* [8, 9], which provides working code for various micro-controller platforms.

Hardware setup: All examples use the Proteus-IV evaluation board (EV-Board) connected to a PC via USB, with some tutorials also requiring a smartphone. Before starting any tutorial, ensure your evaluation board is properly configured:

- **JP1 - Mode Selection:** Set the jumpers for *MODE_0* (P1.05) and *MODE_1* (P2.05) according to each tutorial's requirements. These pins determine which operating mode the module boots into.
- **JP2 - UART Connection:** Install jumpers on *RXD*, *TXD*, and if using flow control, also */RTS* and */CTS*. These connect the module's UART to the USB interface.

The mode pin settings are only read during boot-up. If you change the jumpers, you must reset the module for the changes to take effect.

Each tutorial specifies exactly which jumper settings to use, what messages to expect, and how to verify successful operation. Let's begin!

13.1. Module configuration

13.1.1. Configuration in command mode

This tutorial shows you how to configure the Proteus-IV module in command mode. We will read the current UART baud rate, change it, and verify the change after a reset. This is a fundamental skill for customizing your module's settings.

What you'll need:

- Proteus-IV EV-Board connected to your PC via USB, with the *MODE_0* and *MODE_1* pin pulled to LOW (leave mode pin jumpers on JP1 of EV-Board unconnected)
- WE UART Terminal program [12] (or similar terminal software)
- Terminal configured to 115200 Baud, 8 data bits, no parity, 1 stop bit (8N1), with flow control enabled

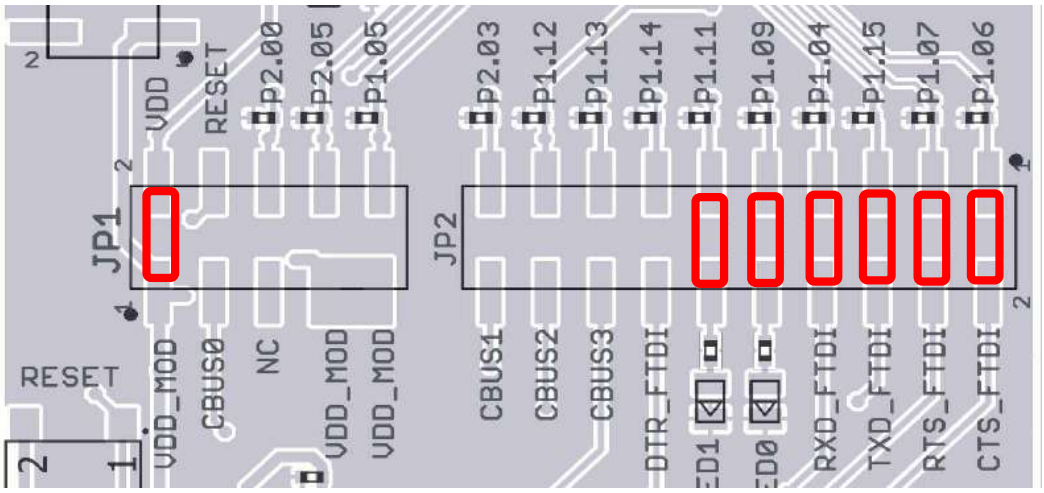


Figure 37: Proteus-IV EV-Board configured for command mode

Step 1: Connect and verify the module is ready First, connect your EV-Board to your PC. Open the WE UART Terminal and configure it with the settings above. After powering on or resetting the module, you should receive a startup indication:

Description	Command/Response
⇐ Module sends startup indication	0x02 0x81 0x01 0x00 **0x01** 0x83

This `CMD_STARTUP_IND` message tells us the module has booted successfully in command mode (**0x01**).

Step 2: Stop advertising to prepare for configuration Before changing settings, it's good practice to stop any ongoing radio activity. We'll disable advertising with `CMD_ADVERTISING_REQ`:

Description	Command/Response
⇒ Send disable advertising command	0x02 0x12 0x01 0x00 0x00 0x11
⇐ Module confirms	0x02 0x52 0x01 0x00 **0x00** 0x51

The status byte **0x00** in the `CMD_ADVERTISING_CNF` confirms advertising has been disabled successfully.

Step 3: Read the current UART baud rate Now let's check what baud rate is currently configured. We use `CMD_GET_REQ` with settings index **11** (`UART_ConfigIndex`):

Description	Command/Response
⇒ Request UART config	0x02 0x10 0x01 0x00 **0x0B** 0x18
⇐ Module responds	0x02 0x50 0x02 0x00 0x00 **0x17** 0x47

The parameter byte **0x17** (decimal 23) tells us the current configuration is 115200 Baud with flow control. You can find all possible values in the section `UART_ConfigIndex` and available user settings in Table 35.

Step 4: Change the UART baud rate Let's change the baud rate to 57600 Baud with flow control (index **0x13** = 19). Use `CMD_SET_REQ`:

Description	Command/Response
⇒ Set new UART config	0x02 0x11 0x02 0x00 0x0B **0x13** 0x09
⇐ Module confirms	0x02 0x51 0x01 0x00 0x00 0x52
⇐ Module restarts	0x02 0x81 0x01 0x00 0x01 0x83

Important: The module automatically resets after changing the UART settings. After the reset, the module is now communicating at 57600 Baud!

Step 5: Verify the change Close your terminal connection, change your terminal software to 57600 Baud, and reconnect. Now perform a manual reset by pulling the */RESET* pin LOW for at least 10 ms, then releasing it to HIGH. You should see the startup indication again. Let's verify the new setting:

Description	Command/Response
⇒ Request UART config again	0x02 0x10 0x01 0x00 0x0B 0x18
⇐ Module responds	0x02 0x50 0x02 0x00 0x00 **0x13** 0x43

Perfect! The parameter byte is now **0x13** (decimal 19), confirming our new setting.

What we accomplished: Congratulations! You have successfully:

- Connected to the module in command mode
- Stopped advertising to prepare for configuration
- Read a user setting from RRAM memory (UART baud rate)
- Modified the user setting and saved it to RRAM
- Verified the change persists after a reset

This same process can be used to modify any user setting listed in chapter `User settings - Module configuration values`. Remember that settings stored in RRAM persist even when the module is powered off, making them ideal for your permanent configuration.

13.1.2. Configuration in config mode

This tutorial demonstrates how to use config mode for module configuration. Config mode is especially useful when you don't know the current UART settings or need to restore a module to a known state. In this mode, the UART always runs at 9600 Baud with no parity and no flow control, regardless of the stored settings.

What you'll need:

- Proteus-IV EV-Board connected to your PC via USB
- WE UART Terminal program [12] (or similar terminal software)
- Terminal configured to 9600 Baud, 8 data bits, no parity, 1 stop bit (8N1), no flow control
- Access to *MODE_0* and *MODE_1* pins on the EV-Board

Step 1: Enter config mode To boot into config mode, we need to set the mode pins correctly before powering on or resetting the module:

- Set *MODE_0* pin to HIGH (set mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to LOW (unset mode 1 pin jumper on JP1 on EV-Board)

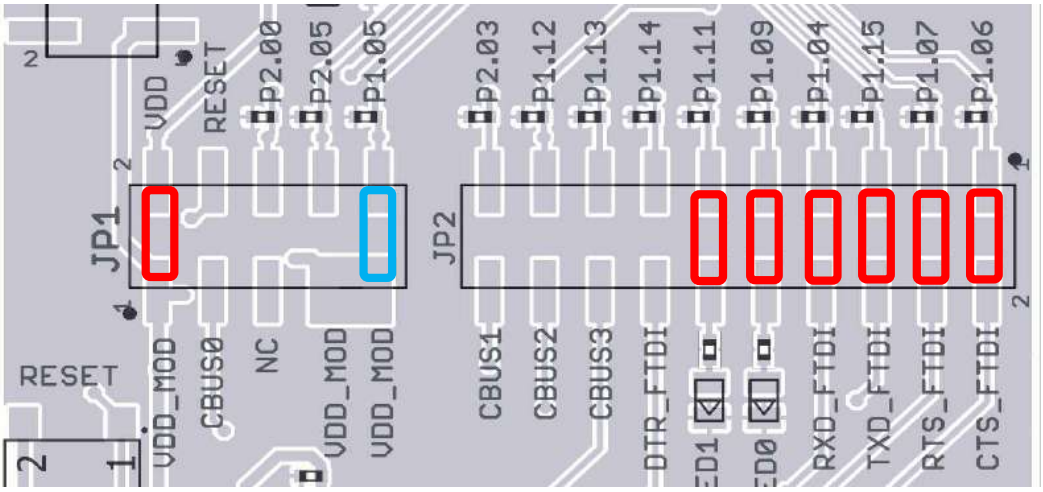


Figure 38: Proteus-IV EV-Board configured for config mode

Now apply power or perform a reset. The module will boot in config mode. Open your terminal at 9600 Baud and you should see:

Description	Command/Response
⇐ Module sends startup indication	0x02 0x81 0x01 0x00 **0x02** 0x80

The mode byte **0x02** in this `CMD_STARTUP_IND` confirms we're in config mode. In this mode, the radio is turned off and only configuration commands are available.

Step 2: Read the current TX power Let's check the module's transmit power setting. We use `CMD_GET_REQ` with settings index 17 (**0x11**), `RF_TXPower`:

Description	Command/Response
⇒ Request TX power	0x02 0x10 0x01 0x00 **0x11** 0x02
⇐ Module responds	0x02 0x50 0x02 0x00 0x00 **0x08** 0x58

The parameter byte **0x08** tells us the TX power is currently set to +8 dBm. TX power values are stored in two's complement notation.

Step 3: Change the TX power Let's reduce the TX power to +4 dBm to save energy. We use `CMD_SET_REQ`, with the parameter value **0x04**:

Description	Command/Response
⇒ Set TX power to +4 dBm	0x02 0x11 0x02 0x00 0x11 **0x04** 0x04
⇐ Module confirms	0x02 0x51 0x01 0x00 0x00 0x52
⇐ Module restarts	0x02 0x81 0x01 0x00 0x02 0x82

The module automatically restarts after changing settings. Notice it boots back into config mode (mode byte 0x02) because the mode pins are still held in the config mode position.

Step 4: Verify the new TX power setting Let's confirm the change was saved:

Description	Command/Response
⇒ Request TX power again	0x02 0x10 0x01 0x00 0x11 0x02
⇐ Module responds	0x02 0x50 0x02 0x00 0x00 **0x04** 0x54

Excellent! The parameter is now **0x04**, confirming the TX power is set to +4 dBm.

Step 5: Restart in command mode Now that configuration is complete, let's switch back to command mode for normal operation:

- Set both *MODE_0* and *MODE_1* pins to LOW (unset both mode pin jumpers on JP1 on EV-Board)
- Perform a reset

The module will now boot in command mode at whatever UART baud rate is configured (remember, config mode always uses 9600 Baud, but command mode uses the configured rate).

Description	Command/Response
⇐ Module sends startup indication	0x02 0x81 0x01 0x00 **0x01** 0x83

The mode byte **0x01** confirms we're now in command mode. Don't forget to adjust your terminal's baud rate to match the module's configured UART speed!

What we accomplished: Congratulations! You have successfully:

- Entered config mode using the mode pins
- Read a radio parameter (TX power) from RRAM
- Modified and saved the TX power setting
- Verified the change persisted after reset

- Switched back to command mode for normal operation

Config mode is particularly valuable when:

- You don't know the current UART baud rate
- You need to recover a module with unknown settings
- You want to configure the module before enabling radio operation
- You need a predictable UART configuration (always 9600 Baud)

Remember to keep the mode pins at the desired levels throughout the entire configuration process. The mode selection only happens during boot-up.

13.2. Connection setup and data exchange

13.2.1. Peripheral: Transparent mode

This tutorial shows you how to use the Proteus-IV in transparent mode, which provides the simplest way to transmit data wirelessly. In transparent mode, you simply send data to the module's UART, and it automatically transmits it via Bluetooth® LE to any connected device. No commands are needed - just pure data transfer!

Note that a Proteus-IV running in transparent mode can only act as peripheral. For central operation, command mode is required.

What you'll need:

- Proteus-IV EV-Board
- Smartphone or tablet with WE Bluetooth LE Terminal app [3, 4] installed
- WE UART Terminal program [12] (or similar terminal software) on your PC
- Terminal configured to 115200 Baud, 8N1, with flow control enabled

Understanding transparent mode: In transparent mode, the module acts as a wireless serial port. Data you send to the UART is automatically packaged and transmitted via Bluetooth® LE. When the module receives data wirelessly, it outputs it directly to the UART. The ETX (End of Transmission) characters trigger transmission-by default these are 0x0D 0x0A (carriage return + line feed, or "\r\n" in many programming languages).

Step 1: Enter transparent mode Set up the mode pins on your EV-Board:

- Set *MODE_0* pin to LOW (unset mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to HIGH (set mode 1 pin jumper on JP1 on EV-Board)

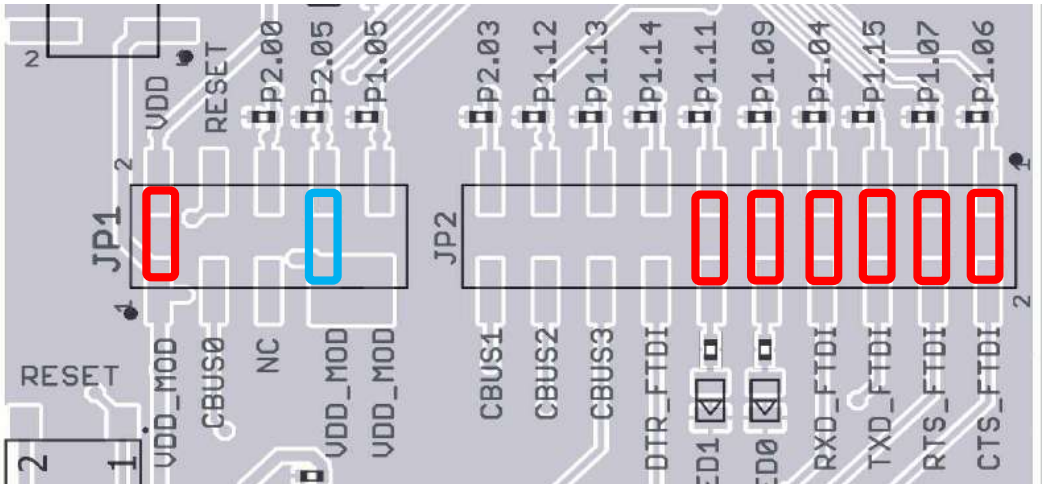


Figure 39: Proteus-IV EV-Board configured for transparent mode

Power on or reset the module. Open your terminal at 115200 Baud. You should see nothing on the terminal tool.
The module is now in transparent mode! Notice *LED_0* starts blinking (1 second on, 1 second off)-this indicates the module is advertising and ready for connections.

Step 2: Connect from your smartphone Open the WE Bluetooth LE Terminal app on your smartphone. You should see your module in the scan results. The default name is "Proteus-IV" unless you've changed it. Tap to connect.
Watch *LED_1* on your EV-Board-it will turn on solid once the connection is established. This indicates an active Bluetooth® LE link.

Step 3: Send data from PC to smartphone Now let's send some data wirelessly! In your PC terminal, type:

```
Hello from PC\r\n
```

Important: You must include the `\r\n` (carriage return and line feed) at the end. These are the ETX characters that trigger the module to transmit the data.
On your smartphone's WE Bluetooth LE Terminal app, you should immediately see:

```
Hello from PC
```

The data has been transmitted wirelessly! The transmitted \r\n will be shown as a new line in WE Bluetooth LE Terminal app. The ETX characters (`\r\n`) can be removed from the radio data, if the corresponding bit is set in the `UART_TransparentETXConfig` setting.

Step 4: Send data from smartphone to PC Now let's go the other direction. In the WE Bluetooth LE Terminal app, type a message and send it. For example:

```
Hello from phone
```

In your PC terminal, you should see:

```
Hello from phone
```

The module can automatically add the ETX characters (`\r\n`) to the received data before sending it to the UART, if the corresponding bit is set in the `UART_TransparentETXConfig` setting.

Understanding the data flow: Here's what happens internally:
PC to smartphone:

1. You type: `Hello from PC\r\n`
2. Module receives bytes via UART and buffers them
3. When ETX (`\r\n`) is detected, transmission is triggered
4. Module sends "Hello from PC" via Bluetooth® LE (ETX removed by default)
5. Smartphone receives and displays the message

Smartphone to PC:

1. Smartphone sends: `Hello from phone`
2. Module receives via Bluetooth® LE
3. Module adds ETX (`\r\n`) to the data
4. Module sends `Hello from phone\r\n` to UART
5. Your terminal displays the message

Tips for using transparent mode:

- **Always end messages with ETX:** Data is only transmitted when the ETX sequence (`\r\n`) is received
- **Watch the LEDs:** *LED_0* blinks during advertising, *LED_1* is solid during an active connection
- **Enable flow control for high data rates:** If sending large amounts of data rapidly, enable UART flow control in the settings to prevent data loss
- **Maximum packet size:** By default, you can send up to 244 bytes per packet
- **ETX configuration:** You can customize ETX behavior using `UART_TransparentETXConfig`, `UART_TransparentReceiveETX`, and `UART_TransparentTransmitETX`

What we accomplished: Congratulations! You have successfully:

- Booted the Proteus-IV in transparent mode
- Connected to the module from a smartphone
- Sent data wirelessly from PC to smartphone
- Received data wirelessly from smartphone to PC
- Understood how ETX characters trigger data transmission

Transparent mode is perfect for simple wireless serial communication. You don't need to worry about commands or protocols-just send your data with the appropriate ETX characters, and the module handles everything else automatically!

