

USER MANUAL

PROTEUS-III-SPI

2611011024010

VERSION 1.6

FEBRUARY 19, 2024

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

Revision history

| Manual version | FW version | HW version | Notes | Date |
|----------------|------------|------------|--|---------------|
| 1.0 | 1.2.0 | 1.2 | <ul style="list-style-type: none"> Initial release | February 2021 |
| 1.1 | 1.3.0 | 1.2 | <ul style="list-style-type: none"> New features of firmware version 1.3.0. Please refer to chapter Firmware history. | July 2021 |
| 1.2 | 1.4.0 | 1.2 | <ul style="list-style-type: none"> New features of firmware version 1.4.0. Please refer to chapter Firmware history. Added overview of helpful application notes | July 2022 |
| 1.3 | 1.4.0 | 1.2 | <ul style="list-style-type: none"> Added radiation characteristics in chapter 16.2 Updated firmware update description in chapter 13.2.1 Added new radio certification for UKCA in chapter 22.6 and India in chapter 22.10 Added chapter on radio compatibility to other Proteus radio modules in chapter 5.13 | May 2023 |
| 1.4 | 1.4.0 | 1.2 | <ul style="list-style-type: none"> Added certificates in addition to the required compliance statements in chapter Regulatory compliance information. | June 2023 |

| | | | | |
|-----|-------|-----|--|---------------|
| 1.5 | 1.5.0 | 1.5 | <ul style="list-style-type: none"> • Added known issue KI-001 in chapter 14.1 • Added information concerning new firmware version in chapter 14 Firmware history • Extended chapter DIS - Device information service with description how to make the firmware version visible on the radio | December 2023 |
| 1.6 | 1.5.0 | 1.5 | <ul style="list-style-type: none"> • Added NCC compliance statement for Taiwan chapter 22.11 | February 2024 |

★ For firmware history see chapter Firmware history

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

Contents

| | |
|--|-----------|
| Overview of helpful application notes | 12 |
| 1. Introduction | 14 |
| 1.1. Operational description | 14 |
| 1.1.1. Key features | 15 |
| 1.1.2. Connectivity | 16 |
| 1.2. Block diagram | 17 |
| 1.3. Ordering information | 17 |
| 2. Electrical specifications | 18 |
| 2.1. Recommended operating conditions | 18 |
| 2.2. Absolute maximum ratings | 18 |
| 2.3. Power consumption | 19 |
| 2.3.1. Static | 19 |
| 2.3.2. Dynamic | 20 |
| 2.4. Radio characteristics | 22 |
| 2.5. Pin characteristics | 23 |
| 3. Pinout | 24 |
| 4. Quick start | 27 |
| 4.1. Minimal pin connections | 27 |
| 4.2. Antenna connection | 29 |
| 4.2.1. On-board PCB antenna | 29 |
| 4.2.2. External antenna | 29 |
| 4.3. Power up | 30 |
| 4.4. Test the SPI interface | 30 |
| 4.5. Quickstart example | 31 |
| 5. Functional description | 34 |
| 5.1. Operation modes | 34 |
| 5.2. Radio module states | 34 |
| 5.3. State indication using the LED pins | 36 |
| 5.4. Sleep mode | 36 |
| 5.5. Identification of a Proteus-III-SPI device on the radio | 37 |
| 5.6. Connection based data transmission, with or without security | 37 |
| 5.6.1. Further information for a secure connection setup | 38 |
| 5.6.1.1. Just works mode | 38 |
| 5.6.1.2. StaticPasskey mode | 41 |
| 5.6.1.3. LescPasskey mode | 44 |
| 5.6.1.4. LescNumComp mode | 47 |
| 5.6.1.5. Bonding | 50 |
| 5.7. Unidirectional connectionless data transmission using Beacons | 54 |
| 5.8. Energy-efficient distance estimation solutions | 55 |
| 5.9. Configure the module for low power consumption | 55 |
| 5.10. Start the direct test mode (DTM) | 56 |
| 5.11. Using the 2 MBit and LE Coded phy | 59 |

| | | |
|-----------|--|-----------|
| 5.12. | Connection setup using LE Coded phy | 59 |
| 5.13. | Radio compatibility to other Proteus devices | 62 |
| 6. | Host connection | 63 |
| 6.1. | Serial interface: SPI | 63 |
| 6.1.1. | SPI communication details in command mode | 64 |
| 6.1.2. | Shared WAKE_UP/SPI_INT pin | 66 |
| 7. | The command interface | 68 |
| 7.1. | Scan for other modules in range | 69 |
| 7.1.1. | CMD_SCANSTART_REQ | 69 |
| 7.1.2. | CMD_SCANSTOP_REQ | 69 |
| 7.1.3. | CMD_GETDEVICES_REQ | 70 |
| 7.1.3.1. | Example 1 | 71 |
| 7.1.4. | CMD_RSSI_IND | 71 |
| 7.1.5. | CMD_BEACON_RSP | 72 |
| 7.2. | Setup connections | 73 |
| 7.2.1. | CMD_CONNECT_REQ | 73 |
| 7.2.2. | CMD_CONNECT_IND | 73 |
| 7.2.3. | CMD_SECURITY_IND | 74 |
| 7.2.4. | CMD_CHANNELOPEN_RSP | 74 |
| 7.2.5. | CMD_DISCONNECT_REQ | 74 |
| 7.2.6. | CMD_DISCONNECT_IND | 75 |
| 7.2.7. | CMD_PHYUPDATE_REQ | 75 |
| 7.2.8. | CMD_PHYUPDATE_IND | 76 |
| 7.2.9. | CMD_PASSKEY_REQ | 76 |
| 7.2.10. | CMD_PASSKEY_IND | 77 |
| 7.2.11. | CMD_DISPLAY_PASSKEY_IND | 77 |
| 7.2.12. | CMD_NUMERIC_COMP_REQ | 78 |
| 7.2.13. | CMD_GETBONDS_REQ | 78 |
| 7.2.13.1. | Example 1 | 79 |
| 7.2.14. | CMD_DELETEBONDS_REQ | 79 |
| 7.2.14.1. | Example 1 | 80 |
| 7.2.14.2. | Example 2 | 80 |
| 7.2.15. | CMD_ALLOWUNBONDEDCONNECTIONS_REQ | 81 |
| 7.3. | Transmit and receive data | 82 |
| 7.3.1. | CMD_DATA_REQ | 82 |
| 7.3.2. | CMD_TXCOMPLETE_RSP | 82 |
| 7.3.3. | CMD_DATA_IND | 83 |
| 7.3.4. | CMD_SETBEACON_REQ | 83 |
| 7.3.5. | CMD_BEACON_IND | 83 |
| 7.4. | Configuring the module and modifying the device settings | 85 |
| 7.4.1. | CMD_SET_REQ | 85 |
| 7.4.1.1. | Example 1 | 86 |
| 7.4.1.2. | Example 2 | 86 |
| 7.4.2. | CMD_GET_REQ | 87 |
| 7.4.2.1. | Example 1 | 87 |

| | | |
|-----------|--|------------|
| 7.5. | Manage the device state | 88 |
| 7.5.1. | CMD_GETSTATE_REQ | 88 |
| 7.5.1.1. | Example 1 | 89 |
| 7.5.2. | CMD_RESET_REQ | 89 |
| 7.5.3. | CMD_SLEEP_REQ | 89 |
| 7.5.4. | CMD_SLEEP_IND | 90 |
| 7.5.5. | CMD_FACTORYRESET_REQ | 90 |
| 7.5.6. | CMD_BOOTLOADER_REQ | 91 |
| 7.6. | Run the Bluetooth test modes | 93 |
| 7.6.1. | CMD_DTMSTART_REQ | 93 |
| 7.6.2. | CMD_DTM_REQ | 93 |
| 7.6.2.1. | Example: Transmission, 16 times 0x0F, channel 0 | 95 |
| 7.6.2.2. | Example: Receiver, channel 0 | 96 |
| 7.6.2.3. | Example: Transmission, carrier test, channel 0 | 97 |
| 7.6.2.4. | Example: Set TX power to -4 dBm | 97 |
| 7.6.2.5. | Example: Set PHY to 2 MBit mode | 98 |
| 7.7. | Switching GPIOs by remote control | 99 |
| 7.7.1. | CMD_GPIO_LOCAL_WRITECONFIG_REQ | 99 |
| 7.7.1.1. | Example: Configure two GPIOs to output high | 100 |
| 7.7.2. | CMD_GPIO_LOCAL_READCONFIG_REQ | 102 |
| 7.7.2.1. | Example: Read the current GPIO configuration | 103 |
| 7.7.3. | CMD_GPIO_REMOTE_WRITECONFIG_REQ | 104 |
| 7.7.3.1. | Example: Configure two GPIOs of the connected remote device to output high | 105 |
| 7.7.4. | CMD_GPIO_REMOTE_READCONFIG_REQ | 107 |
| 7.7.4.1. | Example: Read the current GPIO configuration of the connected remote device | 108 |
| 7.7.5. | CMD_GPIO_REMOTE_WRITE_REQ | 109 |
| 7.7.5.1. | Example: Set a remote output GPIO to low | 110 |
| 7.7.6. | CMD_GPIO_REMOTE_READ_REQ | 111 |
| 7.7.6.1. | Example: Read the values of remote GPIOs | 112 |
| 7.7.7. | CMD_GPIO_LOCAL_WRITE_REQ | 113 |
| 7.7.7.1. | Example: Set a local output GPIO to low | 114 |
| 7.7.8. | CMD_GPIO_LOCAL_READ_REQ | 115 |
| 7.7.8.1. | Example: Read the values of local GPIOs | 116 |
| 7.7.9. | CMD_GPIO_REMOTE_WRITECONFIG_IND | 117 |
| 7.7.9.1. | Example: Two GPIOs have been configured by the connected re- mote device to output high | 117 |
| 7.7.10. | CMD_GPIO_REMOTE_WRITE_IND | 118 |
| 7.7.10.1. | Example: GPIOs have been written via remote access | 118 |
| 7.7.11. | CMD_GPIO_LOCAL_WRITE_IND | 119 |
| 7.7.11.1. | Example: GPIOs of the remote device have been written by its local host | 119 |
| 7.8. | Message overview | 120 |
| 8. | UserSettings - Module configuration values | 124 |
| 8.1. | FS_DeviceInfo: Read the chip type and OS version | 124 |
| 8.1.1. | Example 1 | 125 |

| | | |
|---------|---|-----|
| 8.2. | FS_FWVersion: Read the firmware version | 126 |
| 8.2.1. | Example 1 | 126 |
| 8.3. | FS_MAC: Read the MAC address | 127 |
| 8.3.1. | Example 1 | 127 |
| 8.4. | FS_BTMAC: Modify the Bluetooth conform MAC address | 128 |
| 8.4.1. | Example 1 | 128 |
| 8.4.2. | Example 2 | 128 |
| 8.5. | FS_SerialNumber: Read the serial number of the module | 130 |
| 8.5.1. | Example 1 | 130 |
| 8.6. | RF_DeviceName: Modify the device name | 131 |
| 8.6.1. | Example 1 | 131 |
| 8.6.2. | Example 2 | 131 |
| 8.7. | RF_StaticPasskey: Modify the static passkey | 133 |
| 8.7.1. | Example 1 | 133 |
| 8.7.2. | Example 2 | 133 |
| 8.8. | RF_SecFlags: Modify the security settings | 134 |
| 8.8.1. | Example 1 | 135 |
| 8.8.2. | Example 2 | 136 |
| 8.9. | RF_ScanFlags: Modify the scan behavior | 137 |
| 8.9.1. | Example 1 | 137 |
| 8.9.2. | Example 2 | 137 |
| 8.10. | RF_BeaconFlags: Interpret the advertising data | 139 |
| 8.10.1. | Example 1 | 139 |
| 8.10.2. | Example 2 | 140 |
| 8.11. | RF_AdvertisingTimeout: Modify the advertising timeout | 141 |
| 8.11.1. | Example 1 | 141 |
| 8.11.2. | Example 2 | 141 |
| 8.12. | RF_AdvertisingFlags: Configure the advertising packet | 142 |
| 8.12.1. | Example 1 | 142 |
| 8.12.2. | Example 2 | 143 |
| 8.13. | RF_ScanFactor: Modify the scan factor | 144 |
| 8.13.1. | Example 1 | 144 |
| 8.13.2. | Example 2 | 144 |
| 8.14. | RF_ScanTiming: Modify the scan timing | 145 |
| 8.14.1. | Example 1 | 146 |
| 8.14.2. | Example 2 | 146 |
| 8.15. | RF_ConnectionTiming: Modify the connection timing | 147 |
| 8.15.1. | Example 1 | 148 |
| 8.15.2. | Example 2 | 148 |
| 8.16. | RF_TXPower: Modify the output power | 150 |
| 8.16.1. | Example 1 | 150 |
| 8.16.2. | Example 2 | 150 |
| 8.17. | RF_SPPBaseUUID: Configure the SPP base UUID | 152 |
| 8.17.1. | Example 1 | 152 |
| 8.17.2. | Example 2 | 152 |
| 8.18. | RF_SPPServiceUUID: Configure the SPP service UUID | 154 |
| 8.18.1. | Example 1 | 154 |
| 8.18.2. | Example 2 | 154 |

| | | |
|------------|--|------------|
| 8.19. | RF_SPPRXUUID: Configure the SPP RX UUID | 155 |
| 8.19.1. | Example 1 | 155 |
| 8.19.2. | Example 2 | 155 |
| 8.20. | RF_SPPTXUUID: Configure the SPP TX UUID | 156 |
| 8.20.1. | Example 1 | 156 |
| 8.20.2. | Example 2 | 156 |
| 8.21. | RF_Appearance: Configure the appearance of the device | 157 |
| 8.21.1. | Example 1 | 157 |
| 8.21.2. | Example 2 | 157 |
| 8.22. | SPI_ConfigIndex: Modify the SPI mode | 158 |
| 8.22.1. | Example 1 | 158 |
| 8.22.2. | Example 2 | 158 |
| 8.23. | RF_CompanyIdentifier: Configure the company identifier of the device | 160 |
| 8.23.1. | Example 1 | 160 |
| 8.23.2. | Example 2 | 160 |
| 8.24. | CFG_Flags: Configure the module | 161 |
| 8.24.1. | Example 1 | 161 |
| 8.24.2. | Example 2 | 161 |
| 8.25. | DIS_ManufacturerName: Configure the manufacturer name | 163 |
| 8.25.1. | Example 1 | 163 |
| 8.25.2. | Example 2 | 163 |
| 8.26. | DIS_ModelNumber: Configure the model number | 165 |
| 8.26.1. | Example 1 | 165 |
| 8.26.2. | Example 2 | 165 |
| 8.27. | DIS_SerialNumber: Configure the serial number | 166 |
| 8.27.1. | Example 1 | 166 |
| 8.27.2. | Example 2 | 166 |
| 8.28. | DIS_HWVersion: Configure the HW version | 167 |
| 8.28.1. | Example 1 | 167 |
| 8.28.2. | Example 2 | 167 |
| 8.29. | DIS_SWVersion: Configure the SW version | 168 |
| 8.29.1. | Example 1 | 168 |
| 8.29.2. | Example 2 | 168 |
| 8.30. | DIS_Flags: Configure the device information service | 169 |
| 8.30.1. | Example 1 | 169 |
| 8.30.2. | Example 2 | 169 |
| 9. | Timing parameters | 173 |
| 9.1. | Reset and sleep | 173 |
| 9.2. | Bluetooth LE timing parameters | 173 |
| 9.3. | Connection establishment | 173 |
| 9.4. | Connection based data transmission | 174 |
| 9.4.1. | Maximum data throughput | 175 |
| 10. | Remote GPIO control | 177 |
| 10.1. | PWM | 182 |
| 10.2. | Supported GPIO_IDs for remote and local control | 183 |

| | |
|--|------------|
| 11. Customizing the Proteus-III-SPI | 184 |
| 11.1. DIS - Device information service | 184 |
| 11.1.1. Make the module's firmware version visible on the radio | 184 |
| 11.2. UUID | 184 |
| 11.3. Appearance | 185 |
| 12. Custom firmware | 186 |
| 12.1. Custom configuration of standard firmware | 186 |
| 12.2. Customer specific firmware | 186 |
| 12.3. Customer firmware | 186 |
| 12.4. Contact for firmware requests | 187 |
| 13. Firmware updates | 188 |
| 13.1. Firmware flashing using the production interface | 188 |
| 13.2. Firmware update using the Proteus-III-SPI OTA bootloader | 188 |
| 13.2.1. Firmware update steps using the nRF Device Firmware Update app . . | 190 |
| 14. Firmware history | 191 |
| 14.1. Known issues | 192 |
| 15. Design in guide | 193 |
| 15.1. Advice for schematic and layout | 193 |
| 15.2. Dimensioning of the micro strip antenna line | 195 |
| 15.3. Antenna solutions | 196 |
| 15.3.1. Wire antenna | 197 |
| 15.3.2. Chip antenna | 197 |
| 15.3.3. PCB antenna | 197 |
| 15.3.4. Antennas provided by Würth Elektronik eiSos | 198 |
| 15.3.4.1. 2600130021 - Himalia - 2.4 GHz dipole antenna | 198 |
| 16. Reference design | 199 |
| 16.1. Mini EV-Board | 200 |
| 16.2. Radiation characteristic of the module's internal antenna | 202 |
| 16.3. Trace design | 203 |
| 17. Manufacturing information | 207 |
| 17.1. Moisture sensitivity level | 207 |
| 17.2. Soldering | 207 |
| 17.2.1. Reflow soldering | 207 |
| 17.2.2. Cleaning | 209 |
| 17.2.3. Potting and coating | 209 |
| 17.2.4. Other notations | 209 |
| 17.3. ESD handling | 210 |
| 17.4. Safety recommendations | 210 |
| 18. Physical specifications | 212 |
| 18.1. Dimensions | 212 |
| 18.2. Weight | 212 |
| 18.3. Module drawing | 213 |

| | |
|---|------------|
| 18.4. Footprint WE-FP-4+ | 214 |
| 18.5. Antenna free area | 214 |
| 19. Marking | 215 |
| 19.1. Lot number | 215 |
| 19.2. General labeling information | 216 |
| 20. Information for explosion protection | 217 |
| 21. Bluetooth SIG listing/qualification | 218 |
| 22. Regulatory compliance information | 219 |
| 22.1. Important notice EU | 219 |
| 22.2. Important notice FCC | 219 |
| 22.3. Conformity assessment of the final product | 219 |
| 22.4. Exemption clause | 219 |
| 22.5. EU Declaration of conformity | 220 |
| 22.6. UK Declaration of conformity | 221 |
| 22.7. FCC Compliance Statement (US) | 222 |
| 22.7.1. FCC certificate | 222 |
| 22.8. IC Compliance Statement (Canada) | 223 |
| 22.8.1. IC certificate | 223 |
| 22.9. FCC and IC requirements to OEM integrators | 224 |
| 22.9.1. Pre-certified antennas | 225 |
| 22.10. ETA-WPC (India) | 226 |
| 22.10.1. ETA-WPC certificate | 226 |
| 22.11. NCC Compliance Statement (Taiwan) | 228 |
| 22.11.1. NCC certificate | 228 |
| 23. References | 230 |
| 24. Important notes | 231 |
| 24.1. General customer responsibility | 231 |
| 24.2. Customer responsibility related to specific, in particular safety-relevant applications | 231 |
| 24.3. Best care and attention | 231 |
| 24.4. Customer support for product specifications | 231 |
| 24.5. Product improvements | 232 |
| 24.6. Product life cycle | 232 |
| 24.7. Property rights | 232 |
| 24.8. General terms and conditions | 232 |
| 25. Legal notice | 233 |
| 25.1. Exclusion of liability | 233 |
| 25.2. Suitability in customer applications | 233 |
| 25.3. Trademarks | 233 |
| 25.4. Usage restriction | 233 |

| | |
|---|------------|
| 26. License terms | 235 |
| 26.1. Limited license | 235 |
| 26.2. Usage and obligations | 235 |
| 26.3. Ownership | 236 |
| 26.4. Firmware update(s) | 236 |
| 26.5. Disclaimer of warranty | 236 |
| 26.6. Limitation of liability | 237 |
| 26.7. Applicable law and jurisdiction | 237 |
| 26.8. Severability clause | 237 |
| 26.9. Miscellaneous | 237 |
| A. Additional CRC8 Information | 240 |
| A.1. Example CRC8 Implementation | 240 |
| A.2. CRC8 Test Vectors | 240 |
| B. Example codes for host integration | 241 |

Overview of helpful application notes

Application note ANR004 - Peripheral only mode

<http://www.we-online.com/ANR004>

The Bluetooth® LE modules Proteus-I,-II,-III provide the so called "peripheral only mode", that supports a serial cable replacement by offering a transparent UART bridge functionality. This document explains how to set the module into the corresponding operation mode and how to establish a Bluetooth® LE connection to a Bluetooth® LE enabled central device.

Application note ANR006 - Proteus High throughput mode

<http://www.we-online.com/ANR006>

The Proteus-II and Proteus-III provide the so called "high throughput mode". This mode sends several data packets per connection interval to increase the data throughput to a remote Bluetooth® LE device. This application note describes how to set the radio module in this mode, and how to test it in a module-to-module setup. It presents measurements and test scenarios for throughput measurements.

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 ANR009 - Proteus-III Advanced developer guide

<http://www.we-online.com/ANR009>

This advanced developer guide covers the details on the Proteus-III radio module that are required to implement compatible App for smart devices. It covers the documentation on the SPP-like Bluetooth® LE profile, the used protocols and data coding for arbitrary user payload. In addition all information required to develop a custom firmware on the Proteus module hardware platform are provided within.

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 ANR014 - Proteus-I,-II,-III Quickstart

<http://www.we-online.com/ANR014>

This application note describes how to set up a Bluetooth® LE connection between one of a Proteus-I,-II,-III and a Bluetooth® LE enabled device, like a smart phone. Furthermore the data transmission via Bluetooth® LE is presented.

Application note ANR019 - Proteus-III UART vs. SPI - a comparison

<http://www.we-online.com/ANR019>

This application note shows the differences, advantages and disadvantages between the Proteus-III module with UART interface compared to the module variant with SPI slave interface.

Application note ANR020 - Proteus-III Remote GPIO control

<http://www.we-online.com/ANR020>

The Proteus-III module offers six remote controllable GPIOs that can be configured as input, output and PWM. This application note describes that feature which provides the possibility to perform simple and quick hostless operation for simple applications.

Application note ANR026 - Proteus beacons

<http://www.we-online.com/ANR026>

Besides the standard Bluetooth® LE connection based data transmission, it is possible to transmit data via Bluetooth® LE without an active connection in a broadcast message, called "Beacon". This application note describes what beacons are and how they can be used.

Application note ANR027 - Bluetooth listing guide

<http://www.we-online.com/ANR027>

Every product containing Bluetooth® technology needs to be listed at the Bluetooth® SIG (special interest group). This application note explains the steps to be done to gain a Bluetooth® listing 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, that has been adapted by Würth Elektronik eiSos, as well as for firmware written by customer.

1. Introduction

1.1. Operational description

The Proteus-III-SPI module is a radio sub module/device for wireless communication between devices such as control systems, remote controls, sensors et cetera. On the basis of Bluetooth® LE 5.1 [1] it offers a fast and secure data transmission of data packages between two or more parties (point to point topology). A serial peripheral interface (SPI) is available for communication with the host system.

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

Even with it's small dimensions of 8 x 12 mm the Proteus-III-SPI provides a strongly miniaturized integrated PCB antenna. Beside it is possible to connect an external antenna if high radio ranges are of interest.

The main functionality is accessible through pads with edge castellation. This offers easy prototype building as it is suitable for hand soldering. More optional GPIOs without enlarging the size are accessible through land grid pads that can only be connected through reflow process.

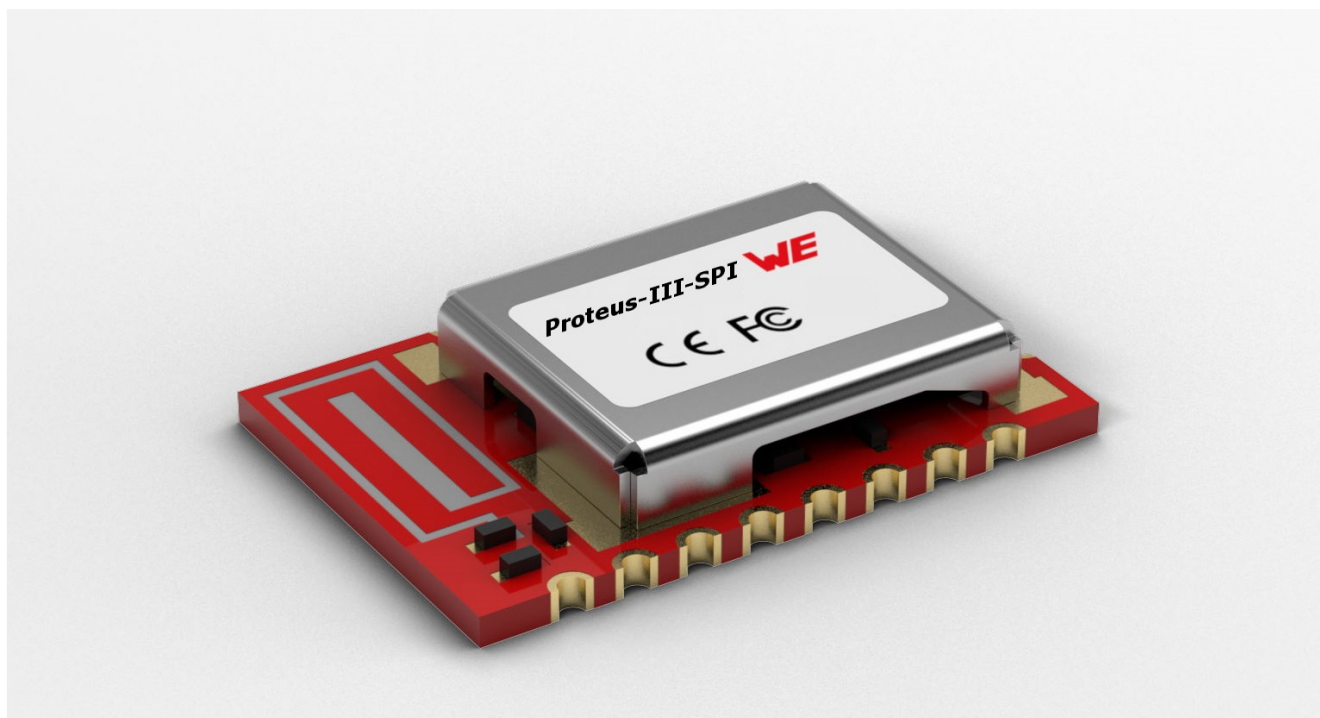


Figure 1: Proteus-III-SPI

1.1.1. Key features

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

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

Remote GPIOs: The Proteus-III-SPI firmware allows to switch free module GPIOs via remote control. More information can be found in chapter 10.

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

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

Fast serial interface: The Proteus-III-SPI offers a SPI-interface and the slave role to communicate with a host (SPI master) using a simple command interface.

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

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

High throughput mode: The Proteus-III-SPI contains the so called "High throughput mode" that allows to send up to 4 data packets per connection interval. This increases significantly the throughput. This mode and its usage is described in application note ANR006 [2].

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

Flexible wired interfacing: The Proteus-III-SPI is equipped with extra pins suited for custom device/sensor connection. With help of these, a tailored firmware can be developed which is optimized to the customer's needs. The pins can be configured to various functions such as UART, SPI, I2C, ADC, PWM, NFC and GPIO.

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

Additional Bluetooth® 5 radio modes: The Proteus-III-SPI provides the advanced radio modes 2 MBit mode for faster data transmission and the LE coded mode, that allows long range data transmissions. For more information, see chapter 5.11.

Long range connect: For backward compatibility reasons, a Bluetooth® LE connection is setup using the legacy 1 MBit radio mode and can then be updated to long range mode. The Proteus-III-SPI allows in addition to setup the connection immediately using the long range mode, such that even connections can be initiated on high distances. More information can be found in chapter 5.12.

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

1.1.2. Connectivity

The Bluetooth® LE standard allows to setup a network with various Bluetooth® LE devices from different manufacturers. To be able to communicate with Proteus-III-SPI devices, the WE SPP-like profile must be known and implemented by all network participants.

The advanced developer guide of Proteus-III-SPI (application note ANR009 [3]) contains the design data of the WE SPP-like profile, to implement it for example in smart phone apps.

1.2. Block diagram

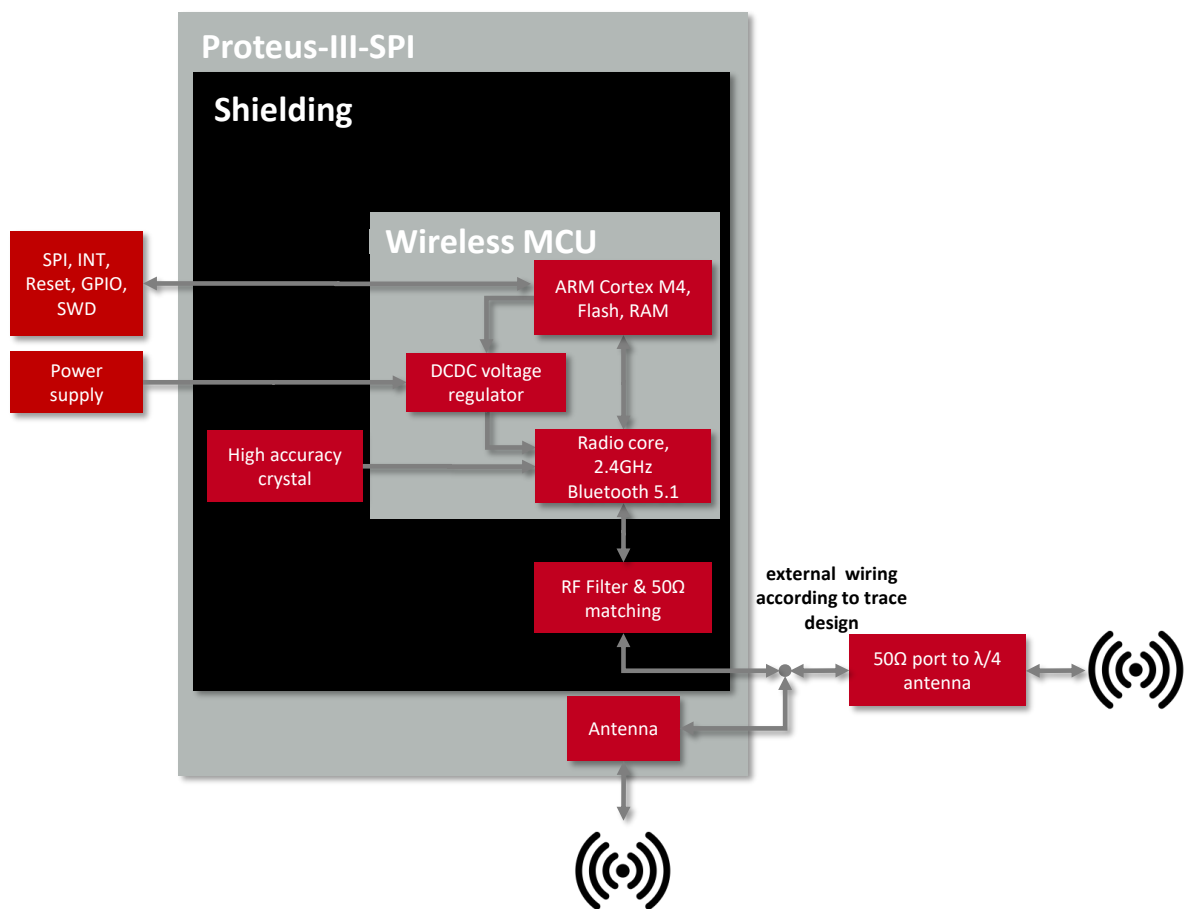


Figure 2: Block diagram of the module

1.3. Ordering information

| WE order code | Description |
|---------------|--|
| 2611011024010 | Proteus-III-SPI Bluetooth® LE Module, Tape & Reel (4-wire SPI + interrupt pin) |
| 2611011024000 | Proteus-III Bluetooth® LE Module, Tape & Reel (UART) |
| 2611119024011 | Proteus-III-SPI mini EV board |
| 2611036024001 | USB Dongle Proteus-III USB radio stick, integrated antenna |

Table 1: Ordering information

2. Electrical specifications

As not otherwise stated measured on the evaluation board Proteus-III-SPI-EV with $T = 25\text{ °C}$, $V_{DDS} = 3\text{ V}$, $f = 2.44\text{GHz}$, internal DC-DC converter in use.