13.2.2. Peripheral: Command mode

This tutorial demonstrates using the Proteus-IV as a peripheral (server) in command mode. This gives you full control over the connection process, security settings, and data transmission. We'll connect to the module using the WE Bluetooth LE Terminal app and exchange data using JustWorks security mode.

What you'll need:

- Proteus-IV EV-Board
- Smartphone or tablet with WE Bluetooth LE Terminal app [3, 4] installed
- WE UART Terminal program [12] (or similar terminal software) on your PC
- Terminal configured to 115200 Baud, 8N1, with flow control enabled

Step 1: Enter command mode Set up the mode pins on your EV-Board:

- Set *MODE_0* pin to LOW (unset mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to LOW (unset mode 1 pin jumper on JP1 on EV-Board)

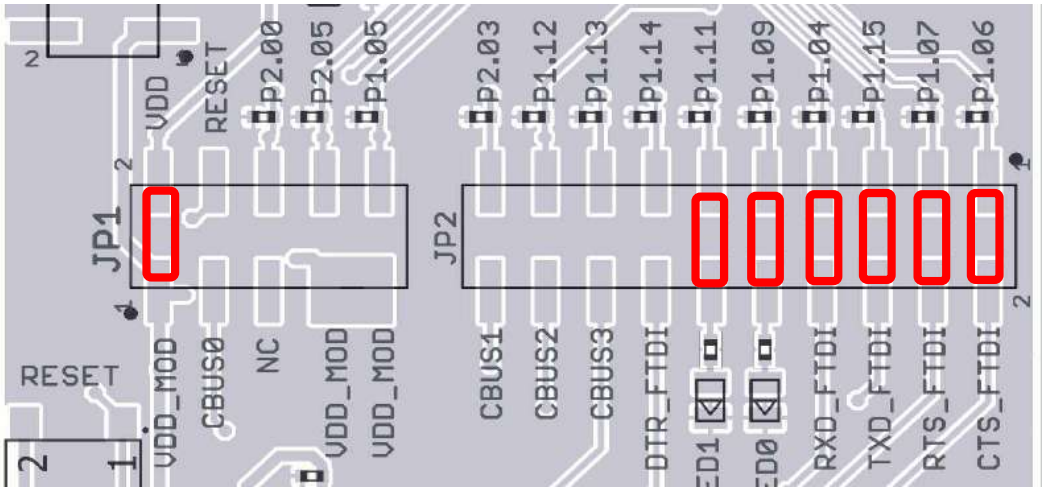


Figure 40: Proteus-IV EV-Board configured for command mode

Power on or reset the module. Open your terminal at 115200 Baud and you should see:

Description	Command/Response
⇐ Module sends startup indication	0x02 0x81 0x01 0x00 **0x01** 0x83

The mode byte **0x01** confirms command mode. Now let's start advertising so other devices can find us:

Description	Command/Response
⇒ Enable advertising	0x02 0x12 0x01 0x00 0x01 0x10
⇐ Module confirms	0x02 0x52 0x01 0x00 0x00 0x51

LED_0 should now start blinking (1 second on, 1 second off), indicating the module is advertising. Your module is now discoverable as "Proteus-IV" (or whatever name you've configured).

Step 2: Verify security settings Let's confirm we're using JustWorks security mode (no passkey required). Check the `RF_SecFlags` setting:

Description	Command/Response
⇒ Request security flags	0x02 0x10 0x01 0x00 0x0C 0x1F
⇐ Module responds	0x02 0x50 0x02 0x00 0x00 **0x01** 0x51

The value **0x01** indicates "No IO capabilities, minimum level L2"-this is the JustWorks mode where connections are established automatically without user interaction.

Step 3: Connect from your smartphone Open the WE Bluetooth LE Terminal app on your smartphone and scan for devices. You should see "Proteus-IV" in the list. Tap to connect.
On your PC terminal, you'll see a series of messages as the connection is established:

Description	Command/Response
⇐ Connection established	0x02 0x86 0x09 0x00 **0x00** 0x00 0x00 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX CS
⇐ Security established	0x02 0x88 0x03 0x00 **0x00** 0x00 0x02 0x8B
⇐ Link opened	0x02 0xC6 0x02 0x00 **0x00** 0x01 0xC7

Let's break down what happened:

- `CMD_CONNECT_IND`: Physical connection established with Conn_ID **0x00**, status 0x00 (success). The six bytes before the checksum are the smartphone's Bluetooth® MAC address.
- `CMD_SECURITY_IND`: Security negotiation completed. Status 0x00 (success), security level 0x02 (L2 - encrypted but no authentication).
- `CMD_LINKOPEN_RSP`: The Bluetooth® LE link is now fully open. Status 0x01 means link opened-you can now exchange data!

Watch *LED_1* on your EV-Board - it should now be solid ON, indicating an active connection.

Step 4: Send data from module to smartphone Let's send a message from the Proteus-IV to your smartphone. We'll send "Hello Phone!" (0x48 0x65 0x6C 0x6C 0x6F 0x20 0x50 0x68 0x6F 0x6E 0x65 0x21):

Description	Command/Response
⇒ Send data to Conn_ID 0x00	0x02 0x04 0x0D 0x00 0x00 0x48 0x65 0x6C 0x6C 0x6F 0x20 0x50 0x68 0x6F 0x6E 0x65 0x21 0x4A
⇐ Request accepted and data transmitted	0x02 0x44 0x01 0x00 **0x00** 0x47

On your smartphone, the WE Bluetooth LE Terminal app should display:

```
Hello Phone!
```

The responses tell us:

- `CMD_DATA_CNF`: Status **0x00** means the request was accepted and the module has sent the data

Step 5: Receive data from smartphone Now send a message from your smartphone to the module. In the WE Bluetooth LE Terminal app, type and send "Hello Module!"
On your PC terminal, you'll receive:

Description	Command/Response
⇐ Data received	0x02 0x84 0x0F 0x00 0x00 0xXX 0x48 0x65 0x6C 0x6C 0x6F 0x20 0x4D 0x6F 0x64 0x75 0x6C 0x65 0x21 CS

This `CMD_DATA_IND` message contains:

- Conn_ID: 0x00 (the connection that sent the data)
- RSSI: 0xXX (signal strength in dBm, two's complement)
- Payload: 0x48 0x65... = "Hello Module!" in ASCII

Step 6: Disconnect When you're finished, disconnect from your smartphone app. You'll see:

Description	Command/Response
⇐ Disconnected	0x02 0x87 0x02 0x00 0x00 **0x16** 0x94

The `CMD_DISCONNECT_IND` shows Conn_ID 0x00 and reason **0x16** (host terminated connection). *LED_1* turns off, and *LED_0* starts blinking again as the module returns to advertising mode.

Understanding command mode benefits: Command mode gives you complete control:

- **Connection management:** You know exactly when connections are established or dropped
- **Connection tracking:** The Conn_ID lets you manage multiple simultaneous connections
- **Confirmation feedback:** Every action gets a confirmation response
- **RSSI information:** You can monitor signal strength with each received packet
- **Security control:** You can implement various security modes and handle passkey requests
- **Flexible data transmission:** Send data to specific connections or broadcast to all connected devices

What we accomplished: Congratulations! You have successfully:

- Started the module in command mode as a peripheral
- Enabled advertising to make the module discoverable
- Established a secure connection using JustWorks security
- Sent data from module to smartphone with confirmation
- Received data from smartphone with RSSI information

- Observed the connection lifecycle through LED indicators
- Handled disconnection properly

Command mode is ideal when you need full control and visibility over Bluetooth® LE operations. It's perfect for applications requiring connection management, multiple simultaneous connections, or integration with your own protocol layer.

13.2.3. Central: Command mode

This tutorial shows how to use the Proteus-IV as a central (client) device in command mode. The central role is the active partner that initiates connections to peripheral devices. We'll scan for nearby Bluetooth® LE devices, connect to another Proteus-IV module, and exchange data.

What you'll need:

- Two Proteus-IV EV-Boards (we'll call them "Module A" and "Module B")
- Two WE UART Terminal instances [12] on your PC
- Both terminals configured to 115200 Baud, 8N1, with flow control enabled

Setup overview:

- **Module A** will act as the central (scanner/initiator)
- **Module B** will act as the peripheral (advertiser/responder)

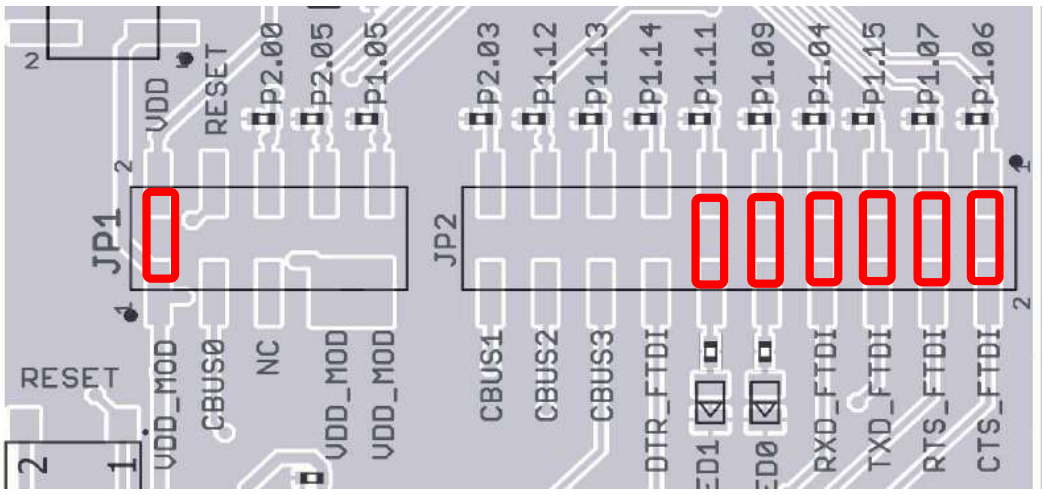


Figure 41: Proteus-IV EV-Board configured for command mode

Step 1: Prepare Module B (Peripheral) Set up the mode pins on your EV-Board to enter command mode:

- Set *MODE_0* pin to LOW (unset mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to LOW (unset mode 1 pin jumper on JP1 on EV-Board)

Power on or reset the module. Open your terminal at 115200 Baud and you should see:

Description	Command/Response
⇐ Module B sends startup	0x02 0x81 0x01 0x00 0x01 0x83

Start advertising on Module B:

Description	Command/Response
⇒ Enable advertising	0x02 0x12 0x01 0x00 0x01 0x10
⇐ Module B confirms	0x02 0x52 0x01 0x00 0x00 0x51

Module B is now advertising. Its *LED_0* should be blinking. Note down or remember Module B's Bluetooth® MAC address-we'll need it shortly. If you don't know it, you can read it:

Description	Command/Response
⇒ Request BTMAC	0x02 0x10 0x01 0x00 0x04 0x17
⇐ Module B responds	0x02 0x50 0x07 0x00 0x00 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX CS

The six bytes (0xXX...) are the Bluetooth® MAC address in LSB-first order. For this example, let's assume Module B's address is **0x0018DAFFFFFF**.

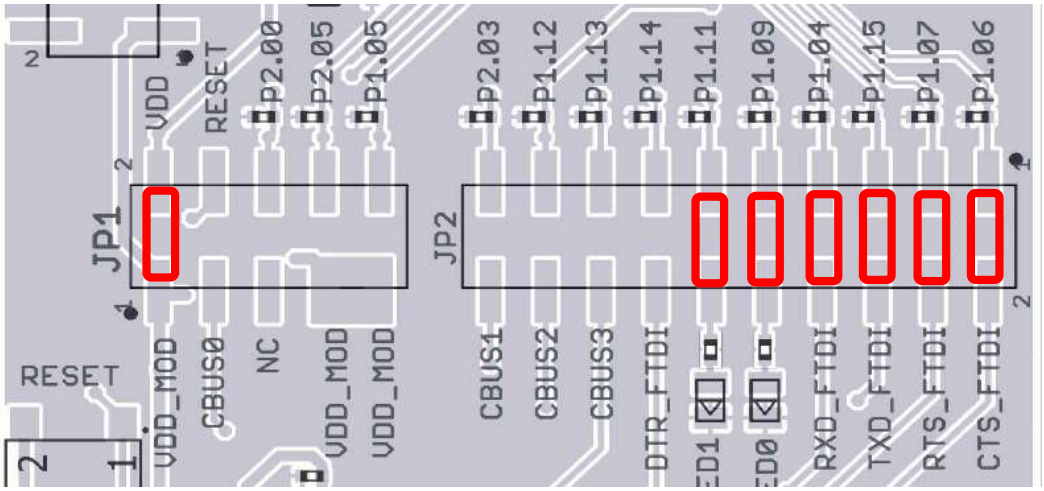


Figure 42: Proteus-IV EV-Board configured for command mode

Step 2: Prepare Module A (Central) Set up the mode pins on your EV-Board to enter command mode:

- Set *MODE_0* pin to LOW (unset mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to LOW (unset mode 1 pin jumper on JP1 on EV-Board)

Power on or reset the module. Open your terminal at 115200 Baud and you should see:

Description	Command/Response
⇐ Module A sends startup	0x02 0x81 0x01 0x00 0x01 0x83

Module A is ready in command mode.

Step 3: Scan for devices with Module A Let's scan for nearby Bluetooth® LE devices. Start scanning:

Description	Command/Response
⇒ Start scan	0x02 0x09 0x01 0x00 0x01 0x0B
⇐ Module A confirms	0x02 0x49 0x01 0x00 0x00 0x4A

The scan is now active. Module A will send a `CMD_SCAN_IND` message for each device it discovers:

Description	Command/Response
⇐ Device found	0x02 0x89 0x13 0x00 0x00 **0xFF 0xFF 0xFF 0xDA 0x18 0x00** 0xC5 0x05 0x50 0x72 0x6F 0x74 0x65 0x75 0x73 0x2D 0x49 0x56 0x0D

This `CMD_SCAN_IND` tells us:

- Address type: 0x00 (public)
- BTMAC: **0xFF 0xFF 0xFF 0xDA 0x18 0x00** (LSB first = 0x0018DAFFFFFF)
- RSSI: 0xC5 (-59 dBm in two's complement)
- Device name length: 0x0A (10 bytes)
- Device name: 0x50 0x72 0x6F 0x74 0x65 0x75 0x73 0x2D 0x49 0x56 = "Proteus-IV"

Perfect! We found Module B. You may see other devices as well if there are more Bluetooth® LE devices nearby.
After a few seconds, stop scanning:

Description	Command/Response
⇒ Stop scan	0x02 0x09 0x01 0x00 0x00 0x0A
⇐ Module A confirms	0x02 0x49 0x01 0x00 0x00 0x4A

Step 4: Connect to Module B Now let's connect to Module B using its MAC address (0x0018DAFFFFFF). Remember to use LSB-first notation:

Description	Command/Response
⇒ Connect request	0x02 0x06 0x07 0x00 0x00 **0xFF 0xFF 0xFF 0xDA 0x18 0x00** 0xB9
⇐ Request accepted	0x02 0x46 0x01 0x00 0x00 0x45

The `CMD_CONNECT_CNF` status 0x00 means Module A accepted the request and is now attempting to connect. Watch both modules - you'll see a sequence of messages as the connection is established:
On Module A (Central):

Description	Command/Response
⇐ Connection established	0x02 0x86 0x09 0x00 0x00 0x00 0x00 **0xFF 0xFF 0xFF 0xDA 0x18 0x00** 0x7E
⇐ Security established	0x02 0x88 0x03 0x00 0x00 0x00 0x02 0x8B
⇐ Link opened	0x02 0xC6 0x02 0x00 0x00 0x01 0xC7

On Module B (Peripheral):

Description	Command/Response
⇐ Connection established	0x02 0x86 0x09 0x00 0x00 0x00 0x00 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX CS
⇐ Security established	0x02 0x88 0x03 0x00 0x00 0x00 0x02 0x8B
⇐ Link opened	0x02 0xC6 0x02 0x00 0x00 0x01 0xC7

Both modules show *LED_1* solid ON, indicating an active connection. The connection is using Conn_ID 0x00 on both sides.