2.1. Recommended operating conditions

| Description | Min. | Typ. | Max. | Unit |
|--|------------------|------|------|------|
| Ambient temperature | -40 | 25 | 85 | °C |
| Supply voltage (V_{DDS}) | 1.8 ¹ | 3 | 3.6 | V |
| Supply rise time (0V to $\geq 1.7\text{V}$) | | | 60 | ms |

Table 2: Recommended operating conditions



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



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

2.2. Absolute maximum ratings

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

Table 3: Absolute maximum ratings

¹Power fail comparator is set to 1.8 V ($\pm 5\%$) to avoid flash fail due to voltage drop.

2.3. Power consumption

2.3.1. Static

| Continuous test mode | Min. | Typ. | Max. | Unit |
|--|------|-------------------|------|------|
| TX current consumption at RF_TXPower = 8 | | 16.4 ¹ | | mA |
| TX current consumption at RF_TXPower = 0 | | 6.4 ¹ | | mA |
| RX current consumption | | 6.25 ¹ | | mA |
| TX current consumption at RF_TXPower = 8 | | 18.9 ² | | mA |
| TX current consumption at RF_TXPower = 0 | | 8 ² | | mA |
| RX current consumption | | 7.7 ² | | mA |
| Sleep (system off mode) | | 0.4 | | μA |

Table 4: Power consumption for 100% transmission/reception



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

¹ Transmitter only with DC/DC converter from nRF52 data sheet, CPU current not included.

² Full module power consumption.

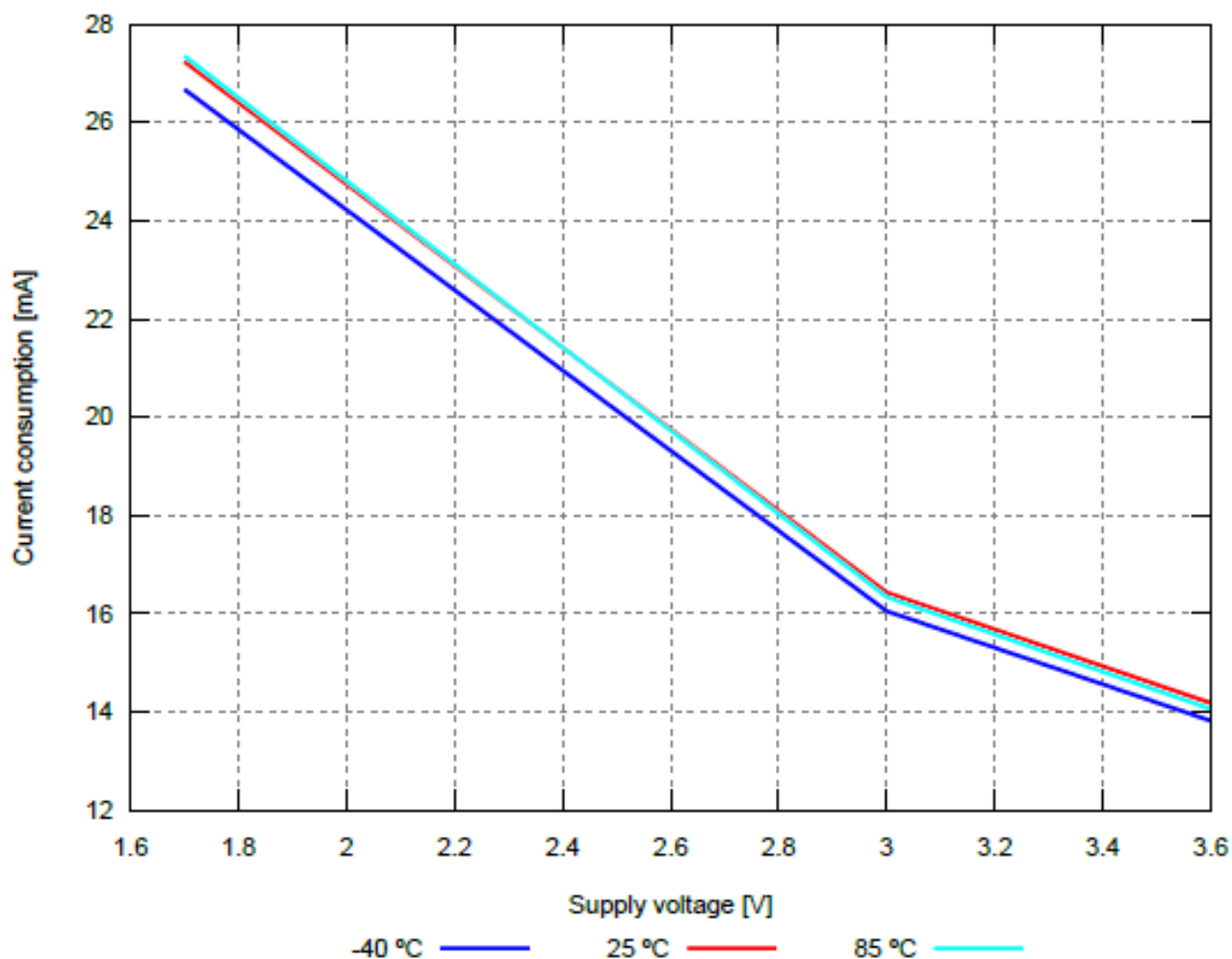


Figure 3: Radio transmitting @ 8 dBm output power, 1 Mbps Bluetooth® LE mode, Clock = HFXO, Regulator = DC/DC (typical values)

2.3.2. Dynamic

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

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

Test setup

| | |
|------------|-----------------|
| Chip | nRF52840 QIAAC0 |
| Softdevice | s140 6.1.0 |
| Voltage | 3.0 V |
| Regulator | DCDC |

BLE event details

| | |
|----------|----------|
| Interval | 45.00 ms |
| Length | 3.61 ms |

Data transmission

| | |
|------------------|--------|
| On air data rate | 1 Mbps |
|------------------|--------|

Current consumption

| | |
|------------------------------|---------------------|
| BLE event total charge | 13.29 μC |
| LF clock calibration current | 1.0 μA |
| Idle current | 3.1 μA |
| Total average current | 299 μA |

ckiges Ausschneiden

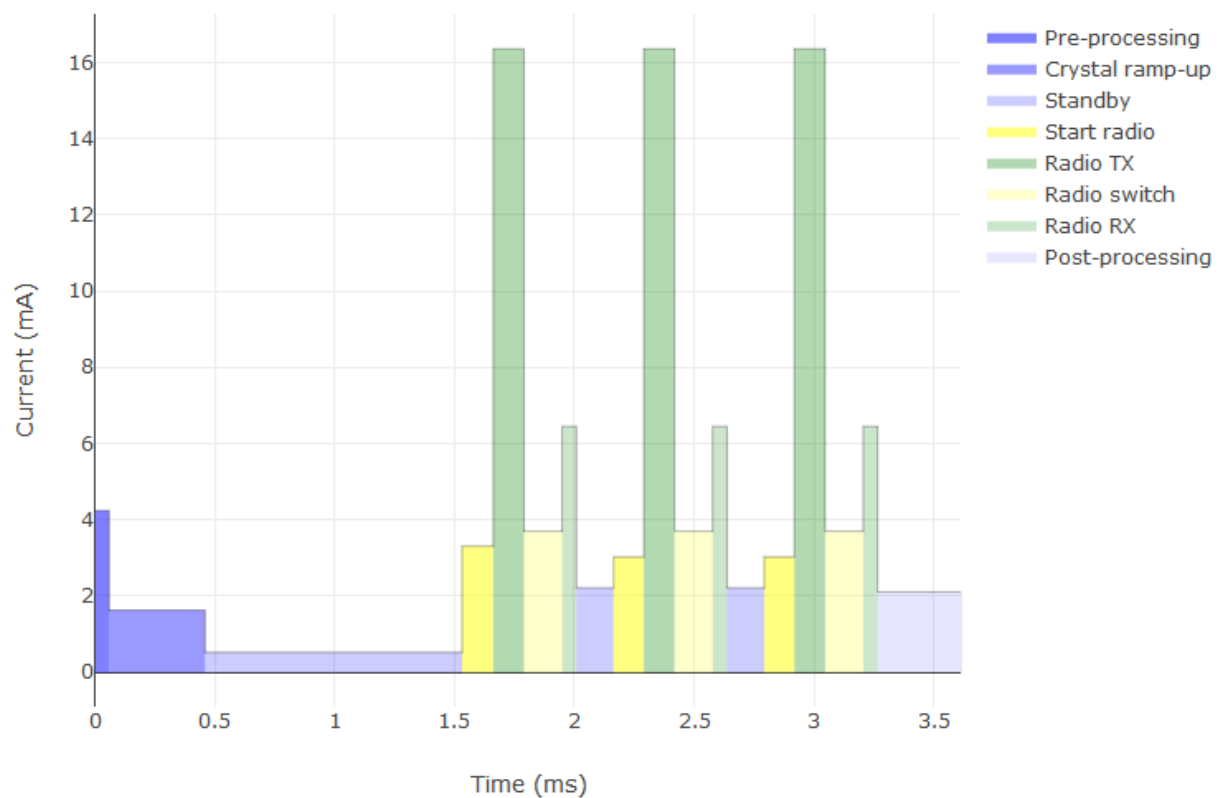


Figure 4: Current consumption calculation in advertising mode with 40 ms advertising interval with 8 dBm output power, UART/SPI disabled

2.4. Radio characteristics

Specifications of timing and RSSI value

| Description | Min. | Typ. | Max. | Unit |
|---|------|------|------|---------|
| RSSI accuracy valid range (± 2 dB) | -90 | | -20 | dBm |
| Enable TX or RX delay | | 140 | | μ s |
| Enable TX or RX delay (fast mode) | | 40 | | μ s |
| Disable TX delay | | 6 | | μ s |
| Disable RX delay | | 0 | | μ s |

Table 5: Timing and RSSI

| Description | Typ. | Unit |
|--|------|------|
| Output power ($R_{F_TXPower} = 8$, conducted) | +6 | dBm |
| Output power integrated antenna ($R_{F_TXPower} = 8$, e.r.p.) | +4 | dBm |
| Input sensitivity conducted (BER=1E-3, 1Mbps) | -92 | dBm |
| Input sensitivity integrated antenna (BER=1E-3, 1Mbps) | -90 | dBm |

Table 6: Transmit and receive power

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

| Description | Typ. | Unit |
|-------------------------|------|------|
| 1 Mbit Phy (PER 1%) | -90 | dBm |
| 2 Mbit Phy (PER 1%) | -87 | dBm |
| LE coded S=2 (PER 1%) | -94 | dBm |
| LE coded S=8 (PER 1%) | -97 | dBm |

Table 7: Sensitivity at different data rates

2.5. Pin characteristics

Specifications from nRF52 data sheet

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

Table 8: Pin characteristics

3. Pinout

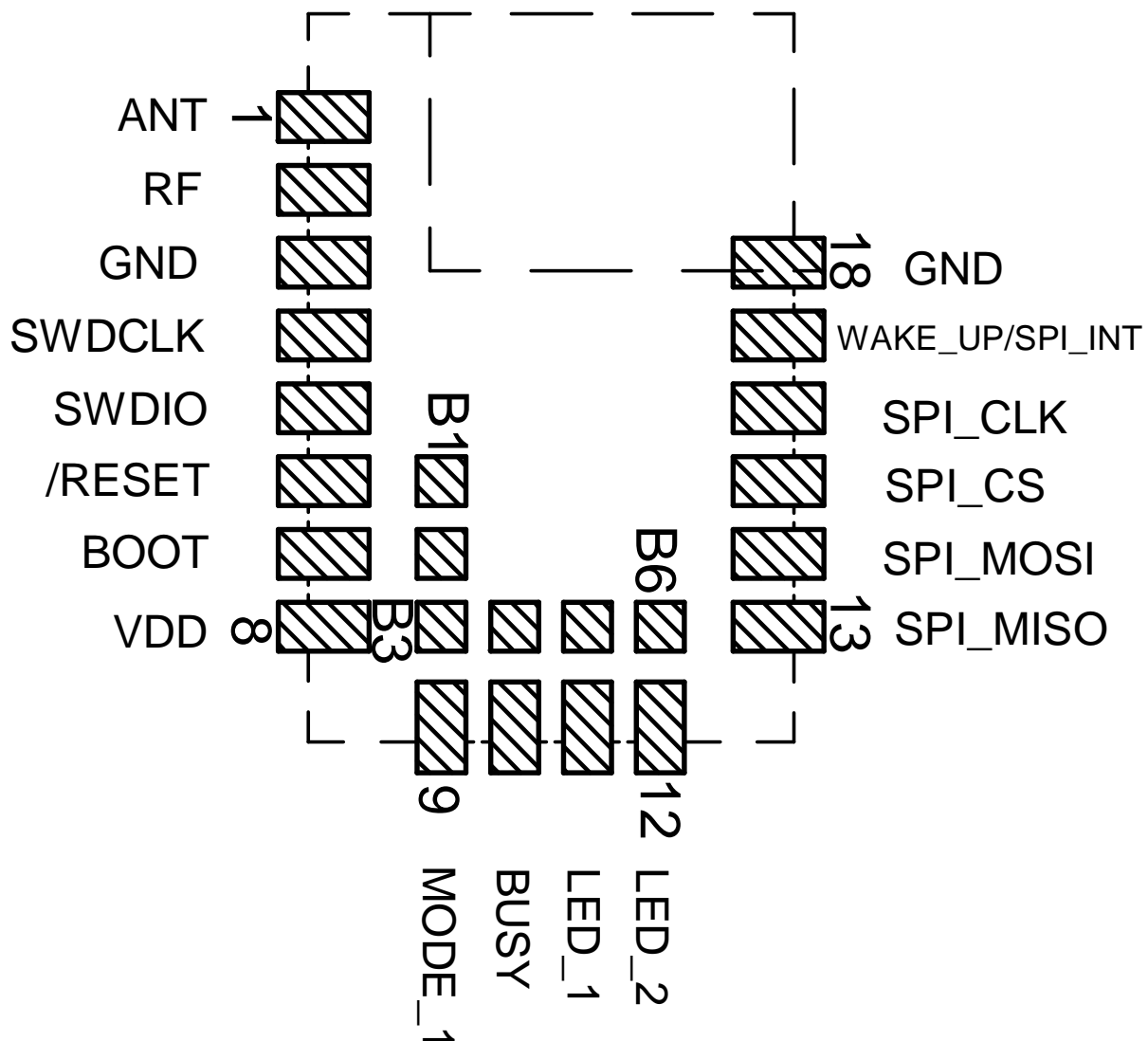


Figure 5: Pinout (top view)



The main functionality is accessible through pad 1 - 18 with edge castellation. This offers easy prototype building as it is suitable for hand soldering. More optional GPIOs without enlarging the size are accessible through the land grid pads B1 - B6 that can only be connected through re-flow process.

| No | μC Pin | Designation | I/O | Description |
|----|------------------------|-----------------|--------|---|
| 1 | | <i>ANT</i> | I/O | RF connection to PCB antenna. (see section 4.2) |
| 2 | | <i>RF</i> | I/O | 50 Ω RF connection through radio front end to transceiver part of chipset. (see section 4.2) |
| 3 | | <i>GND</i> | Supply | Ground |
| 4 | | <i>SWDCLK</i> | Input | Serial wire clock (SWD Interface). Uses internal pull down resistor. Do not connect if not needed. |
| 5 | | <i>SWDIO</i> | Input | Serial wire input/output (SWD Interface). Uses internal pull up resistor. Do not connect if not needed. |
| 6 | P0.18 | <i>/RESET</i> | Input | Reset pin. A low signal resets the module. Uses internal pull up resistor. |
| 7 | P0.02 | <i>BOOT</i> | Input | Boot pin. A low signal during and short after reset starts the module in OTA bootloader mode. Uses internal pull up resistor ¹ . Do not connect if not needed. |
| 8 | | <i>VDD</i> | Supply | Supply voltage |
| 9 | P0.19 | <i>MODE_1</i> | Input | Operation mode pin with internal pull down resistor ¹ during start-up. Do not connect. |
| 10 | P0.22 | <i>BUSY</i> | Output | Do not connect. |
| 11 | P0.00/XL1 ² | <i>LED_1</i> | Output | Indicates the module state (active high). Do not connect if not needed. |
| 12 | P0.01/XL2 ² | <i>LED_2</i> | Output | Indicates the module state (active high). Do not connect if not needed. |
| 13 | P1.08 | <i>SPI_MISO</i> | Output | SPI Master in, slave out signal. In sleep mode, this pin is high. |
| 14 | P1.09 | <i>SPI_MOSI</i> | Input | SPI Master out, slave in signal. |
| 15 | P0.11 | <i>SPI_CS</i> | Input | SPI chip select signal. |

Table 9: Pinout, first part

¹Internal pull ups or pull downs are configured at startup by the firmware installed in the SoC. The pull up on the */RESET* pin cannot be disabled by firmware.

²Pins available to connect an external crystal in custom firmware. The standard firmware of Proteus-III-SPI does not implement this function.

³NFC pins available for NFC function in custom firmware. The standard firmware of Proteus-III-SPI does not implement this function.

| No | µC Pin | Designation | I/O | Description |
|----|-------------------------|------------------------|----------------|---|
| 16 | P0.12 | <i>SPI_CLK</i> | Input | SPI clock signal. |
| 17 | P0.03 | <i>WAKE_UP/SPI_INT</i> | Input / Output | Shared pin for wake-up from sleep mode, and SPI data indication. See chapters 5.4 and 6.1.2 for more details. |
| 18 | | <i>GND</i> | Supply | Ground |
| B1 | P0.09/NFC1 ³ | <i>B1</i> | GPIO | Pin for remote GPIO access. Do not connect, if not needed. |
| B2 | P0.10/NFC2 ³ | <i>B2</i> | GPIO | Pin for remote GPIO access. Do not connect, if not needed. |
| B3 | P0.23 | <i>B3</i> | GPIO | Pin for remote GPIO access. Do not connect, if not needed. |
| B4 | P1.00 | <i>B4</i> | GPIO | Pin for remote GPIO access. Do not connect, if not needed. |
| B5 | P0.21 | <i>B5</i> | GPIO | Pin for remote GPIO access. Do not connect, if not needed. |
| B6 | P0.07 | <i>B6</i> | GPIO | Pin for remote GPIO access. Do not connect, if not needed. |

Table 10: Pinout, second part

4. Quick start

4.1. Minimal pin connections

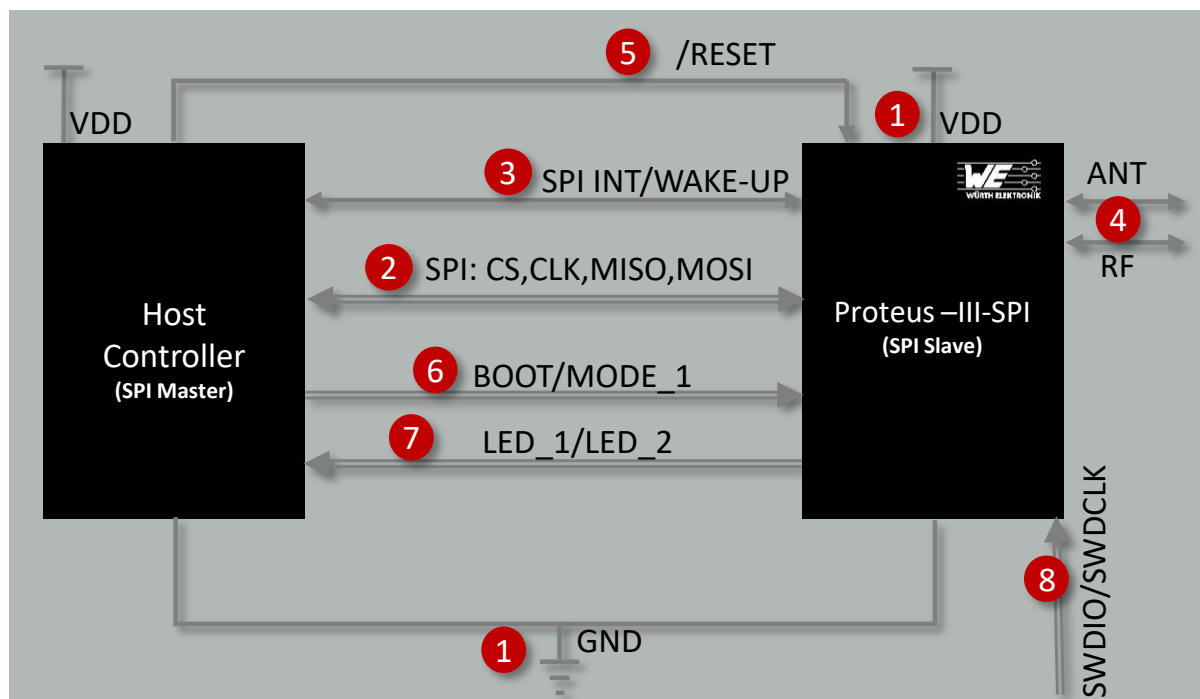


Figure 6: Minimal pin connections

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

1. Supply voltage and ground
Connect the *VDD* and *GND* pins to supply the radio module with power.
2. SPI interface to the host
Connect the SPI pins *SPI_CS*, *SPI_CLK*, *SPI_MOSI* and *SPI_MISO* to the host to control the module via host.
3. SPI interrupt and wake-up from sleep
Connect the shared pin *SPI_INT / WAKE_UP* to the host.
4. Antenna connection
The antenna configuration must be done. See chapter 4.2 for details.
5. Reset
Connect the */RESET* pin to the host to allow a hard reset of the module.
6. (Optional) FOTA and mode selection

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

7. (Optional) Status indication

Connect the *LED_1* and *LED_2* pins to the host controller to allow easy indication of the status.

8. (Optional) Flash and debug interface

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



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

4.2. Antenna connection

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

4.2.1. On-board PCB antenna

The Proteus-III-SPI 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-III-SPI. In this configuration, the module does not require any additional RF circuitry. For US and Canada, please refer to the trace design in chapter 16.3.

4.2.2. External antenna

For applications that use an external antenna, the Proteus-III-SPI 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 16 for further information.



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.

4.3. Power up

After powering the module the */RESET* pin shall be hold for another Δt of 1ms after the *VDD* is stable to ensure a safe start-up. The module will send a *CMD_GETSTATE_CNF* (0x02 41 02 00 01 01 41) to indicate "ready for operation" after the */RESET* pin was released.



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

4.4. Test the SPI interface

To test the SPI transaction in command mode, perform a pin reset by pulling the */RESET* pin down for at least 1 ms and release it again. To do so, the reset button can be simply pressed, in case a Proteus-III-SPI mini evaluation board is used. After this reset, the Proteus-III-SPI restarts and wants to transmit a *CMD_GETSTATE_CNF* (0x02 41 02 00 01 01 41) message to the connected host. Thus, the Proteus-III-SPI sets the interrupt pin *SPI_INT* to its "active level" to signalize that the module has data available to be fetched by the host.



The default "active level" of the *SPI_INT* is HIGH. It can be configured using the user setting *SPI_ConfigIndex*.

Next, the host controller must read the first 4 bytes via SPI from the radio module, interpret the length field of the command frame and read the remaining bytes (payload and check sum). For a detailed description, please see figure 11 in the chapter *Host connection*.

If the transfer of SPI data from radio module to the host was been run successfully, the direction host to radio module can be tested. Thus, sent for example a *CMD_GETSTATE_REQ* (0x02 01 00 00 03) command to the module, and check if it responds with a *CMD_GETSTATE_CNF* (0x02 41 02 00 01 01 41) message. To do so, sent the whole frame 0x02 01 00 00 03 in one SPI transaction to the radio module, as shown in figure 10 in the chapter *Host connection*. If this has been successful, the radio module responds again with a *CMD_GETSTATE_CNF* message, as described previously (by pulling the *SPI_INT* to its active state, reading 4 bytes, interpreting the length field, reading the remaining bytes).

4.5. Quickstart example

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



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

Connect the two modules to a microcontroller with SPI interface (see chapter 6).



To reproduce the following sequence, note that the MAC address FS_BTMAC of every module is different. Thus it has to be replaced in the example commands below. In addition, the checksum has to be adjusted, when adapting any command. The command structure and checksum calculation is described in chapter 7.

Connection setup and first data transmission

1. Power-up the modules and make their SPIs accessible by the host(s). After the power-up or after reset the following sequence is available in the radio module to be fetched by the host.

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

2. Request the FS_BTMAC of both modules.

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

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

| Info | Module A | Module B |
|---|---|---|
| ⇒ Request CMD_CONNECT_REQ with FS_BTMAC of module B | 02 06 06 00 11 00 00 DA 18 00 D1 | |
| ⇐ Response CMD_CONNECT_CNF: Request understood, try to connect now | 02 46 01 00 00 45 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 86 07 00 00 11 00 00 DA 18 00 50 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 86 07 00 00 55 00 00 DA 18 00 14 |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | 02 C6 08 00 00 11 00 00 DA 18 00 F3 E3 | |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | | 02 C6 08 00 00 55 00 00 DA 18 00 F3 A7 |

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



The RSSI values will be different in your tests.

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

5. Reply with "EFGH" to module B.

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

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

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

5. Functional description

5.1. Operation modes

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

- The **command mode**, where the Proteus-III-SPI can be controlled by the host controller via commands. The command mode allows to use all central and peripheral function of the radio module. The functions of the radio module, like data transmission or configuration tasks, can be triggered by predefined commands (see chapter 7) that are sent as telegrams over the SPI interface.

5.2. Radio module states

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

When the Proteus-III-SPI is powered up, it starts in `ACTION_IDLE` state. In this state the module advertises (Bluetooth® LE role "peripheral"), such that other devices in range can detect it and connect to it.

The `ACTION_IDLE` state also allows to switch to `ACTION_SCANNING` state, where the module stops advertising and scans for other advertising Bluetooth® LE devices in range.

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

The `ACTION_CONNECTED` state can be entered, either by getting a connection request from another Bluetooth® LE device, or by setting up a connection itself. In this case, it stops advertising and data can be transmitted and received to/from the connected Bluetooth® LE device. This state remains active as long as the module does not disconnect itself, and no disconnection request from the connected remote device is received.

When disconnecting, the module goes to `ACTION_IDLE` state and starts advertising again, to be ready for the next connection setup.

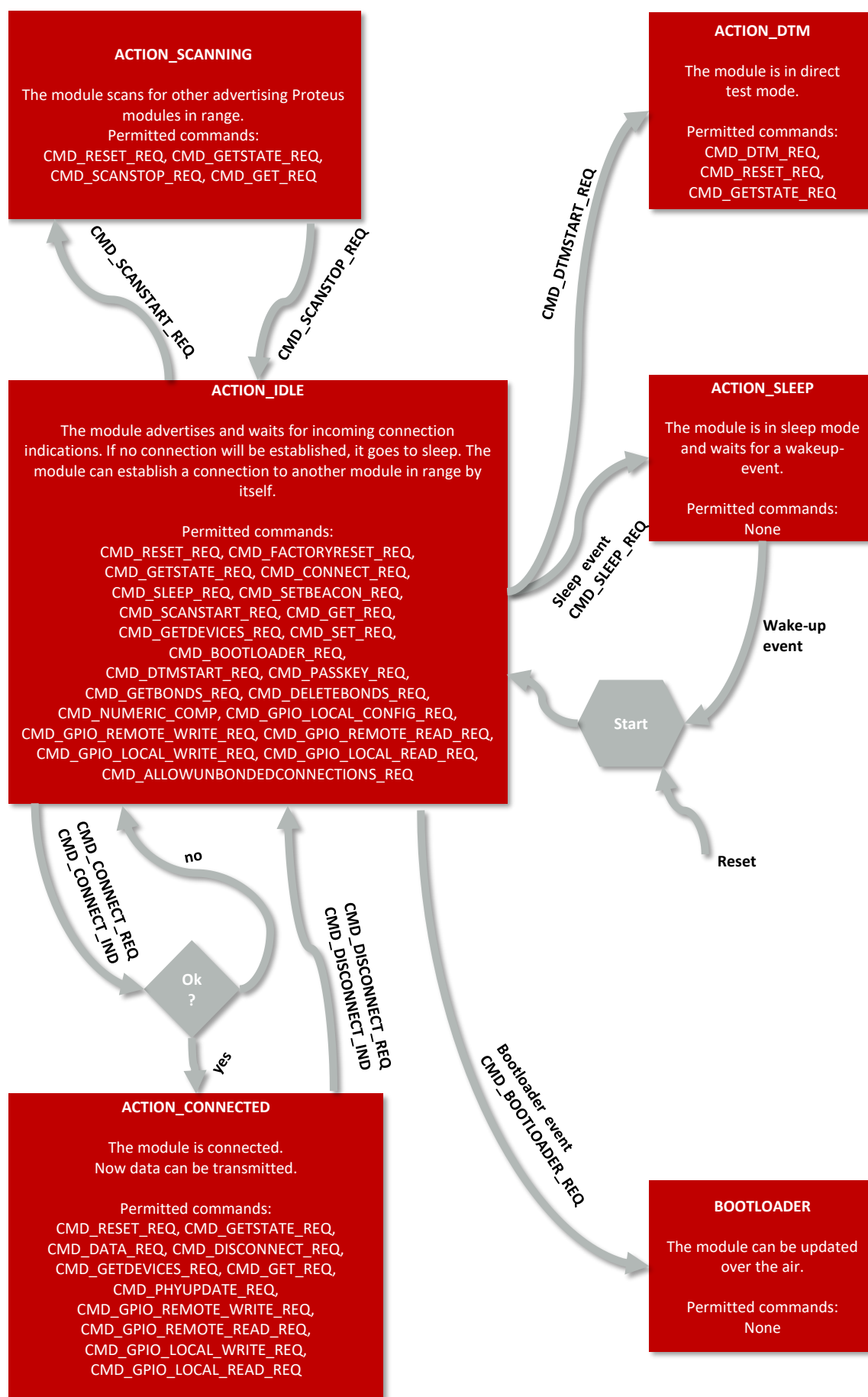


Figure 7: State overview

5.3. State indication using the LED pins

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

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

Table 17: LED behavior of the Proteus-III-SPI

5.4. Sleep mode

Especially for battery-powered devices the ACTION_SLEEP mode (system-off mode) supports very low power consumption. It can be entered by sending the command CMD_SLEEP_REQ to the module. As response, the module will send a CMD_SLEEP_CNF and then enter the ACTION_SLEEP mode.

In ACTION_SLEEP mode, the GPIO pins *B1-B6* are set to input during the sleep period. Furthermore the SPI is disabled. Thus the module will not receive or transmit any data. To prevent leakage current, the host shall pull the SPI pins to their default voltage levels and the shared *WAKE_UP/SPI_INT* pin to its "inactive level".

As the *WAKE_UP* pin and *SPI_INT* pin are shared, the radio module sets this pin to input function when it goes to sleep mode to detect a wake-up trigger from the host. As soon as this happened, the radio module can be woken up by setting the voltage level of the *WAKE_UP* pin to its "active level" for at least 5 ms before setting it back to its "inactive level". The "active level" is defined by means of the user setting SPI_ConfigIndex (the default value is "active high").

The module then restarts completely, so that all volatile settings are set to default. In this case the *WAKE_UP/SPI_INT* pin of the radio module is set to output function to signalize to the host if there is data on the radio module that is supposed to be sent via SPI to the host controller. A CMD_GETSTATE_CNF will be sent when the module is ready for operation again.

5.5. Identification of a Proteus-III-SPI device on the radio

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

In ACTION_SCANNING state a module listens to the data packets of all advertising devices in range and stores their FS_BTMAC to an internal data base. With help of this FS_BTMAC a connection to the corresponding device can then be established using the CMD_CONNECT_REQ command. To simplify the identification of Proteus-III-SPI devices on the RF-interface a short user-defined name (see RF_DeviceName) can be given to the module, which is also part of the advertising packet.



The FS_BTMAC consists of the Würth Elektronik eiSos MAC ID 0x0018DA followed by the module's serial number FS_SerialNumber.

5.6. Connection based data transmission, with or without security

In the Bluetooth® LE standard the data transmission typically is connection based. A connection between two devices can be secured or unsecured (default configuration). In any case, each data packet transmitted is acknowledged on the link layer, such that it is resent as long as it has not been received from the receiver. The following lines describe how to run the connection setup and data transmission using the Proteus-III-SPI.

If module A is supposed to setup a connection with module B, the host of module A must send the command CMD_CONNECT_REQ including the FS_BTMAC of module B to module A. If the FS_BTMAC of module B is unknown, a radio scan shall be run before by module A to discover all available Bluetooth® LE enabled devices in range.

After sending the command CMD_CONNECT_REQ, the module answers with a CMD_CONNECT_CNF to signal that the request has been understood and the module tries to establish the connection. If module B cannot be found on the air within a timeout, module A outputs a CMD_CONNECT_IND with "failed" as status. Otherwise, as soon as the physical connection has been set up successfully, module A and B output a CMD_CONNECT_IND message with the status of the successful connection and LED_1 turns on.

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

After the physical connection has been setup successfully the modules exchange their services (WE SPP-like). As soon as this has finished successfully, a CMD_CHANNELOPEN_RSP message is sent to the host indicating that the connection is ready for data transmission. In addition, LED_2 turns on.

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

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

As soon as one module closes the connection using a CMD_DISCONNECT_REQ, both devices will

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

An example of setting up an unsecured connection is shown in chapter 4.5. How to setup a secured connection is shown in the preceding chapters. See also the application note "ANR009 advanced developer guide" to get detailed information about the connection setup with foreign devices.

5.6.1. Further information for a secure connection setup

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



When connecting from a Proteus-III-SPI to another Proteus radio module, you shall not use different security modes.



To get further information about the secured connection setup, when using a foreign device (i.e. mobile phone with a custom APP), please refer to the Proteus-III-SPI application note "ANR009 advanced developer guide".

5.6.1.1. Just works mode

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

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

1. Power-up the modules and make their SPIs accessible by the host(s). After the power-up or after reset the following sequence is available in the radio module to be fetched by the host.

| Info | Module A | Module B |
|---|----------------------|----------------------|
| ⇐ Response <code>CMD_GETSTATE_CNF</code> : Module A started in <code>ACTION_IDLE</code> mode. | 02 41 02 00 01 01 41 | |
| ⇐ Response <code>CMD_GETSTATE_CNF</code> : Module B started in <code>ACTION_IDLE</code> mode. | | 02 41 02 00 01 01 41 |

2. Request the FS_BTMAC of both modules.

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

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

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

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

| Info | Module A | Module B |
|--|--|--|
| ⇒ Request CMD_CONNECT_REQ with FS_BTMAC of module B | 02 06 06 00 11 00 00 DA 18 00 D1 | |
| ⇐ Response CMD_CONNECT_CNF: Request understood, try to connect now | 02 46 01 00 00 45 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 86 07 00 00 11 00 00 DA 18 00 50 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 86 07 00 00 55 00 00 DA 18 00 14 |
| ⇐ Indication CMD_SECURITY_IND, status 0x02 (encrypted link, pairing, no bonding), with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 88 07 00 02 11 00 00 DA 18 00 5C | |

| | | |
|---|--|---|
| ⇐ Indication CMD_SECURITY_IND, status 0x02 (encrypted link, pairing, no bonding), with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 88 07 00 02 55 00 00 DA 18 00 18 |
|---|--|---|

| Info | Module A | Module B |
|---|--|--|
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | 02 C6 08 00 00 11 00 00 DA 18 00 F3 EC | |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | | 02 C6 08 00 00 55 00 00 DA 18 00 F3 A8 |

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



The RSSI values will be different in your tests.

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

6. Reply with "EFGH" to module B.

| Info | Module A | Module B |
|--|---|----------|
| ⇒ Request CMD_DATA_REQ: Send " EFGH " to module B | 02 04 04 00 45 46 47 48 0E | |
| ⇐ Response CMD_DATA_CNF: Request received, send data now | 02 44 01 00 00 47 | |

| | | |
|--|-------------------|---|
| ⇐ Indication CMD_DATA_IND: Received string "EFGH" from FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm) | | 02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7 |
| ⇐ Response CMD_TXCOMPLETE_RSP: Data transmitted successfully | 02 C4 01 00 00 C7 | |

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

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

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

5.6.1.2. StaticPaskey mode

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

Example: Secured connection with security method "StaticPaskey"

1. Power-up the modules and make their SPIs accessible by the host(s). After the power-up or after reset the following sequence is available in the radio module to be fetched by the host.

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

2. Request the FS_BTMAC of both modules.

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

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

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

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

| Info | Module A | Module B |
|--|--|--|
| ⇒ Request CMD_CONNECT_REQ with FS_BTMAC of module B | 02 06 06 00 11 00 00 DA 18 00 D1 | |
| ⇐ Response CMD_CONNECT_CNF: Request understood, try to connect now | 02 46 01 00 00 45 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 86 07 00 00 11 00 00 DA 18 00 50 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 86 07 00 00 55 00 00 DA 18 00 14 |
| ⇐ Indication CMD_PASSKEY_IND to ask for the pass key | 02 8D 07 00 00 11 00 00 DA 18 00 5B | |
| ⇒ Answer with the CMD_PASSKEY_REQ and the pass key "123123" | 02 0D 06 00 31 32 33 31 32 33 09 | |
| ⇐ Response CMD_PASSKEY_CNF: Pass key ok | 02 4D 01 00 00 4E | |

| | | |
|--|--|--|
| ⇐ Indication CMD_SECURITY_IND, status 0x02 (encrypted link, pairing, no bonding), with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 88 07 00 02 11 00 00 DA 18 00 5C | |
| ⇐ Indication CMD_SECURITY_IND, status 0x02 (encrypted link, pairing, no bonding), with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 88 07 00 02 55 00 00 DA 18 00 18 |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | 02 C6 08 00 00 11 00 00 DA 18 00 F3 EC | |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | | 02 C6 08 00 00 55 00 00 DA 18 00 F3 A8 |

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



The RSSI values will be different in your tests.

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

6. Reply with "EFGH" to module B.

| Info | Module A | Module B |
|--|---|----------|
| ⇒ Request CMD_DATA_REQ: Send " EFGH " to module B | 02 04 04 00 45 46 47 48 0E | |
| ⇐ Response CMD_DATA_CNF: Request received, send data now | 02 44 01 00 00 47 | |

| | | |
|--|-------------------|---|
| ⇐ Indication CMD_DATA_IND: Received string "EFGH" from FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm) | | 02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7 |
| ⇐ Response CMD_TXCOMPLETE_RSP: Data transmitted successfully | 02 C4 01 00 00 C7 | |

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

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

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

5.6.1.3. LescPasskey mode

In case of the "LescPasskey" mode, a pass key has to be entered at the central side that has to match the pass key of the peripheral. Here the Proteus-III-SPI uses a pass key in the peripheral role that is generated by the LESC method (low energy secure connection) when a connection is initiated. When using this method, the peripheral device outputs the new generated pass key (see CMD_DISPLAY_PASSKEY_IND) when a connection setup has been initiated. At the same time the central device requests its host to enter this pass key (see CMD_PASSKEY_IND). In this case the pass key of the peripheral has to be entered on central side using the CMD_PASSKEY_REQ command. If the entered pass key is correct, the channel will be opened for data transmission. Otherwise, the connection will be rejected.

Example: Secured connection with security method "LescPasskey"

1. Power-up the modules and make their SPIs accessible by the host(s). After the power-up or after reset the following sequence is available in the radio module to be fetched by the host.

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

2. Request the FS_BTMAC of both modules.

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

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

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

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

| Info | Module A | Module B |
|--|--|---|
| ⇒ Request CMD_CONNECT_REQ with FS_BTMAC of module B | 02 06 06 00 11 00 00 DA 18 00 D1 | |
| ⇐ Response CMD_CONNECT_CNF: Request understood, try to connect now | 02 46 01 00 00 45 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 86 07 00 00 11 00 00 DA 18 00 50 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 86 07 00 00 55 00 00 DA 18 00 14 |
| ⇐ Indication CMD_DISPLAY_PASKEY_IND to display the new generated pass key "429943" | | 02 A4 0D 00 00 55 00 00 DA 18 00 34 32 39 39 34 33 3D |
| ⇐ Indication CMD_PASKEY_IND to ask for the pass key | 02 8D 07 00 00 11 00 00 DA 18 00 5B | |

| | | |
|---|--|--|
| ⇒ Answer with the CMD_PASSKEY_REQ and the pass key "429943" | 02 0D 06 00 34 32 39 39 34 33 08 | |
| ⇐ Response CMD_PASSKEY_CNF: Pass key ok | 02 4D 01 00 00 4E | |
| ⇐ Indication CMD_SECURITY_IND, status 0x02 (encrypted link, pairing, no bonding), with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 88 07 00 02 11 00 00 DA 18 00 5C | |
| ⇐ Indication CMD_SECURITY_IND, status 0x02 (encrypted link, pairing, no bonding), with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 88 07 00 02 55 00 00 DA 18 00 18 |

| Info | Module A | Module B |
|---|---|---|
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | 02 C6 08 00 00 11 00 00 DA 18 00 F3 EC | |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | | 02 C6 08 00 00 55 00 00 DA 18 00 F3 A8 |

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



The RSSI values will be different in your tests.

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

6. Reply with "EFGH" to module B.

| Info | Module A | Module B |
|------|----------|----------|
|------|----------|----------|

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

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

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

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

5.6.1.4. LescNumComp mode

In case of the "LescNumComp" mode, a pass key is displayed on peripheral and central side. Both, the central and peripheral device must confirm that both keys are equal. Here the Proteus-III-SPI uses a pass key that is generated by the LESC method (low energy secure connection) when a connection is initiated. When using this method, the peripheral and central device output the new generated pass key (see CMD_DISPLAY_PASKEY_IND) when a connection setup has been initiated. Both, the central and peripheral device request their hosts to confirm that both keys coincide (see CMD_NUMERIC_COMP_REQ). If both devices confirmed the key, the channel will be opened for data transmission. Otherwise, the connection will be rejected.

Example: Secured connection with security method "LescNumComp"

1. Power-up the modules and make their SPIs accessible by the host(s). After the power-up or after reset the following sequence is available in the radio module to be fetched by the host.

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

| | | |
|--|--|----------------------|
| ⇒ Response CMD_GETSTATE_CNF: Module B started in ACTION_IDLE mode. | | 02 41 02 00 01 01 41 |
|--|--|----------------------|

2. Request the FS_BTMAC of both modules.

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

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

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

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

| Info | Module A | Module B |
|--|--|--|
| ⇒ Request CMD_CONNECT_REQ with FS_BTMAC of module B | 02 06 06 00 11 00 00 DA 18 00 D1 | |
| ⇒ Response CMD_CONNECT_CNF: Request understood, try to connect now | 02 46 01 00 00 45 | |
| ⇒ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 86 07 00 00 11 00 00 DA 18 00 50 | |
| ⇒ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 86 07 00 00 55 00 00 DA 18 00 14 |

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

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



The RSSI values will be different in your tests.

| Info | Module A | Module B |
|---|----------|-------------------------------|
| ⇒ Request CMD_DATA_REQ: Send "ABCD" to module A | | 02 04 04 00 41 42 43 44 06 |

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

6. Reply with "EFGH" to module B.

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

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

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

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

5.6.1.5. Bonding

The SECFLAGS_BONDING_ENABLE flag in the RF_SecFlags user setting allows enabling the bonding feature. This feature stores the keys that are exchanged during the pairing phase in a connection setup. With this, subsequent connections to bonded devices can be established without renegotiation. Bonding data of up to 32 devices will be stored in the flash.

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

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

1. Power-up the modules and make their SPIs accessible by the host(s). After the power-up or after reset the following sequence is available in the radio module to be fetched by the host.

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

2. Request the FS_BTMAC of both modules.

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

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

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

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

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

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

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

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

| Info | Module A | Module B |
|---|-------------------------------------|----------|
| ⇒ Request CMD_CONNECT_REQ with FS_BTMAC of module B | 02 06 06 00 11 00 00 DA 18 00 D1 | |

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

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

5.7. Unidirectional connectionless data transmission using Beacons

Besides the connection-based type of data transmission described in the previous section, there exists a second method that uses so called Beacons. In this case, up to 19 bytes of user data can be placed in the Bluetooth® LE scan response packet, which is broadcasted frequently without acknowledgement and without security during advertising.

If a Proteus-III-SPI is supposed to broadcast some user data the command `CMD_SETBEACON_REQ` places the payload data, that is marked as "manufacturer data" combined with the Würth Elektronik eiSos company identifier 0x031A, in the scan response packet (see also application note ANR026).

If a second Proteus-III-SPI, which has its Beacon-function enabled (see `RF_BeaconFlags`), is in the operating state `ACTION_SCANNING`, the scan response packet and the containing beacon data is received. Filtering the beacon messages can be enabled or disabled using the user setting `RF_BeaconFlags`.

After the reception of the beacon data, it is output to the connected host using a `CMD_BEACON_IND` message.

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

This method is very suitable for sensor networks, which frequently send their data to data collectors. Especially when using a slow `RF_ScanTiming` mode, data can be transmitted in very energy efficient way. Please check the settings `RF_AdvertisingTimeout` and the advertising interval in `RF_ScanTiming` to configure the frequency and interval of transmissions which will have an influence on the current consumption of the module.



Please note that the transmission and reception of Beacons are not supported, if `RF_AdvertisingFlags` is 2 or "Long range connection mode" bit is set in `CFG_Flags`.

| Info | Module A | Module B |
|--|----------------------------------|----------------------|
| ⇐ Reset both modules using <code>/RESET</code> pin, <code>CMD_GETSTATE_CNF</code> | 02 41 02 00 01 01 41 | 02 41 02 00 01 01 41 |
| ⇒ Configure <code>RF_BeaconFlags</code> using <code>CMD_SET_REQ</code> to "beacon rx enabled, no filter" | | 02 11 02 00 0E 01 1E |
| ⇐ <code>CMD_SET_CNF</code> from module B | | 02 51 01 00 00 52 |
| ⇐ Module B reset such that the change in the user setting takes effect (<code>CMD_GETSTATE_CNF</code>) | | 02 41 02 00 01 01 41 |
| ⇒ Activate scanning on module B | | 02 09 00 00 0B |
| ⇐ Response <code>CMD_SCANSTART_CNF</code> | | 02 49 01 00 00 4A |
| ⇒ <code>CMD_SETBEACON_REQ</code> , content "Hallo" | 02 0C 05 00 48 61 6C 6C 6F 4D | |
| ⇐ <code>CMD_SETBEACON_CNF</code> | 02 4C 01 00 00 4F | |

| | | |
|--|----------------------|--|
| ⇐ Receiving multiple CMD_BEACON_IND | | 02 8C 0C 00 02 00 00 DA 18 00 B5 48 61 6C 6C 6F B1 02 8C 0C 00 02 00 00 DA 18 00 B1 48 61 6C 6C 6F B5 |
| : | : | : |
| ⇒ Deactivate scanning on module B, CMD_SCANSTOP_REQ | | 02 0A 00 00 08 |
| ⇐ Response CMD_SCANSTOP_CNF | | 02 4A 01 00 00 49 |
| ⇒ Reset module A (disable sending beacons), CMD_RESET_REQ | 02 00 00 00 02 | |
| ⇐ Response CMD_RESET_CNF | 02 40 01 00 00 43 | |
| ⇐ Response CMD_GETSTATE_CNF | 02 41 02 00 01 01 41 | |

5.8. Energy-efficient distance estimation solutions

The transmitted advertising packet contains its TX power value. This value in combination with the RSSI value of the received advertising packet can be used to estimate the distance between the modules. Using a suitable triangulation algorithm and multiple receivers or transmitters, a position can be approximately determined.

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

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

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

5.9. Configure the module for low power consumption

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

- **CMD_SLEEP_REQ:** This command puts the module into ACTION_SLEEP mode, where it consumes the lowest current (<1μA). In this case, both the SPI and the Bluetooth® LE interface are shut down.
- **RF_TXPower:** This setting can be used to configure the output power of the module. Reducing the output power saves energy.
- **RF_ScanTiming** and **RF_ScanFactor:** These settings define the timing behavior of the module, when advertising or scanning. The less often the module sends advertising packets or scans, the less current is consumed.

- **RF_ConnectionTiming:** This setting defines the timing behavior of the module during connection setup and during an open connection. The less often the connected modules communicate with each other, the less current is consumed.
- The on-board nRF52 SoC is running in debug mode. This will not occur if the pins are connected as described in this manual.
- The 2 MBit radio modes transmits data packets faster. Using it reduces the current consumption slightly.



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

5.10. Start the direct test mode (DTM)

The direct test mode (DTM) enables the test functions described in Bluetooth® Specification. The purpose of DTM is to test the operation of the radio at the physical level, such as transmission power and receiver sensitivity, frequency offset and drift, modulation characteristics, packet error rate and inter modulation performance.

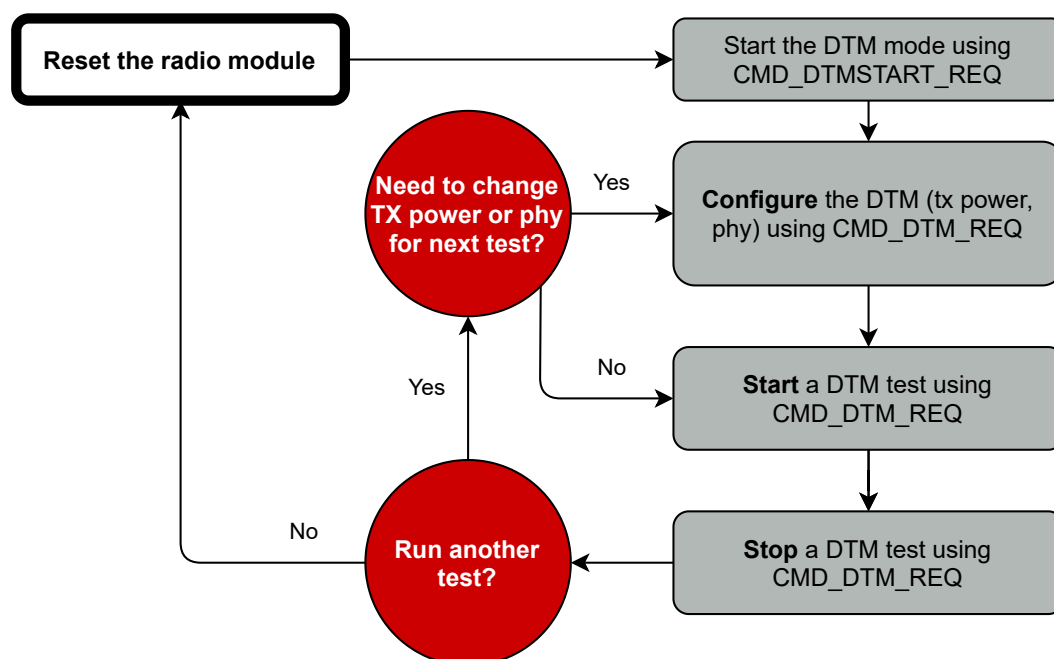


Figure 8: DTM flow chart

Conformance tests of the nRF52 with the DTM application are carried out by dedicated test equipment. To get access to the test functions the `CMD_DTMSTART_REQ`¹ shall be used first. This command restarts the module in direct test mode. A `CMD_GETSTATE_CNF` message confirms that

¹Please note that the Smart Commander [4] PC tool in version 1.3.1.0 or newer supports the DTM function.

the DTM has been started successfully.

Now the CMD_DTM_REQ can be used to configure output power and phy of the subsequent tests. Next, to start and stop the test functions, please use again the CMD_DTM_REQ command. After a test has been started, it has to be stopped before a next test can be run.

Example: Transmission test on channel 0 with Bit pattern 0x0F

The goal of this example is to show how the DTM, and in specific the transmission/reception test, can be run. Here fore we need to connect two modules, start the transmission test on one module and start the reception test on the second module. In this section, all packet data from or to the modules is given in **hexadecimal notation**.

All steps are described in the following:

- First, restart the modules in DTM mode.

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

- Now both modules are ready for the DTM configuration and operation.
- (Optional) Set both modules to 2 MBit mode (default setting after start-up is 1 MBit).

| Info | Module A | Module B |
|--|-------------------------------|-------------------------------|
| ⇒ Request CMD_DTM_REQ to set module A to 2 MBit mode | 02 1E 04 00 00 02 02 00 18 | |
| ⇐ Response CMD_DTM_CNF: Success | 02 5E 03 00 00 00 00 5F | |
| ⇒ Request CMD_DTM_REQ to set module B to 2 MBit mode | | 02 1E 04 00 00 02 02 00 18 |
| ⇐ Response CMD_DTM_CNF: Success | | 02 5E 03 00 00 00 00 5F |

- (Optional) Set the output power to 4 dBm (default setting after start-up is 8 dBm).

| Info | Module A | Module B |
|------|----------|----------|
|------|----------|----------|

| | | |
|---|-------------------------------|-------------------------------|
| ⇒ Request CMD_DTM_REQ to set module A to 8 dBm output power | 02 1E 04 00 02 04 02 03 1F | |
| ⇐ Response CMD_DTM_CNF: Success | 02 5E 03 00 00 00 00 5F | |
| ⇒ Request CMD_DTM_REQ to set module B to 8 dBm output power | | 02 1E 04 00 02 04 02 03 1F |
| ⇐ Response CMD_DTM_CNF: Success | | 02 5E 03 00 00 00 00 5F |

- After configuration the tests can start. First start the transmission test.

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

- Start the reception test.

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

- Stop both tests again.

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

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

5.11. Using the 2 MBit and LE Coded phy

Bluetooth® 5 allows to transmit data with 2 MBit data rate as well as in LE Coded mode. The LE Coded mode is the so called "Long range mode" that has been invented with Bluetooth® 5.0. It uses the Direct Sequence Spread Spectrum (DSSS) technique that spreads the signal and thus generates redundant informations. On the receiver side, it uses the Forward Error Correction (FEC) technique to use the redundancy to correct a received perturbed signal. The combination of both, the DSSS and FEC, enable higher ranges in data transmission.

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



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

5.12. Connection setup using LE Coded phy

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

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

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



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

Example: Configure the device for long range connection setup

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

1. Power-up the modules and make their SPIs accessible by the host(s). After the power-up or after reset the following sequence is available in the radio module to be fetched by the host.

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

2. Request the FS_BTMAC of both modules.

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

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

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

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

| Info | Module A | Module B |
|--|--|----------|
| ⇒ Request CMD_CONNECT_REQ with FS_BTMAC of module B | 02 06 06 00 11 00 00 DA 18 00 D1 | |
| ⇐ Response CMD_CONNECT_CNF: Request understood, try to connect now | 02 46 01 00 00 45 | |
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 | 02 86 07 00 00 11 00 00 DA 18 00 50 | |

| | | |
|---|---|---|
| ⇐ Indication CMD_CONNECT_IND: Physical connection established successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 | | 02 86 07 00 00 55 00 00 DA 18 00 14 |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | 02 C6 08 00 00 11 00 00 DA 18 00 F3 EC | |
| ⇐ Indication CMD_CHANNELOPEN_RSP: Channel opened successfully to module with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet | | 02 C6 08 00 00 55 00 00 DA 18 00 F3 A8 |

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



The RSSI values will be different in your tests.

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

6. Reply with "EFGH" to module B.

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

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

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

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

5.13. Radio compatibility to other Proteus devices

The Proteus-III-SPI uses the same radio settings as Proteus-I and -II, and is therefore fully compatible on the radio.

Only Proteus-e has a different format of the advertising packets, which forces the Proteus-III-SPI to use special scanner settings to be able to scan for Proteus-e radio modules. Before running the scan on Proteus-III-SPI, first configure the following settings:

- RF_AdvertisingFlags to 0x02 (format 2)
- RF_ScanFlags to 0x01 (active scan enabled)

6. Host connection

6.1. Serial interface: SPI

The Proteus-III-SPI acts as SPI slave and provides a 4-wire SPI interface using the pins *SPI_CS*, *SPI_CLK*, *SPI_MOSI* and *SPI_MISO*. In addition to that, the pin *SPI_INT* is used to signalize that the module (SPI slave) has data available. If the level of this pin is set to its active state, the module awaits the chip select and clock signal from the connected host (SPI master), such that the available data can be transferred from the slave to the master device.

The SPI configuration (SPI mode and *SPI_INT* active level) can be configured by means of the user setting *SPI_ConfigIndex*. The default setting is SPI mode 2 and *SPI_INT* "active high". The maximum supported SPI clock is 8 Mbit/s.

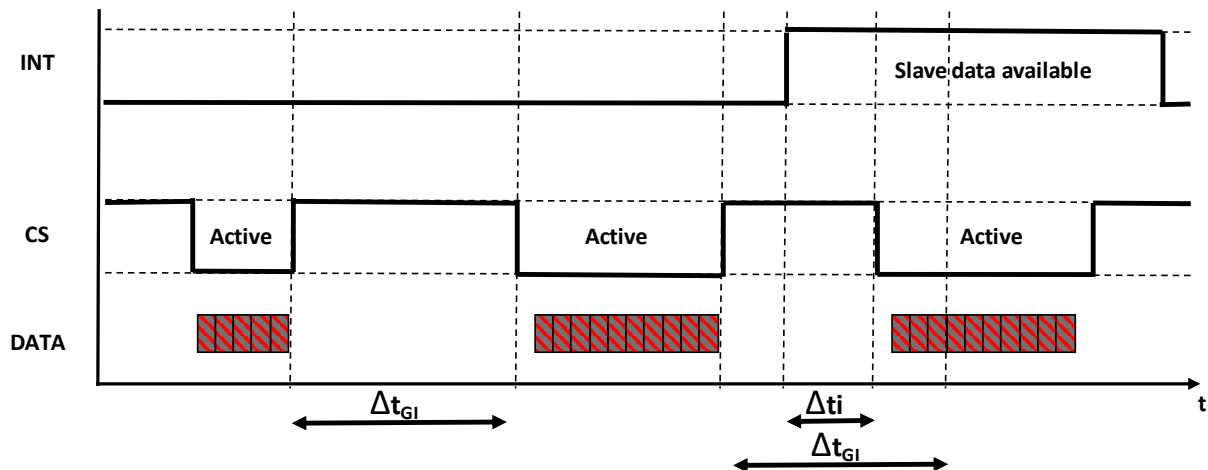
The following timing values are used within this chapter:

| Name | Description | Min. | Typ. | Max. | Unit |
|-----------------|--|------|------|------|---------|
| Δt_{GI} | (Guard interval) Time between <i>SPI_CS</i> gets inactive and <i>SPI_CS</i> is allowed to get active again | 150 | 500 | - | μs |
| Δt_i | Time between <i>SPI_INT</i> gets active and <i>SPI_CS</i> is allowed to get active | 0 | 0 | - | μs |
| Δt_s | Time between <i>SPI_CS</i> gets active and first <i>SPI_CLK</i> signal is allowed | 3.5 | 5 | - | μs |
| Δt_b | Time between the two packet fragments in case of data transmission from module to host | 0 | 0 | - | μs |
| Δt_e | Time between <i>SPI_CS</i> gets inactive and <i>SPI_INT</i> is updated | 23 | 50 | - | μs |

To be sure that the radio module is ready to receive data, the *SPI_CS* is allowed to get active not before

- Δt_{GI} after the *SPI_CS* line got inactive OR
- Δt_i after the *SPI_INT* line got active.

SPI packet timing

Figure 9: SPI packet timing, example *SPI_INT* "active high"

6.1.1. SPI communication details in command mode

To transmit data from the host (SPI master) to the radio module (SPI slave), the full command frame must be sent in a single SPI transaction. The radio module returns 0xFF on the *SPI_MISO* line, if no data in the module is available to transmit (*SPI_INT* is inactive). As soon as the *SPI_CS* line is returned to its default state by the host, the module interpretes the received command. If the command is valid (see chapter 7 for details of the command structure), it is interpreted. Otherwise it's discarded without notification. A SPI transfer must not exceed a length of 1024 bytes.

Figure 10 shows an example of such a transaction, where the *SPI_INT* is active high.



The full frame, starting with STX and ending with the checksum (see chapter 7), must not exceed an overall size of 1024 bytes.

Host/Master → Module/Slave
Command Mode

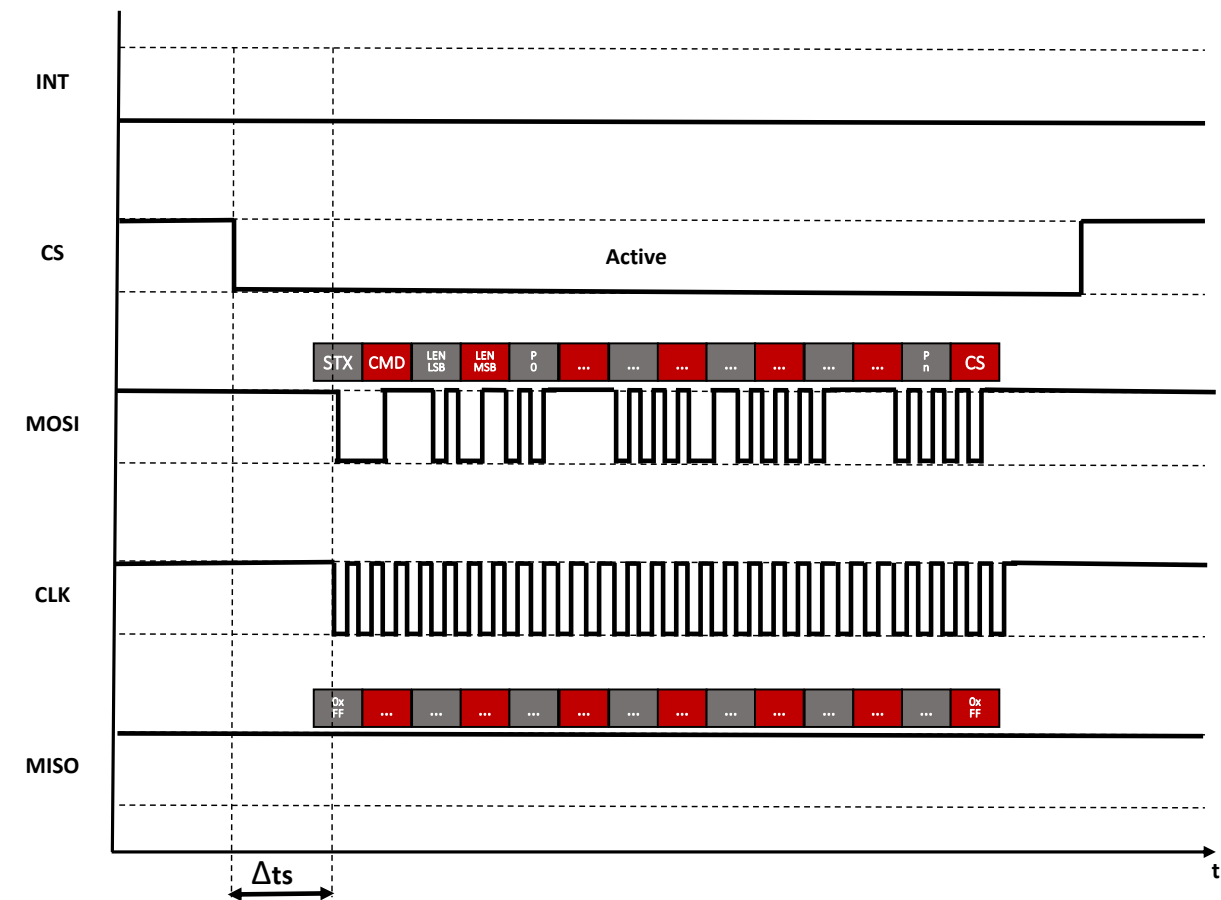


Figure 10: SPI master to slave communication, example *SPI_INT* "active high"

If the radio module (SPI slave) has data to transmit to the host (SPI master), the radio module sets the *SPI_INT* to its active voltage level. To do so, first the host must read 4 bytes from the radio module and interpret the length field of the command frame. Then, the remaining "length + 1" bytes can be read in a second SPI transaction.



It is important that the *SPI_CS* line stays active during the transmission of the two packet fragments. Otherwise the packet is discarded on the radio module as soon as the *SPI_CS* is put to its default state.

Figure 11 shows an example of such a transaction, where the *SPI_INT* is active high.