Step 5: Exchange data Now the fun part - let's send data in both directions!
From Module A to Module B: Send "Central says hi!" from Module A:

Description	Command/Response
⇒ Send from Module A	0x02 0x04 0x10 0x00 0x00 0x43 0x65 0x6E 0x74 0x72 0x61 0x6C 0x20 0x73 0x61 0x79 0x73 0x20 0x68 0x69 0x21 0x6C
⇐ Request accepted and data transmitted	0x02 0x44 0x01 0x00 0x00 0x47

On Module B's terminal, you'll see the received data:

Description	Command/Response
⇐ Data received	0x02 0x84 0x12 0x00 0x00 0xXX 0x43 0x65 0x6E 0x74 0x72 0x61 0x6C 0x20 0x73 0x61 0x79 0x73 0x20 0x68 0x69 0x21 CS

From Module B to Module A: Send "Peripheral replies!" from Module B:

Description	Command/Response
⇒ Send from Module B	0x02 0x04 0x13 0x00 0x00 0x50 0x65 0x72 0x69 0x70 0x68 0x65 0x72 0x61 0x6C 0x20 0x72 0x65 0x70 0x6C 0x69 0x65 0x73 0x21 0x17
⇐ Request accepted and data transmitted	0x02 0x44 0x01 0x00 0x00 0x47

On Module A's terminal, you'll receive:

Description	Command/Response
⇐ Data received	0x02 0x84 0x15 0x00 0x00 0xXX 0x50 0x65 0x72 0x69 0x70 0x68 0x65 0x72 0x61 0x6C 0x20 0x72 0x65 0x70 0x6C 0x69 0x65 0x73 0x21 CS

Data flows bidirectionally between the modules!

Step 6: Disconnect When finished, disconnect from Module A (the central initiates the disconnection):

Description	Command/Response
⇒ Disconnect	0x02 0x07 0x01 0x00 0x00 0x06
⇐ Request accepted	0x02 0x47 0x01 0x00 0x00 0x44
⇐ Disconnected	0x02 0x87 0x02 0x00 0x00 0x13 0x94

Module B will also receive a disconnect indication:

Description	Command/Response
⇐ Disconnected	0x02 0x87 0x02 0x00 0x00 0x13 0x94

Both modules' *LED_1* turns off. Module B's *LED_0* starts blinking again as it resumes advertising.

What we accomplished: Congratulations! You have successfully:

- Set up one module as central and another as peripheral
- Scanned for nearby Bluetooth® LE devices
- Identified a specific device by its MAC address and name
- Initiated a connection from the central to the peripheral

- Exchanged data bidirectionally between both modules
- Properly disconnected and cleaned up the connection

The central role is essential for:

- **Active device discovery:** Scanning and selecting which devices to connect to
- **Connection initiation:** The central decides when and where to connect
- **Network topologies:** A central can connect to multiple peripherals
- **Data collection:** Ideal for gathering data from multiple sensor nodes
- **Gateway applications:** Acting as a hub that connects to multiple devices

13.3. Others

13.3.1. Low power mode - Sleep

This tutorial demonstrates how to use sleep mode for ultra-low power consumption. Sleep mode is perfect for battery-powered applications where the device spends most of its time idle and only wakes up periodically or when triggered by an external event.

What you'll need:

- Proteus-IV EV-Board
- WE UART Terminal program [12]
- Terminal configured to 115200 Baud, 8N1, with flow control enabled
- Optional: Multimeter to measure current consumption

Understanding sleep mode: In sleep (system-off) mode, the Proteus-IV shuts down almost completely:

- UART interface is disabled
- Bluetooth® radio is turned off
- Most internal circuitry is powered down
- Current consumption drops to microampere levels
- All volatile settings and RAM contents are lost

The module can only be woken from sleep mode by a hardware reset on the */RESET* pin.

Step 1: Boot the module and verify it's ready Set up the mode pins on your EV-Board:

- Set *MODE_0* pin to LOW (unset mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to LOW (unset mode 1 pin jumper on JP1 on EV-Board)

Power on or reset the module. Open your terminal at 115200 Baud and you should see:

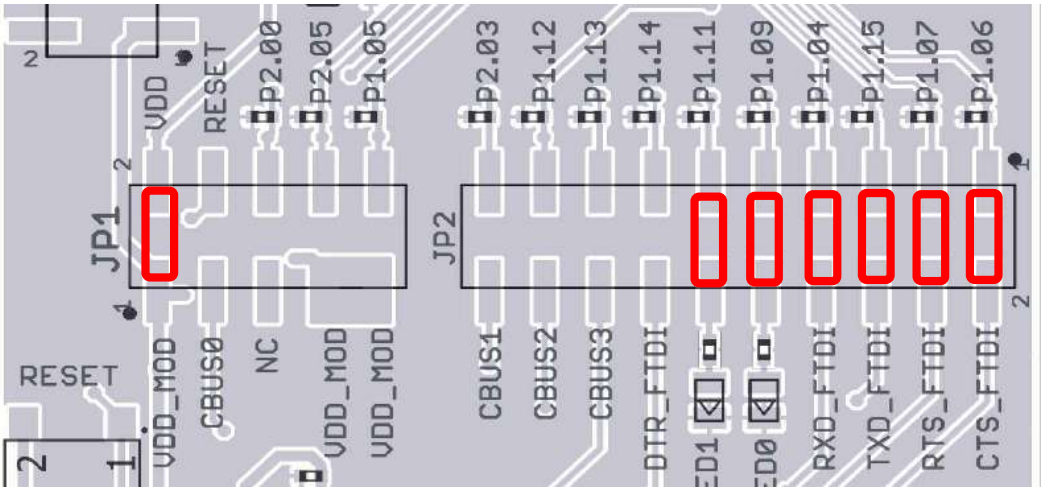


Figure 43: Proteus-IV EV-Board configured for command mode

Description	Command/Response
⇐ Module sends startup	0x02 0x81 0x01 0x00 0x01 0x83

The module is in command mode and ready for operation.

Step 2: Check the module state Before entering sleep mode, let's verify the current state:

Description	Command/Response
⇒ Get state	0x02 0x01 0x00 0x00 0x03
⇐ Module responds	0x02 0x41 0x03 0x00 0x00 0x01 0x01 0x40

The `CMD_GETSTATE_CNF` shows:

- Status: 0x00 (success)
- Module mode: 0x01 (command)
- More info: 0x01 (advertising, no scanning, not connected)

Step 3: Ensure no active connections Sleep mode can only be entered when no Bluetooth® connections are active. If you had any connections, disconnect them first. Since we have no connections, we can proceed directly to sleep.

Step 4: Enter sleep mode Send the sleep command:

Description	Command/Response
⇒ Sleep request	0x02 0x02 0x00 0x00 0x00
⇐ Request accepted	0x02 0x42 0x01 0x00 **0x00** 0x41

The `CMD_SLEEP_CNF` status **0x00** confirms the module will now enter sleep mode. Within a few milliseconds:

- All LEDs turn off
- UART communication stops
- Current consumption drops dramatically
- The module is now in deep sleep

If you're measuring current, you should see consumption drop to just a few microamperes.

Step 5: Wake up the module The module will remain in sleep mode indefinitely until you wake it. To wake the module, perform a pin reset:

1. Apply a LOW signal to the */RESET* pin
2. Hold LOW for at least 10 milliseconds
3. Release back to HIGH

The module will now boot up completely:

Description	Command/Response
⇐ Module sends startup	0x02 0x81 0x01 0x00 0x01 0x83

The module is awake and ready for operation again. Note that all volatile settings have been reset to their defaults - the module has performed a complete reboot.

Step 6: Verify the module is operational Let's confirm the module is working normally by checking its state:

Description	Command/Response
⇒ Get state	0x02 0x01 0x00 0x00 0x03
⇐ Module responds	0x02 0x41 0x03 0x00 0x00 0x01 0x01 0x40

Perfect! The module is fully operational again.

Typical use cases for sleep mode:

- **Sensor nodes:** Wake up periodically to read sensors and transmit data, then sleep
- **Event-driven devices:** Sleep until an external interrupt (like a button press) wakes the module
- **Battery-powered applications:** Maximize battery life by sleeping when not actively communicating

Important considerations:

- **No connections allowed:** You must disconnect all Bluetooth® connections before entering sleep mode
- **Hardware wake-up only:** The module can only be woken by the */RESET* pin
- **Complete reset:** All volatile settings and RAM contents are lost during sleep
- **RRAM settings persist:** User settings stored in RRAM (like baud rate, TX power) are retained
- **External circuitry needed:** You need external hardware (microcontroller, button) to control the wake-up

What we accomplished: Congratulations! You have successfully:

- Verified the module state before entering sleep mode
- Sent the sleep command and entered ultra-low power mode
- Observed the dramatic reduction in power consumption
- Woken the module using a hardware reset
- Confirmed the module operates normally after wake-up
- Understood when and how to use sleep mode effectively

Sleep mode is one of the most powerful features for battery-operated IoT devices. Combined with efficient advertising intervals and connection parameters, you can create devices that run for years on a single battery!

13.3.2. Firmware (over the air) update

Please refer to chapter `Firmware updates` for detailed information about firmware updates via the FOTA (Firmware Over-The-Air) mode.
The FOTA mode enables wireless firmware updates using the Simple Management Protocol (SMP) Bluetooth® LE service. To enter FOTA mode set the mode pins as follows and reset the radio module:

- Set *MODE_0* pin to HIGH (set mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to HIGH (set mode 1 pin jumper on JP1 on EV-Board)

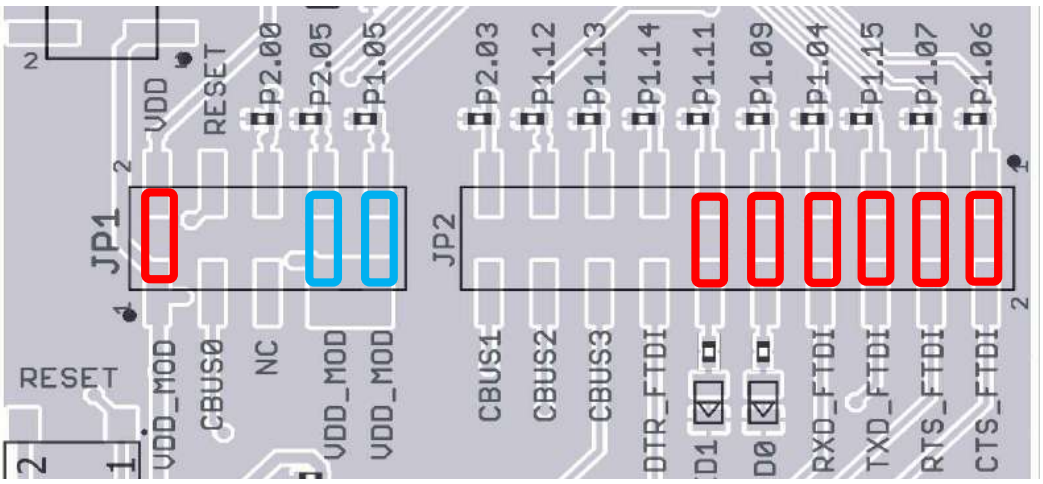


Figure 44: Proteus-IV EV-Board configured for FOTA mode

Once in FOTA mode, you can use compatible tools like nRF Connect Device Manager to upload new firmware wirelessly.

13.3.3. Beaconing - Raw advertising data

This tutorial shows how to transmit custom advertising data, such as iBeacon or Eddystone beacon formats. By customizing the advertising packet, you can broadcast information without requiring connections-perfect for proximity detection, navigation, and information broadcasting.

What you'll need:

- Proteus-IV EV-Board
- WE UART Terminal program [12]
- Terminal configured to 115200 Baud, 8N1
- Smartphone with beacon scanner app (like nRF Connect or Beacon Scanner)

Understanding beacon mode: Beacons broadcast data in advertising packets without establishing connections. By using `CMD_SETRAM_REQ` to modify `RF_AdvertisingData`, you can transmit custom beacon formats. Note that once you set custom advertising data, the standard content (device name, UUIDs) is replaced entirely.

Step 1: Enter command mode Set up the mode pins on your EV-Board:

- Set *MODE_0* pin to LOW (unset mode 0 pin jumper on JP1 on EV-Board)
- Set *MODE_1* pin to LOW (unset mode 1 pin jumper on JP1 on EV-Board)

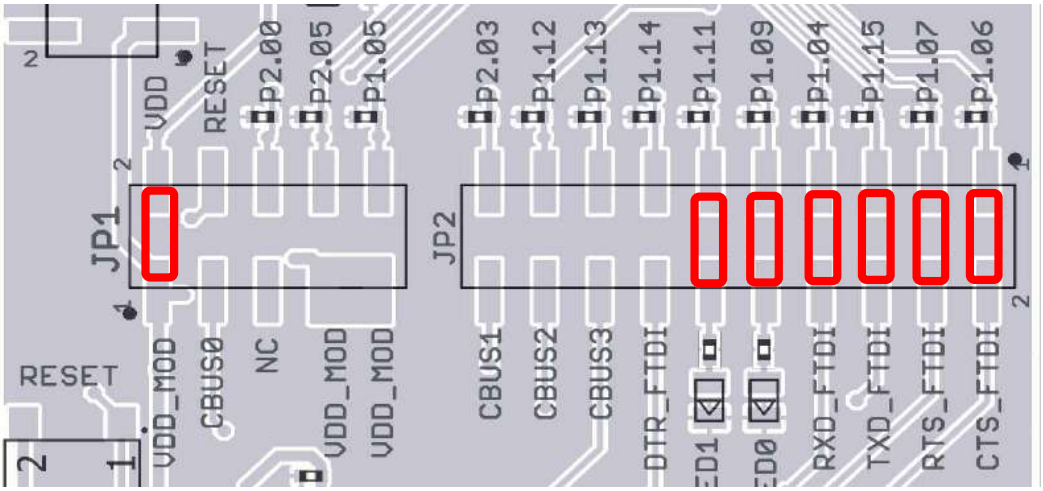


Figure 45: Proteus-IV EV-Board configured for command mode

Power on or reset the module. Open your terminal at 115200 Baud and you should see:

Description	Command/Response
⇐ Module sends startup	0x02 0x81 0x01 0x00 0x01 0x83

Step 2: Apply useful module configurations (optional) In case of pure beaconing applications, it may be useful apply special settings:

- Block peer devices from connecting to the radio module. To enable unconnectable advertising mode, bit 4 has to be set in the user settings `CFG_Flags`. To do so, read back the `CFG_Flags`, set bit 4 and write the modified value to the radio module:

Description	Command/Response
⇒ Read `CFG_Flags`	0x02 0x10 0x01 0x00 0x05 0x16
⇐ Read back the value **0x000B**	0x02 0x50 0x03 0x00 0x00 **0x0B 0x00** 0x5A

  Enabling bit 4 means to set the `CFG_Flags` to 0x001B:

Description	Command/Response
⇒ Write **0x001B** to CFG_Flags	0x02 0x11 0x03 0x00 0x05 **0x1B 0x00** 0x0E
⇐ Module confirms	0x02 0x51 0x01 0x00 0x00 0x52
⇐ Module sends startup	0x02 0x81 0x01 0x00 0x01 0x83

- Slow down the advertising interval (RF_AdvertisingInterval) to save power. Here we choose the advertising interval of 1000-1000 ms (1600/0x0640 - 1600/0x0640):

Description	Command/Response
⇒ Write to the RF_AdvertisingInterval	0x02 0x11 0x05 0x00 0x07 0x40 0x06 0x40 0x06 0x11
⇐ Module confirms	0x02 0x51 0x01 0x00 0x00 0x52
⇐ Module sends startup	0x02 0x81 0x01 0x00 0x01 0x83

Step 3: Create iBeacon advertising data We'll create an iBeacon packet. The iBeacon format consists of:

- Flags: 0x02 0x01 0x06
- Manufacturer data: 0x1A 0xFF 0x4C 0x00 0x02 0x15
- Proximity UUID: 16 bytes (we'll use: E2C56DB5-DFFB-48D2-B060-D0F5A71096E0)
- Major: 0x00 0x01
- Minor: 0x00 0x02
- TX Power: 0xC5 (-59 dBm)

Use CMD_SETRAM_REQ with settings index 13 to set this data temporarily in RAM:

Description	Command/Response
⇒ Set iBeacon data	0x02 0x21 0x1F 0x00 0x0D **0x02 0x01 0x06 0x1A 0xFF 0x4C 0x00 0x02 0x15 0xE2 0xC5 0x6D 0xB5 0xDF 0xFB 0x48 0xD2 0xB0 0x60 0xD0 0xF5 0xA7 0x10 0x96 0xE0 0x00 0x01 0x00 0x02 0xC5** 0x39
⇐ Setting updated	0x02 0x61 0x01 0x00 0x00 0x62

The status 0x00 confirms the advertising data has been set in RAM. The module is now broadcasting as an iBeacon!
If CMD_SET_REQ is used instead of CMD_SETRAM_REQ, the data will be stored in RRAM and be reloaded after each boot-up.

In case of periodically new beacon data, use CMD_SETRAM_REQ instead of CMD_SET_REQ as RRAM can be written only 10000 times.

Step 4: Disable UART (optional) In case of pure beaconing applications, it may be useful to switch off the radio module's UART to save power:

Description	Command/Response
⇒ Send CMD_UARTDISABLE_REQ	0x02 0x1B 0x00 0x00 0x19
⇐ Module confirms	0x02 0x5B 0x 01 0x00 0x00 0x58

Alternatively, the module can be rebooted in transparent mode. In transparent mode, the UART is switched off after reboot. To use this simple option, make sure, that the custom advertising data has been stored in RRAM (by CMD_SET_REQ) and that the "Autostart advertising" bit is set in the CFG_Flags (enabled by default).

Step 5: Scan for the beacon Open a beacon scanner app on your smartphone (like nRF Connect). You should see your iBeacon with:

- Type: iBeacon
- UUID: E2C56DB5-DFFB-48D2-B060-D0F5A71096E0
- Major: 1
- Minor: 2
- TX Power: -59 dBm

The module will NOT appear with its device name "Proteus-IV" anymore-the custom advertising data has completely replaced the standard content.

Important considerations:

- **Temporary vs permanent:** Use CMD_SETRAM_REQ for temporary changes (lost on reset) or CMD_SET_REQ for permanent changes (saved to RRAM)
- **Format compliance:** Ensure your beacon data follows Bluetooth® specification format rules (see chapter 7.4.5)
- **No device name:** With custom advertising data, the module won't show its device name in standard scans

- **Cannot connect:** When advertising as a pure beacon without service UUIDs, some devices may not be able to connect
- **Scan response:** You can also customize the scan response packet using `RF_ScanResponseData`

Other beacon formats: You can implement other beacon formats like:

- **Eddystone UID:** For broadcasting unique identifiers
- **Eddystone URL:** For broadcasting URLs
- **Eddystone TLM:** For broadcasting telemetry data
- **AltBeacon:** Open-source beacon standard
- **Custom formats:** Design your own beacon protocol

Refer to the respective beacon specifications for exact data format requirements.

What we accomplished: Congratulations! You have successfully:

- Modified advertising data to create an iBeacon
- Broadcast beacon data without requiring connections
- Detected the beacon using a smartphone app
- Understood the difference between volatile RAM and non-volatile RRAM settings
- Restored standard advertising mode

Beacon mode is ideal for:

- Indoor navigation and wayfinding
- Proximity marketing and notifications
- Asset tracking and inventory management
- Contactless information broadcasting
- Museum exhibits and interactive displays

The beacon approach maximizes battery life since no connections are required, and you can broadcast to unlimited receivers simultaneously!

14. Custom firmware and configuration

Any kind of configuration and firmware, which is provided as Intel hex file, can be programmed on the radio module at Würth Elektronik eiSos production site.

In case of interest, please contact your Business Development Manager (BDM) or *WCS@we-online.com*.

14.1. Custom configuration of standard firmware

The configuration of the standard firmware includes adoption of the non-volatile settings to customer requirements and creating a customized product based on 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.

14.2. Customer specific firmware

A customer specific firmware may include "Custom configuration of standard firmware" plus additional options or functions and tasks that are customer specific and not part of the standard firmware.
Further scheduled firmware updates of the standard firmware will not be applied to this variant automatically. Applying updates or further functions require a customer request and release procedure.
This also results in a customer exclusive module with a unique ordering number.
An example for this level of customization are functions like host-less operation where the module will perform data generation (e.g. by reading a SPI or I^2C sensor) and cyclic transmission of this data to a data collector, while sleeping or being passive most of the time.
Also replacing UART with SPI as host communication interface is classified as 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.

14.3. 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 an Intel hex file) will be implemented into the module's production process at our production site.

This also results in a customer exclusive module with a unique ordering number.
The additional information needed for this type of customer firmware, such as hardware specific details and details towards the development of such firmware are not available for the public and can only be made available to qualified customers.

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

15. Firmware updates

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

15.1. Firmware flashing using the production interface

Most Würth Elektronik eiSos wireless connectivity modules offer a production interface (e.g. JTAG, SWD, Spy-Bi-Wire) for module flash access. Depending on the product, this interface can be used by customers to erase the entire chip and install their own firmware.

Using the production interface is not intended to perform updates of Würth Elektronik eiSos standard product firmware[1].

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

Any certification, declaration, listing and qualification becomes invalid if the production interface is used by a customer. Some products, in their documentation, state exceptions to this invalidation under certain conditions.

Customers shall make the product specific firmware update interface available to their application. These methods will use a wired (UART, SPI, etc.) or wireless (Bluetooth® LE, WiFi, etc.) communication interface of the module to allow updating the product's firmware. Details are described in the next sections.

15.2. Firmware update using the Proteus-IV OTA bootloader

This method offers a possibility to update the standard product firmware[1] over the air (OTA). Any other firmware (i.e. custom firmware, Bluetooth® test firmware (DTM), ...) must be flashed via the production interface (see chapter 15.1).

For the OTA firmware update, the MCUboot bootloader is integrated into the Proteus-IV's firmware, which will communicate over the Bluetooth® LE interface using MCUmgr's Simple Management Protocol (SMP).
For this reason, a .zip-file can be provided, which contains the signed firmware package in an encrypted and authenticated format.

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

[1] The standard product firmware is the firmware described in this user manual.

Version of the firmware before the update	Version of the new firmware	App
1.x.x	1.x.x	nRF Connect Device Manager [13, 14]

Table 40: Compatibility matrix

By default, the SMP service is deactivated to prevent unauthorized firmware updates. To enable FOTA functionality, one of the following two conditions has to be satisfied:

1. In command mode, send the command `CMD_BOOTLOADER_REQ` to the module to enable SMP service for FOTA.
2. During a reset and while restarting, a high signal has to be present on the *MODE* pins of the module to enable SMP service for FOTA.

The FOTA mode has been enabled successfully if the SMP service is available with the UUID 8D53DC1D-1DB7-4CD3-868B-8A527460AA84. After the SMP service has been enabled, the module continues normal operation while making the firmware update service available via Bluetooth® LE.
Now, any Bluetooth® LE device hosting an application that understands the commands of the MCUmgr SMP protocol can connect in order to update the Proteus-IV firmware. The nRF Connect Device Manager application [13, 14] is such an application. For more details, please refer to chapter `15.2.1`.

The implemented MCUboot 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 on how much of the firmware shall be updated (application only or complete update).

Only signed firmware packages by Würth Elektronik eiSos will be accepted by the Proteus-IV radio module. The signature ensures the zip package cannot be used for other products and provides authentication.

15.2.1. Firmware update steps using the nRF Connect Device Manager app

If the radio module Proteus-IV has been set to FOTA mode (SMP service enabled), the nRF Connect Device Manager app [13, 14] can be used to perform the OTA firmware update.

- Install the nRF Connect Device Manager app on your mobile device (Android or iOS).
- When opening the Device Manager app, it will not find any devices by default. This is because the app by default filters for an SMP service UUID. Disable the filter by unchecking the checkbox to see all available devices.
- Select your Proteus-IV device from the list to connect to it.

- Observe the features listed in the menu bar at the bottom. These are SMP services. Select the "Image" tab for firmware update functionality.
- Transfer the .zip archive file containing the firmware to your mobile device using your preferred method.
- In the Image tab, press "Select file" and choose the .zip archive file that contains the FOTA image.

If the SMP service is not available, please check whether the module has been set to FOTA mode by using one of the methods described above.

- Press "Start" to begin the update process. You can choose between the following update modes:
  - **Test Only**: The firmware is uploaded and temporarily activated on the device. This allows you to test the new firmware functionality without permanently replacing the old firmware. If the device is reset or power-cycled, it will automatically revert to the previous firmware version. *Use case*: Safe testing of new firmware before permanent installation.
  - **Confirm Only (Recommended for most users)**: The firmware is uploaded, activated, and permanently applied in one step. Once the update completes successfully, the new firmware becomes the permanent firmware even after device resets. *Use case*: Direct permanent update when you're confident the new firmware will work correctly.

This mode does not provide automatic recovery if the new firmware fails to boot properly. If a problem occurs, manual re-flashing may be required.

  - **Test and Confirm (Advanced users only)**: This mode combines both testing and confirmation steps, providing the highest level of safety with automatic error recovery. The firmware is uploaded, tested, and then automatically confirmed if it boots successfully. *Special requirement*: For this mode to work properly, the device must remain in OTA (bootloader) mode even after reset. This can only be achieved by keeping the *MODE* pins held high during the entire update process, or using the app to set the device in persistent bootloader mode via the hardware pins. *Use case*: Maximum safety for critical applications where firmware rollback capability is essential.
  - **Upload Only**: Not supported - This mode is not available for standard firmware updates.

  For a safe and recommended workflow, use the following sequence:

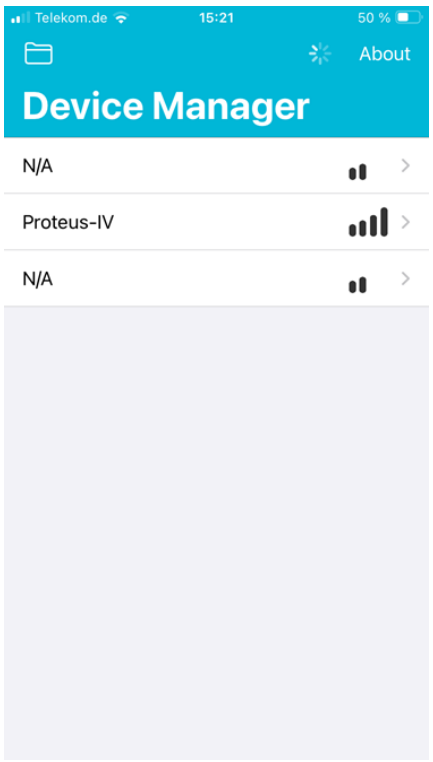
  1. **First update**: Use *Test Only* mode to verify that the new firmware functions correctly.
  2. **Second update**: Use *Confirm Only* mode to permanently apply the new firmware.

This two-step approach provides safety testing while remaining simple to execute, without requiring special hardware pin configurations.

Mode Selection Guidelines:

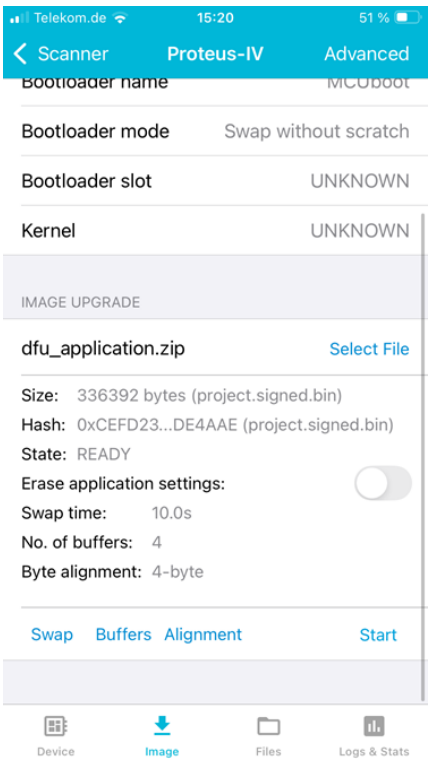
- **New to firmware updates**: Use the recommended two-step workflow (Test Only → Confirm Only)
- **Experienced users with critical applications**: Consider Test and Confirm mode if you can manage the hardware pin requirements
- **Production environments**: Confirm Only mode for direct updates when firmware has been pre-validated
- **Development/testing**: Test Only mode for iterative firmware development

- Wait until the transfer process has finished. The transfer speed depends on the mobile device and its operating system.
- When the process is completed, the device will restart with the new firmware.

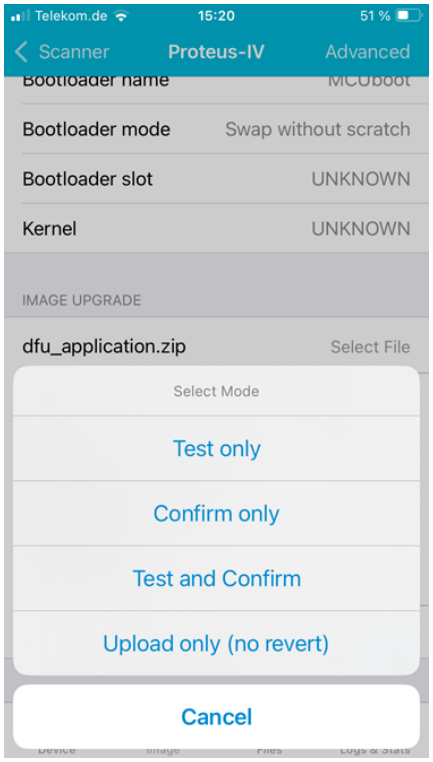


a) Device list

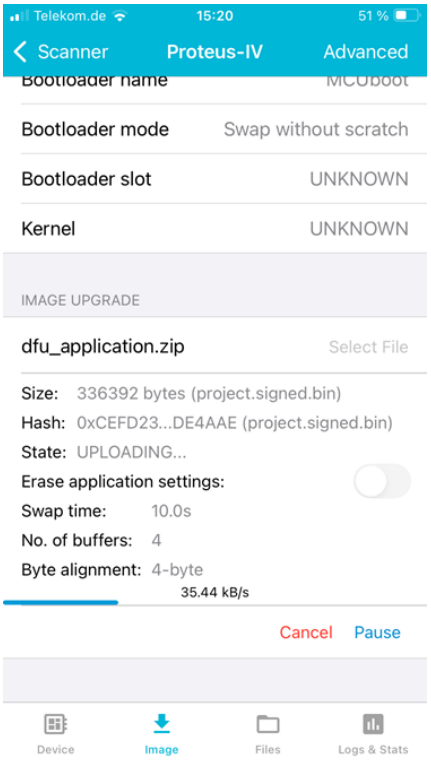
b) Device connected



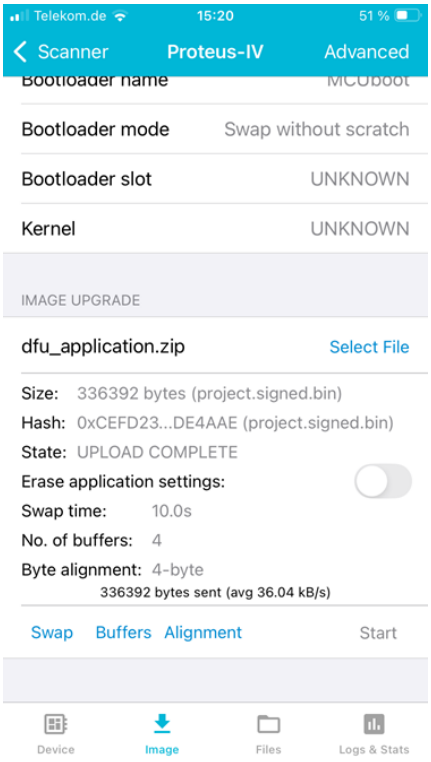
c) File selection



d) Mode selection



e) Upload progress



f) Upload complete

Figure 46: nRF Connect Device Manager app firmware update process

16. Firmware history

Version 1.0.0 "Release"

- Initial version

Version 1.1.0 "Release"

- Relax scan filter to improve connectivity to other Proteus devices

17. Hardware history

Version 2.0 "Release"

- Initial version

18. Antenna connection

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

18.1. On-board PCB antenna

The Proteus-IV has an on-board PCB antenna optimized for strong miniaturization operating in the 2.4 GHz frequency band. A simple short between the pins *RF* and *ANT* feeds the RF output of the module to the on-board antenna of the Proteus-IV. In this configuration, the module does not require any additional RF circuitry. The RF pin of module can be coupled to on-board PCB antenna or an external antenna.

- For the on-board PCB antenna: 22 pF shall be assembled on C2.
  - If additional tuning is needed in the end application, C8 and C12 can be assembled.
  - The exact values of C8 and C12 shall be specified in the end application corresponding to the individual need.

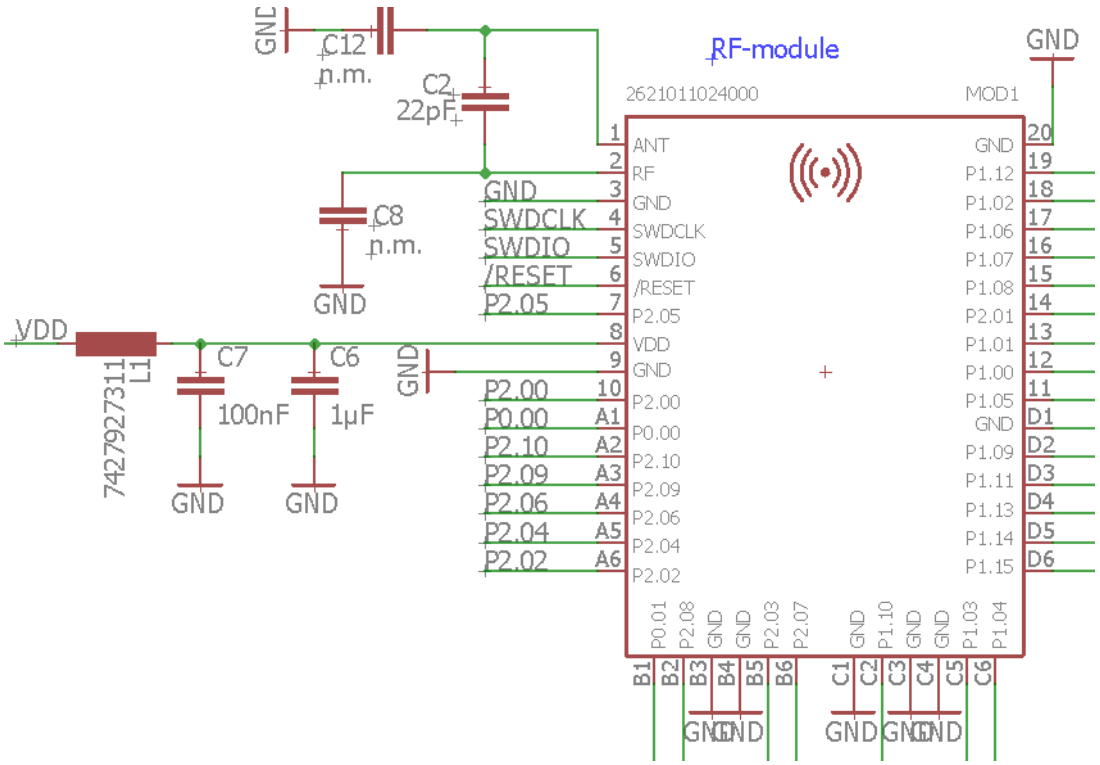


Figure 47: On-board PCB antenna

18.2. External antenna

For applications that use an external antenna, the Proteus-IV provides a 50 Ω RF signal on pin *RF* of the module. In this configuration, pin *ANT* of the module has to be connected to ground and pin *RF* to the external antenna via 50 Ω feed line. Refer to chapter 20 for further information.