Module/Slave → Host/Master
Command Mode

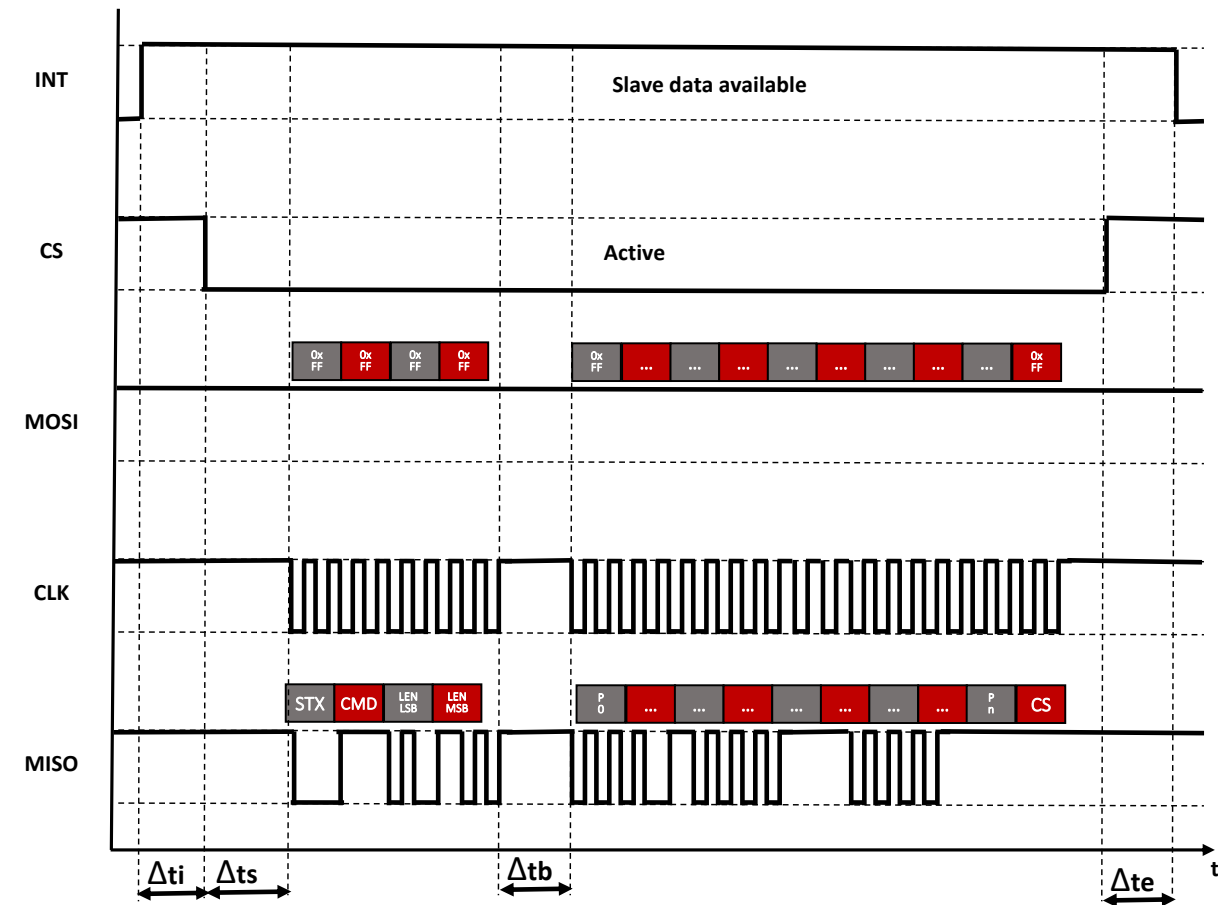


Figure 11: SPI slave to master communication, example *SPI_INT* "active high"

Ideally, the *SPI_MOSI* line is set to 0xFF if data is read from the radio module. But in case, the host wants to transmit a command to the radio module in parallel, it can do so by putting its data onto the *SPI_MOSI* line. Please be aware that the host must provide enough clock cycles (CLK_{max}) for the data transmission from host to radio module ($CLK_{host2slave}$) and vice versa ($CLK_{slave2host}$):

$$CLK_{max} = \text{Max}(CLK_{host2slave}, CLK_{slave2host})$$

As soon as the *SPI_CS* line is returned to its default state by the host, the module interpretes the received command. If the command is valid (see chapter 7 for details of the command structure), it is interpreted. Otherwise it's discarded without notification. A SPI transfer must not exceed a length of 1024 bytes.

6.1.2. Shared WAKE_UP/SPI_INT pin

As the *WAKE_UP/SPI_INT* is shared, the host controller must handle this pin dependent on the radio module state.

In case the radio module is sleeping (i.e. after a *CMD_SLEEP_REQ*), the *WAKE_UP/SPI_INT* pin of the radio module is set to input with a pull resistor to its "inactive level". Thus, to not wake-up the radio module immediately, the *WAKE_UP/SPI_INT* pin of the host controller must be set to

output with its "inactive level" as output power level. The "active level" is defined by means of the user setting `SPI_ConfigIndex` (by default it is "active high").

If the radio module is running (after a reset or wake-up), the *WAKE_UP/SPI_INT* pin of the host controller must be set to input with no pull resistor to detect if there are messages in the radio module available that are supposed to be sent via SPI to the host controller (as shown in figure 11).

7. The command interface

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

The commands of the command interface can be divided into 3 groups:

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

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

Start signal: 0x02 (1 Byte)

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

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

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

Checksum: Byte wise XOR combination of all preceding Bytes including the start signal, i.e.
 $0x02 \oplus \text{Command} \oplus \text{Length} \oplus \text{Payload} = \text{CS}$



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*.



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

7.1. Scan for other modules in range

7.1.1. CMD_SCANSTART_REQ

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

Format:

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

Response (CMD_SCANSTART_CNF):

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

Status:

0x00: Request received, will start scan now

0x01: Operation failed

0xFF: Operation not permitted

7.1.2. CMD_SCANSTOP_REQ

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

Format:

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

Response (CMD_SCANSTOP_CNF):

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

Status:

0x00: Request received, will stop scan now

0x01: Operation failed

0xFF: Operation not permitted

7.1.3. CMD_GETDEVICES_REQ

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

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

Response (CMD_GETDEVICES_CNF):

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

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

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

Status:

0x00: Request received

0x01: Operation failed

0xFF: Operation not permitted



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



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



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



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



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

7.1.3.1. Example 1

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

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

Response:

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

During the last scan two devices have been detected:

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

7.1.4. CMD_RSSI_IND

This telegram indicates the reception of an advertising packet sent by another Proteus-III-SPI module. It can be used to realize a position sensing application. This data can only be received, when the module is in ACTION_SCANNING mode (passive scan is sufficient) and the corresponding value in the RF_BeaconFlags is set.

Besides the FS_BTMAC, the RSSI value of the advertising packet and the transmission power of

the sending device are output. Both, the RSSI value and the TX power are in two's complement notation.

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

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

Format:

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

7.1.5. CMD_BEACON_RSP

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

Besides the FS_BTMAC, the RSSI value of the advertising packet and the raw data is output.

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



The format of the raw advertising data is described in the Bluetooth® specification version 5.1 | Vol 3, Part C, section "Advertising and scan response data format".

Format:

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

7.2. Setup connections

7.2.1. CMD_CONNECT_REQ

This command tries to setup a connection to the Proteus-III-SPI, which is identified by the FS_BTMAC used in the command. After the module prints a CMD_CONNECT_CNF to confirm that the request was received, the indication message CMD_CONNECT_IND follows which determines whether the connection request was accepted by the other device.

In case of enabled security features (see the setting RF_SecFlags) a CMD_SECURITY_IND is output in addition.

As soon as the connection setup has been completed and all services have been discovered successfully a CMD_CHANNELOPEN_RSP is sent to the host. Now data may be sent using the CMD_DATA_REQ.

Format:

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

Response (CMD_CONNECT_CNF):

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

Status:

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

0x01: Operation failed

0xFF: Operation not permitted

7.2.2. CMD_CONNECT_IND

This telegram indicates the connection status and, in case of success, the FS_BTMAC of the connected device. This indication message is the result of a connection request (CMD_CONNECT_REQ).

Format (connected successfully):

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

Format (failed to connect):

| Start signal | Command | Length | Status | CS |
|--------------|---------|-----------|--------|--------|
| 0x02 | 0x86 | 0x01 0x00 | 0x01 | 1 Byte |

Status:

0x00: Physical connection established successfully

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

7.2.3. CMD_SECURITY_IND

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

Format:

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

Status:

0x00: Encrypted link to previously bonded device established

0x01: Bonding successful, encrypted link established

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

7.2.4. CMD_CHANNELOPEN_RSP

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

Format:

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

Status:

0x00: Success

7.2.5. CMD_DISCONNECT_REQ

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

Format:

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

Response (CMD_DISCONNECT_CNF):

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

Status:

0x00: Request received, try to disconnect

0x01: Operation failed

0xFF: Operation not permitted

7.2.6. CMD_DISCONNECT_IND

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

Format:

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

Reason:

0x08: Connection timeout

0x13: User terminated connection

0x16: Host terminated connection

0x3B: Connection interval unacceptable

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

0x3E: Connection setup failed

7.2.7. CMD_PHYUPDATE_REQ

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

Format:

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

PHY:

0x01: 1 MBit PHY

0x02: 2 MBit PHY

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

Response (CMD_PHYUPDATE_CNF):

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

Status:

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

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

0xFF: Operation not permitted

7.2.8. CMD_PHYUPDATE_IND

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

Format in case of success:

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

PHY Rx/PHY Tx:

0x01: Using 1 MBit PHY now

0x02: Using 2 MBit PHY now

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

Format in case of failure:

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

Info:

0x1A: Unsupported feature of remote device

7.2.9. CMD_PASSKEY_REQ

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

Format:

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

Response (CMD_PASKEY_CNF):

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

Status:

0x00: Pass key accepted and pass key request answered

0x01: Operation failed, due to invalid pass key

0xFF: Operation not permitted

7.2.10. CMD_PASKEY_IND

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

Format:

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

Status:

0x00: Success

7.2.11. CMD_DISPLAY_PASKEY_IND

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

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

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

Format:

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

Action:

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

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

7.2.12. CMD_NUMERIC_COMP_REQ

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

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

Format:

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

Status:

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

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

0xFF: Operation not permitted

7.2.13. CMD_GETBONDS_REQ

This command requests the MAC addresses of all bonded devices.

Format:

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

Response (CMD_GETBONDS_CNF):

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

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

| Bond_ID | BTMAC |
|---------|---------|
| 2 Bytes | 6 Bytes |

Status:

0x00: Request successfully processed

0x01: Operation failed

0xFF: Operation not permitted



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

7.2.13.1. Example 1

Request for the bonding data of the devices in database.

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

Response:

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

Two devices have been bonded before:

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

7.2.14. CMD_DELETEBONDS_REQ

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

Format:

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

Response (CMD_DELETEBONDS_CNF):

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

Status:

0x00: Request successfully processed

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

0xFF: Operation not permitted

7.2.14.1. Example 1

Request to remove all bonding data.

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

Response:

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

Successfully removed all bonding information.

7.2.14.2. Example 2

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

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

Response:

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

Successfully removed the bonding information.

7.2.15. CMD_ALLOWUNBONDEDCONNECTIONS_REQ

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

Format:

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

Response (CMD_ALLOWUNBONDEDCONNECTIONS_CNF):

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

Status:

0x00: Request successfully processed

0x01: Operation failed

0xFF: Operation not permitted

7.3. Transmit and receive data

7.3.1. CMD_DATA_REQ

This command provides the simple data transfer between two connected modules. Transmission takes place to the previously connected device(s). This command is suitable for transmission for a point-to-point connection. The number of payload data Bytes is negotiated during the connection phase. It can be maximal 243 Bytes, but at least 19 Bytes.

When the data is processed by the module a CMD_DATA_CNF is sent to the host. Additionally a CMD_TXCOMPLETE_RSP will follow as soon as the data has been sent.

The receiving Proteus-III-SPI will get a CMD_DATA_IND message containing the transmitted payload data.

In "high throughput mode" the length of data packets may be up to 964 Bytes. Format:

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

Response (CMD_DATA_CNF):

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

Status:

0x00: Request received, will send data now

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

0xFF: Operation not permitted

7.3.2. CMD_TXCOMPLETE_RSP

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

Format:

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

Status:

0x00: Data transmitted successfully

0x01: Data transmission failed

7.3.3. CMD_DATA_IND

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

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

Format:

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

7.3.4. CMD_SETBEACON_REQ

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

It can be received by any scanning Proteus-III-SPI with Beacon-function enabled (see RF_BeaconFlags). The receiving module will output a CMD_BEACON_IND indication message containing the transmitted data. See chapter 5.7 for more information.

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

Format:

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

Response (CMD_SETBEACON_CNF):

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

Status:

0x00: Request received, will place data now

0x01: Operation failed

0xFF: Operation not permitted

7.3.5. CMD_BEACON_IND

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

The data received via the RF-interface can be found in the payload of the CMD_BEACON_IND

telegram. Besides this, the FS_BTMAC of the sending device and the RSSI value of the data packet are output as well. The RSSI value is output in two's complement notation.

Format:

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

7.4. Configuring the module and modifying the device settings



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

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

7.4.1. `CMD_SET_REQ`

This command enables direct manipulation of the parameters in the module's settings in flash. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 76.

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



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



The flash memory used to store these settings has a limited count of write cycles of minimum 10.000. Try to avoid performing periodic `CMD_SET_REQ` as each command will use one write cycle.



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



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

Recommendation: First, verify the configuration of the module with `CMD_GET_REQ` and only then apply a `CMD_SET_REQ` if required to avoid unnecessary flash cycles.

Format:

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

Response (CMD_SET_CNF):

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

Status:

0x00: Request received, settings set successfully

0x01: Operation failed due to invalid parameter

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

0x05: Supply voltage below 1.8V $\pm 5\%$. Please apply correct supply voltage, reset and retry.

0xFF: Operation not permitted

7.4.1.1. Example 1

Setting the advertising time RF_AdvertisingTimeout to 180 seconds.

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

Response:

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

Setting was set successfully.

7.4.1.2. Example 2

Setting the static pass key RF_StaticPasskey to "123456".

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

Response:

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

Setting was set successfully.

7.4.2. CMD_GET_REQ

This command can be used to query individual setting parameters in flash. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 76.

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

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

Format:

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

Response (CMD_GET_CNF):

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

Status:

0x00: Request received, read out of setting successful

0x01: Operation failed

0xFF: Operation not permitted

7.4.2.1. Example 1

Request the current static pass key RF_StaticPasskey.

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

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

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

Setting was read successfully.

7.5. Manage the device state

7.5.1. CMD_GETSTATE_REQ

This command returns the current state of the module.



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

Format:

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

Response (CMD_GETSTATE_CNF):

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

Module role:

0x00: No role

0x01: Peripheral

0x02: Central

0x10: Direct test mode (DTM)

Other: Reserved

Module action:

0x00: No action

0x01: Idle (advertising)

0x02: Scanning

0x03: Connected

0x04: Sleep (system-off mode)

0x05: Direct test mode

More info:

- If module action is "Connected"
 - 6 Bytes FS_BTMAC address of the connected device
 - 1 Byte indicating the maximum payload of the connection
- Otherwise, more info is empty

7.5.1.1. Example 1

Get the current state of the module.

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

Response:

| Start signal | Command 0x40 | Length | Module role | Module actions | More info | CS |
|--------------|-------------------|-----------|----------------|-------------------|---|------|
| 0x02 | 0x41 | 0x09 0x00 | 0x02 | 0x03 | 0x11 0x00 0x00 0xDA 0x18 0x00 0xF3 | 0x6B |

The module is connected to another module with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00.
The MTU of the connection is **243** Bytes.

7.5.2. CMD_RESET_REQ

This command triggers a software reset of the module.

Format:

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

Response (CMD_RESET_CNF):

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

Status:

0x00: Request received, will perform reset now

0x01: Operation failed

0xFF: Operation not permitted

7.5.3. CMD_SLEEP_REQ

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

Format:

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

Response (CMD_SLEEP_CNF):

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

Status:

0x00: Request received, will go to sleep now

0x01: Operation failed

0xFF: Operation not permitted



Please note that the *WAKE_UP* pin has a second function. If the module is not in *ACTION_SLEEP* mode, this pin is used to indicate that the module has SPI data, that has to be transmitted to the host as soon as the host requests for it.

7.5.4. CMD_SLEEP_IND

This indication is sent by the module when the *RF_AdvertisingTimeout* has expired without a connection to the module.

Format:

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

Status:

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

7.5.5. CMD_FACTORYRESET_REQ

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



Please note that also the GPIO configuration specified in chapter 10 is reset to default.

Format:

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

Response (CMD_FACTORYRESET_CNF):

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

Status:

0x00: Request received, will perform factory reset now

0x01: Operation failed

0xFF: Operation not permitted



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



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



This command also removes all bonding data.

7.5.6. CMD_BOOTLOADER_REQ

This command resets the module and starts the OTA bootloader.



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



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



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

Format:

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

Response (CMD_BOOTLOADER_CNF):

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

Status:

0x00: Request received, will start bootloader now

0x01: Operation failed

0xFF: Operation not permitted

7.6. Run the Bluetooth test modes

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

7.6.1. CMD_DTMSTART_REQ

This command restarts the module in direct test mode (DTM). When starting in DTM mode, a CMD_GETSTATE_CNF message follows which indicates that the test mode has been enabled successfully. Now the CMD_DTM_REQ can be used to start and stop various test modes. Performing a reset will leave the DTM and restart the module in the ACTION_IDLE state.

Format:

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

Response (CMD_DTMSTART_CNF):

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

Status:

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

0x01: Operation failed

0xFF: Operation not permitted

7.6.2. CMD_DTM_REQ

This command starts and stops various test modes. To be able to run these test modes, the DTM has to be enabled first using the CMD_DTMSTART_REQ. After a test has been started, it has to be stopped first before a next test can be run.

The default TX power value is 8 dBm, the allowed range is from -40 up to +8 dBm (see chapter 8.16 for valid TX power values). The valid range for channel is 0...39.



Configuring the phy and TX power can be only done, when the test is stopped (no TX or RX test ongoing).

Format:

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

Command code:

0x00: DTM setup

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

0x01: Start RX test

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

0x02: Start TX test

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

| Vendor option | Vendor command | Payload |
|--|--------------------|------------------------------|
| Frequency = (2402 + Channel * 2) MHz to be used for TX | 0x00: Carrier test | 0x03: Vendor specific |
| TX power -40 up to +8 dBm (see chapter 8.16 for valid TX power values) | 0x02: Set TX power | 0x03: Vendor specific |

0x03: Stop last test

| Channel | Length | Payload |
|---------|--------|---------|
| 0x00 | 0x00 | 0x00 |

Response (CMD_DTM_CNF):

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

Status:

0x00: Request received

0x01: Operation failed

0x03: Busy

0xFF: Operation not permitted

Result:

0x0000: Test success

0x0001: Test failed

0x8000 + n: Received n packets during RX test



See also the example in chapter 5.10.

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

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

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

Response:

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

Test started successfully. Now stop the test again.

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

Response:

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

Test stopped successfully and received 0 packets.

7.6.2.2. Example: Receiver, channel 0

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

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

Response:

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

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

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

Response:

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

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

7.6.2.3. Example: Transmission, carrier test, channel 0

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

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

Response:

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

See the previous example to stop the test again.

7.6.2.4. Example: Set TX power to -4 dBm

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

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

Response:

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

7.6.2.5. Example: Set PHY to 2 MBit mode

Set the phy to 2 MBit mode:

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

Response:

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

7.7. Switching GPIOs by remote control

This chapter contains the commands to use the GPIO feature of the Proteus-III-SPI. Please refer to chapter 10 for a detailed description.

7.7.1. CMD_GPIO_LOCAL_WRITECONFIG_REQ

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



The flash memory used to store these settings has a limited count of write cycles of minimum 10.000. Try to avoid performing periodic CMD_GPIO_LOCAL_WRITECONFIG_REQ as each command will use one write cycle.

Format:

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

Response (CMD_GPIO_LOCAL_WRITECONFIG_CNF):

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

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_LOCAL_WRITECONFIG_REQ block structure

Each **Block** has the following format:

| Length | GPIO_ID | Function | Values |
|--------|---------|----------|-------------------|
| 1 Byte | 1 Byte | 1 Byte | (Length - 2) Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Function:

- 0x00:** GPIO disconnected
- 0x01:** GPIO works as input
- 0x02:** GPIO works as output
- 0x03:** GPIO works as PWM

Values:

- if **Function** is disconnected, Length is 0x03:
0x00: value field must use 0x00.
- if **Function** is input, Length is 0x03:
0x00: GPIO has no pull resistor
0x01: GPIO has internal pull down resistor
0x02: GPIO has internal pull up resistor
- if **Function** is output, Length is 0x03:
0x00: GPIO is output LOW
0x01: GPIO is output HIGH
- if **Function** is PWM, Length is 0x05 (see chapter 10.1):
Byte 0 and 1: LSB first uint16 PWM period in ms (1 - 500 ms)
Byte 2: Ratio (0x00=0%,... , 0xFE=100%)

CMD_GPIO_LOCAL_WRITECONFIG_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Status |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Status:

- 0x00:** Success
- 0x01:** Failed

7.7.1.1. Example: Configure two GPIOs to output high

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

| Start signal | Command | Length | Block ₁ | Block ₂ | CS |
|--------------|---------|-----------|----------------------------|----------------------------|------|
| 0x02 | 0x25 | 0x08 0x00 | 0x03 0x01 0x02 0x01 | 0x03 0x02 0x02 0x01 | 0x2C |

Response:

| Start signal | Command 0x40 | Length | Status | Block ₁ | Block ₂ | CS |
|--------------|----------------|-----------|--------|-----------------------|-----------------------|------|
| 0x02 | 0x65 | 0x07 0x00 | 0x00 | 0x02 0x01 0x00 | 0x02 0x02 0x00 | 0x63 |

Configured both GPIOs with success.

7.7.2. CMD_GPIO_LOCAL_READCONFIG_REQ

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

Format:

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

Response (CMD_GPIO_LOCAL_READCONFIG_CNF):

| Start signal | Command 0x40 | Length | Status | Block ₁ | ... | Block _n | CS |
|--------------|----------------|---------|--------|--------------------|-----|--------------------|--------|
| 0x02 | 0x6B | 2 Bytes | 1 Byte | x Bytes | | x Bytes | 1 Byte |

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_LOCAL_READCONFIG_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Function | Values |
|--------|---------|----------|-------------------|
| 1 Byte | 1 Byte | 1 Byte | (Length - 2) Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Function:

0x00: GPIO is disconnected

0x01: GPIO works as input

0x02: GPIO works as output

0x03: GPIO works as PWM

Values:

- if **Function** is disconnected, Length is 0x02:

Values field is not used in this block

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

0x00: GPIO has no pull resistor

0x01: GPIO has pull down resistor

- 0x02:** GPIO has pull up resistor
- if **Function** is output, Length is 0x03:
 - 0x00:** GPIO is output LOW
 - 0x01:** GPIO is output HIGH
- if **Function** is PWM, Length is 0x05 (see chapter 10.1):
 - Byte 0 and 1:** LSB first uint16 PWM period in ms (1 - 500 ms)
 - Byte 2:** Ratio (0x00=0%,..., 0xFE=100%)

7.7.2.1. Example: Read the current GPIO configuration

Read the current configuration:

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

Response:

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

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

7.7.3. CMD_GPIO_REMOTE_WRITECONFIG_REQ

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



The flash memory used to store these settings has a limited count of write cycles of minimum 10.000. Try to avoid performing periodic CMD_GPIO_REMOTE_WRITECONFIG_REQ as each command will use one write cycle.

Format:

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

Response (CMD_GPIO_REMOTE_WRITECONFIG_CNF):

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

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_REMOTE_WRITECONFIG_REQ block structure

Each **Block** has the following format:

| Length | GPIO_ID | Function | Values |
|--------|---------|----------|-------------------|
| 1 Byte | 1 Byte | 1 Byte | (Length - 2) Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Function:

0x00: GPIO disconnected

0x01: GPIO works as input

0x02: GPIO works as output

0x03: GPIO works as PWM

Values:

- if **Function** is disconnected, Length is 0x03:
0x00: value field must use 0x00.
- if **Function** is input, Length is 0x03:
0x00: GPIO has no pull resistor
0x01: GPIO has internal pull down resistor
0x02: GPIO has internal pull up resistor
- if **Function** is output, Length is 0x03:
0x00: GPIO is output LOW
0x01: GPIO is output HIGH
- if **Function** is PWM, Length is 0x05 (see chapter 10.1):
Byte 0 and 1: LSB first uint16 PWM period in ms (1 - 500 ms)
Byte 2: Ratio (0x00=0%,... , 0xFE=100%)

CMD_GPIO_REMOTE_WRITECONFIG_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Status |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Status:

0x00: Success

0x01: Failed

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

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

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

| Start signal | Command | Length | Block ₁ | Block ₂ | CS |
|--------------|---------|-----------|----------------------------|----------------------------|------|
| 0x02 | 0x28 | 0x08 0x00 | 0x03 0x01 0x02 0x01 | 0x03 0x02 0x02 0x01 | 0x21 |

Response:

| Start signal | Command 0x40 | Length | Status | Block ₁ | Block ₂ | CS |
|--------------|----------------|-----------|--------|--------------------------|--------------------------|------|
| 0x02 | 0x68 | 0x07 0x00 | 0x00 | 0x02 0x01 0x00 | 0x02 0x02 0x00 | 0x6E |

Configured both GPIOs with success.

7.7.4. CMD_GPIO_REMOTE_READCONFIG_REQ

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

Format:

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

Response (CMD_GPIO_REMOTE_READCONFIG_CNF):

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

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_REMOTE_READCONFIG_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Function | Values |
|--------|---------|----------|-------------------|
| 1 Byte | 1 Byte | 1 Byte | (Length - 2) Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Function:

0x00: GPIO is disconnected

0x01: GPIO works as input

0x02: GPIO works as output

0x03: GPIO works as PWM

Values:

- if **Function** is disconnected, Length is 0x02:
Values field is not used in this block
- if **Function** is input, Length is 0x03:
0x00: GPIO has no pull resistor

- 0x01:** GPIO has pull down resistor
- 0x02:** GPIO has pull up resistor
- if **Function** is output, Length is 0x03:
 - 0x00:** GPIO is output LOW
 - 0x01:** GPIO is output HIGH
- if **Function** is PWM, Length is 0x05 (see chapter 10.1):
 - Byte 0 and 1:** LSB first uint16 PWM period in ms (1 - 500 ms)
 - Byte 2:** Ratio (0x00=0%,... , 0xFE=100%)

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

Read the current GPIO configuration of the connected remote device:

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

Response:

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

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

7.7.5. CMD_GPIO_REMOTE_WRITE_REQ

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

Format:

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

Response (CMD_GPIO_REMOTE_WRITE_CNF):

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

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via Bluetooth® LE)

CMD_GPIO_REMOTE_WRITE_REQ block structure

Each **Block** has the following format:

| Length | GPIO_ID | Value |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Value:

- if **Function** is output
 - 0x00:** Set GPIO to LOW
 - 0x01:** Set GPIO to HIGH
- if **Function** is PWM
 - Byte 0:** Ratio (0x00=0%, ..., 0xFE=100%)

CMD_GPIO_REMOTE_WRITE_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Status |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Status:

0x00: Success

0x01: Failed

7.7.5.1. Example: Set a remote output GPIO to low

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

| Start signal | Command | Length | Block ₁ | CS |
|--------------|---------|-----------|-----------------------|------|
| 0x02 | 0x29 | 0x03 0x00 | 0x02 0x01 0x00 | 0x2B |

Response:

| Start signal | Command 0x40 | Length | Status | Block ₁ | CS |
|--------------|----------------|-----------|--------|-----------------------|------|
| 0x02 | 0x69 | 0x04 0x00 | 0x00 | 0x02 0x01 0x00 | 0x6C |

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

7.7.6. CMD_GPIO_REMOTE_READ_REQ

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

Format:

| Start signal | Command | Length | Block ₁ | CS |
|--------------|---------|---------|--------------------|--------|
| 0x02 | 0x2A | 2 Bytes | x Bytes | 1 Byte |

Response (CMD_GPIO_REMOTE_READ_CNF):

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

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via Bluetooth® LE)

CMD_GPIO_REMOTE_READ_REQ block structure

Each **Block** has the following format:

| Length | GPIO_ID ₁ | ... | GPIO_ID _n |
|---------|----------------------|-----|----------------------|
| 1 Bytes | 1 Byte | | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

CMD_GPIO_REMOTE_READ_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Value |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Value:

- if **Function** is output or input
 - 0x00:** The remote GPIO is LOW.
 - 0x01:** The remote GPIO is HIGH.
 - 0xFF:** Failed reading remote GPIO value.
- if **Function** is PWM
 - 0xFF:** Failed reading remote GPIO value.
 - Other:** Ratio (0x00=0%,... , 0xFE=100%)

7.7.6.1. Example: Read the values of remote GPIOs

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

| Start signal | Command | Length | Block ₁ | CS |
|--------------|---------|-----------|-----------------------|------|
| 0x02 | 0x2A | 0x03 0x00 | 0x02 0x01 0x02 | 0x2A |

Response:

| Start signal | Command 0x40 | Length | Status | Block ₁ | Block ₂ | CS |
|--------------|----------------|-----------|--------|-----------------------|-----------------------|------|
| 0x02 | 0x6A | 0x07 0x00 | 0x00 | 0x02 0x01 0x00 | 0x02 0x02 0x01 | 0x6D |

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

7.7.7. CMD_GPIO_LOCAL_WRITE_REQ

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

Format:

| Start signal | Command | Length | Block ₁ | ... | Block _n | CS |
|--------------|---------|---------|--------------------|-----|--------------------|--------|
| 0x02 | 0x26 | 2 Bytes | x Bytes | | x Bytes | 1 Byte |

Response (CMD_GPIO_LOCAL_WRITE_CNF):

| Start signal | Command 0x40 | Length | Status | Block ₁ | ... | Block _n | CS |
|--------------|----------------|---------|--------|--------------------|-----|--------------------|--------|
| 0x02 | 0x66 | 2 Bytes | 1 Byte | x Bytes | | x Bytes | 1 Byte |

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via Bluetooth® LE)

CMD_GPIO_LOCAL_WRITE_REQ block structure

Each **Block** has the following format:

| Length | GPIO_ID | Value |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Value:

- if **Function** is output
 - 0x00:** Set GPIO to LOW
 - 0x01:** Set GPIO to HIGH
- if **Function** is PWM
 - Byte 0:** Ratio (0x00=0%, ..., 0xFE=100%)

CMD_GPIO_LOCAL_WRITE_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Status |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Status:

0x00: Success

0x01: Failed

7.7.7.1. Example: Set a local output GPIO to low

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

| Start signal | Command | Length | Block ₁ | CS |
|--------------|---------|-----------|-----------------------|------|
| 0x02 | 0x26 | 0x03 0x00 | 0x02 0x01 0x00 | 0x24 |

Response:

| Start signal | Command 0x40 | Length | Status | Block ₁ | CS |
|--------------|----------------|-----------|--------|-----------------------|------|
| 0x02 | 0x66 | 0x04 0x00 | 0x00 | 0x02 0x01 0x00 | 0x63 |

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

7.7.8. CMD_GPIO_LOCAL_READ_REQ

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

Format:

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

Response (CMD_GPIO_LOCAL_READ_CNF):

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

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via Bluetooth® LE)

CMD_GPIO_LOCAL_READ_REQ block structure

Each **Block** has the following format:

| Length | GPIO_ID ₁ | ... | GPIO_ID _n |
|---------|----------------------|-----|----------------------|
| 1 Bytes | 1 Byte | | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

CMD_GPIO_LOCAL_READ_CNF block structure

Each **Block** has the following format:

| Length | GPIO_ID | Value |
|--------|---------|--------|
| 0x02 | 1 Byte | 1 Byte |

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 10.2

Value:

- if **Function** is output or input
 - 0x00**: The remote GPIO is LOW.
 - 0x01**: The remote GPIO is HIGH.
 - 0xFF**: Failed reading GPIO value.
- if **Function** is PWM
 - 0xFF**: Failed reading GPIO value.
 - Other**: Ratio (0x00=0%,..., 0xFE=100%)

7.7.8.1. Example: Read the values of local GPIOs

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

| Start signal | Command | Length | Block ₁ | CS |
|--------------|---------|-----------|-----------------------|------|
| 0x02 | 0x27 | 0x03 0x00 | 0x02 0x01 0x02 | 0x27 |

Response:

| Start signal | Command 0x40 | Length | Status | Block ₁ | Block ₂ | CS |
|--------------|----------------|-----------|--------|--------------------------|--------------------------|------|
| 0x02 | 0x67 | 0x07 0x00 | 0x00 | 0x02 0x01 0x00 | 0x02 0x02 0x01 | 0x60 |

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

7.7.9. CMD_GPIO_REMOTE_WRITECONFIG_IND

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



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

Format:

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

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

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

| Start signal | Command | Length | Block ₁ | Block ₂ | CS |
|--------------|---------|-----------|----------------------------|----------------------------|----|
| 0x02 | 0xA8 | 0x08 0x00 | 0x03 0x01 0x02 0x01 | 0x03 0x02 0x02 0x01 | A1 |