- For the external antenna: 22 pF shall be assembled on C1.
  - If additional tuning is needed in the end application, C8 and C11 can be assembled.
  - The exact values of C8 and C11 shall be specified in the end application corresponding to the individual need.

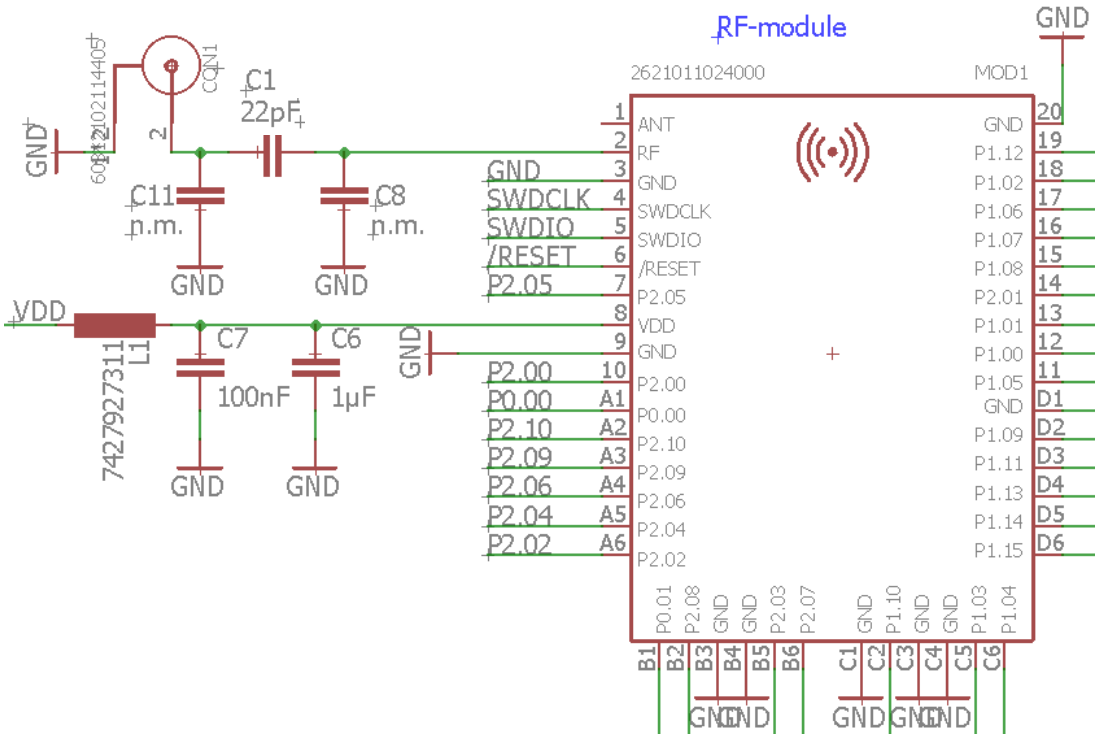


Figure 48: External antenna connection

The use cases for the integrated antenna are miniaturization and re-use of module certifications for the end-application. The use cases for the external antenna are optimization of radio range spending more space for the antenna and differentiated antenna for example when metal housings are used.

19. Design in guide

19.1. Advice for schematic and layout

For users with less RF experience it is advisable to closely copy the relating EV-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.
- 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.
- 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 malfunction. The reset pin of the module shall be held on LOW logic level whenever the VDD is not stable or below the minimum operating Voltage.
  - Special care must be taken in case of battery powered systems.
- 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.

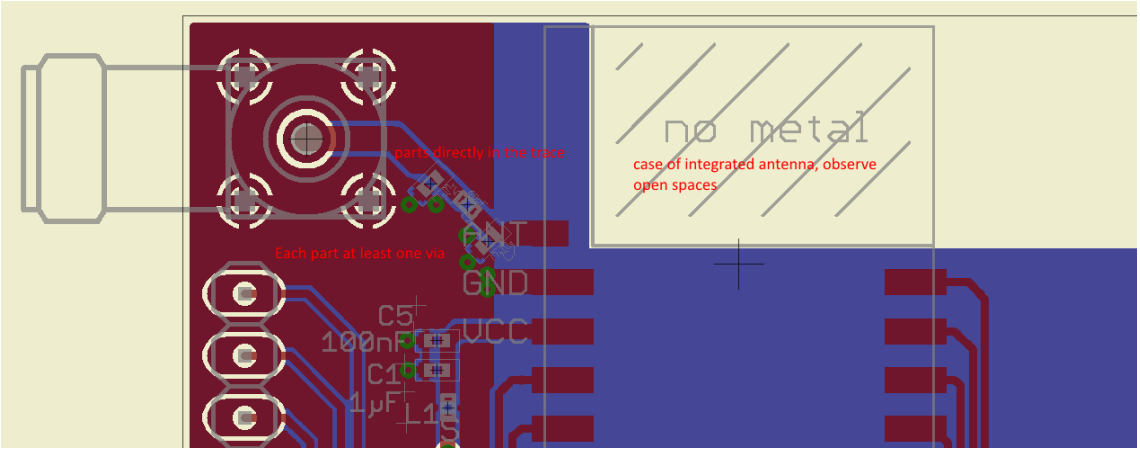


Figure 49: Layout

- In case of integrated antennas it is required to have areas free from ground. This area should be copied from the EV-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.

Fixed values can not be recommended, as these depend on the circumstances of the application (main power source, interferences etc.).

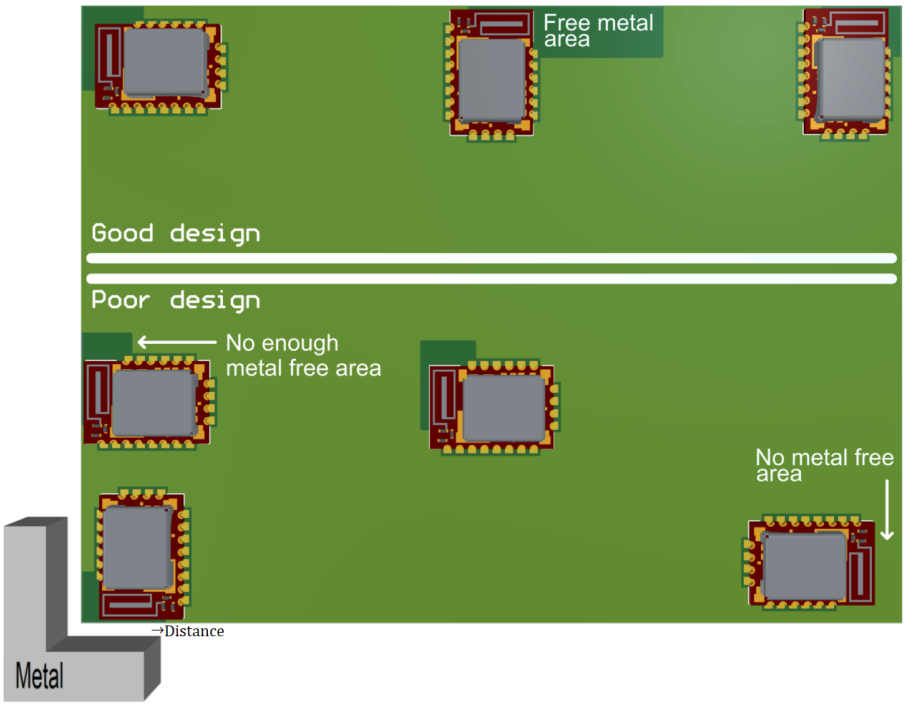


Figure 50: Placement of the module with integrated antenna

19.2. Designing the antenna connection

The antenna should be connected with a 50 Ω line. This is needed to obtain impedance matching to the module and avoids reflections. Here we show as an example how to calculate the dimensions of a 50 Ω line in form of a micro strip above ground, as this is easiest to calculate. Other connections like coplanar or strip line are more complicated to calculate but can offer more robustness to EMC. There are free calculation tools available in the internet.

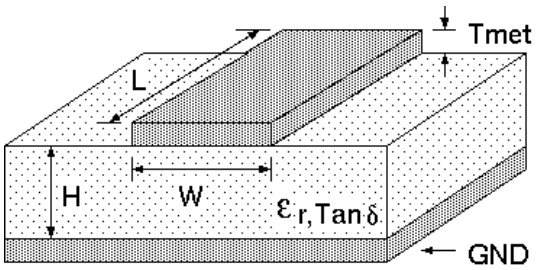


Figure 51: Dimensioning the antenna connection as micro strip

The width W for a micro strip can be calculated using the following equation:

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

Example:
A FR4 material with ε_r = 4.3, a height H = 1000 µm and a copper thickness of T_{met} = 18 µm will lead to a trace width of W ~ 1.9 mm. To ease the calculation of the micro strip line (or e.g. a

coplanar) many calculators can be found in the internet.

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

19.3. Antenna solutions

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

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

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

19.3.1. 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.

19.3.2. 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.

19.3.3. 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 EV 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.

19.3.4. Antennas provided by Würth Elektronik eiSos

Besides the radio modules Würth Elektronik eiSos provides various antennas tailored for the different frequency bands. The recommended single external antennas are shown in the subsequent chapters.

In case integrated multilayer chip antennas are needed because of space limitations, please refer to *https://www.we-online.com/en/components/products/WE-MCA*.

19.3.4.1. 2600130021 - Himalia dipole antenna

Figure 52: Himalia dipole antenna

Due to the fact that the antenna has dipole topology, there is no need for an additional ground plane. Nevertheless, the specification was measured edge mounted and 90 ° bent on a 100 x 100 mm ground plane.

Specification	Value
Frequency range [GHz]	2.4 – 2.5
Impedance [Ω]	50
VSWR	≤ 2:1
Polarization	Linear
Radiation	Omni-Directional
Peak Gain [dBi]	2.8
Average Gain [dBi]	-0.6
Efficiency	85 %
Dimensions (L x d) [mm]	83.1 x 10
Weight [g]	7.4
Connector	SMA plug
Operating temp. [°C]	-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 it is possible to remove it only through professional installation.

20. Reference design

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

Further information concerning the use of the EV-Board can be found in the manual of the Proteus-IV EV-Board.

20.1. EV-Board

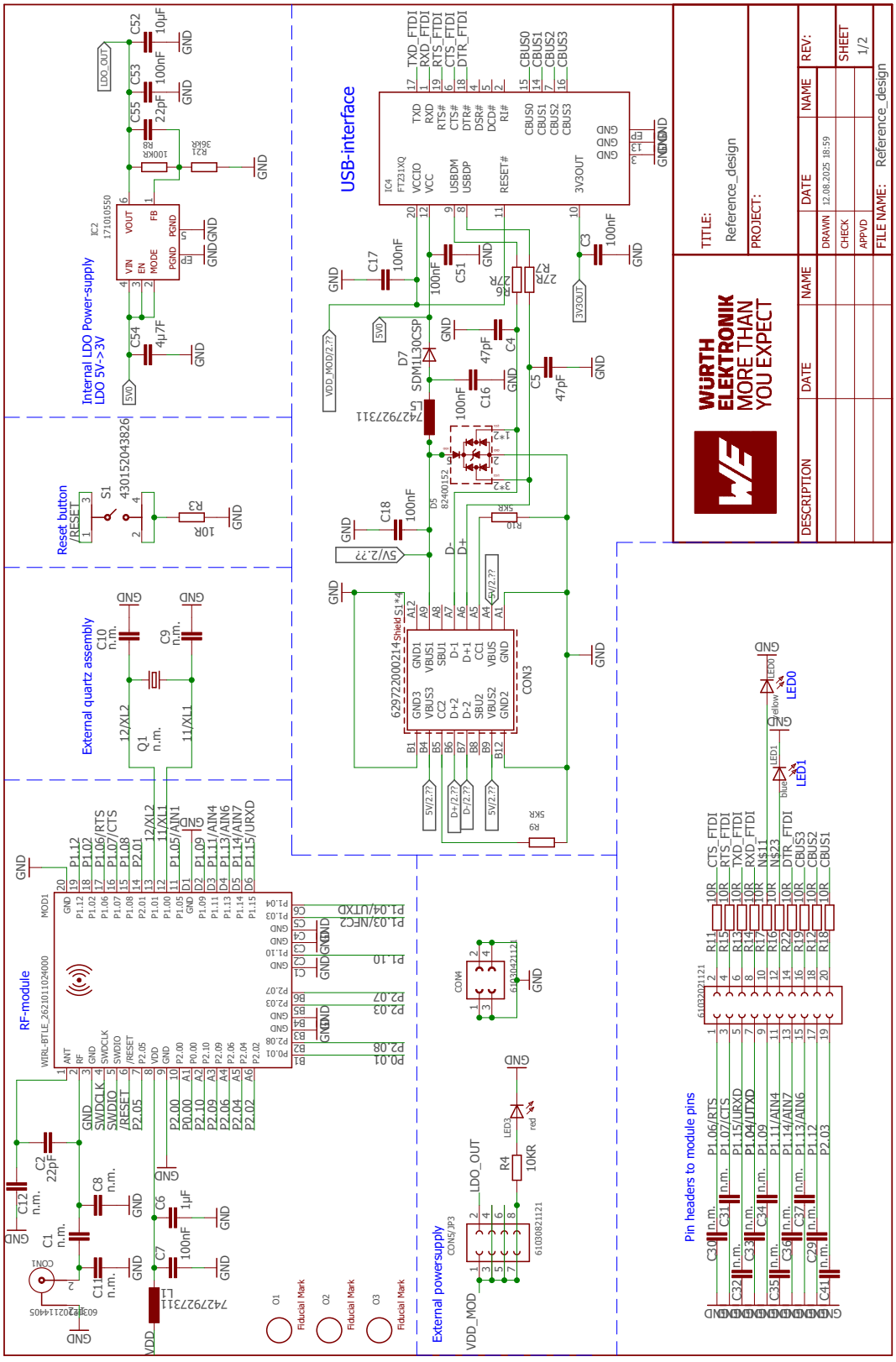


Figure 53: Reference design: schematic page 1

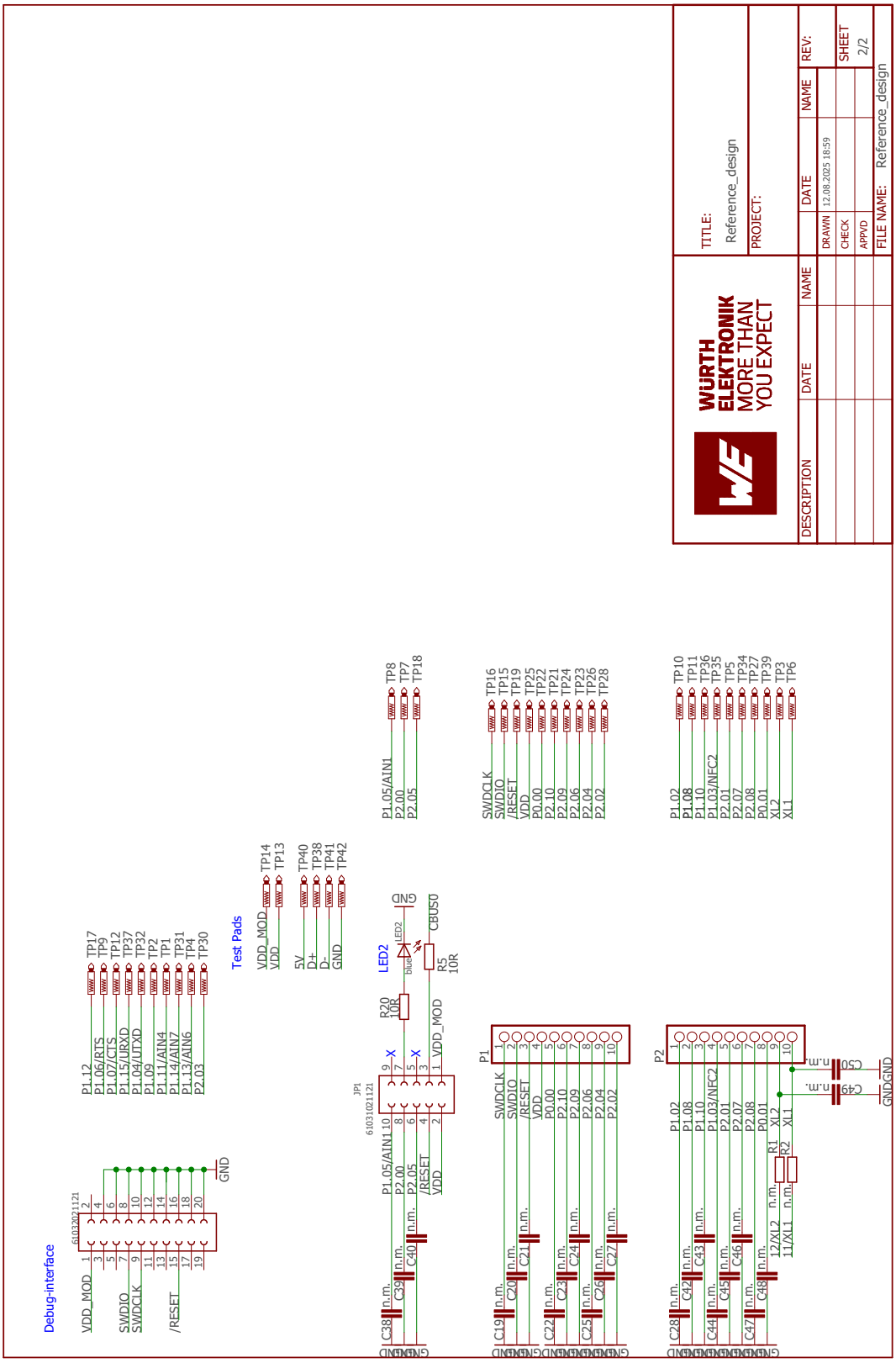


Figure 54: Reference design: schematic page 2

20.2. Layout

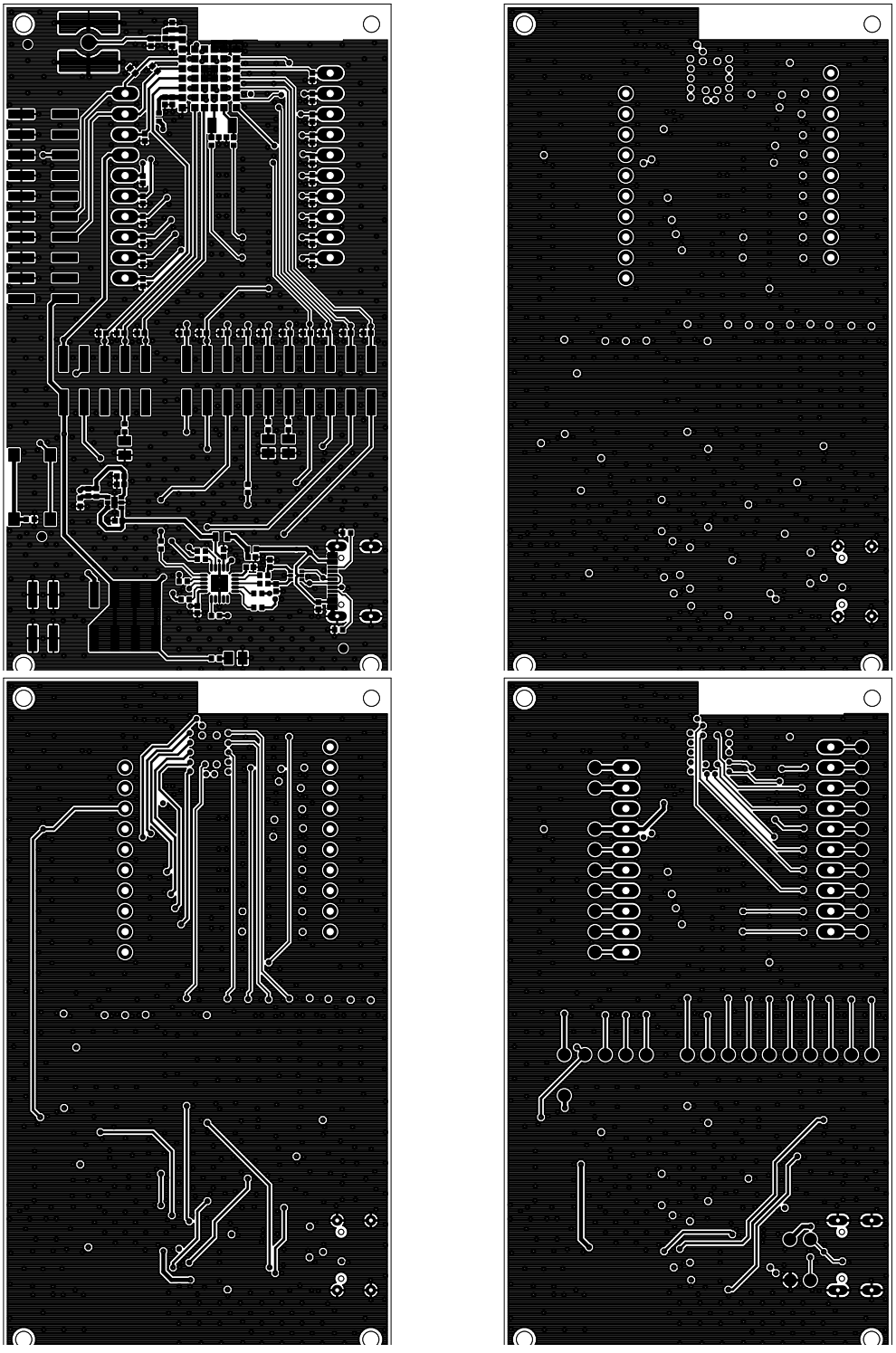


Figure 55: Top layer (top left), second layer (top right), third layer (bottom left), bottom layer (bottom right)

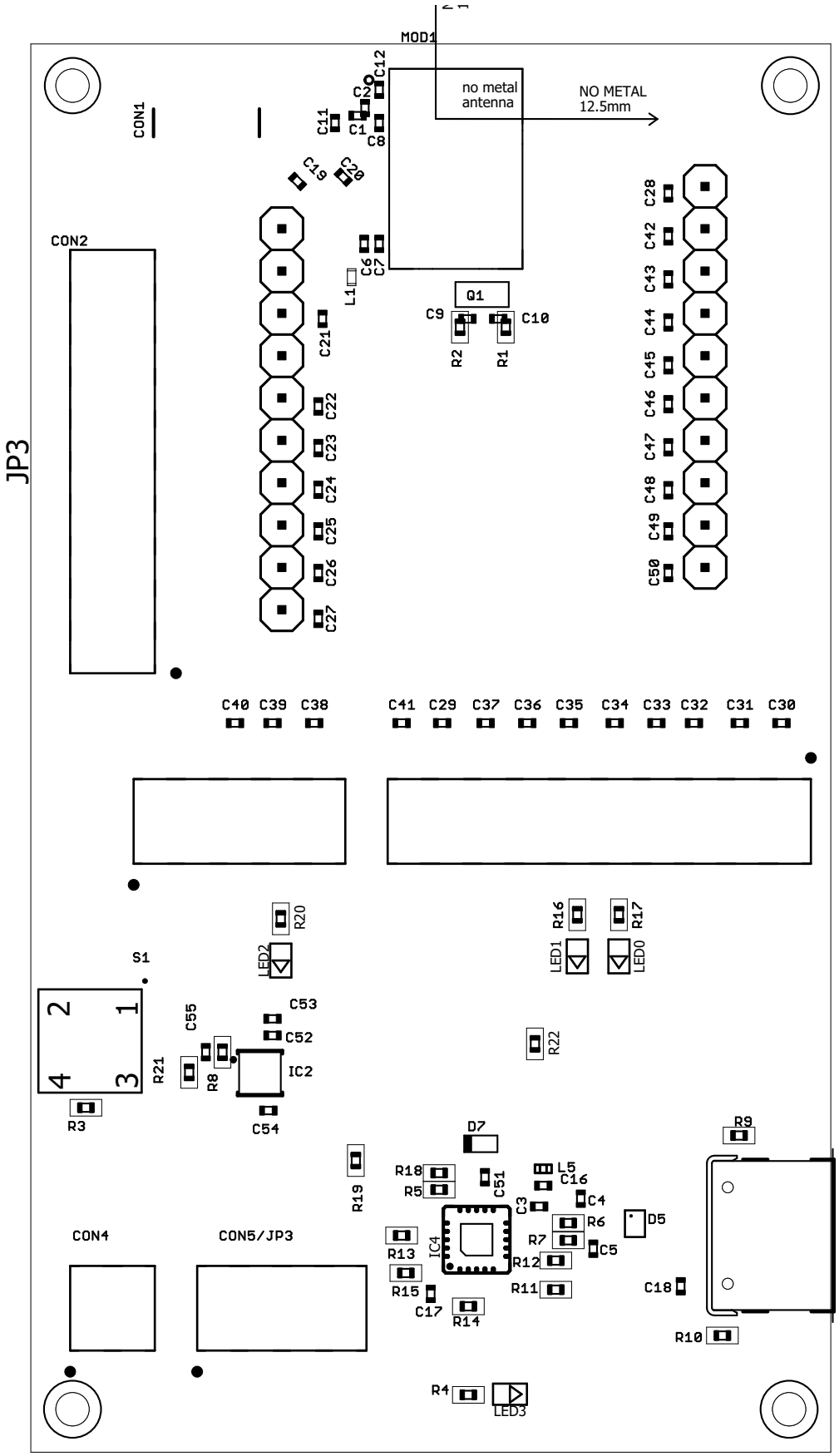


Figure 56: Reference design assembly plan

20.3. Trace design

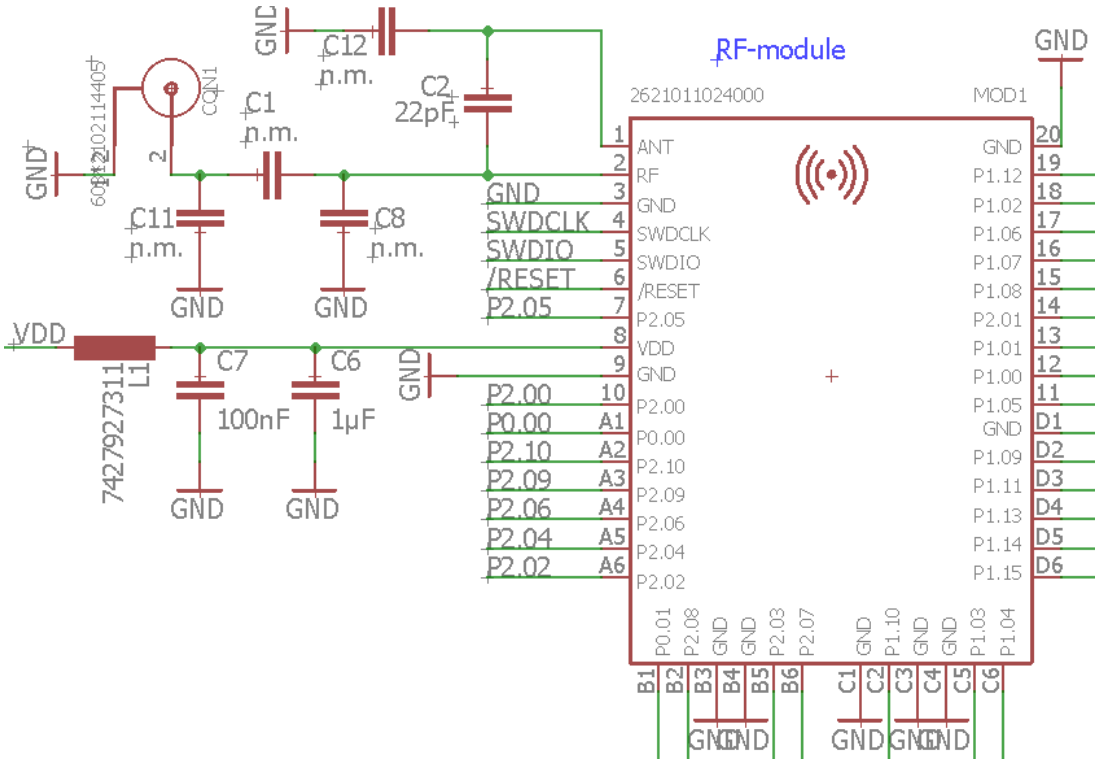


Figure 57: Trace design: Schematic

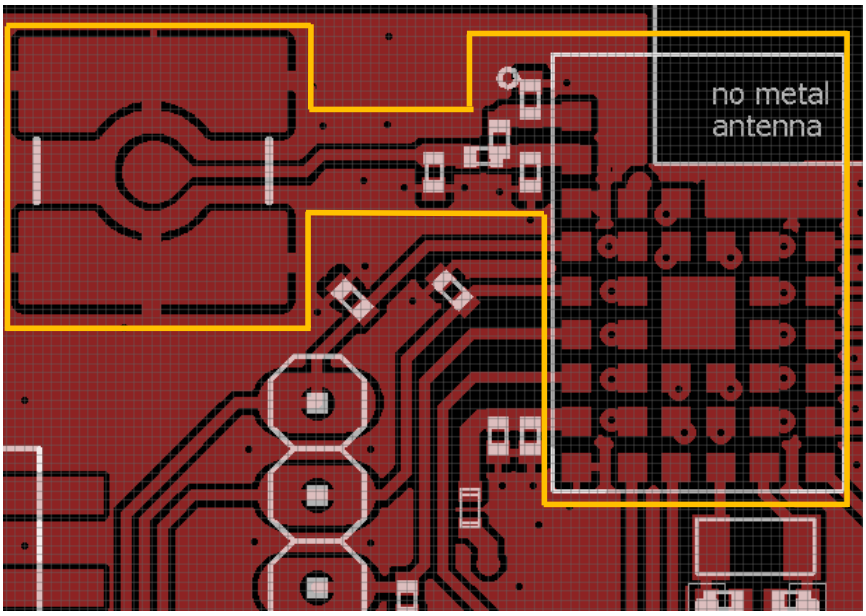


Figure 58: Trace design: Layout

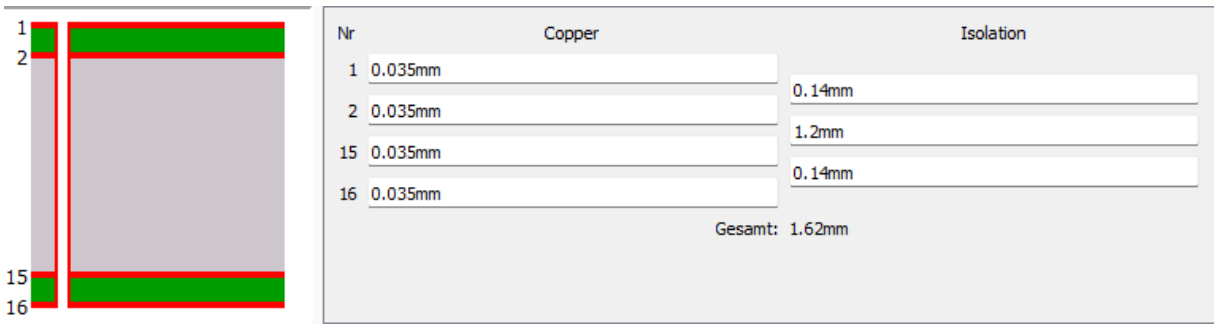


Figure 59: Trace design: Stack-up

- Top layer is used for routing, filled with ground plane except area under the module and antenna free area.
- Second layer is filled with ground plane, except the antenna free area.
- Third layer is the supply layer, except antenna free area. Some routing is allowed, not dividing the supply layer in to many or too small parts.
- Bottom layer is used for routing and filled with ground.

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

21. Manufacturing information

21.1. 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*.

21.2. Soldering

21.2.1. Reflow soldering

Attention must be paid on the thickness of the solder resist between the host PCB top side and the modules bottom side. Only lead-free assembly is recommended according to JEDEC J-STD020.

Profile feature		Value
Preheat temperature, min	$T_{S\ Min}$	150 °C
Preheat temperature, max	$T_{S\ Max}$	200 °C
Preheat time from $T_{S\ Min}$ to $T_{S\ Max}$	t_S	60 - 120 s
Ramp-up rate (T_L to T_P)		3 °C/s max.
Liquidous temperature	T_L	217 °C
Time t_L maintained above T_L	t_L	60 - 150 s
Peak package body temperature	T_P	260 °C
Time within 5 °C of actual peak temperature	t_P	20 - 30 s
Ramp-down rate (T_P to T_L)		6 °C/s max.
Time 20 °C to T_P		8 min max.

Table 41: Classification reflow soldering profile, 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.