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

7.7.10. CMD_GPIO_REMOTE_WRITE_IND

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



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

Format:

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

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

7.7.10.1. Example: GPIOs have been written via remote access

| Start signal | Command | Length | Block ₁ | Block ₂ | CS |
|--------------|---------|-----------|-----------------------|-----------------------|------|
| 0x02 | 0xA9 | 0x06 0x00 | 0x02 0x01 0x00 | 0x02 0x02 0x01 | 0xAE |

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

7.7.11. CMD_GPIO_LOCAL_WRITE_IND

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



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

Format:

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

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

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

| Start signal | Command | Length | Block ₁ | Block ₂ | CS |
|--------------|---------|-----------|-----------------------|-----------------------|------|
| 0x02 | 0xA6 | 0x06 0x00 | 0x02 0x01 0x00 | 0x02 0x02 0x01 | 0xA1 |

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

7.8. Message overview

| Start signal | CMD | Message name | Short description | Chapter |
|--------------|------|--------------------------------|---|---------|
| 0x02 | 0x00 | CMD_RESET_REQ | Reset the module | 7.5.2 |
| 0x02 | 0x01 | CMD_GETSTATE_REQ | Request the current module state | 7.5.1 |
| 0x02 | 0x02 | CMD_SLEEP_REQ | Go to sleep | 7.5.3 |
| 0x02 | 0x04 | CMD_DATA_REQ | Send data to the connected device | 7.3.1 |
| 0x02 | 0x06 | CMD_CONNECT_REQ | Setup a connection with another device | 7.2.1 |
| 0x02 | 0x07 | CMD_DISCONNECT_REQ | Close the connection | 7.2.5 |
| 0x02 | 0x09 | CMD_SCANSTART_REQ | Start scan | 7.1.1 |
| 0x02 | 0x0A | CMD_SCANSTOP_REQ | Stop scan | 7.1.2 |
| 0x02 | 0x0B | CMD_GETDEVICES_REQ | Request the scanned/detected devices | 7.1.3 |
| 0x02 | 0x0C | CMD_SETBEACON_REQ | Place data in scan response packet | 7.3.4 |
| 0x02 | 0x0D | CMD_PASSKEY_REQ | Respond to a pass key request | 7.2.9 |
| 0x02 | 0x0E | CMD_DELETEBONDS_REQ | Delete bonding information | 7.2.14 |
| 0x02 | 0x0F | CMD_GETBONDS_REQ | Read the MACs of bonded devices | 7.2.13 |
| 0x02 | 0x10 | CMD_GET_REQ | Read the module settings in flash | 7.4.2 |
| 0x02 | 0x11 | CMD_SET_REQ | Modify the module settings in flash | 7.4.1 |
| 0x02 | 0x1A | CMD_PHYUPDATE_REQ | Update the PHY | 7.2.7 |
| 0x02 | 0x1C | CMD_FACTORYRESET_REQ | Perform a factory reset | 7.5.5 |
| 0x02 | 0x1D | CMD_DTMSTART_REQ | Enable the direct test mode | 7.6.1 |
| 0x02 | 0x1E | CMD_DTM_REQ | Start/stop a test of the direct test mode | 7.6.2 |
| 0x02 | 0x1F | CMD_BOOTLOADER_REQ | Switch to the bootloader | 7.5.6 |
| 0x02 | 0x24 | CMD_NUMERIC_COMP_REQ | Confirm/reject the displayed pass key | 7.2.12 |
| 0x02 | 0x25 | CMD_GPIO_LOCAL_WRITECONFIG_REQ | Configure the free GPIOs for remote control | 7.7.1 |
| 0x02 | 0x26 | CMD_GPIO_LOCAL_WRITE_REQ | Set the output value of a output GPIO of the current device | 7.7.7 |

| | | | | |
|------|------|----------------------------------|---|--------|
| 0x02 | 0x27 | CMD_GPIO_LOCAL_READ_REQ | Read the value of a GPIO of the current device | 7.7.8 |
| 0x02 | 0x28 | CMD_GPIO_REMOTE_WRITECONFIG_REQ | Configure the free GPIOs of the remote device for remote control | 7.7.3 |
| 0x02 | 0x29 | CMD_GPIO_REMOTE_WRITE_REQ | Set the output value of a output GPIO of a remote device | 7.7.5 |
| 0x02 | 0x2A | CMD_GPIO_REMOTE_READ_REQ | Read the value of a GPIO of a remote device | 7.7.6 |
| 0x02 | 0x2B | CMD_GPIO_LOCAL_READCONFIG_REQ | Read the GPIO configuration | 7.7.2 |
| 0x02 | 0x2C | CMD_GPIO_REMOTE_READCONFIG_REQ | Read the GPIO configuration of the connected remote device | 7.7.4 |
| 0x02 | 0x2D | CMD_ALLOWUNBONDEDCONNECTIONS_REQ | Temporarily allow the connection setup from unbonded peer devices | 7.2.15 |

Table 68: Message overview: Requests

| Start signal | CMD | Message name | Short description | Chapter |
|--------------|------|---------------------|--|---------|
| 0x02 | 0x40 | CMD_RESET_CNF | Reset request received | 7.5.2 |
| 0x02 | 0x41 | CMD_GETSTATE_CNF | Return the current module state | 7.5.1 |
| 0x02 | 0x42 | CMD_SLEEP_CNF | Sleep request received | 7.5.3 |
| 0x02 | 0x44 | CMD_DATA_CNF | Data transmission request received | 7.3.1 |
| 0x02 | 0x46 | CMD_CONNECT_CNF | Connection setup request received | 7.2.1 |
| 0x02 | 0x47 | CMD_DISCONNECT_CNF | Disconnection request received | 7.2.5 |
| 0x02 | 0x49 | CMD_SCANSTART_CNF | Scan started | 7.1.1 |
| 0x02 | 0x4A | CMD_SCANSTOP_CNF | Scan stopped | 7.1.2 |
| 0x02 | 0x4B | CMD_GETDEVICES_CNF | Output the scanned/detected devices | 7.1.3 |
| 0x02 | 0x4C | CMD_SETBEACON_CNF | Data is placed in scan response packet | 7.3.4 |
| 0x02 | 0x4D | CMD_PASSKEY_CNF | Received the pass key | 7.2.9 |
| 0x02 | 0x4E | CMD_DELETEBONDS_CNF | Deleted bonding information | 7.2.14 |
| 0x02 | 0x4F | CMD_GETBONDS_CNF | Return the MAC of all bonded devices | 7.2.13 |
| 0x02 | 0x50 | CMD_GET_CNF | Return the requested module flash settings | 7.4.2 |

| | | | | |
|------|------|----------------------------------|---|--------|
| 0x02 | 0x51 | CMD_SET_CNF | Module flash settings have been modified | 7.4.1 |
| 0x02 | 0x5A | CMD_PHYUPDATE_CNF | Update Phy request received | 7.2.7 |
| 0x02 | 0x5C | CMD_FACTORYRESET_CNF | Factory reset request received | 7.5.5 |
| 0x02 | 0x5D | CMD_DTMSTART_CNF | Enable the direct test mode now | 7.6.1 |
| 0x02 | 0x5E | CMD_DTM_CNF | Test of direct test mode started/stopped | 7.6.2 |
| 0x02 | 0x5F | CMD_BOOTLOADER_CNF | Will switch to bootloader now | 7.5.6 |
| 0x02 | 0x64 | CMD_NUMERIC_COMP_CNF | CMD_NUMERIC_COMP_REQ accepted | 7.2.12 |
| 0x02 | 0x65 | CMD_GPIO_LOCAL_WRITECONFIG_CNF | Configuration of a local GPIO for remote control done | 7.7.1 |
| 0x02 | 0x66 | CMD_GPIO_LOCAL_WRITE_CNF | Output value of a local GPIO set | 7.7.7 |
| 0x02 | 0x67 | CMD_GPIO_LOCAL_READ_CNF | Value of a local GPIO read | 7.7.8 |
| 0x02 | 0x68 | CMD_GPIO_REMOTE_WRITECONFIG_CNF | Configuration of a remote GPIO for remote control done | 7.7.3 |
| 0x02 | 0x69 | CMD_GPIO_REMOTE_WRITE_CNF | Output value of a remote GPIO set | 7.7.5 |
| 0x02 | 0x6A | CMD_GPIO_REMOTE_READ_CNF | Value of a remote GPIO read | 7.7.6 |
| 0x02 | 0x6B | CMD_GPIO_LOCAL_READCONFIG_CNF | Returns the GPIO configuration | 7.7.2 |
| 0x02 | 0x6C | CMD_GPIO_REMOTE_READCONFIG_CNF | Returns the GPIO configuration of the connected remote device | 7.7.4 |
| 0x02 | 0x6D | CMD_ALLOWUNBONDEDCONNECTIONS_CNF | Temporarily allowed the connection setup from unbonded peer devices | 7.2.15 |

Table 69: Message overview: Confirmations

| Start signal | CMD | Message name | Short description | Chapter |
|--------------|------|--------------------|--|---------|
| 0x02 | 0x82 | CMD_SLEEP_IND | State will be changed to ACTION_SLEEP | 7.5.4 |
| 0x02 | 0x84 | CMD_DATA_IND | Data has been received | 7.3.3 |
| 0x02 | 0x86 | CMD_CONNECT_IND | Connection established | 7.2.2 |
| 0x02 | 0x87 | CMD_DISCONNECT_IND | Disconnected | 7.2.6 |
| 0x02 | 0x88 | CMD_SECURITY_IND | Secured connection established | 7.2.3 |
| 0x02 | 0x8B | CMD_RSSI_IND | Proteus-III-SPI advertising package detected | 7.1.4 |
| 0x02 | 0x8C | CMD_BEACON_IND | Received Beacon data | 7.3.5 |
| 0x02 | 0x8D | CMD_PASSKEY_IND | Received a pass key request | 7.2.10 |

| | | | | |
|------|------|---------------------------------|---|--------|
| 0x02 | 0x9A | CMD_PHYUPDATE_IND | PHY has been updated | 7.2.8 |
| 0x02 | 0xA4 | CMD_DISPLAY_PASSKEY_IND | Display pass key | 7.2.11 |
| 0x02 | 0xA6 | CMD_GPIO_LOCAL_WRITE_IND | Local host has written the GPIOs of the remote device | 7.7.11 |
| 0x02 | 0xA8 | CMD_GPIO_REMOTE_WRITECONFIG_IND | Remote device has configured the GPIOs of the module | 7.7.9 |
| 0x02 | 0xA9 | CMD_GPIO_REMOTE_WRITE_IND | Remote device has written the GPIOs of the module | 7.7.10 |
| 0x02 | 0xC4 | CMD_TXCOMPLETE_RSP | Data has been sent | 7.3.2 |
| 0x02 | 0xC6 | CMD_CHANNELOPEN_RSP | Channel open, data transmission possible | 7.2.4 |
| 0x02 | 0xCC | CMD_BEACON_RSP | Advertising package detected | 7.1.5 |

Table 70: Message overview: Indications

8. UserSettings - Module configuration values

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



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



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

8.1. FS_DeviceInfo: Read the chip type and OS version

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

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

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

OS version:

0x00B6 : Softdevice S140 6.1.1.

0x00C1 : Softdevice S140 7.0.0.

0x00CA : Softdevice S140 7.0.1.

Package variant:

0x2004: QFN - QI

0x2005: WLCSP - CK

Chip ID:

0x00052840: nRF52840

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

Table 71: nRF52840 IC revision overview

8.1.1. Example 1

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

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

Response CMD_GET_CNF: Successfully read out the device info (with Byte order changed to MSB first):

OS version = 0x00B6 (Softdevice S140 6.1.1)

Build code = 0x41414300 (AAC0)

Package variant = 0x2004 (QFN)

Chip ID = 0x00052840

Please note that LSB is transmitted first in case of parameters with more than 1 Byte length.

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

8.2. FS_FWVersion: Read the firmware version

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

This setting contains the firmware version of the module.

8.2.1. Example 1

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

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

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

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

8.3. FS_MAC: Read the MAC address

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

This setting contains the unique MAC address of the module.

8.3.1. Example 1

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

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

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

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

8.4. FS_BTMAC: 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, it is set to the MAC address bought by Würth Elektronik eiSos which consists of the Würth Elektronik eiSos MAC ID 0x0018DA followed by the FS_SerialNumber of the module.

Nevertheless, 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 association [5]. If set to 0xFFFFFFFFFFFF, the Proteus-III-SPI uses the default address mentioned above.

Please note that LSB is transmitted first in all commands.

8.4.1. 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 |

8.4.2. 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 |

8.5. FS_SerialNumber: Read the serial number of the module

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

This setting contains the serial number of the module.

8.5.1. Example 1

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

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

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

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

8.6. RF_DeviceName: Modify the device name

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|---------------|--------------------|---------------|-------------|-----------------|
| 2 | RF_DeviceName | See description | "Prot3" | read/write | 1-31 |



This parameter is using MSB first notation.

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



The maximum size of the device name that fits into an advertising packet is 5 Bytes in default configuration. With help of the user setting RF_AdvertisingFlags it can be extended. Longer device names will be shortened and declared as "Shortened Local Name" in the advertising packet. The full device name is included in the GAP.

8.6.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.6.2. Example 2

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

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

Response CMD_GET_CNF: Successfully read out the module 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 |

8.7. RF_StaticPaskey: Modify the static paskey

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|-----------------|--------------------|---------------|-------------|-----------------|
| 18 | RF_StaticPaskey | See description | "123123" | read/write | 6 |

This setting determines the static pass key of the peripheral device used for authentication. If the static pass key security mode is enabled by the peripheral, this key must be entered in the central device. In case of a Proteus-III-SPI central, the command to enter this pass key during connection setup is the CMD_PASKEY_REQ.

The permissible characters are ranging from 0x30 to 0x39 (both included) which are ASCII numbers (0-9). This is due to the fact that mobile phones prefer numbers only for the passkey.

8.7.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.7.2. Example 2

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

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

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

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

8.8. RF_SecFlags: Modify the security settings

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|-------------|--------------------|---------------|-------------|-----------------|
| 12 | RF_SecFlags | See description | 0 | read/write | 1 |

This 8-Bit field configures security settings of the module. Chapter 5.6 contains further information about secure connections.



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



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



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

| Bit no. | Description | | |
|---------|--|-----------------------------------|--|
| 2 : 0 | Security mode configuration. Depending on its value, different modes are chosen when setting up a secure connection. In firmware version 2.1.0 and newer the peripheral decides which is the minimum security level to access its data. | | |
| | 0x0 | No security | Data is transmitted without authentication and encryption. |
| | 0x1 | LESC just works level 1.2 | Each time a connection is established, new random keys are exchanged in advance to use them for data encryption. This mode uses the "just works" method with keys generated by the LESK method (low energy elliptic curve) |
| | 0x2 | Just works level 1.2 | Each time a connection is established, new random keys are exchanged in advance to use them for data encryption. This mode uses the "just works" method. |
| | 0x3 | Static pass key level 1.3 | For authentication, the RF_StaticPasskey is used. If the peripheral uses this method, the central device must enter the correct passkey to finalize the connection. |
| | 0x4 | LESC numeric comparison level 1.4 | For authentication, the peripheral and central device output a pass key. The central and peripheral device must confirm the pass key to finalize the connection in case both keys coincide. Otherwise reject it to cancel the connection. The pass key is generated using the LESK method (low energy elliptic curve). |
| | 0x5 | LESC pass key level 1.4 | For authentication, the peripheral outputs a passkey. The central device must enter this pass key to finalize the connection. The pass key is generated using the LESK method (low energy elliptic curve). |
| | others | | Reserved |
| 3 | SECFLAGS_BONDING_ENABLE: If this Bit is set, bonding is enabled when using one of the pairing methods. Bonding data of up to 32 devices will be stored in the flash. If bonding storage is full, the bonding information that has not been used for the longest period will be removed. | | |
| 4 | SECFLAGS_BONDEDCONNECTIONSONLY_ENABLE: If this Bit is set, only bonded peer devices are allowed to connect. All connection requests from any unbonded peer device are rejected. In case this restriction shall be disabled temporarily to setup a bonding to a new peer device, use the command CMD_ALLOWUNBONDEDCONNECTIONS_REQ, which temporarily disables this restriction. If this feature is enabled, the maximum number of bonded devices is reduced to 8. | | |
| 7 : 5 | Reserved | | |

Table 72: Security configuration flags

8.8.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.8.2. Example 2

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

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

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

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

8.9. RF_ScanFlags: Modify the scan behavior

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|--------------|--------------------|---------------|-------------|-----------------|
| 13 | RF_ScanFlags | See description | 0 | read/write | 1 |

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

| Bit no. | Description |
|---------|--|
| 0 | If this Bit is set, an active scan is performed when using CMD_SCANSTART_REQ. In this case, after receiving an advertising packet a scan request is sent to the advertising module that returns a scan response containing additional information. For the communication of Proteus-III-SPI modules, active scanning is only needed when using Beacons. In this case, it is enabled automatically by the firmware. Note that if receiving raw Beacon data is enabled (RF_BeaconFlags is set to 0x02) and raw scan response packets should be received as well, it is necessary to set this bit manually. Also note that active scanning increases the current consumption. |
| 7 : 1 | Reserved |

Table 73: Scan configuration flags

8.9.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.9.2. Example 2

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

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

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



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

8.10. RF_BeaconFlags: Interpret the advertising data

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|----------------|--------------------|---------------|-------------|-----------------|
| 14 | RF_BeaconFlags | See description | 0 | read/write | 1 |

This field configures the reception of Beacons.

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

Table 74: Beacon configuration flags



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



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

8.10.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.10.2. Example 2

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

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

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

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

8.11. RF_AdvertisingTimeout: Modify the advertising timeout

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

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



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

8.11.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.11.2. Example 2

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

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

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

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

8.12. RF_AdvertisingFlags: Configure the advertising packet

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|---------------------|--------------------|---------------|-------------|-----------------|
| 29 | RF_AdvertisingFlags | 0,1,2 | 0 | read/write | 1 |

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

| Bit no. | Description |
|---------|---|
| 1 : 0 | Define the content of the advertising packet |
| | 0x0 |
| | Command mode: Advertising packet contains 5 bytes device name RF_DeviceName and the TX power and the UUID. |
| | Peripheral only mode: Advertising packet contains 8 bytes device name generated by the FS_BTMAC (A-123456 in case the FS_BTMAC is 0x0018DA123456) and the UUID. |
| | 0x1 |
| 0x2 | Advertising packet contains 8 bytes device name RF_DeviceName and the UUID. |
| | Advertising packet contains 26 bytes device name RF_DeviceName and the TX power (only in case the RF_DeviceName is equal or lower than 23 bytes). The UUID is part of the scan response packet, that can be only received by an active scan (see user settings RF_ScanFlags). Using this option, the transmission of beacons is not more supported. |
| | Others |
| 7 : 2 | Reserved. |

Table 75: Advertising packet configuration flags

8.12.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.12.2. Example 2

Request the RF_AdvertisingFlags using CMD_GET_REQ with settings index 29:

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

Response CMD_GET_CNF: Successfully read out the value 0x00.

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

8.13. RF_ScanFactor: Modify the scan factor

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

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

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

8.13.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.13.2. Example 2

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

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

Response CMD_GET_CNF: Successfully read out the value 2.

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

8.14. RF_ScanTiming: Modify the scan timing

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

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

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

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

Further information:

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

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



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



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

8.14.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.14.2. Example 2

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

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

Response CMD_GET_CNF: Successfully read out the value 4.

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

8.15. RF_ConnectionTiming: Modify the connection timing

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|---------------------|--------------------|---------------|-------------|-----------------|
| 8 | RF_ConnectionTiming | 0 - 12 | 2 | read/write | 1 |

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

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

| RF_ConnectionTiming | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|------------------------------------|-----|-----|----|-----|-----|-----|------|------|------|
| Minimum connection interval [ms] | 7.5 | 7.5 | 15 | 30 | 45 | 100 | 195 | 750 | 2000 |
| Maximum connection interval [ms] | 7.5 | 30 | 75 | 100 | 250 | 500 | 1000 | 1995 | 4000 |
| Connection supervision timeout [s] | 4 | 4 | 4 | 4 | 4 | 4 | 6 | 6 | 25 |

| RF_ConnectionTiming | 9 | 10 | 11 | 12 |
|------------------------------------|-------|-------|----|----|
| Minimum connection interval [ms] | 7.5 | 11.25 | 15 | 15 |
| Maximum connection interval [ms] | 11.25 | 20 | 15 | 30 |
| Connection supervision timeout [s] | 4 | 4 | 4 | 4 |



Please note that the smallest minimum connection interval supported by Android is 11.25 ms. Thus profile 0 cannot be used on Android devices. Further note that iOS supports only profiles, where its minimum connection interval is 15 ms or a multiple of it.

Further information:

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

connection interval settings the peripheral requests the central to accept the peripheral's settings after 5s. The central accepts these settings, and thus the peripheral's connection interval is used.

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

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



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



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

8.15.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.15.2. Example 2

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

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

Response CMD_GET_CNF: Successfully read out the value 1.

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

8.16. RF_TXPower: Modify the output power

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

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

| Permissible values | | | | | | | |
|-------------------------------|------|------|------|------|------|------|------|
| Decimal [dBm] | -40 | -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 |

8.16.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.16.2. Example 2

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

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

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



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

8.17. RF_SPPBaseUUID: Configure the SPP base UUID

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

Set the base UUID of the WE SPP-like profile. For more information about the UUID definition, please refer to chapter 11 . 2.



Please note that the UUID 0xFB349B5F8000008000100000xxxx0000 (0000xxxx-0000-1000-8000-00805f9b34fb) is reserved for 16 Bit UUIDs and must not be used here.

8.17.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.17.2. Example 2

Request the base UUID of the module using CMD_GET_REQ:

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

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

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

8.18. RF_SPPServiceUUID: Configure the SPP service UUID

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

Set the service UUID of the WE SPP-like profile. For more information about the UUID definition, please refer to chapter 11.2.

The service UUID can be every value, but must be different from RF_SPPTXUUID and RF_SPPRXUUID.

8.18.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.18.2. Example 2

Request the service UUID of the module using CMD_GET_REQ:

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

Response CMD_GET_CNF: Successfully read out the value 0x1234.

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

8.19. RF_SPPRXUUID: Configure the SPP RX UUID

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

Set the RX UUID of the WE SPP-like profile. For more information about the UUID definition, please refer to chapter 11.2.



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

8.19.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.19.2. Example 2

Request the service UUID of the module using CMD_GET_REQ:

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

Response CMD_GET_CNF: Successfully read out the value 0x1234.

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

8.20. RF_SPPTXUUID: Configure the SPP TX UUID

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

Set the TX UUID of the WE SPP-like profile. For more information about the UUID definition, please refer to chapter 11.2.



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

8.20.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.20.2. Example 2

Request the service UUID of the module using CMD_GET_REQ:

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

Response CMD_GET_CNF: Successfully read out the value 0x1234.

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

8.21. RF_Appearance: Configure the appearance of the device

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

The user setting RF_Appearance specifies the appearance of the Bluetooth® devices. It's a 2 Byte field defined by the Bluetooth® SIG. Please check the Bluetooth® assigned numbers [6].

8.21.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.21.2. Example 2

Request the RF_Appearance using CMD_GET_REQ:

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

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

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

8.22. SPI_ConfigIndex: Modify the SPI mode

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|-----------------|--------------------|---------------|-------------|-----------------|
| 30 | SPI_ConfigIndex | See description | 2 | read/write | 1 |

This parameter defines the SPI mode and the *SPI_INT*'s active level. The permissible values are listed in the following table.

| SPI_ConfigIndex | SPI mode | CPOL | CPHA | <i>SPI_INT</i> / <i>WAKE_UP</i> active level |
|-----------------|----------|-----------------|----------------------|--|
| 0 | 0 | Clock Idle Low | First egde sampling | High |
| 1 | 1 | Clock Idle Low | Second egde sampling | High |
| 2 | 2 | Clock Idle High | First egde sampling | High |
| 3 | 3 | Clock Idle High | Second egde sampling | High |
| 4 | 0 | Clock Idle Low | First egde sampling | Low |
| 5 | 1 | Clock Idle Low | Second egde sampling | Low |
| 6 | 2 | Clock Idle High | First egde sampling | Low |
| 7 | 3 | Clock Idle High | Second egde sampling | Low |



After changing the SPI_ConfigIndex using the CMD_SET_REQ the module restarts using the new settings. Therefore don't forget to update the settings of the connected host to be able to further use the module's SPI.

8.22.1. Example 1

Set the SPI_ConfigIndex to 0x00 (SPI mode 0, *SPI_INT* high active) using CMD_SET_REQ with settings index 30

| Start signal | Command | Length | Settings index | Parameter | CS |
|--------------|---------|-----------|----------------|-----------|------|
| 0x02 | 0x11 | 0x02 0x00 | 0x1E | 0x00 | 0x0F |

Response CMD_SET_CNF: Successfully modified the setting.

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

8.22.2. Example 2

Request the SPI_ConfigIndex of the module 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 0x02 (SPI mode 2, *SPI_INT* high active).

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

8.23. RF_CompanyIdentifier: Configure the company identifier of the device

| Settings index | Designation | Permissible values | Default value | Permissions | Number of bytes |
|----------------|----------------------|--------------------|---------------|-------------|-----------------|
| 35 | RF_CompanyIdentifier | 0-65535 | 0x031A | read/write | 2 |

The user setting RF_CompanyIdentifier specifies the company identifier of the Proteus-III-SPI. It's a 2 Byte field defined by the Bluetooth® SIG, which is used in the manufacturer specific fields of the advertising packet (see CMD_SETBEACON_REQ). Please check the Bluetooth® assigned numbers [6].

8.23.1. Example 1

Set the company id to "Motorola" (0x0008) using CMD_SET_REQ with settings index 35

| Start signal | Command | Length | Settings index | Parameter | CS |
|--------------|---------|-----------|----------------|-----------|------|
| 0x02 | 0x11 | 0x03 0x00 | 0x23 | 0x08 0x00 | 0x3B |

Response CMD_SET_CNF: Successfully modified the setting.

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

8.23.2. Example 2

Request the RF_CompanyIdentifier using CMD_GET_REQ:

| Start signal | Command | Length | Settings index | CS |
|--------------|---------|-----------|----------------|------|
| 0x02 | 0x10 | 0x01 0x00 | 0x23 | 0x30 |

Response CMD_GET_CNF: Successfully read out the value 0x031A (Würth Elektronik eiSos).

| Start signal | Command 0x40 | Length | Status | Parameter | CS |
|--------------|----------------|-----------|--------|-----------|------|
| 0x02 | 0x50 | 0x03 0x00 | 0x00 | 0x1A 0x03 | 0x48 |

8.24. CFG_Flags: Configure the module

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

The user setting CFG_Flags specifies various module features.

| Bit no. | Name | Description |
|---------|----------------------------|---|
| 0 | High throughput mode | Set this Bit to 1 to enable the high throughput mode. |
| 1 | Long range connection mode | Set this Bit to 1 to enable the mode using the LE Coded mode during connection setup. |
| 2 | GPIO remote config. | Set this Bit to 1 to block the GPIO configuration via remote device. |
| 4 | Disconnect on mismatch | Set this Bit to 1 to disconnect in case the central device does not respect the peripheral's connection interval. |
| 3, 5-15 | Reserved | Do not modify. |



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

8.24.1. Example 1

Enable the high throughput mode using CMD_SET_REQ with settings index 28

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.24.2. Example 2

Request the CFG_Flags using CMD_GET_REQ:

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

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

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

8.25. DIS_ManufacturerName: Configure the manufacturer name

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

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



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

8.25.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.25.2. Example 2

Request the manufacturer name of the DIS profile using CMD_GET_REQ:

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

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

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

8.26. DIS_ModelNumber: Configure the model number

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

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



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

8.26.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.26.2. Example 2

Request the model number of the DIS profile using CMD_GET_REQ:

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

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

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

8.27. DIS_SerialNumber: Configure the serial number

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

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



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

8.27.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.27.2. Example 2

Request the serial number of the DIS profile using CMD_GET_REQ:

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

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

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

8.28. DIS_HWVersion: Configure the HW version

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

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



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

8.28.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.28.2. Example 2

Request the hardware version of the DIS profile using CMD_GET_REQ:

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

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

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

8.29. DIS_SWVersion: Configure the SW version

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

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



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

8.29.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.29.2. Example 2

Request the software version of the DIS profile using CMD_GET_REQ:

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

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

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

8.30. DIS_Flags: Configure the device information service

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

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

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

8.30.1. Example 1

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

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

Response CMD_SET_CNF: Successfully modified the setting.