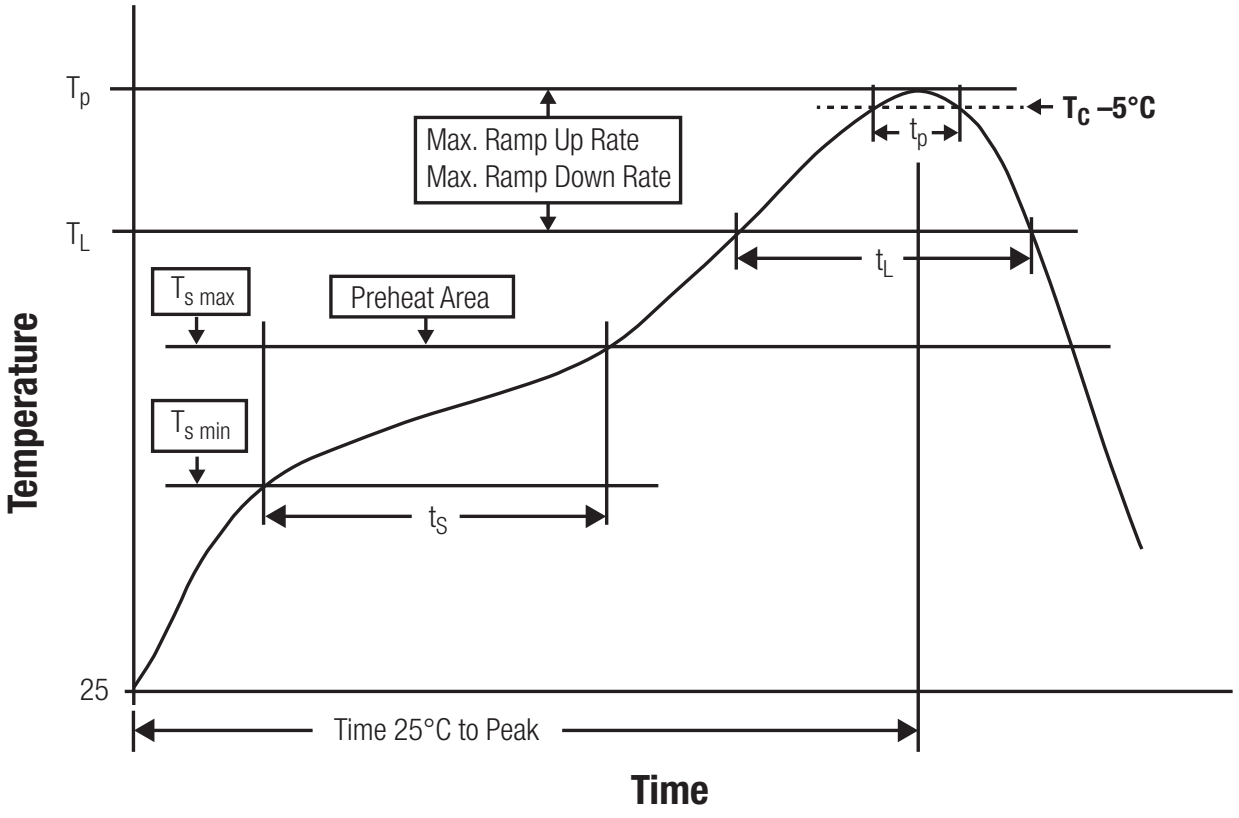


Figure 60: Reflow soldering profile

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

21.2.2. 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.

21.2.3. Potting and coating

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

21.2.4. Other notations

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

21.3. 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 4. 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.

21.4. 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 generate a large amount of heat while transmitting. The manufacturer of the end device must take care of potentially necessary actions for his application.

22. Product testing

22.1. Würth Elektronik eiSos in-house production tests

To achieve a high quality standard, Würth Elektronik eiSos follows a philosophy of supplying fully tested radio modules. At the end of the production process, every unit undergoes an optical inspection. Here the quality of soldering, edge castellation and edge milling is monitored.

If this has been passed, the radio modules are handed over to the automatic test equipment for the electrical characterization. This includes:

- Voltage and current tests to ensure proper electrical performance
- RF characteristics (frequency, spectrum, TX power) measurement and calibration
- Radio communication tests
- Firmware and serial number programming
- Host interface communication tests

The automated testing process is logged for internal quality control. The gained measurement data of each unit is analysed to detect defective parts and investigate the corresponding root cause. Defective radio modules are discarded, in order to guarantee a 100% failure-free delivery to customers.

22.2. EMS production tests

The rigorous in-series production testing ensures that EMS don't need to duplicate firmware tests or measurements. This streamlines the process and eliminates the need for additional testing over analogue and digital interfaces during device production. When it comes to device testing, the ideal focus should be on module assembly quality:

- All module pins are soldered properly on the base PCB
- There are no short circuits
- The mounting process did not damage the module
- The communication between host and radio module is working
- The antenna is connected properly

Simple "Go/No go" tests, like checking the RSSI value, give already a hint if the power supply and antenna have been connected properly.

In addition to such standard testing procedures, radio module integrators have the flexibility to perform additional dedicated tests to thoroughly evaluate the device. Specific tests they can consider are:

- Measure module current consumption in a specified operating state. Deviations from expected results (compared to a "Golden Device") can signal potential issues.

- Perform functional tests, including communication checks with the host controller and verification of interfaces.
- Assess fundamental RF characteristics (modulation accuracy, power levels, spectrum). Verify that the device meets expected performance standards.

23. Physical specifications

23.1. Dimensions

Dimensions
12 x 8 x 2.3 mm

Table 42: Dimensions

23.2. Weight

Weight
< 1 g

Table 43: Weight

23.3. Module drawing

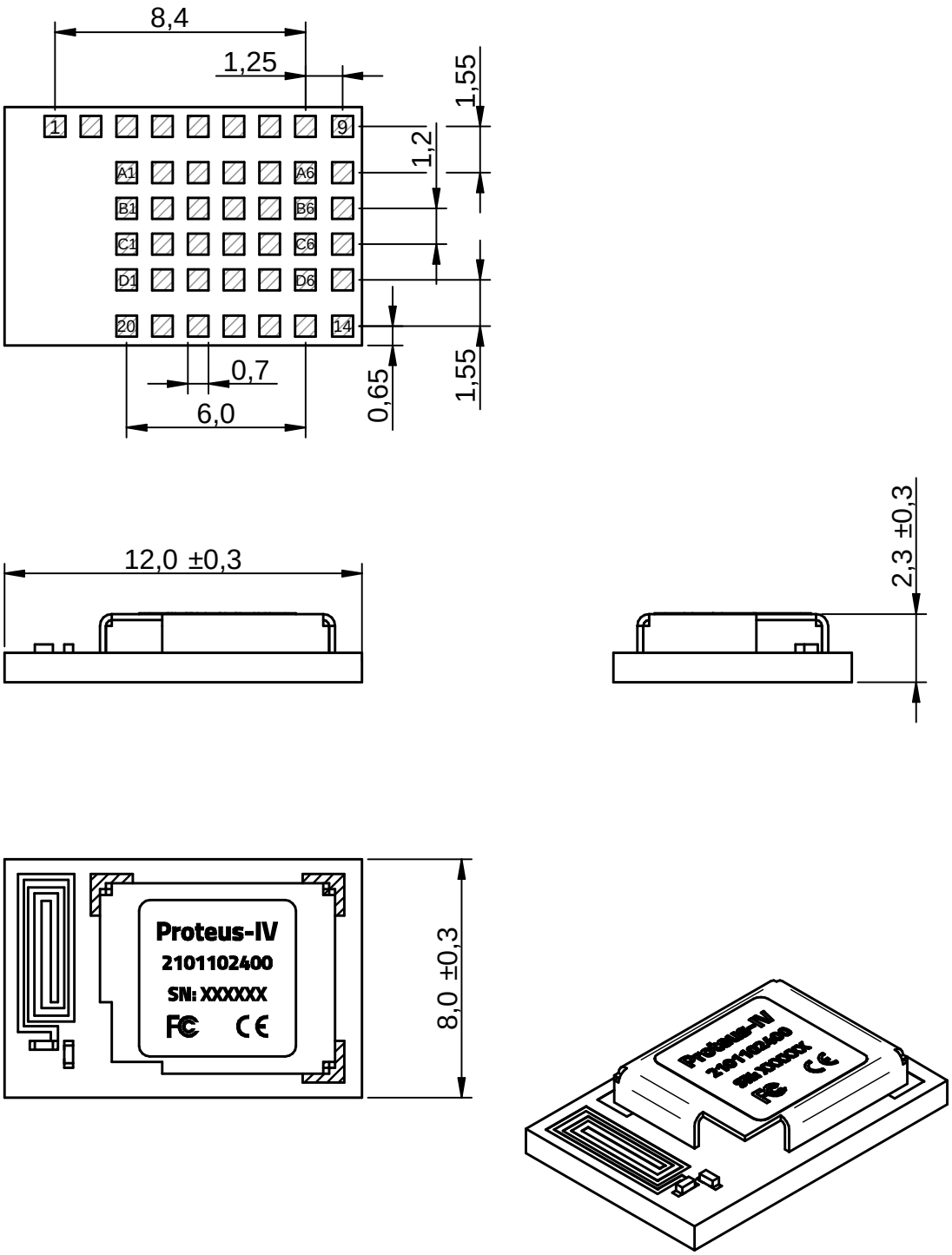


Figure 61: Module dimensions [mm]

23.4. Footprint

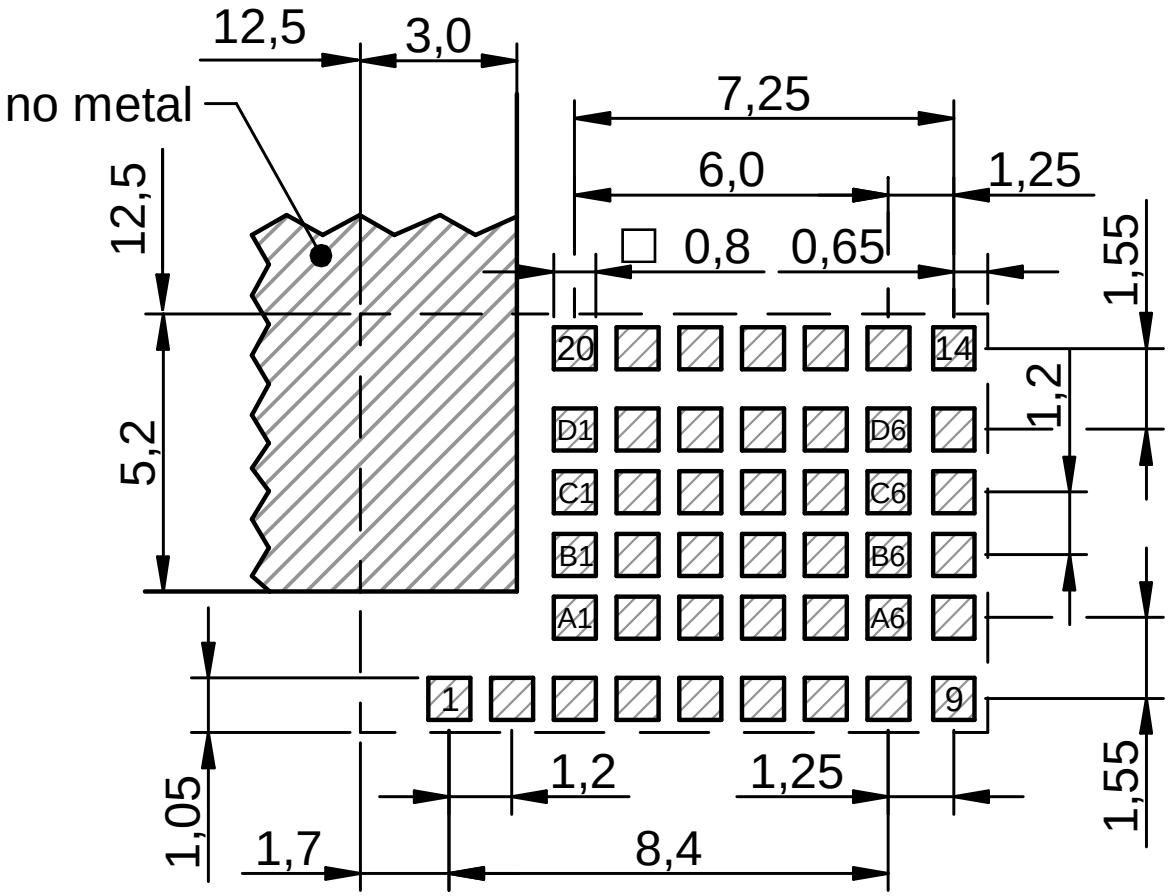


Figure 62: Footprint [mm]

23.5. 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 62). Even though metal parts would influence the characteristic of the antenna, but the direct influence and matching keep an acceptable level.

24. Marking

24.1. 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.

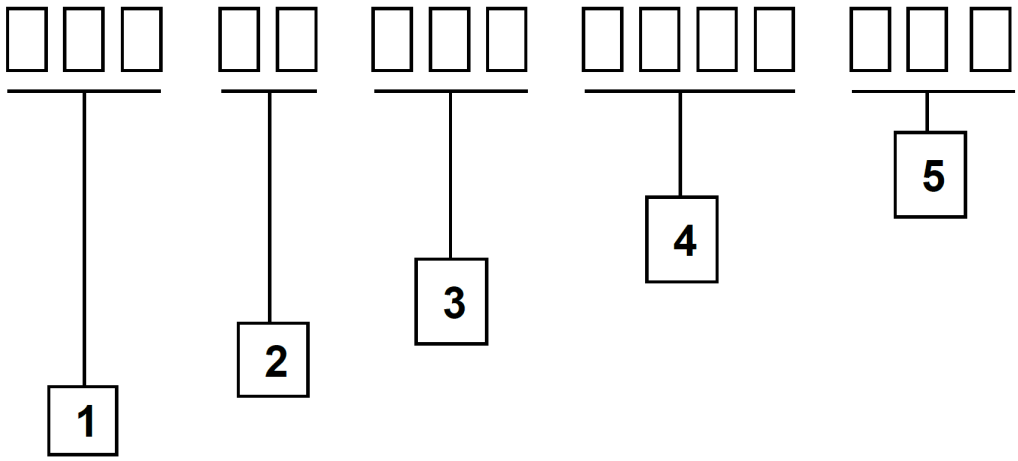


Figure 63: Lot number structure

Block	Information	Example(s)
1	eiSos internal, 3 digits	438
2	eiSos internal, 2 digits	01
3	Radio module 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	Radio module firmware version, 3 digits	V3.2 = 302, V5.13 = 513

Table 44: 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. Block 5 is not applicable for products without firmware.

24.2. General labeling information

Labels of Würth Elektronik eiSos radio modules include several fields. Besides the manufacturer identification, the product's *WE* order code, serial number and certification information are placed on the label. In case of small labels, additional certification marks are placed on the label of the reel.
The information on the label are fixed. Only the serial number changes with each entity of the radio module. For Proteus-IV the label is as follows:

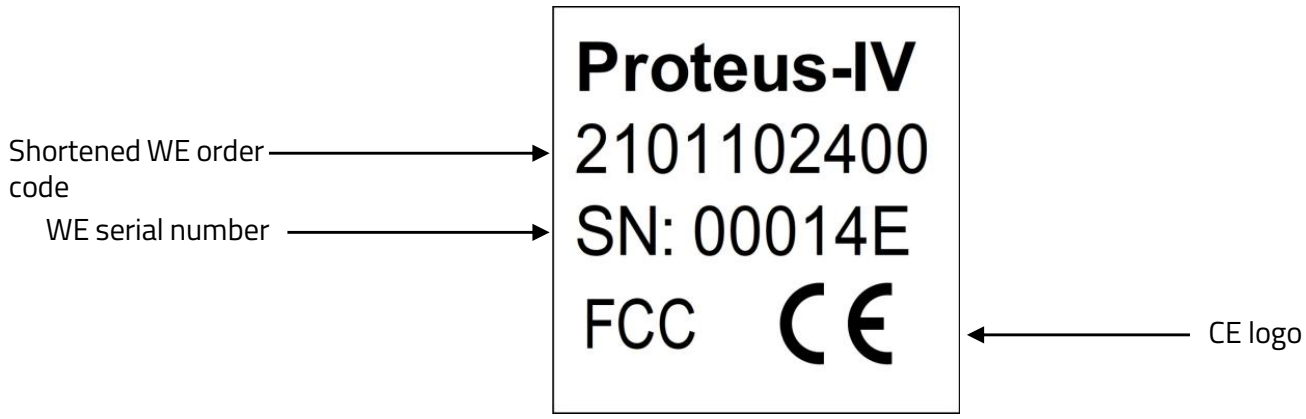


Figure 64: Label of the Proteus-IV

25. Information for explosion protection

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

- The module itself is unfused.
- The maximum output power of the module is 7.5 dBm for external antenna and 2 dBm for internal antenna.
- The total amount of capacitance of all capacitors is 14.62 µF.
- The total amount of inductance of all inductors is 4.71 µH.

26. Bluetooth® SIG listing/qualification

Type	Data
Design name	2621011024000
DN	Q370836
Specification name	6.0
Project type	Core Complete

Each product containing intellectual property of the Bluetooth® Special Interest Group (SIG) must be qualified by the SIG to obtain the corresponding Declaration ID.
Due to the qualification of the Proteus-IV as end product no further Bluetooth® tests are required. The only arising expenses are those for purchasing a Bluetooth® Declaration ID.
To obtain the Bluetooth® qualification of the end device, refer to the application note ANR027 [15].

27. Regulatory compliance information

27.1. Important notice EU

The use of RF frequencies is limited by national regulations. The Proteus-IV has been designed to comply with the RED directive 2014/53/EU of the European Union (EU).
The Proteus-IV can be operated without notification and free of charge in the area of the European Union. However, according to the RED directive, restrictions (e.g. in terms of duty cycle or maximum allowed RF power) may apply.

Modifications (2014/53/EU article 3 (i))
Caution: Changes or modifications for this equipment not expressly approved by Würth Elektronik eiSos may void the CE conformity to operate this equipment.

Since the module itself is not fused the voltage supply shall be fed from a power source which is class PS2 according to EN 62368-1.

27.2. Important notice UKCA

The UK's government has laid legislation to continue recognition of current EU requirements for a range of product regulations, including the CE marking. The Radio Equipment Regulation 2017/1206 is within the scope of this announcement, among others.
Consequently, the Proteus-IV can be sold and utilized in the UK with the CE marking, without the need of UKCA declaration of conformity or UKCA marking.

Source: *https://www.gov.uk/guidance/ce-marking*

27.3. Important notice FCC

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

27.4. Conformity assessment of the final product

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

27.5. 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.

27.6. EU Declaration of conformity

EU DECLARATION OF CONFORMITY

Radio equipment: **2621011024000**

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: 2621011024000

The object of the declaration described above is in conformity with the relevant Union harmonisation legislation Directive 2014/53/EU. 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 V3.3.1 (2024-09)
EN 62479 : 2010
EN 62368-1:2014 + AC:2015 + A11:2017 + AC:2017
2011/65/EU with its amending Annex II EU 2015/863

i.A. G. Erlandt

Trier, 12th of August 2025
Place and date of issue

27.7. RED-DA Cybersecurity statement

Cybersecurity as per articles 3.3d, 3.3e and 3.3f of the Radio Equipment Directive Delegated Act. The RED-DA mandates to comply to the EN 18031-1, 18031-2 and 18031-3 in order to fulfill the requirements of the cybersecurity chapters (d, e and f).

- EN 18031-1: Common security requirements for radio equipment - Part 1: Internet connected radio equipment
- EN 18031-2: Common security requirements for radio equipment - Part 2: Radio equipment processing data, namely internet connected radio equipment, childcare radio equipment, toys radio equipment and wearable radio equipment
- EN 18031-3: Common security requirements for radio equipment - Part 3: Internet connected radio equipment processing virtual money or monetary value

Requirements	Statement and conditions
(d) Radio equipment does not harm the network or its functioning nor misuses network resources, thereby causing an unacceptable degradation of service	"Not applicable": The product is not capable itself to communicate over the internet. The product is only able to communicate via the following protocols and interfaces. None of the protocols contained in the product are "internet-connectable". Radio communication protocols: This Bluetooth (2.1, 4.x, 5.x, 6.x) product does not support or include the "Internet Protocol Support Profile". Bluetooth is a set of radio standards (e.g. Bluetooth Classic, Bluetooth Low Energy, Bluetooth MESH, Bluetooth LE Audio, ...) maintained by the Bluetooth SIG. Host Interface, wired: The host interface of the product does not support internet connectivity. UART is used as a wired communication and control channel towards the customers host.
(e) Radio equipment incorporates safeguards to ensure that the personal data and privacy of the user and of the subscriber are protected	"Not applicable": The product is not internet connected. The product does not pose a risk to the users or subscribers privacy, as it does not store or process any personal data.
(f) Radio equipment supports certain features ensuring protection from fraud	"Not applicable": The product is not internet connected. The product does not pose a risk of fraud because it does not store or process financial data or enables financial transactions.

27.8. FCC Compliance Statement (US)

FCC ID: R7T2101102

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.

TCB

GRANT OF EQUIPMENT AUTHORIZATION

TCB

Certification

Issued Under the Authority of the

Federal Communications Commission

By:

cetecom advanced GmbH
Untertuerkheimer Strasse 6-10
Saarbruecken, 66117
Germany

Date of Grant: 09/04/2025

Application Dated: 09/04/2025

Wuerth Elektronik eiSos GmbH & Co KG

Max-Eyth-Strasse 1

Waldenburg, 74638
Germany

Attention: Gudrun Eckhardt , Teamleader Hardware Development

NOT TRANSFERABLE

EQUIPMENT AUTHORIZATION is hereby issued to the named GRANTEE, and is VALID ONLY for the equipment identified hereon for use under the Commission's Rules and Regulations listed below.

FCC IDENTIFIER:	R7T2101102
Name of Grantee:	Wuerth Elektronik eiSos GmbH & Co KG
Equipment Class:	**Digital Transmission System**
Notes:	**Bluetooth LE radio module**
Modular Type:	**Single Modular**

Grant Notes	FCC Rule Parts	Frequency Range (MHZ)	Output Watts	Frequency Tolerance	Emission Designator
	15C	**2402.0 - 2480.0**	**0.00646**		

Output power listed is peak conducted. The module is only sold to OEMs.

27.9. IC Compliance Statement (Canada)

Certification Number: 5136A-2101102
HVIN: 2101102
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.

cetecom advanced GmbH
CAB Identifier DE0001

based on the Comprehensive Economic and Trade Agreement (CETA) between the European Community and Canada

DAkkS Deutsche Akkreditierungsstelle D-ZE-12047-01-00

TECHNICAL ACCEPTANCE CERTIFICATE - CANADA

Certificate Holder	**Würth Elektronik eiSos GmbH & Co. KG** Max-Eyth-Str. 1 74638, Waldenburg Germany
ISED Certification Number	5136A-2101102
Product Description	Bluetooth LE radio module
cetecom advanced Registration No.	1-9394/25-03-08
Wireless Test Lab Address	**cetecom advanced GmbH** Untertuerkheimer Str. 6-10 66117 Saarbruecken Germany Phone:49 681 598 0 Website: www.cetecomadvanced.com
Wireless Test Lab ID	3462C

This certificate is considered valid based on the following conditions:
- The certified products must continue to comply with the latest issue of all applicable ISED standards
- This certificate does not constitute a radio licence, where required by the applicable ISED standard(s), a radio license must be obtained from an ISED regional office prior to product operation
- The certified products shall not be manufactured, imported, distributed, leased, offered for sale, or sold unless the product certification information is listed on ISED's Radio Equipment List (REL)
- For ISED and/or CB audit purposes, sample of certified product shall be made available to ISED and/or CB

Ce certificat est considéré valide sur la base des conditions suivantes :
- Les produits certifiés doivent continuer à être conformes à la dernière version de toutes les normes d'ISDE applicables.
- Ce certificat ne constitue pas une licence radio. Si une licence radio est requise selon la ou les normes d'ISDE applicables, la licence radio doit être obtenue auprès du bureau régional d'ISDE avant l'exploitation du produit.
- Les produits certifiés ne doivent pas être fabriqués, importés, distribués, loués, mis en vente ou vendus à moins que les informations relatives à la certification du produit figurent dans la Nomenclature du matériel radio (NMR) d'ISDE.
- À des fins d'audit de la part d'ISDE et/ou de l'OC, des échantillons de produits certifiés doivent être mis à la disposition d'ISDE et/ou de l'OC.

I hereby attest that the subject equipment was tested and found in compliance with the above-noted specification.
J'atteste par la présente que le matériel a fait l'objet d'essai et jugé conforme à la spécification ci-dessus.

Place, date of issue

Saarbruecken, 2025-09-04

FOREIGN CERTIFICATION BODY
CAB ID NO DE0001

cetecom advanced GmbH

.....
Stefan Bös - Reviewer & Decision Maker

This certificate becomes valid when published in REL at :
Le présent certificat n'entre en vigueur qu'après être publié en REL sur :
https://sms-sgs.ic.gc.ca/equipmentSearch/searchRadioEquipments?execution=e1s1&lang=en_CA

cetecom advanced GmbH
Untertuerkheimer Str. 6–10 | 66117 Saarbruecken | Germany | www.cetecomadvanced.com

27.10. FCC and IC requirements to OEM integrators

This module has been granted modular approval. OEM integrators for host products may use the module in their final products without additional FCC/IC (Industry Canada) certification if they meet the following conditions. Otherwise, additional FCC/IC approvals must be obtained. The host product with the module installed must be evaluated for simultaneous transmission requirements.

- The users manual for the host product must clearly indicate the operating requirements and conditions that must be observed to ensure compliance with current FCC/IC RF exposure guidelines.
- A label must be affixed to the outside of the host product with the following statements:
  This device contains FCC ID: R7T2101102
  This equipment contains equipment certified under IC ID: 5136A-2101102
- The final host / module combination may also need to be evaluated against the FCC Part 15B criteria for unintentional radiators in order to be properly authorized for operation as a Part 15 digital device.
- If the final host / module combination is intended for use as a portable device (see classifications below) the host manufacturer is responsible for separate approvals for the SAR requirements from FCC Part 2.1093 and RSS-102.

OEM requirements:
The OEM must ensure that the following conditions are met.

- The Proteus-IV will be used at a distance of at least 10 mm to the human body.
- End users of products, which contain the module, must not have the ability to alter the firmware that governs the operation of the module. The agency grant is valid only when the module is incorporated into a final product by OEM integrators.
- The end-user must not be provided with instructions to remove, adjust or install the module.
- The Original Equipment Manufacturer (OEM) must ensure that FCC labeling requirements are met. This includes a clearly visible label on the outside of the final product. Attaching a label to a removable portion of the final product, such as a battery cover, is not permitted.
- The label must include the following text:
  Contains FCC ID: R7T2101102
  The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
  (i.) this device may not cause harmful interference and
  (ii.) this device must accept any interference received, including interference that may cause undesired operation.

  When the device is so small or for such use that it is not practicable to place the statement

above on it, the information required by this paragraph shall be placed in a prominent location in the instruction manual or pamphlet supplied to the user or, alternatively, shall be placed on the container in which the device is marketed. However, the FCC identifier or the unique identifier, as appropriate, must be displayed on the device.

- The user manual for the end product must also contain the text given above.
  - Changes or modifications not expressly approved could void the user's authority to operate the equipment.
  - The OEM must ensure that timing requirements according to 47 CFR 15.231(a-c) are met.
  - The OEM must sign the OEM Modular Approval Agreement.
  - The module must be used with only the following approved antenna(s).

27.10.1. Pre-certified antennas

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

Product	Certified antenna
Proteus-IV (2621011024000)	PCB antenna included in the Proteus-IV development kit

27.11. ETA-WPC (India)

Registration No: ETA-SD-20251110251 Date: 28-11-2025

The Proteus-IV complies with the provisions on the Equipment Type Approval WPC Wing for India.

सत्यमेव जयते

Government of India
Ministry of Communications
Department of Telecommunications
WPC Wing
Sanchar Bhawan, New Delhi-110001.

[Generation of Equipment Type Approval (ETA) through self-declaration issued under O.M. No. ETA-WPC /Policy/2018-19 dated 26 February, 2019].

THIS ETA IS ISSUED FOR A SINGLE MODEL WITH MODEL NAME Proteus-IV (2621011024000)

Registration No: ETA-SD-20251110251 Date: 28-11-2025

I). Details of Applicant and Parameters of Equipment:

1.	Name & Address of the first Applicant. (Indian Manufacturer/ Authorised Indian representative for foreign manufacturer)	WURTH ELECTRONICS SERVICES INDIA PRIVATE LIMITED, Ground and 1st Floor, No. 3, Prestige Sterling Square, Madras Bank Road, Next to Airlines Hotel, Banglore, Bengaluru Bangalore Urban, Karnataka, 560001, Bangalore Urban,KARNATAKA,560001
2.	Equipment category	Bluetooth Low Energy Module
3.	Make	Wurth Elektronik eiSos GmbH & Co. KG,Germany
4.	Model	Proteus-IV (2621011024000)
5.	Frequency range(s) of Equipment	1. 2402-2480 MHz
6.	Max output power/Field strength/PSD	1. E.I.R.P. (dBm). 4.5

7.	Applicable Gazette Notification(s)	1. 45 (E) Dated 28-01-2005	
8.	RF Test Report details:-		
	Name&Address /Country of accredited laboratory issuing the RF test report	Accreditation Certificate Reference/Number	Test Report No. and Date
	cetecom advanced GmbH & Untertuerkheimer Str. 6-10, 66117 Saarbruecken	D-PL-12047-01-00	1-9394-25-03-03_TR1-R02 & 11-06-2025
	cetecom advanced GmbH & Untertuerkheimer Strasse 6 10 66117 Saarbruecken / Germany	D-PL-12047-01-00	1-9394-25-03-10_TR1-R01 & 19-05-2025

II). Terms and Conditions

(i). This certificate will not be valid in case any change in the above parameters and not conforming to the Gazette Notification mentioned in sl.no 7 above.
(ii). Use of such equipment has been exempted from licensing requirement vide Gazette Notification mentioned in sl. no. 7, on Non-interference,Non-protectionand sharing (non-exclusive) basis.
(iii). Use of such equipment in case not conforming to above notification will require a specific wireless operating license, as applicable from this Ministry.
(iv). Field units of WPC Wing reserve the right for sample check/audit carried out for the purpose of RF analysis/spectrum monitoring in view to avoid interference to other wireless users and ensure compliance of technical parameters mentioned in sl no. 5,6&7.
(v). This certificate is valid only for equipment which are exempted from import licensing requirements as per the Import Policy of DGFT and for import of such device, a self-declaration based, system generated (Saralsanchar) Import undertaking/ permission is required.
(vi). The applicant is liable for prosecution under Indian Law in case of any wrong declaration/ submission of ingenuine RF test report(s) for issue of ETA through Self-Declaration.

Note:
1. Once ETA through self-declaration is generated for a model, subsequently it may be utilized by other person(s) for import/usage purpose in India.

2. The importers of above model shall comply with other import related requirements, if any, with Customs.

This is Self-generated certificate. Hence, no signature is required. It may be downloaded/verified from the website https://saralsanchar.gov.in.

28. References

[1] Bluetooth®. Bluetooth® Core Specification, version 6.0. https://www.bluetooth.com/specifications/specs/core-specification-6-0/.

[2] Nordic Semiconductor. Nordic nRF54L15 resources. https://www.nordicsemi.com/products/nrf54l15.

[3] Würth Elektronik. WE Bluetooth LE Terminal app for Android. https://play.google.com/store/apps/details?id=com.eisos.android.terminal.

[4] Würth Elektronik. WE Bluetooth LE Terminal app for iOS. https://apps.apple.com/de/app/proteus-connect/id1533941485.

[5] Würth Elektronik. Source code of WE Bluetooth LE Terminal app (cross platform). https://github.com/WurthElektronik/Proteus-Connect.

[6] Eddystone Beacon. https://en.wikipedia.org/wiki/Eddystone_(Google).

[7] iBeacon. https://en.wikipedia.org/wiki/IBeacon.

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

[9] Würth Elektronik. Application note 8 - Wireless connectivity SDK. http://www.we-online.com/ANR008.

[10] IEEE Registration Authority. https://standards.ieee.org/products-programs/regauth/.

[11] Bluetooth®. Bluetooth® Assigned numbers. https://www.bluetooth.com/specifications/assigned-numbers/.

[12] Würth Elektronik. WE UART Terminal PC tool (Smart Commander). https://www.we-online.de/wcs-software.

[13] Nordic Semiconductor. nrf connect device manager. https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrfconnectdevicemanager.

[14] Nordic Semiconductor. nrf connect device manager. https://apps.apple.com/us/app/nrf-connect-device-manager/id1519423539.

[15] Würth Elektronik. Application note 27 - Bluetooth listing guide. http://www.we-online.com/ANR027.

29. Important notes

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

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.

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 source code 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.

Best care and attention
Any product-specific data sheets, manuals, application notes, PCNs, warnings and cautions must be strictly observed in the most recent versions and matching to the products revisions. These documents can be downloaded from the product specific sections on the wireless connectivity and sensors homepage.

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 Business Development Engineer (BDM) or the internal sales person in charge should be contacted who will be happy to support in this matter.

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 Business Development Engineer (BDM), the internal sales person or the technical support team in charge should be contacted. The basic responsibility of the customer as per section 29 and 29 remains unaffected.
All software like ¨wireless connectivity SDK¨, ¨Sensor SDK¨ or other source codes as well as all PC software tools are not subject to the Product Change Notification information process.

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 Business Development Engineer (BDM) 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.

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.

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*.

30. Legal notice

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.

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 information 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. KG and its representatives against any damages arising out of the use of any Würth Elektronik eiSos GmbH & Co. KG components in safety-critical applications.

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.

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.

31. License terms

These 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 these license terms are 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 these license terms. You agree to comply with all provisions under these license terms.

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 these 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 these 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 these 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.

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.

Ownership

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

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.

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.

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.

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

Severability clause
If a provision of these 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.

Miscellaneous
Würth Elektronik eiSos reserves the right at any time to change these 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 product, you accept these 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 46: CRC8 Test Vectors

B. Example code 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.com/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

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