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

8.30.2. Example 2

Request the DIS_Flags using CMD_GET_REQ:

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

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



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

| Settings index | Designation | Summary | Permissible values | Default value | Permissions | Number of bytes |
|----------------|-----------------------|--|-------------------------|-----------------|-------------|-----------------|
| 1 | FS_FWVersion | Version of the firmware | - | - | read | 3 |
| 2 | RF_DeviceName | Name of the module | See description | "Prot3" | read/write | 1-31 |
| 3 | FS_MAC | MAC address of the module | - | - | read | 6 |
| 4 | FS_BTMAC | Bluetooth® LE conform MAC address of the module | - | - | read/write | 6 |
| 7 | RF_AdvertisingTimeout | Time [s] after advertising stops. LSB first | 0 (infinite), 1 - 65535 | 0 | read/write | 2 |
| 8 | RF_ConnectionTiming | Module connection timing configuration | 0 - 12 | 2 | read/write | 1 |
| 9 | RF_ScanTiming | Module advertising and scanning timing configuration | 0 - 11 | 1 | read/write | 1 |
| 10 | RF_ScanFactor | Factor between scan interval and scan window | 1 - 10 | 2 | read/write | 1 |
| 12 | RF_SecFlags | Security settings of the module | See description | 0 | read/write | 1 |
| 13 | RF_ScanFlags | Scan settings of the module | See description | 0 | read/write | 1 |
| 14 | RF_BeaconFlags | Beacon settings of the module | See description | 0 | read/write | 1 |
| 15 | FS_DeviceInfo | Information about the chip | - | - | read | 12 |
| 16 | FS_SerialNumber | Serial number of the module | - | - | read | 3 |
| 17 | RF_TXPower | Output power [dBm] Two's complement | See description | 8 | read/write | 1 |
| 18 | RF_StaticPasskey | 6 digit pass key | See description | "123123" | read/write | 6 |
| 19 | DIS_Flags | Flags for the DIS | 0 - 255 | 0 | read/write | 1 |
| 20 | DIS_ManufacturerName | Manufacturer name field of the DIS | See description | "Default" | read/write | 1-64 |
| 21 | DIS_ModelNumber | Model number field of the DIS | See description | "Default" | read/write | 1-64 |
| 22 | DIS_SerialNumber | Serial number field of the DIS | See description | "Default" | read/write | 1-64 |
| 23 | DIS_HWVersion | HW version field of the DIS | See description | "Default" | read/write | 1-16 |
| 24 | DIS_SWVersion | SW version field of the DIS | See description | "Default" | read/write | 1-16 |
| 25 | RF_Appearance | Appearance | 0-65535 | 0 | read/write | 2 |
| 26 | RF_SPPBaseUUID | Base UUID of the WE SPP-like profile | See description | See description | read/write | 16 |

| | | | | | | |
|----|----------------------|--------------------|-----------------|--------|------------|---|
| 28 | CFG_Flags | CFG Flags | See description | 0 | read/write | 2 |
| 29 | RF_AdvertisingFlags | Advertising Flags | 0,1,2 | 0 | read/write | 1 |
| 30 | SPI_ConfigIndex | SPI mode | See description | 2 | read/write | 1 |
| 32 | RF_SPPServiceUUID | See description | See description | 0x0001 | read/write | 2 |
| 33 | RF_SPPRXUUID | See description | See description | 0x0002 | read/write | 2 |
| 34 | RF_SPPTXUUID | See description | See description | 0x0003 | read/write | 2 |
| 35 | RF_CompanyIdentifier | Company identifier | 0-65535 | 0x031A | read/write | 2 |

Table 76: Table of settings

9. Timing parameters

9.1. Reset and sleep

After power-up, resetting the module or waking the module from sleep a `CMD_GETSTATE_CNF` is sent to the serial interface as soon as the module is ready for operation.

| Description | Typ. | Unit |
|-------------------------|------|------|
| Ready after reset/sleep | 77 | ms |

9.2. Bluetooth LE timing parameters

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

9.3. Connection establishment

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

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

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

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

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



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

9.4. Connection based data transmission

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

The following image shows the command sequence when sending data:

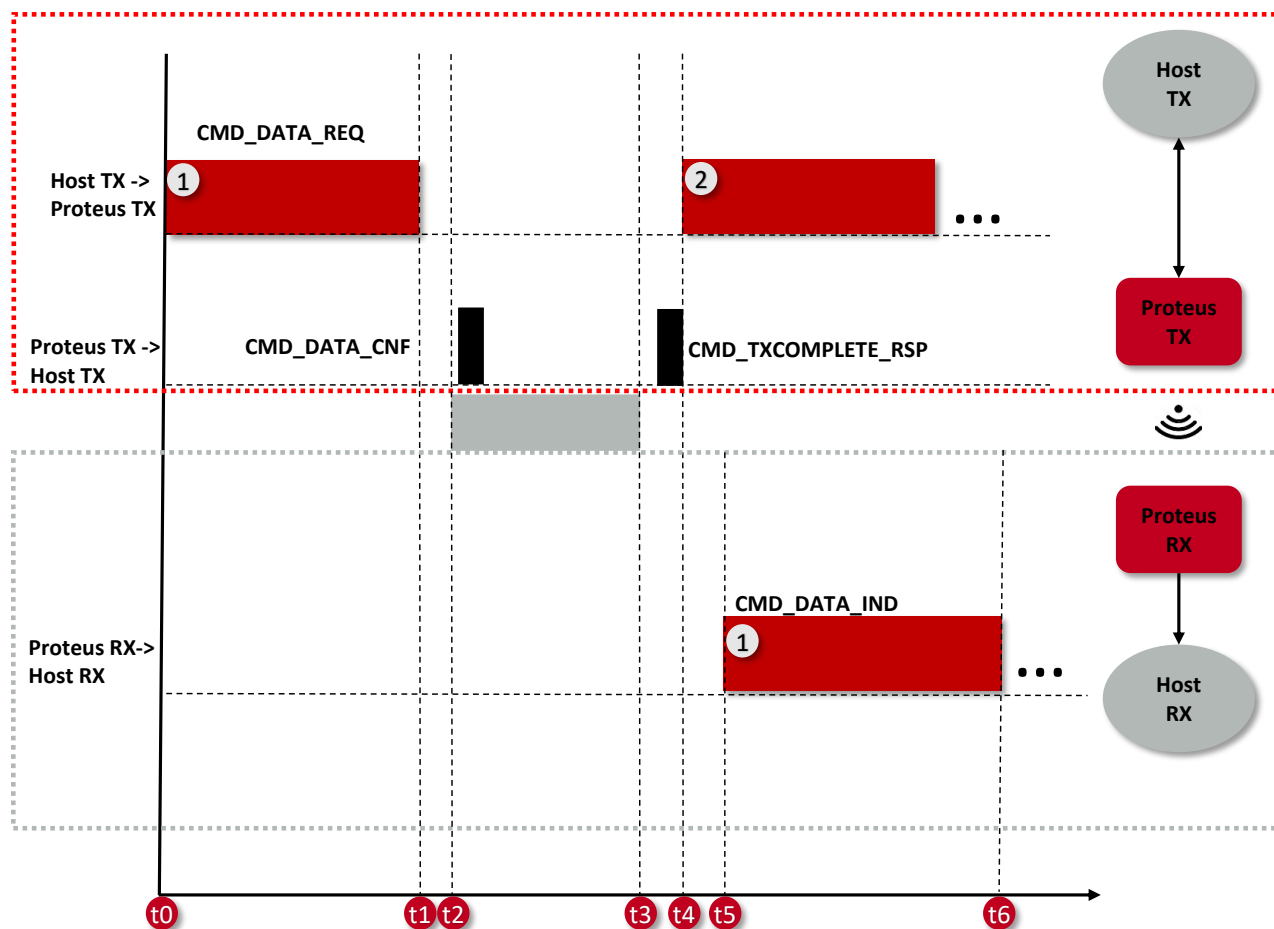


Figure 12: Command sequence when transmitting data

9.4.1. Maximum data throughput

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

- Two Proteus-III-SPI radio modules connected to a fast micro controller (STM32 on NUCLEO-L476RG)
- Radio mode as specified in the table below (125 kb/s (long range mode), 1 Mbit/s or 2 Mbit/s)
- Fastest connection interval of 7.5 ms (RF_ConnectionTiming equals 0)
- SPI datarate of 8MBit
- High throughput mode to transmit 964 Bytes payload (i.e. 4 radio frames, each with 241 byte payload) per connection interval (Bit 0 of CFG_Flags)

| Radio mode | t1-t0 [ms] (SPI RX) | t4-t0 [ms] (Host TX Period) | t6-t5 [ms] (SPI TX) | t6-t0 [ms] (End-to-end delay) | 964/(t4-t0) [kByte/s] (Throughput) |
|------------|------------------------|--------------------------------|------------------------|-------------------------------------|--|
| 125 kb/s | 1.01 | 307 | 1.02 | 306.8 | 3.14 |
| 1 Mb/s | 1.01 | 37.5 | 1.02 | 37.38 | 25.7 |
| 2 Mb/s | 1.01 | 15 | 1.02 | 14.96 | 64.3 |

Table 77: Maximum throughput timings, packet error rate = 0%



Please note that data transmission to/from smart devices typically do not achieve this speed due to latency caused by the smart device and its software and apps or even missing hardware features such as Bluetooth® LE 5.0 full feature support.

10. Remote GPIO control

The Proteus-III-SPI allows to control free GPIOs via remote access. Chapter 7.7 contains the description of the necessary commands.

To use the remote GPIO control feature of the Proteus-III-SPI, the GPIOs of interest must be configured first. This can be done in two ways. Either by the local host (see figure 13), when the radio module is in idle mode (not connected via Bluetooth® LE), or via the connected remote device (see figure 14).

In case of the local host, it must send a `CMD_GPIO_LOCAL_WRITECONFIG_REQ` command to the radio module via SPI. In case of the remote device, it must setup a Bluetooth® LE connection to the remote device first and send a `CMD_GPIO_REMOTE_WRITECONFIG_REQ` command to the radio module via Bluetooth® LE afterwards.

The configuration is stored in flash memory, such that it is retained also after a device restart. It can be reset to default by using the `CMD_FACTORYRESET_REQ` command.

The configuration can be also read out using the respective commands, `CMD_GPIO_LOCAL_READCONFIG_REQ` via local host or `CMD_GPIO_REMOTE_READCONFIG_REQ` via remote device.

If the configuration has been done, the configured GPIOs can be controlled by the local host controller or by any remote device.

To control a GPIO via local host controller just send the respective commands, `CMD_GPIO_LOCAL_WRITE_REQ` for setting GPIO output values (see figure 17), or `CMD_GPIO_LOCAL_READ_REQ` for reading GPIO values (see figure 18). Each time the GPIOs are written via local host, the connected remote device is informed using a `CMD_GPIO_LOCAL_WRITE_IND` message.

To control a GPIO via remote device, first setup a Bluetooth® LE connection to the radio module and send the respective commands, `CMD_GPIO_REMOTE_WRITE_REQ` for setting GPIO output values (see figure 19), or `CMD_GPIO_REMOTE_READ_REQ` for reading GPIO values (see figure 20). Each time the GPIOs are written via remote connection, the local host is informed using a `CMD_GPIO_REMOTE_WRITE_IND` message.

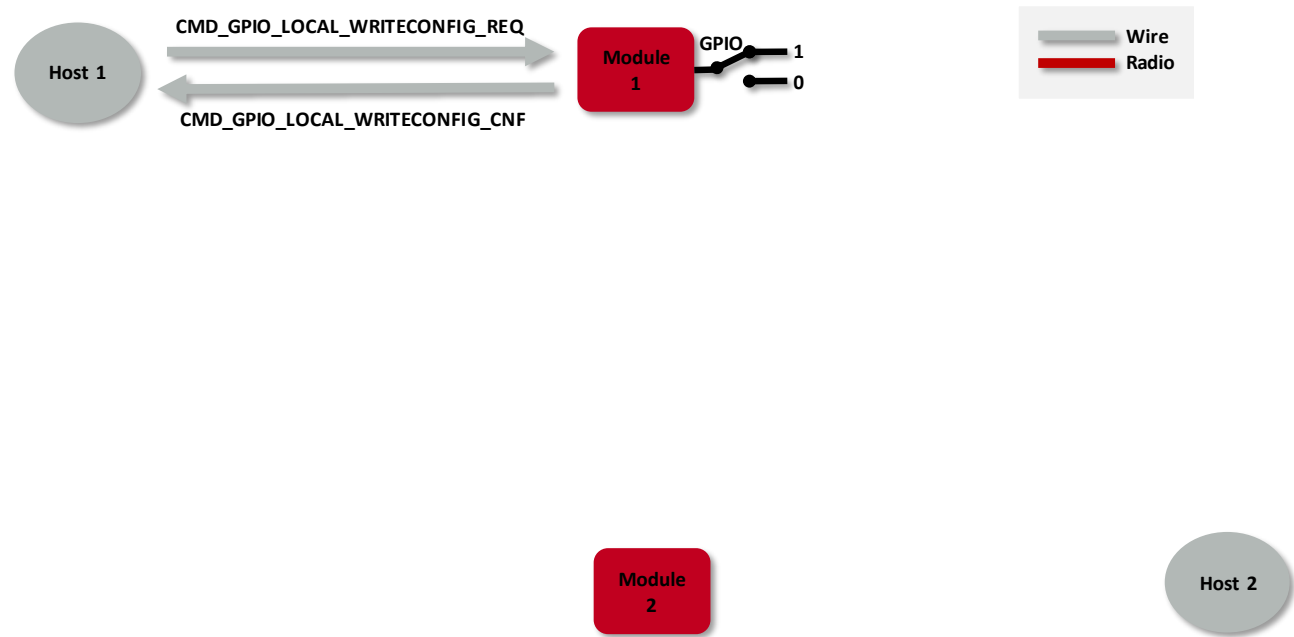


Figure 13: Configure the local GPIOs via local host

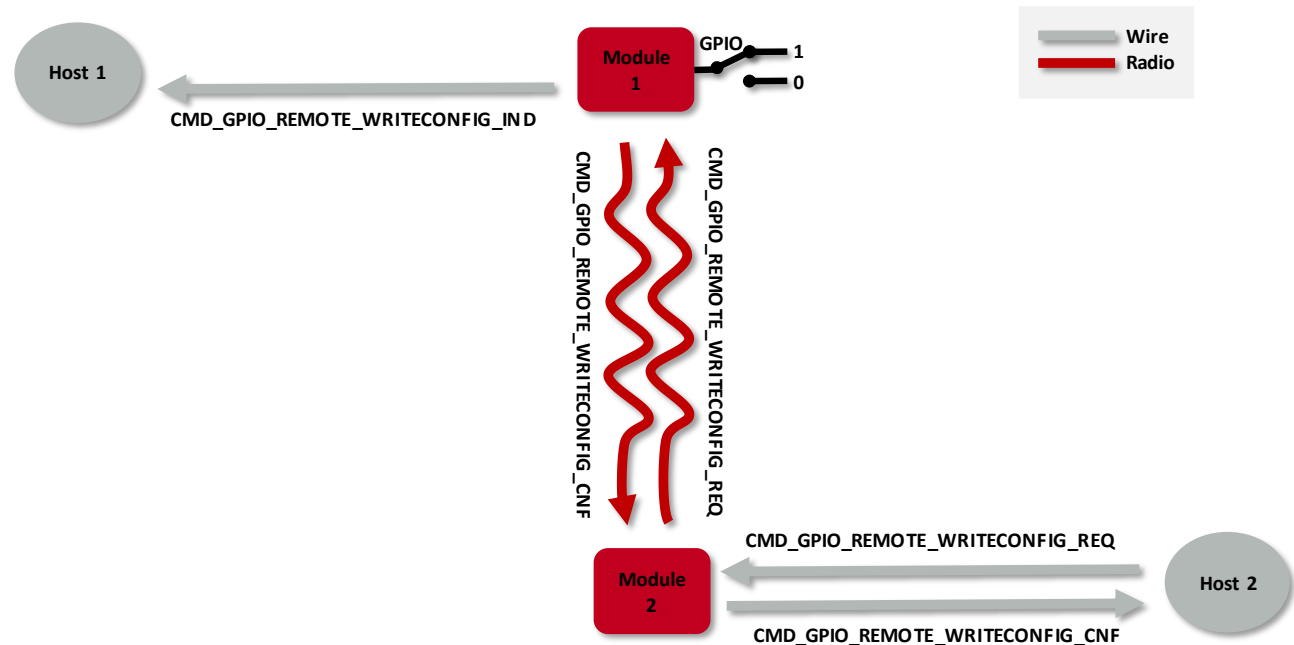


Figure 14: Configure the local GPIOs via remote device host

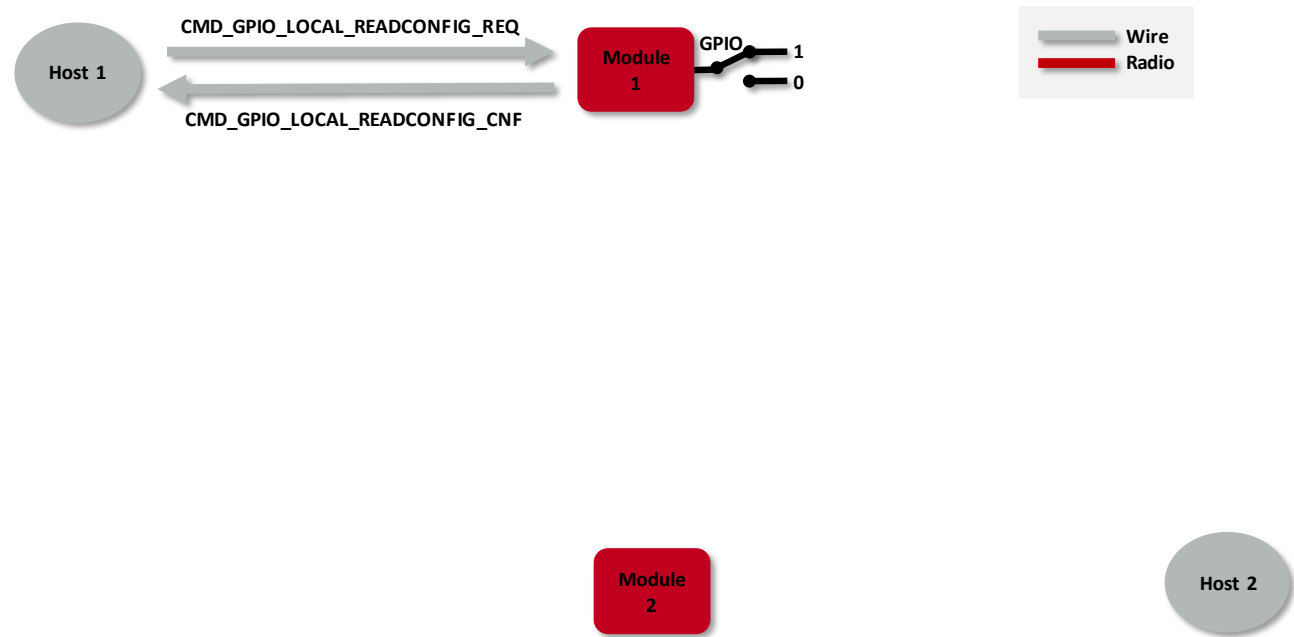


Figure 15: Read the configuration of the local GPIOs via local host

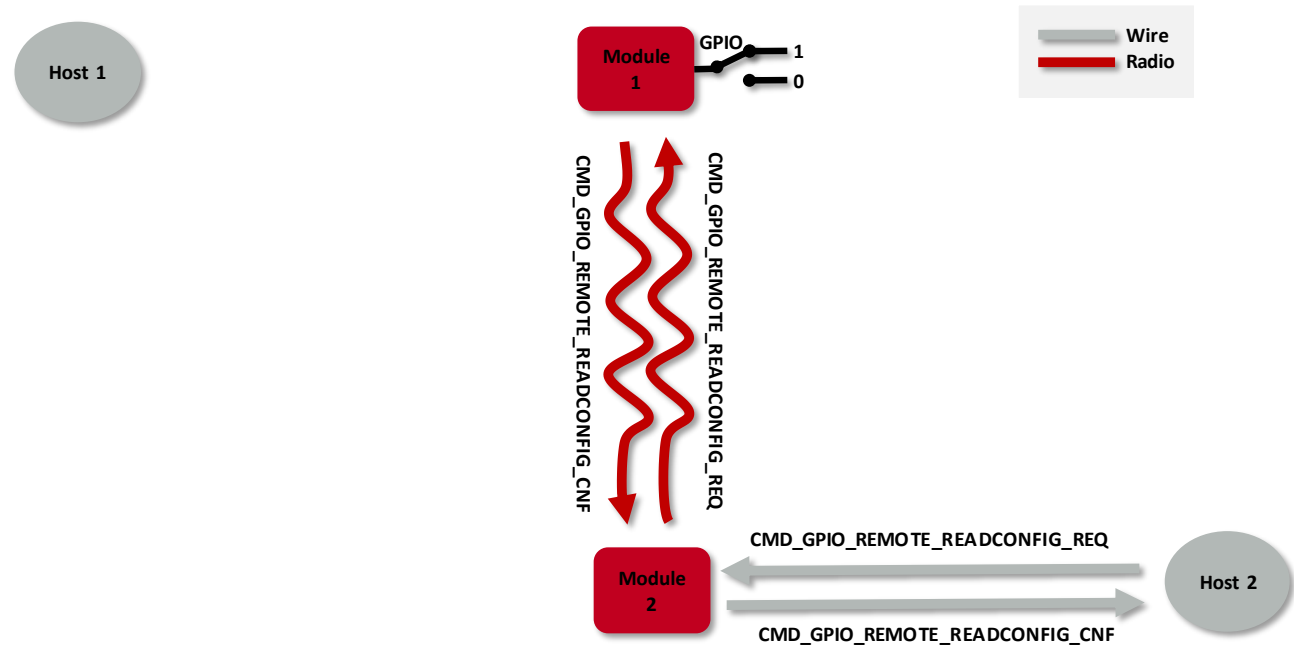


Figure 16: Read the configuration of the local GPIOs via remote device host

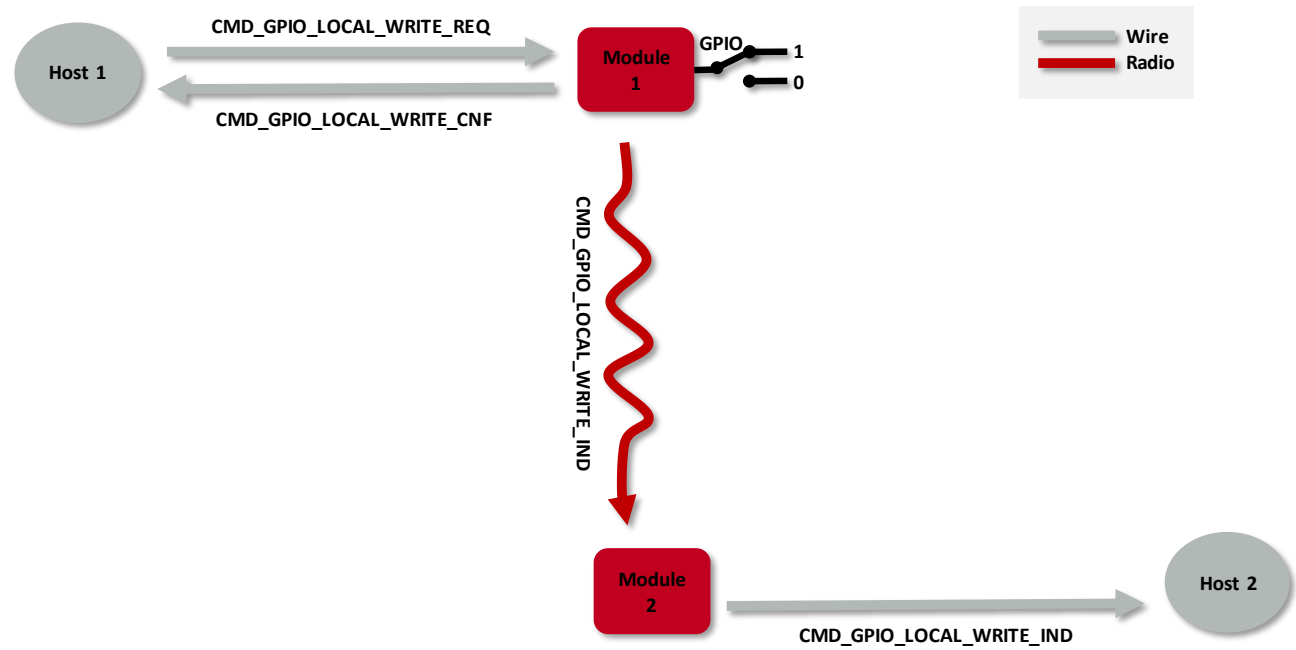


Figure 17: Set the output value of a GPIO via host controller

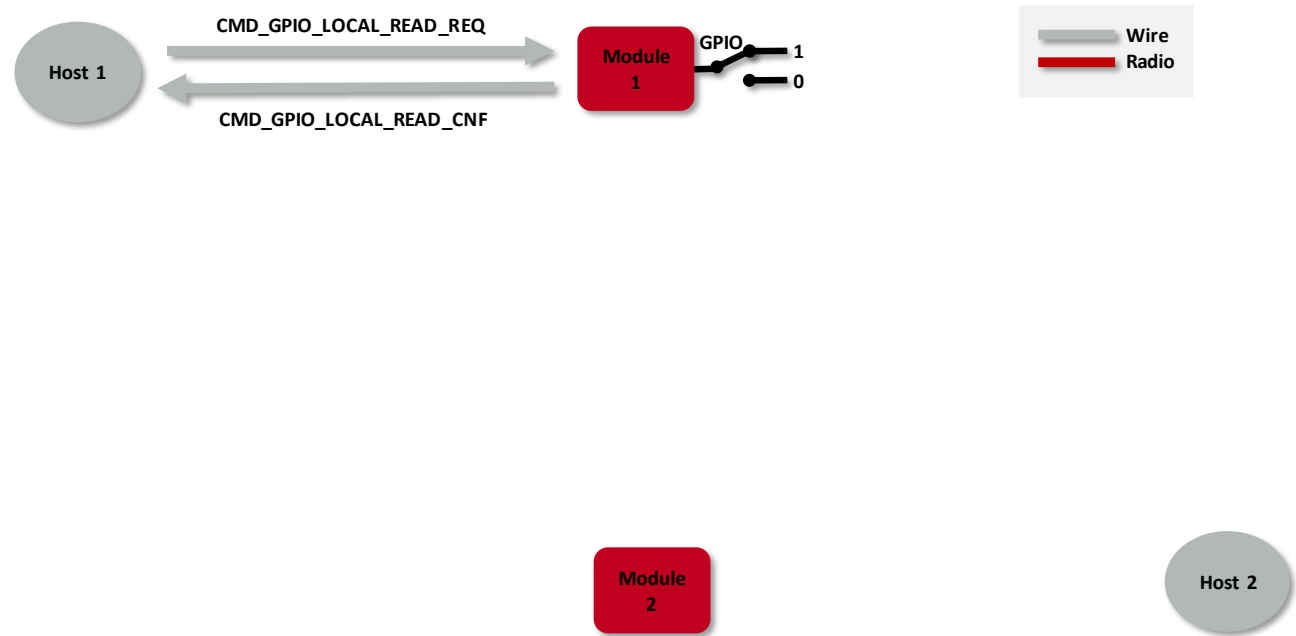


Figure 18: Read the input value of a GPIO via host controller

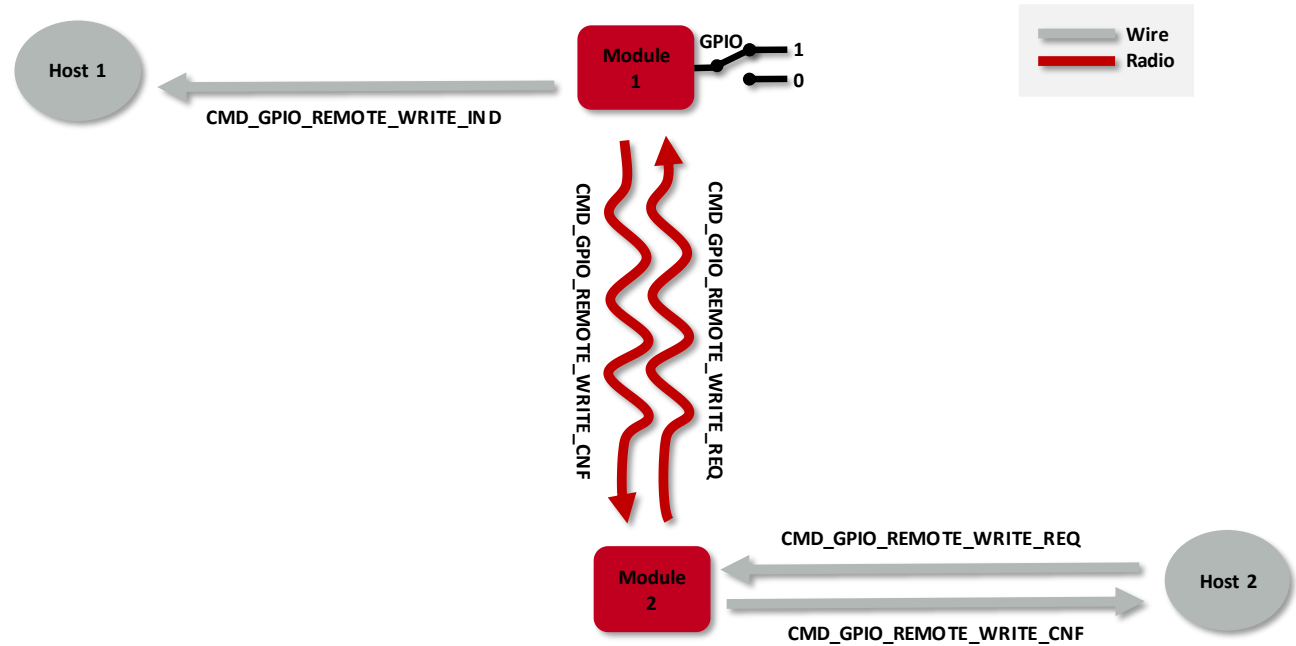


Figure 19: Set the output value of a GPIO via remote device

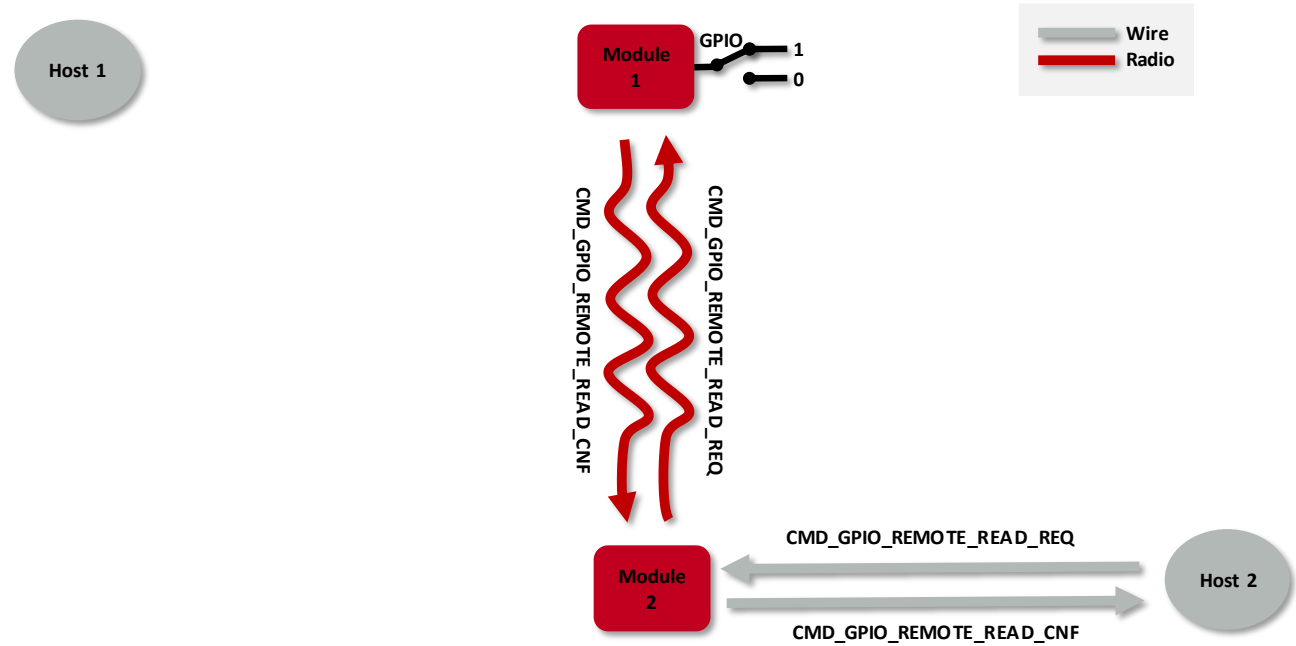


Figure 20: Read the input value of a GPIO via remote device

10.1. PWM

In case a GPIO shall run as PWM, the parameters "ratio" and "period" define its behavior. The parameter "period" defines the period of the PWM signal. The parameter "ratio" defines the ratio between on- and off-time. As an example, a ratio of 0x40 corresponds to 25% on-time and 75% off-time.

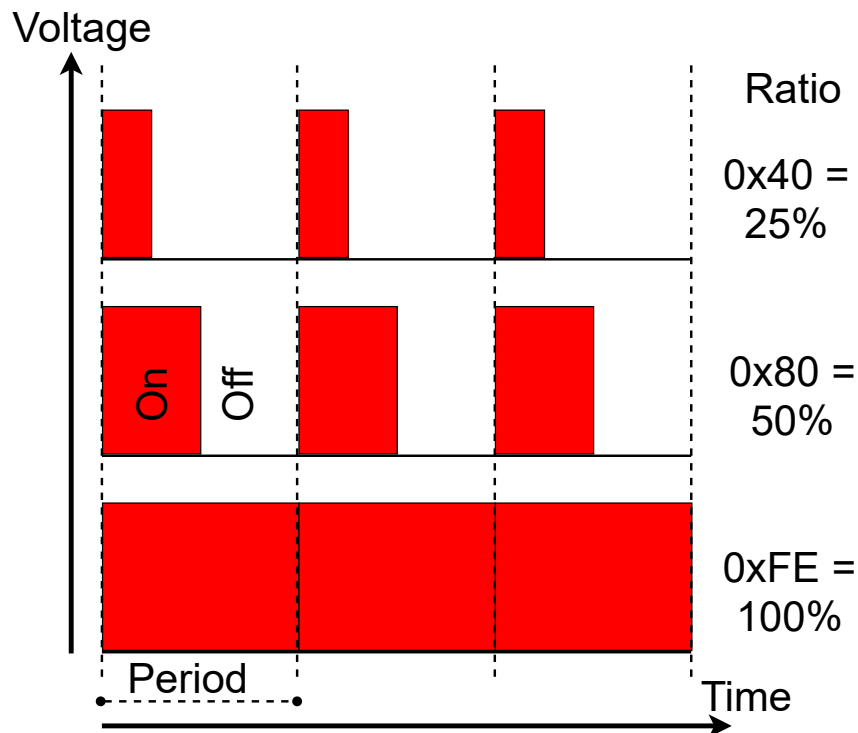
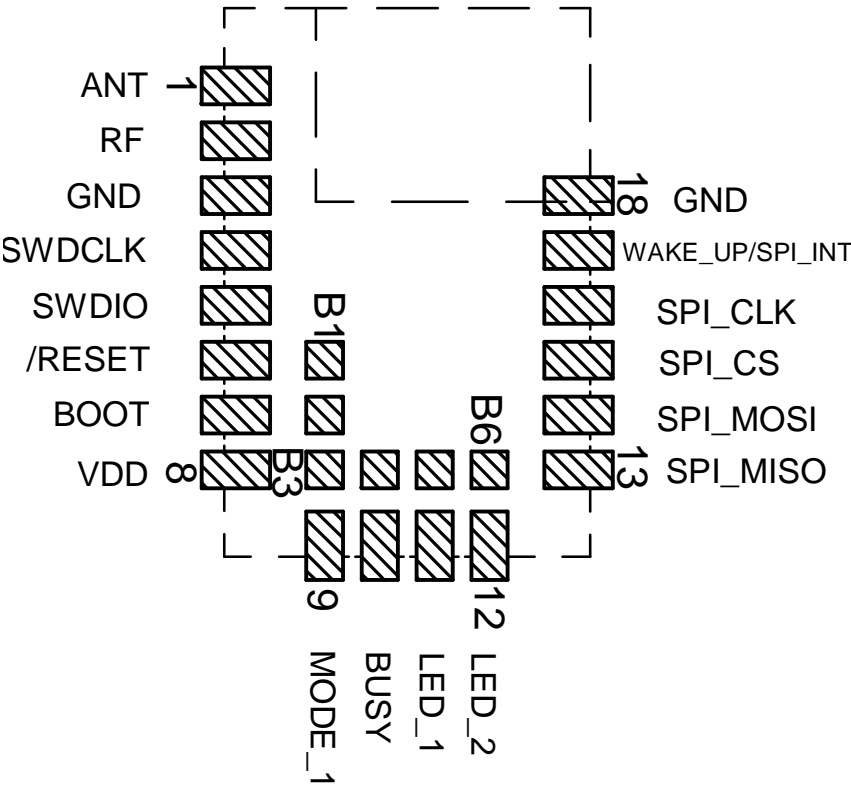


Figure 21: PWM behaviour

10.2. Supported GPIO_IDs for remote and local control

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



| No | GPIO_ID | Supported functions |
|----|---------|---------------------|
| B1 | 1 | Input, Output |
| B2 | 2 | Input, Output |
| B3 | 3 | Input, Output, PWM |
| B4 | 4 | Input, Output, PWM |
| B5 | 5 | Input, Output, PWM |
| B6 | 6 | Input, Output, PWM |

Table 78: Supported GPIO_IDs

11. Customizing the Proteus-III-SPI

11.1. DIS - Device information service

Besides the WE SPP-like profile for data transmission, the Proteus-III-SPI contains the so called Device Information Service (DIS), which is disabled by default. The DIS is a standard Bluetooth® LE profile that is recognized by all devices with Bluetooth® capabilities without the need of special implementations.

This profile exposes manufacturer information about a device and is used to personalize the Proteus-III-SPI to fuse with the custom product. It contains the following fields, that can only be modified by updating the respective user setting using the CMD_SET_REQ command:

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

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

11.1.1. Make the module's firmware version visible on the radio

The Proteus-III-SPI receives firmware updates time by time, that contain optional features and improvements. For applications where the firmware version plays a role, the DIS profile can be used to display the firmware version on the radio.

To do that, first read the firmware version from the module using a CMD_GET_REQ command and the user setting FS_FWVersion. Then write the firmware version as a string to the user setting DIS_SWVersion using CMD_SET_REQ command. Finally, enable the corresponding bit in the user setting DIS_Flags to make the firmware version visible on the radio.

11.2. UUID

The UUID is a unique number identifying a Bluetooth® LE profile and thus describing its functions. The Proteus-III-SPI using its standard UUID is compatible to all devices that implement the WE SPP-like profile, whichever device it is integrated.

To suspend this interoperability, the user settings RF_SPPBaseUUID, RF_SPPServiceUUID, RF_SPPTXUUID and RF_SPPRXUUID can be used to modify the UUID of the WE 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 WE SPP-like profile consists of the 128 bit base UUID plus the 16 bit UUIDs for the underlying characteristics and services:

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

Table 79: UUID default values

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

Example:

With the above mentioned default values, the full UUID calculate as

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

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

11.3. Appearance

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

12. Custom firmware

12.1. Custom configuration of standard firmware

The configuration of the standard firmware includes adoption of the non-volatile Usersettings (see chapter 8) to customer requirements and creating a customized product 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.

12.2. Customer specific firmware

A customer specific firmware may include "Custom configuration of standard firmware" plus additional options or functions and tasks that are customer specific and not part of the standard firmware.

Further scheduled firmware updates of the standard firmware will not be applied to this variant automatically. Applying updates or further functions require a customer request and release procedure.

This also results in a customer exclusive module with a unique ordering number.

An example for this level of customization are functions like host-less operation where the module will perform data generation (e.g. by reading a SPI or I²C sensor) and cyclic transmission of this data to a data collector, while sleeping or being passive most of the time.

Also replacing UART with SPI as host communication interface is classified such a custom specific option.

Certification critical changes need to be re-evaluated by an external qualified measurement laboratory. These critical changes may occur when e.g. changing radio parameters, the channel access method, the duty-cycle or in case of various other functions and options possibly used or changed by a customer specific firmware.

12.3. Customer firmware

A customer firmware is a firmware written and tested by the customer himself or a 3rd party as a customer representative specifically for the hardware platform provided by a module.

This customer firmware (e.g. in form of a Intel hex file) will be implemented into the module's production process at our production 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 firmware cannot be applied to this customer firmware solution without a review and verification.

12.4. Contact for firmware requests

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

13. 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.

13.1. 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.

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, Wi-Fi, etc.) communication interface of the module to allow updating the product's firmware. Details are described in the next sections.

13.2. Firmware update using the Proteus-III-SPI OTA bootloader

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

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

| Version of the firmware before the update | Version of the new firmware | App |
|---|-----------------------------|-----------------------------------|
| 1.x.x | 1.x.x | nRF Device Firmware Update [7, 8] |

Table 80: Compatibility matrix

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

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

The bootloader mode has started successfully if *LED_1* has turned on. After the bootloader has started successfully, the module goes into the advertising mode using the name "DFUProteus-III". Now, any Bluetooth® LE device hosting an application that understands the commands of the Nordic nRF52 Bluetooth® LE DFU Bootloader can connect in order to update the Proteus-III-SPI firmware.

The DFU application of the used App (see Table 80) is such an application. For more details, please refer to chapter 13.2.1. As soon as a connection has been set up, *LED_1* turns off again and *LED_2* turns on.



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



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



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




Only signed firmware packages by Würth Elektronik eiSos will be accepted by the Proteus-III-SPI radio module. Signature and encryption ensure the zip package cannot be used for other products.



This method is only applicable if the Proteus-III-SPI still contains an intact boot-loader.

13.2.1. Firmware update steps using the nRF Device Firmware Update app

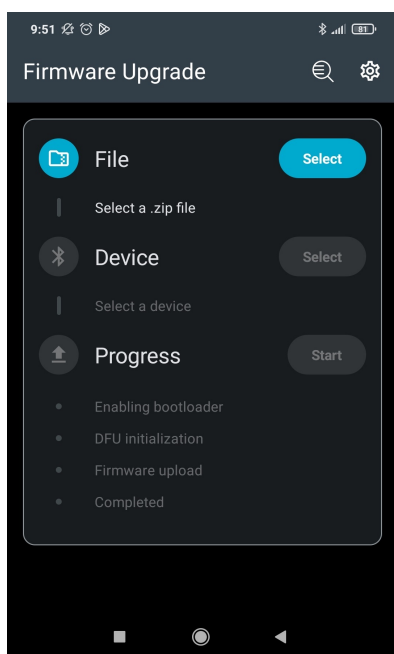
If the radio module Proteus-III-SPI has been set to bootloader mode, the nRF Device Firmware Update app [7, 8]  can be used to perform the OTA firmware update.

- Open the app, press "Select" to choose the file. The file explorer opens where you can select the zip file that contains the FOTA image.
- Then press "Select" to choose the device to update. The app scans for available Bluetooth® LE devices. The Proteus-III-SPI in bootloader mode appears as "DFUxxxx" in the scan list. Select that one.

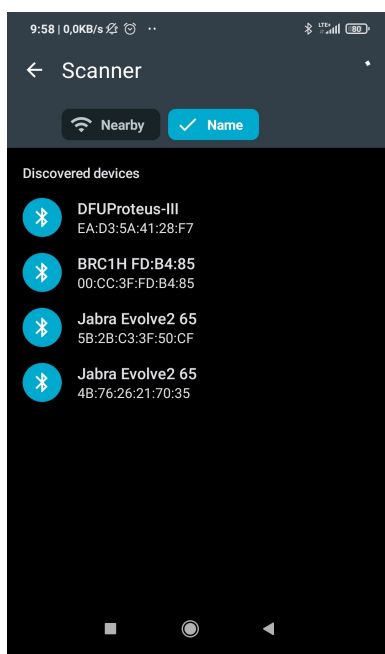


If there is no device named "DFUxxxx" on the radio, please check whether the module has been started in bootloader mode or not.

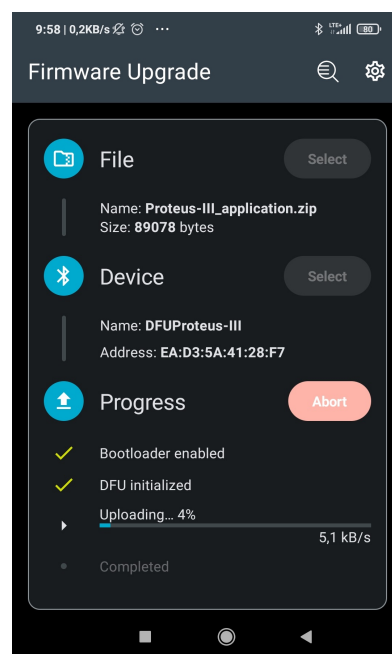
- Then press "Start" to start the update process.
- Wait until it has finished.



a) Select file and device



b) Scan list



c) FOTA running

Figure 22: nRF Device Firmware Update app

14. Firmware history

Version 1.2.0 "Release"

- First production release.

Version 1.3.0 "Release"

- New user settings RF_SPPServiceUUID, RF_SPPTXUUID, RF_SPPRXUUID. Together with the user setting RF_SPPBaseUUID the UUIDs of the SPP-like Bluetooth® LE profile can be customized.
- New option for user setting RF_AdvertisingFlags to extend the device name in the advertising packet to 26 bytes.
- GPIO pins have new options:
 - B3-B6 support PWM.
 - B1-B6 can be configured to default.
 - B1-B6 are set to input, in case the radio module is set to sleep mode using the CMD_SLEEP_REQ.
- New pairing method "Lesc just works" has been added to the user setting RF_SecFlags.
- New option SECFLAGS_BONDEDCONNECTIONSONLY_ENABLE has been added to the user setting RF_SecFlags. This option blocks the connection setup of unbonded devices.
- New command CMD_ALLOWUNBONDEDCONNECTIONS_REQ has been added to temporarily allow a bonding to a new device.

Version 1.4.0 "Release"

- Updated the user setting RF_ConnectionTiming. The new values better support most recent Android and iOS devices.
- Updated the user setting CFG_Flags. A new option has been added that defines whether the Proteus-III-SPI closes the Bluetooth® LE connection in case the connected central (i.e. smart phone) does not respect the configured connection interval settings (see RF_ConnectionTiming).
- Extended the command CMD_GETSTATE_CNF. Additional information has been added to the command in ACTION_CONNECTED state.
- For GPIO pins that support PWM, the maximum PWM ratio 100% has been mapped to 0xFE. When using 0xFF as ratio in CMD_GPIO_REMOTE_WRITECONFIG_REQ, CMD_GPIO_LOCAL_WRITECONFIG_REQ, CMD_GPIO_REMOTE_WRITE_REQ or CMD_GPIO_LOCAL_WRITE_REQ command, the value is internally replaced with 0xFE to guarantee backwards compatibility.
- For GPIO pins that are configured as PWM, the commands CMD_GPIO_REMOTE_READ_REQ and CMD_GPIO_LOCAL_READ_REQ can be used to read the current PWM ratio value.

Version 1.5.0 "Release"

- Added the user setting RF_CompanyIdentifier to modify the company identifier used in beacons

- Added option to set the FS_BTMAC
- Added option to request the maximum transmission unit (MTU) via remote command. For more details, please refer to the Proteus-III-SPI advanced developer guide [3]

14.1. Known issues

| Index | Details | Affected versions |
|--------|---|-------------------|
| KI-001 | <p>Description: Setting the Bluetooth® reserved UUID 0000xxxx-0000-1000-8000-00805f9b34fb as base UUID will result in malfunctioning. The module will not start-up again.</p> <p>Affected functions: User setting RF_SPPBaseUUID and command CMD_SET_REQ.</p> <p>Workaround: Do not use CMD_SET_REQ function to set the user setting RF_SPPBaseUUID to the value 0000xxxx-0000-1000-8000-00805f9b34fb. The device can be recovered by a FOTA (firmware over the air) update.</p> | ≤ 1.4.0 |

15. Design in guide

15.1. Advice for schematic and layout

For users with less RF experience it is advisable to closely copy the relating evaluation board with respect to schematic and layout, as it is a proven design. The layout should be conducted with particular care, because even small deficiencies could affect the radio performance and its range or even the conformity.

The following general advice should be taken into consideration:

- A clean, stable power supply is strongly recommended. Interference, especially oscillation can severely restrain range and conformity.
- Variations in voltage level should be avoided.
- LDOs, properly designed in, usually deliver a proper regulated voltage.
- Blocking capacitors and a ferrite bead in the power supply line can be included to filter and smoothen the supply voltage when necessary.



No fixed values can be recommended, as these depend on the circumstances of the application (main power source, interferences etc.).



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 VCC is not stable or below the minimum operating Voltage.
- Special care must be taken in case of battery powered systems.

- Elements for ESD protection should be placed on all pins that are accessible from the outside and should be placed close to the accessible area. For example, the RF-pin is accessible when using an external antenna and should be protected.
- ESD protection for the antenna connection must be chosen such as to have a minimum effect on the RF signal. For example, a protection diode with low capacitance such as the 8231606A or a 68 nH air-core coil connecting the RF-line to ground give good results.
- Placeholders for optional antenna matching or additional filtering are recommended.
- The antenna path should be kept as short as possible.



Again, no fixed values can be recommended, as they depend on the influencing circumstances of the application (antenna, interferences etc.).

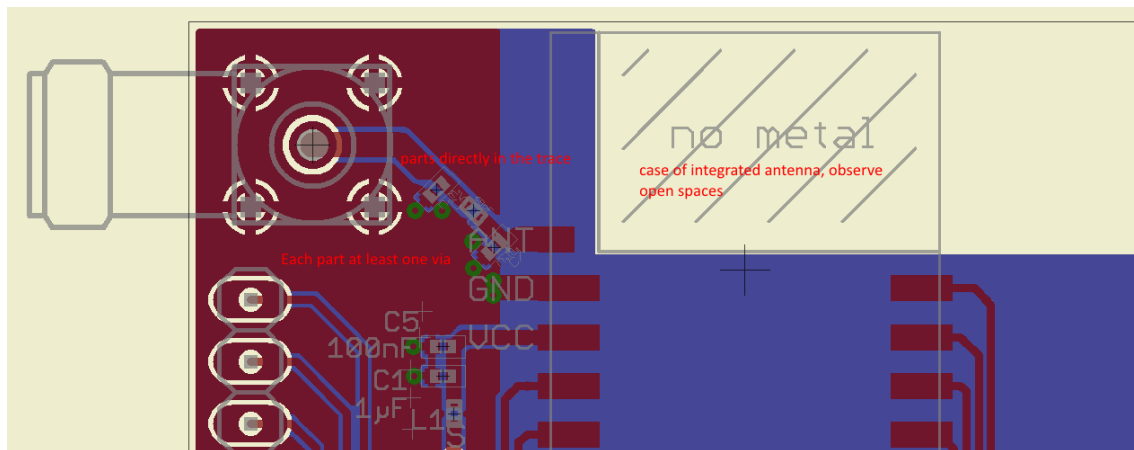


Figure 23: Layout

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

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

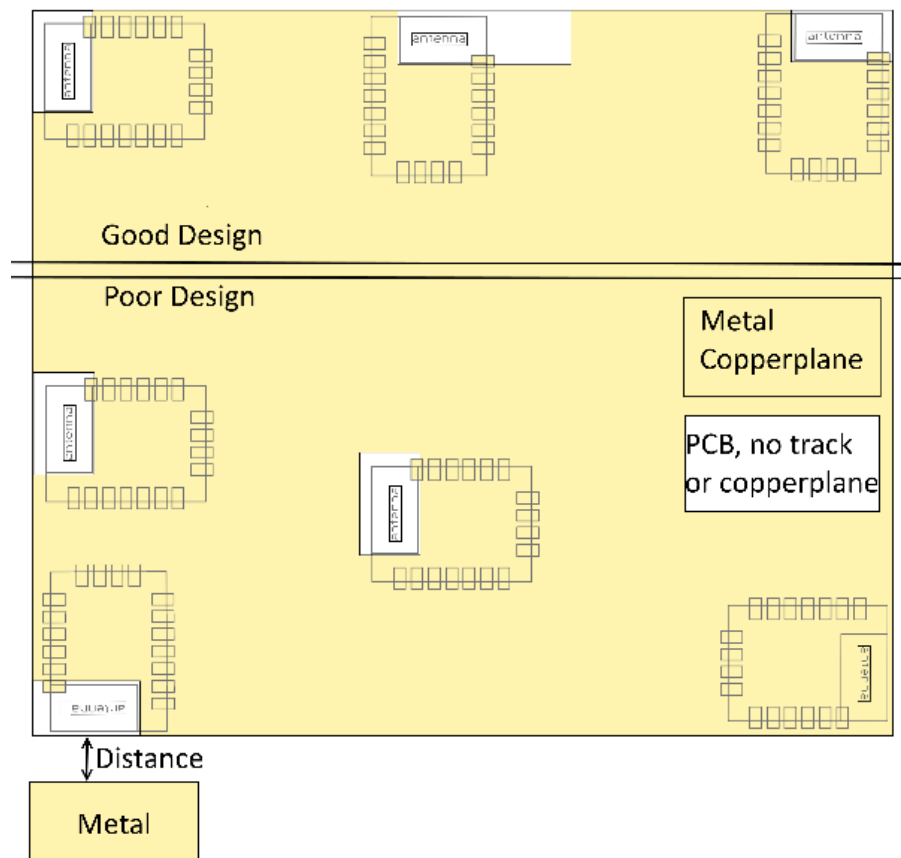


Figure 24: Placement of the module with integrated antenna

15.2. Dimensioning of the micro strip antenna line

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

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

Example:

A FR4 material with $\epsilon_r = 4.3$, a height $H = 1000 \mu\text{m}$ and a copper thickness of $T_{met} = 18 \mu\text{m}$ will

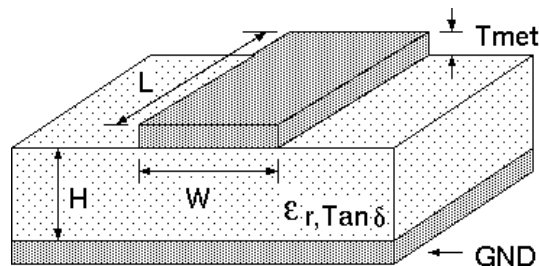


Figure 25: Dimensioning the antenna feed line as micro strip

lead to a trace width of $W \sim 1.9 \text{ mm}$. 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 $3 \times W$ should be observed between the micro strip and other traces / ground.
- The micro strip refers to ground, therefore there has to be the ground plane underneath the trace.
- Keep the feeding line as short as possible.

15.3. Antenna solutions

There exist several kinds of antennas, which are optimized for different needs. Chip antennas are optimized for minimal size requirements but at the expense of range, PCB antennas are optimized for minimal costs, and are generally a compromise between size and range. Both usually fit inside a housing.

Range optimization in general is at the expense of space. Antennas that are bigger in size, so that they would probably not fit in a small housing, are usually equipped with a RF connector. A benefit of this connector may be to use it to lead the RF signal through a metal plate (e.g. metal housing, cabinet).

As a rule of thumb a minimum distance of $\lambda / 10$ (which is 3.5 cm @ 868 MHz and 1.2 cm @ 2.44 GHz) from the antenna to any other metal should be kept. Metal placed further away will not directly influence the behavior of the antenna, but will anyway produce shadowing.



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



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

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

15.3.1. Wire antenna

An effective antenna is a $\lambda/4$ 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 $\lambda/4$ radiator has approximately 40 Ω input impedance. Therefore, matching is not required.

15.3.2. Chip antenna

There are many chip antennas from various manufacturers. The benefit of a chip antenna is obviously the minimal space required and reasonable costs. However, this is often at the expense of range. For the chip antennas, reference designs should be followed as closely as possible, because only in this constellation can the stated performance be achieved.

15.3.3. PCB antenna

PCB antenna designs can be very different. The special attention can be on the miniaturization or on the performance. The benefits of the PCB antenna are their small / not existing (if PCB space is available) costs, however the evaluation of a PCB antenna holds more risk of failure than the use of a finished antenna. Most PCB antenna designs are a compromise of range and space between chip antennas and connector antennas.

15.3.4. Antennas provided by Würth Elektronik eiSos

15.3.4.1. 2600130021 - Himalia - 2.4 GHz dipole antenna



Figure 26: 2.4 GHz dipole-antenna

Due to the fact, that the antenna has dipole topology there is no need for an additional ground plane. Nevertheless the specification was measured edge mounted and 90 ° bent on a 100 x 100 mm ground plane.

| Specification | Value |
|--|------------------|
| Frequency range [GHz] | 2.4 – 2.5 |
| Impedance [Ω] | 50 |
| VSWR | $\leq 2:1$ |
| Polarization | Linear |
| Radiation | Omni-Directional |
| Peak Gain [dBi] | 2.8 |
| Average Gain [dBi] | -0.6 |
| Efficiency | 85 % |
| Dimensions (L x d) [mm] | 83.1 x 10 |
| Weight [g] | 7.4 |
| Connector | SMA plug |
| Operating temp. [$^{\circ}\text{C}$] | -40 – +80 |

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

16. Reference design

The Proteus-III was tested and certified on the corresponding Proteus-III evaluation board. For the European Conformity the evaluation board serves as reference design also for Proteus-III-SPI. For the FCC of the Proteus-III-SPI, the FCC of the Proteus-III can be used by a class 1 permissive change, in case the trace design (chapter 16.3) is respected.

Special care must be taken about proper SPI lines, as there is:

- Proper ground plain for backloop current.
- Short lines.
- Avoiding connectors.

16.1. Mini EV-Board

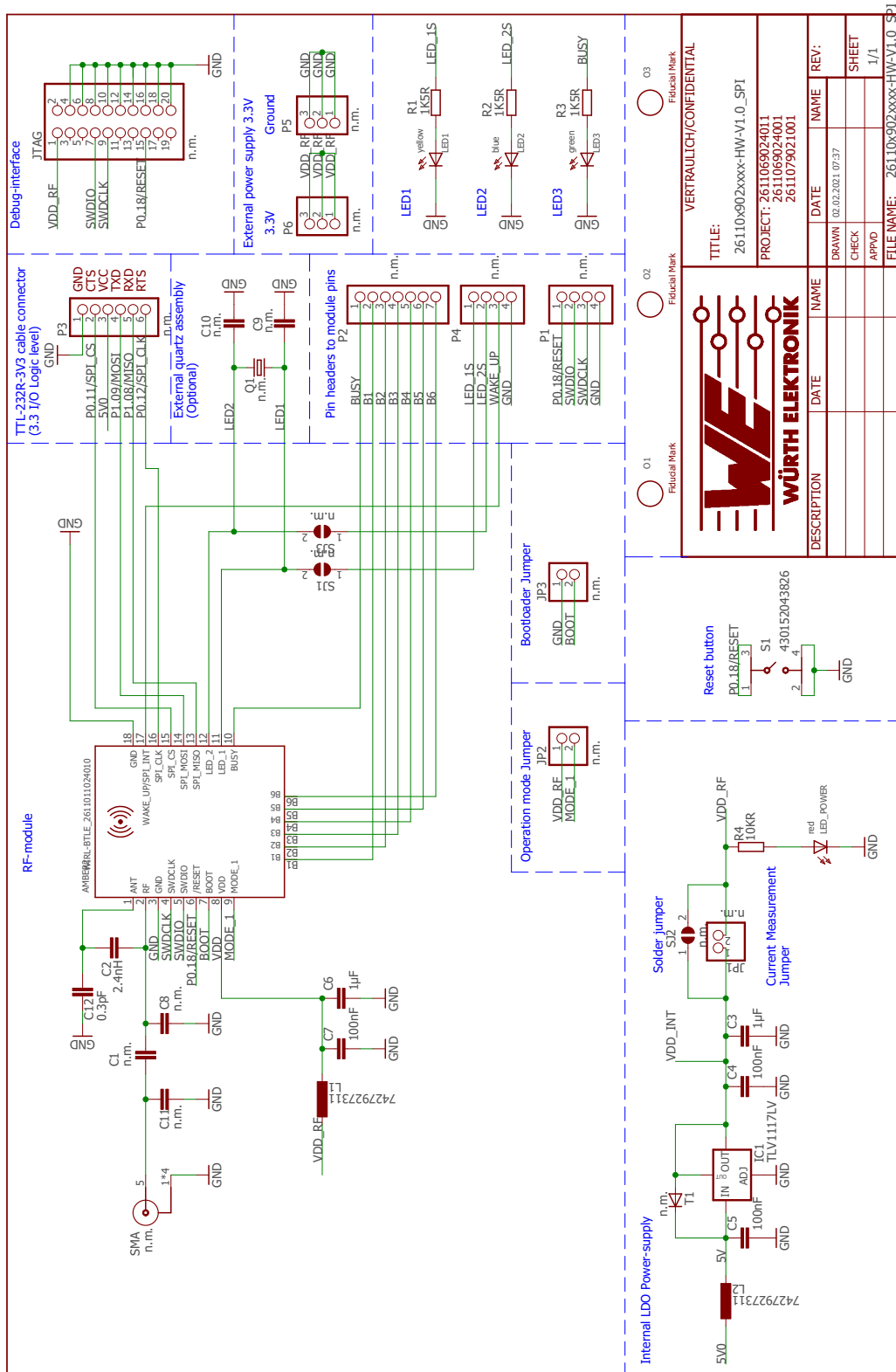


Figure 27: Reference design: Schematic page

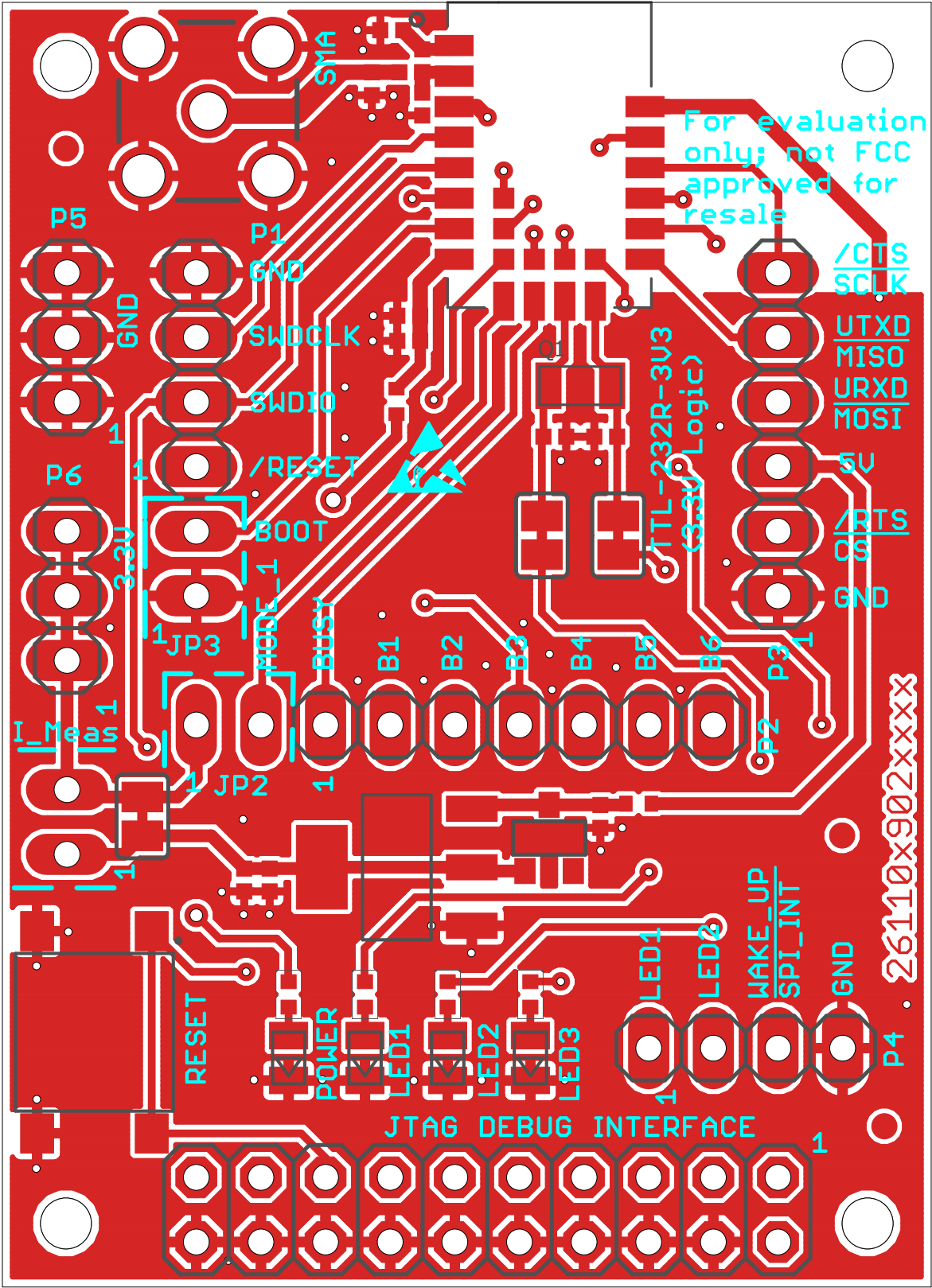


Figure 28: Reference design: Layout

16.2. Radiation characteristic of the module's internal antenna

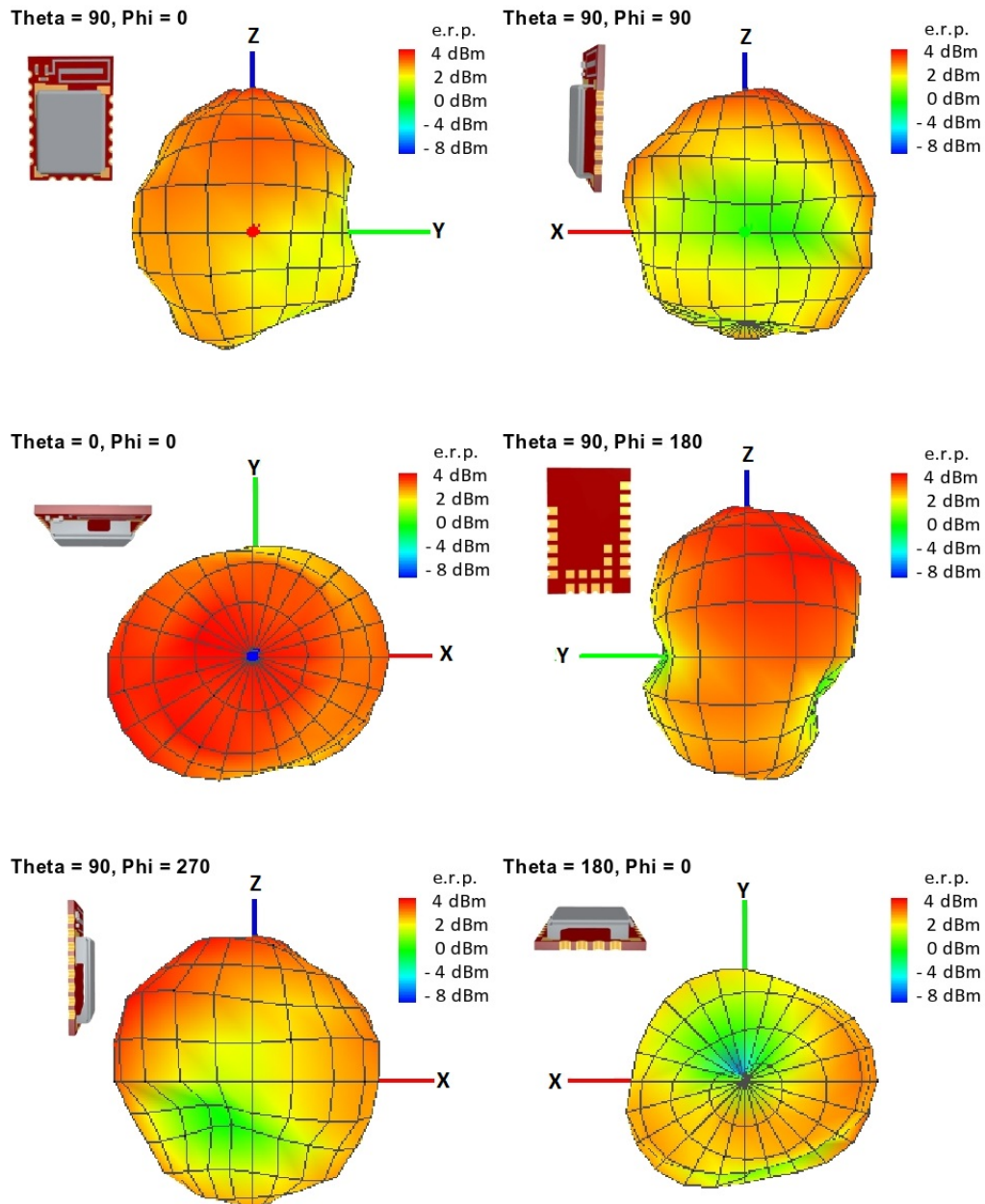


Figure 29: Antenna characteristic of the module with its integrated antenna measured on the official evaluation board (2611149024011)



It is important to be aware that size and shape of the ground plane as well as the placement of module has influence on the radiation pattern.

16.3. Trace design

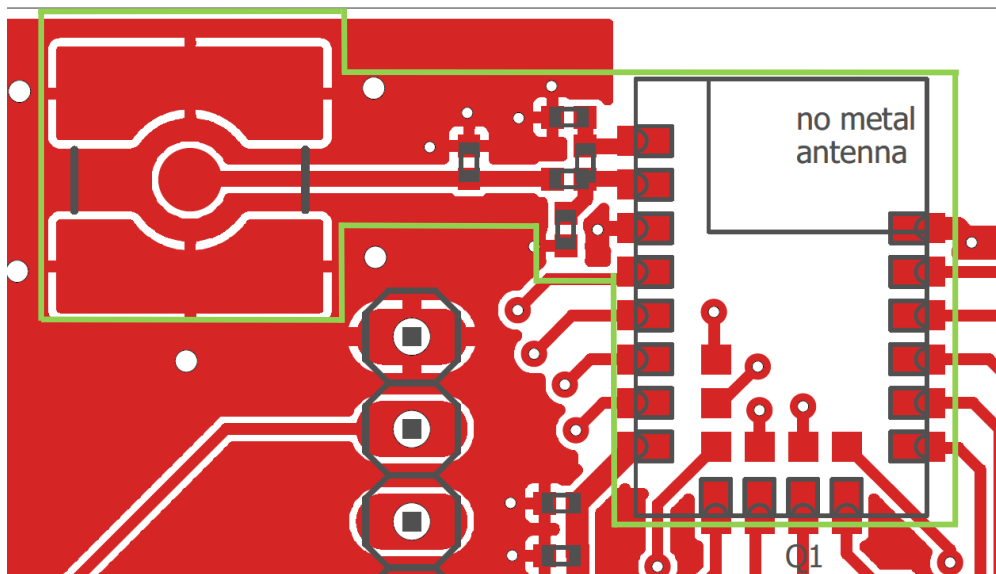


Figure 30: Trace design: Layout

| Nr | Copper | | Isolation | |
|-----------------|-----------|----------|-----------|----------|
| | Thickness | Material | Thickness | Material |
| 1 | 0.035mm | | 10mil | |
| 2 | 0.018mm | | 1mm | |
| 15 | 0.018mm | | 10mil | |
| 16 | 0.035mm | | | |
| Gesamt: 1.614mm | | | | |

Figure 31: Reference design: Stack-up

- Top layer is used for routing, filled with ground plane except area under the module and antenna free area.
- Second layer is filled with ground plane, except the antenna free area.
- Third layer is the supply layer, except antenna free area. Some routing is allowed, not dividing the supply layer in to many or too small parts.
- Bottom layer is used for routing and filled with ground.

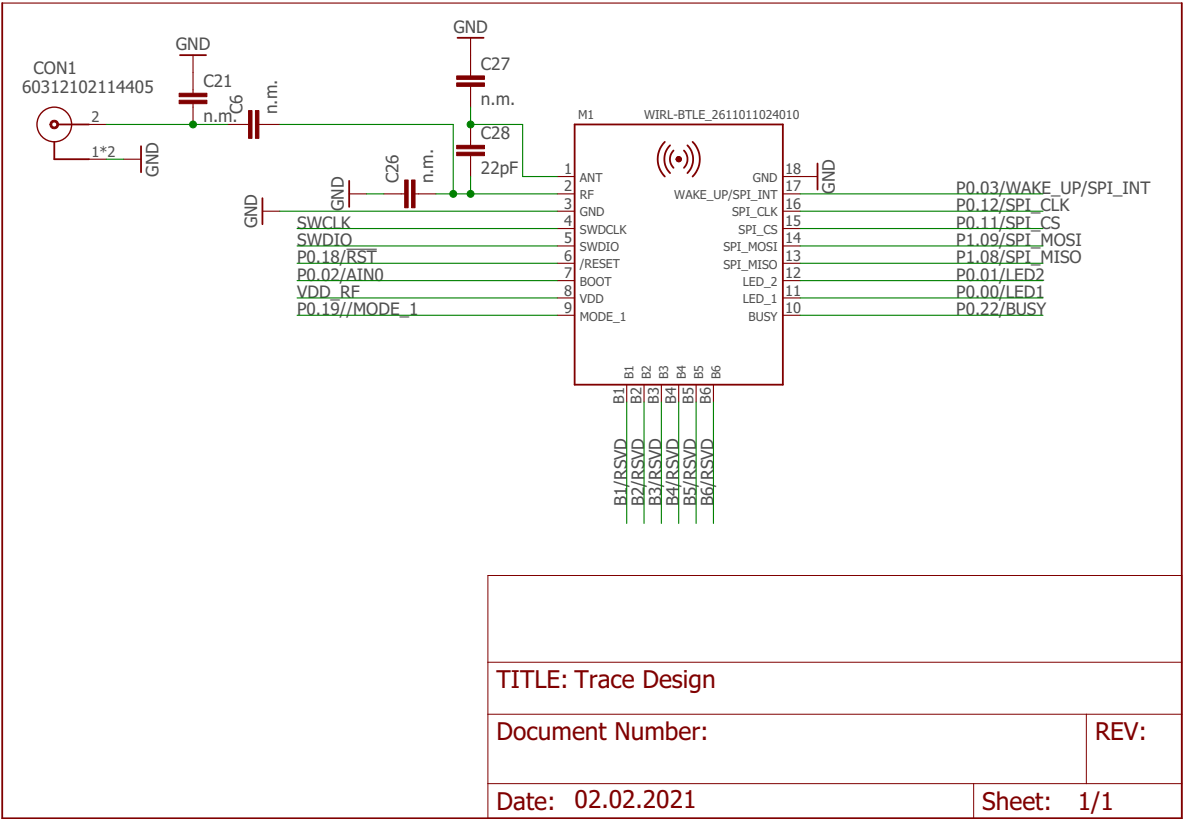


Figure 32: Trace design: Schematic

The *RF* pin of module can be coupled to on-board PCB antenna or an external antenna. Two variants of the Proteus-III-SPI are certified:

- For the on-board PCB antenna: 22 pF shall be assembled on C28.
 - If additional tuning is needed in the end application, C27 and C26 can be assembled.
 - The exact values of C27 and C26 shall be specified in the end application corresponding to the individual need.

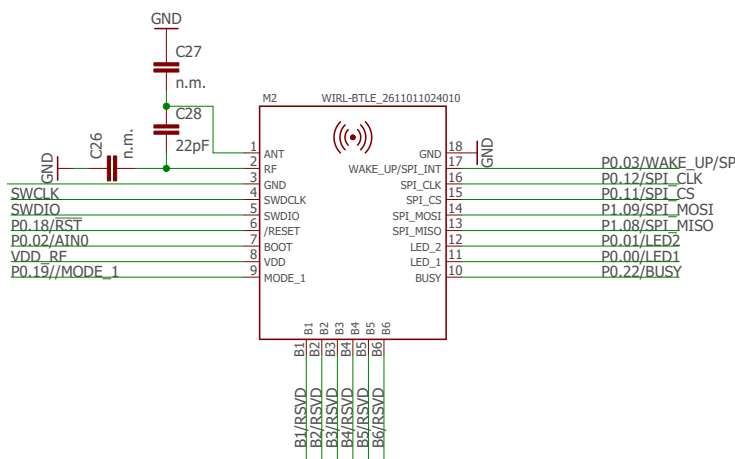


Figure 33: On-board PCB antenna

- For the external antenna: 22 pF shall be assembled on C6.
 - If additional tuning is needed in the end application, C21 and C26 can be assembled.
 - The exact values of C21 and C26 shall be specified in the end application corresponding to the individual need.

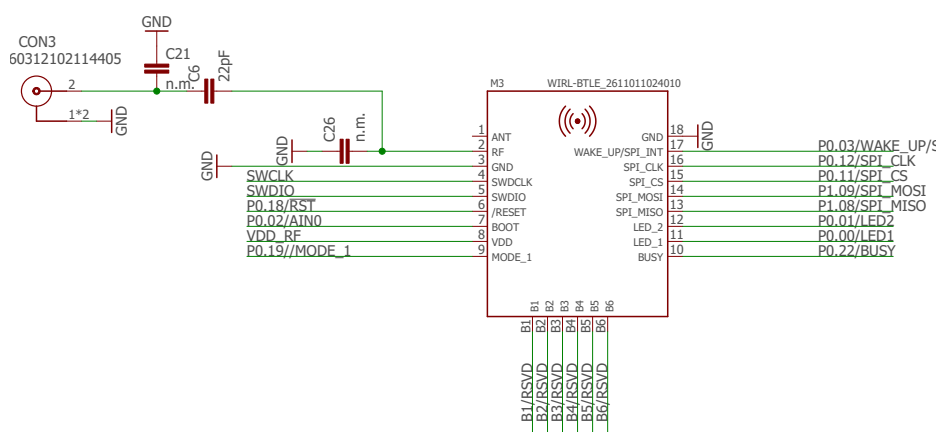


Figure 34: External antenna connection



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

17. Manufacturing information

17.1. Moisture sensitivity level

This wireless connectivity product is categorized as JEDEC Moisture Sensitivity Level 3 (MSL3), which requires special handling.

More information regarding the MSL requirements can be found in the IPC/JEDEC J-STD-020 standard on www.jedec.org.

More information about the handling, picking, shipping and the usage of moisture/reflow and/or process sensitive products can be found in the IPC/JEDEC J-STD-033 standard on www.jedec.org.

17.2. Soldering

17.2.1. Reflow soldering

Attention must be paid on the thickness of the solder resist between the host PCB top side and the modules bottom side. Only lead-free assembly is recommended according to JEDEC J-STD020.

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

Table 82: Classification reflow soldering profile, Note: refer to IPC/JEDEC J-STD-020E

| Package thickness | Volume mm ³ <350 | Volume mm ³ 350-2000 | Volume mm ³ >2000 |
|-------------------|--------------------------------|------------------------------------|---------------------------------|
| < 1.6 mm | 260 °C | 260 °C | 260 °C |
| 1.6 mm - 2.5 mm | 260 °C | 250 °C | 245 °C |
| > 2.5 mm | 250 °C | 245 °C | 245 °C |

Table 83: Package classification reflow temperature, PB-free assembly, Note: refer to IPC/-JEDEC J-STD-020E

It is recommended to solder this module on the last reflow cycle of the PCB. For solder paste use a LFM-48W or Indium based SAC 305 alloy (Sn 96.5 / Ag 3.0 / Cu 0.5 / Indium 8.9HF / Type 3 / 89%) type 3 or higher.

The reflow profile must be adjusted based on the thermal mass of the entire populated PCB, heat transfer efficiency of the reflow oven and the specific type of solder paste used. Based on the specific process and PCB layout the optimal soldering profile must be adjusted and verified. Other soldering methods (e.g. vapor phase) have not been verified and have to be validated by the customer at their own risk. Rework is not recommended.

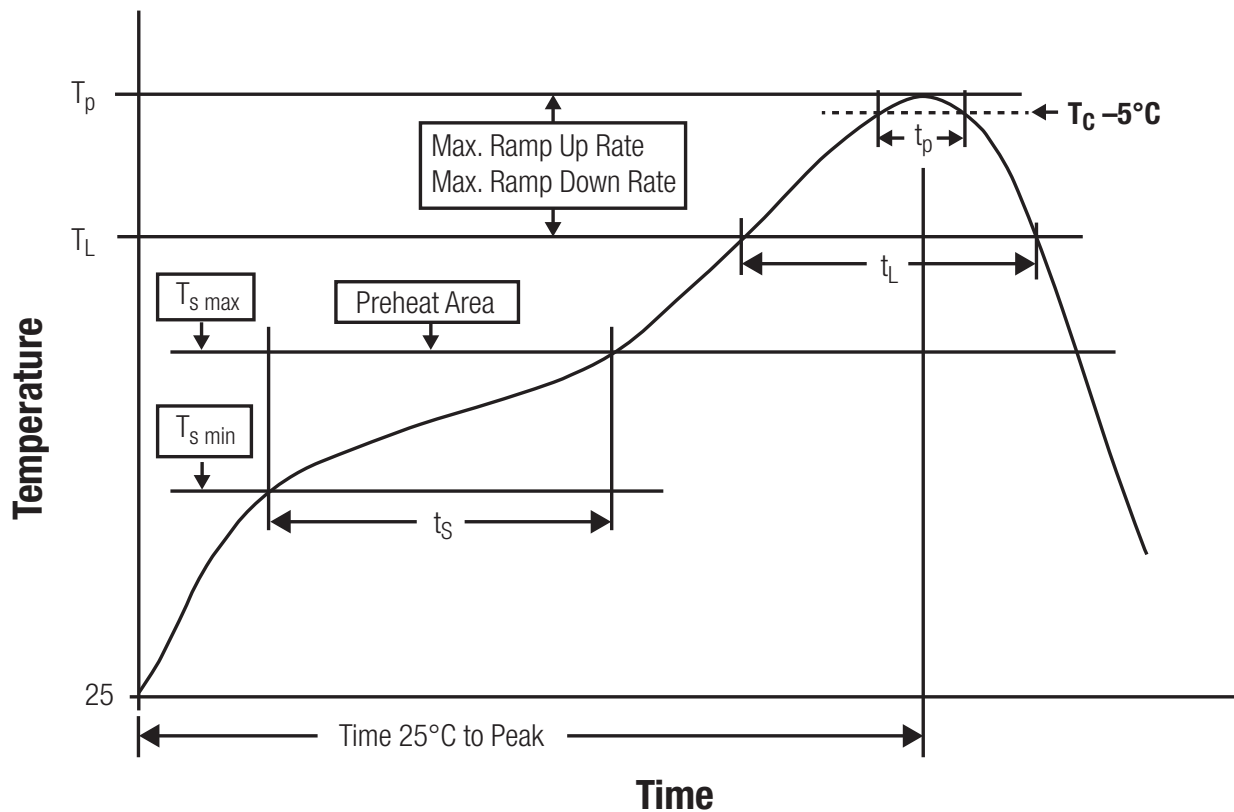


Figure 35: Reflow soldering profile

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

17.2.2. Cleaning

Do not clean the product. Any residue cannot be easily removed by washing. Use a "no clean" soldering paste and do not clean the board after soldering.

- Do not clean the product with water. Capillary effects can draw water into the gap between the host PCB and the module, absorbing water underneath it. If water is trapped inside, it may short-circuit adjoining pads. The water may also destroy the label and ink-jet printed text on it.
- Cleaning processes using alcohol or other organic solvents may draw solder flux residues into the housing, which won't be detected in a post-wash inspection. The solvent may also destroy the label and ink-jet printed text on it.
- Do not use ultrasonic cleaning as it will permanently damage the part, particularly the crystal oscillators.

17.2.3. Potting and coating

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

17.2.4. 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.

17.3. ESD handling

This product is highly sensitive to electrostatic discharge (ESD). As such, always use proper ESD precautions when handling. Make sure to handle the part properly throughout all stages of production, including on the host PCB where the module is installed. For ESD ratings, refer to the module series' maximum ESD section. For more information, refer to the relevant chapter 2. Failing to follow the aforementioned recommendations can result in severe damage to the part.

- the first contact point when handling the PCB is always between the local GND and the host PCB GND, unless there is a galvanic coupling between the local GND (for example work table) and the host PCB GND.
- Before assembling an antenna patch, connect the grounds.
- While handling the RF pin, avoid contact with any charged capacitors and be careful when contacting any materials that can develop charges (for example coaxial cable with around 50-80 pF/m, patch antenna with around 10 pF, soldering iron etc.)
- Do not touch any exposed area of the antenna to avoid electrostatic discharge. Do not let the antenna area be touched in a non ESD-safe manner.
- When soldering, use an ESD-safe soldering iron.

17.4. Safety recommendations

It is your duty to ensure that the product is allowed to be used in the destination country and within the required environment. Usage of the product can be dangerous and must be tested and verified by the end user. Be especially careful of:

- Use in areas with risk of explosion (for example oil refineries, gas stations).
- Use in areas such as airports, aircraft, hospitals, etc., where the product may interfere with other electronic components.

It is the customer's responsibility to ensure compliance with all applicable legal, regulatory and safety-related requirements as well as applicable environmental regulations. Disassembling the product is not allowed. Evidence of tampering will void the warranty.

- Compliance with the instructions in the product manual is recommended for correct product set-up.
- The product must be provided with a consolidated voltage source. The wiring must meet all applicable fire and security prevention standards.
- Handle with care. Avoid touching the pins as there could be ESD damage.

Be careful when working with any external components. When in doubt consult the technical documentation and relevant standards. Always use an antenna with the proper characteristics.



Würth Elektronik eiSos radio modules with high output power of up to 500 mW, as for example the radio module Thebe-II, generate a high amount of warmth while transmitting. The manufacturer of the end device must take care of potentially necessary actions for his application.

18. Physical specifications

18.1. Dimensions

| Dimensions |
|---------------|
| 12 x 8 x 2 mm |

Table 84: Dimensions

18.2. Weight

| Weight |
|--------|
| <1g |

Table 85: Weight

18.3. Module drawing

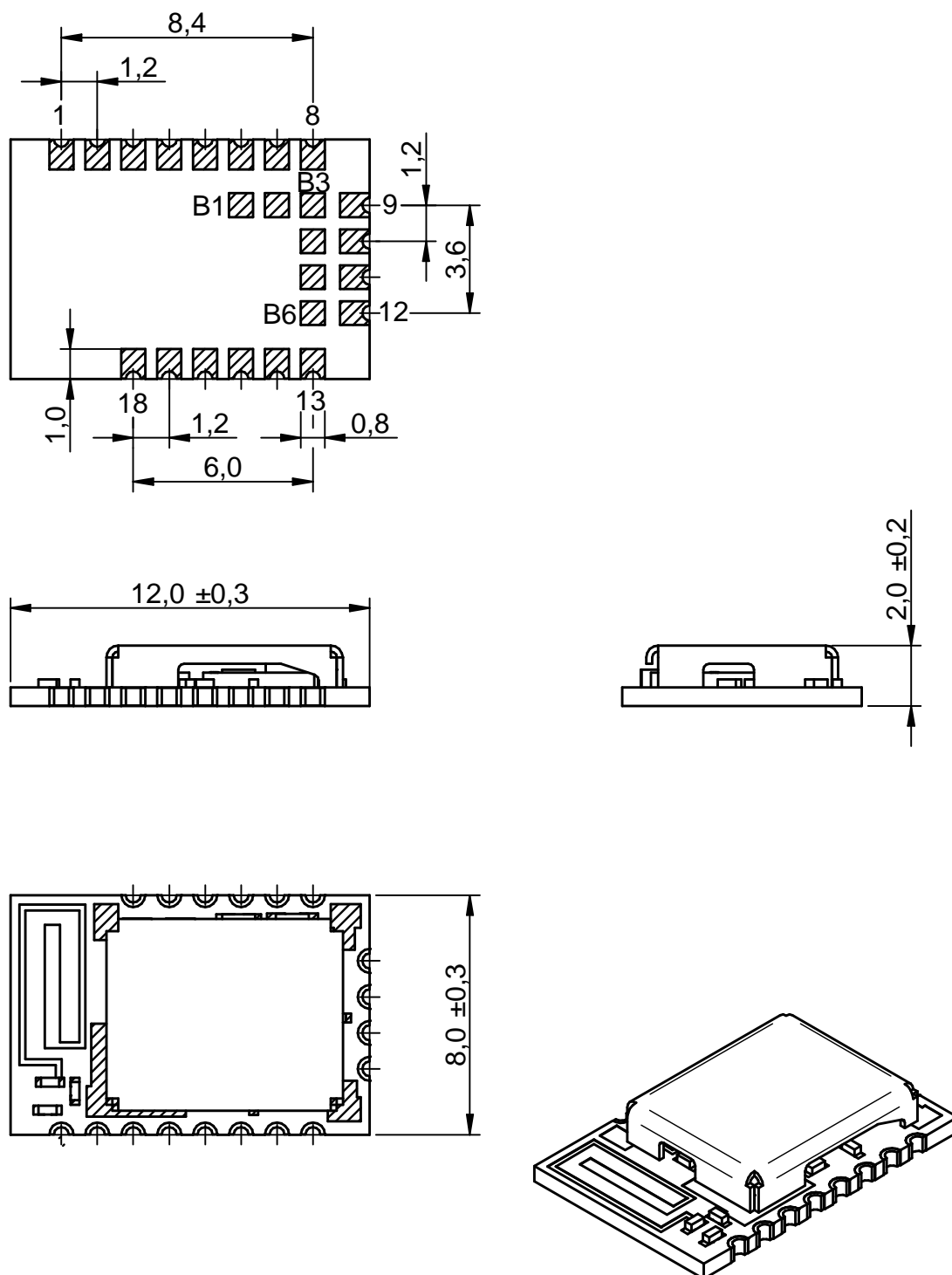


Figure 36: Module dimensions [mm]

18.4. Footprint WE-FP-4+

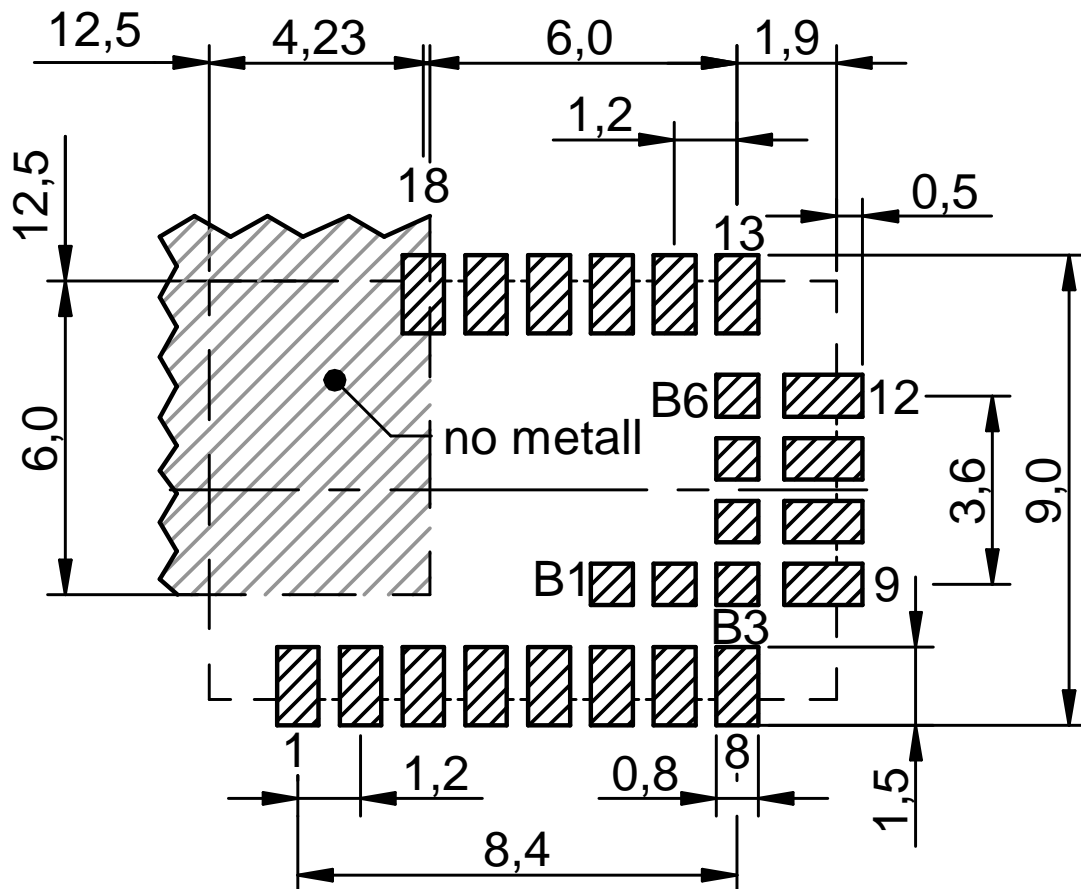


Figure 37: Footprint WE-FP-4+ [mm]

18.5. Antenna free area

To avoid influence and mismatching of the antenna the recommended free area around the antenna should be maintained. As rule of thumb a minimum distance of metal parts to the antenna of $\lambda/10$ should be kept (see figure 37). Even though metal parts would influence the characteristic of the antenna, but the direct influence and matching keep an acceptable level.

19. Marking

19.1. Lot number

The 15 digit lot number is printed in numerical digits as well as in form of a machine readable bar code. It is divided into 5 blocks as shown in the following picture and can be translated according to the following table.

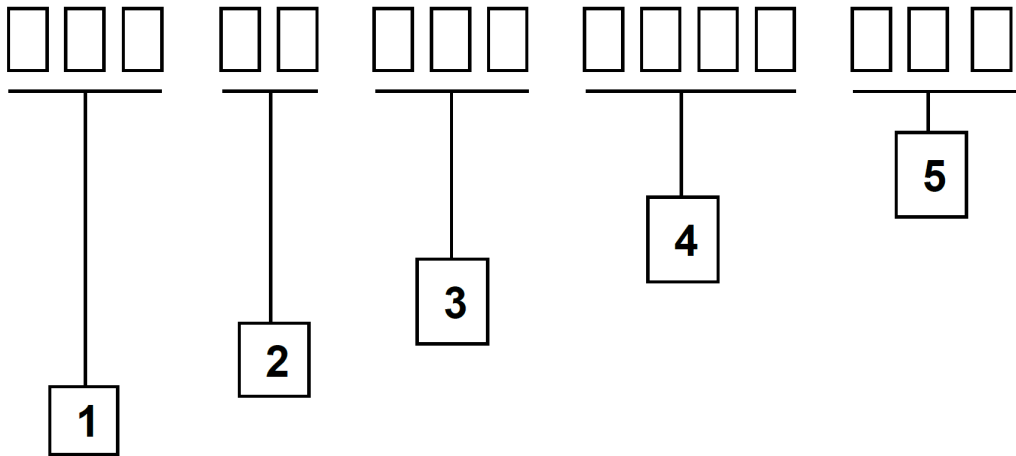


Figure 38: Lot number structure

| Block | Information | Example(s) |
|-------|----------------------------|---|
| 1 | eiSos internal, 3 digits | 439 |
| 2 | eiSos internal, 2 digits | 01 |
| 3 | Hardware version, 3 digits | V2.4 = 024, V12.2 = 122 |
| 4 | Date code, 4 digits | 1703 = week 03 in year 2017, 1816 = week 16 in year 2018 |
| 5 | Firmware version, 3 digits | V3.2 = 302, V5.13 = 513 |

Table 86: Lot number details

As the user can perform a firmware update the printed lot number only shows the factory delivery state. The currently installed firmware can be requested from the module using the corresponding product specific command. The firmware version as well as the hardware version are restricted to show only major and minor version not the patch identifier.

19.2. General labeling information

The module labels may include the following fields:

- Manufacturer identification WE, Würth Elektronik or Würth Elektronik eiSos
- WE Order Code and/or article alias
- Serial number or MAC address
- Certification identifiers (CE, FCC ID, IC, TELEC,...)
- Bar code or 2D code containing the serial number or MAC address

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

In case of small labels, the 3 byte manufacturer identifier (0x0018DA) of the MAC address is not printed on the labels. The 3 byte counter printed on the label can be used with this 0018DA to produce the full MAC address by appending the counter after the manufacturer identifier.

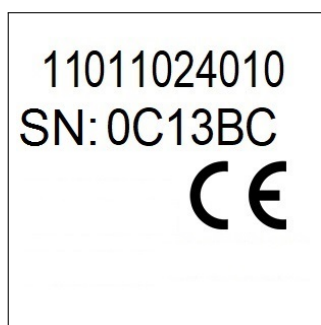


Figure 39: Label of the Proteus-III-SPI

20. 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 6 dBm for external antenna and 4 dBm for internal antenna.
- The total amount of capacitance of all capacitors is 7.2 μF .
- The total amount of inductance of all inductors is 10.025 μH .
- A DC/DC regulator is included in the chip set and used to obtain low power functionality.

21. Bluetooth SIG listing/qualification

| Type | Data |
|--------------------|-----------------|
| Design name | Proteus-III-SPI |
| Declaration ID | D047845 |
| QDID | 141060 |
| Specification name | 5.1 |
| Project type | End product |

Each product containing intellectual property of the Bluetooth® Special Interest Group (SIG) must be qualified by the SIG to obtain the corresponding Declaration ID.

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

To obtain the Bluetooth® listing of the end device, please refer to the application note ANR027 [9].

22. Regulatory compliance information

22.1. Important notice EU

The use of RF frequencies is limited by national regulations. The Proteus-III-SPI has been designed to comply with the RED directive 2014/53/EU of the European Union (EU).

The Proteus-III-SPI 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.



Since the module itself is not fused the voltage supply shall be fed from a power source which is class PS2 according to EN 62368-1.

22.2. Important notice FCC

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

The Proteus-III-SPI can be operated without notification and free of charge in the area of the United States of America. However, according to the FCC Part 15, restrictions (e.g. in terms of maximum allowed RF power and antenna) may apply.

22.3. Conformity assessment of the final product

The Proteus-III-SPI is a subassembly. It is designed to be embedded into other products (products incorporating the Proteus-III-SPI are henceforward referred to as "final products").

It is the responsibility of the manufacturer of the final product to ensure that the final product is in compliance with the essential requirements of the underlying national radio regulations.

The conformity assessment of the subassembly Proteus-III-SPI carried out by Würth Elektronik eiSos does not replace the required conformity assessment of the final product.

22.4. Exemption clause

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

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

22.5. EU Declaration of conformity



EU DECLARATION OF CONFORMITY

Radio equipment: 2611011024010

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: 2611011024010

The object of the declaration described above is in conformity with the relevant Union harmonization legislation: Directive 2014/53/EU and 2011/65/EU.

Following harmonized norms or technical specifications have been applied:

EN 300 328 V2.2.2 (2019-07)
EN 301 489-1 V2.2.3 (2019-11)
EN 301 489-17 V3.2.4 (2020-09)
EN 62479 : 2010
EN 62368-1:2014 + AC:2015 +A11:2019

i.A. G. Eslerdt

Trier, 8th of February 2021
Place and date of issue

22.6. UK Declaration of conformity



DECLARATION OF CONFORMITY

Product: Proteus-III-SPI Bluetooth 5.1 Radio Module
Model/type: 2611011024010

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

The importer: Würth Electronics UK Limited
8 The Vic, MediacityUK
M50 3SP Manchester
United Kingdom

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

Object of the declaration: 2611011024010

The object of the declaration described above is in conformity with the relevant UK Statutory Instrument: The Radio Equipment Regulations 2017 (S.I. 2017 No. 1206).

References to the relevant designated standards used or references to the other technical specifications in relation to which conformity is declared:

EN 300 328 V2.2.2 (2019-07)
EN 301 489-1 V2.2.3 (2019-11)
EN 301 489-17 V3.2.4 (2020-09)
EN 62479 : 2010
EN 62368-1:2014 + AC:2015 +A11:2019
S.I. 2012 No.3032

i.A. G. Eckhardt

Gudrun Eckhardt, Teamleader Hardware Development, Trier, 26th of April 2023
Name, Function, Place and date of issue

22.7. FCC Compliance Statement (US)

FCC ID: R7T1101102

This device complies with Part 15 of the FCC Rules.

Operation is subject to the following two conditions:

- (1) this device may not cause harmful interference, and
 - (2) this device must accept any interference received, including interference that may cause undesired operation.
- (FCC 15.19)

Modifications (FCC 15.21)

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

22.7.1. FCC certificate



| | |
|---|--|
|  | |
| TCB | TCB |
| GRANT OF EQUIPMENT AUTHORIZATION Certification Issued Under the Authority of the Federal Communications Commission By: | |
| Telefication B.V. Edisonstraat 12a Zevenaar, NL-6902 PK Netherlands | |
| Date of Grant: 11/27/2019 Application Dated: 11/06/2019 | |
| Wuerth Elektronik eiSos GmbH & Co KG Max-Eyth-Strasse 1 Waldenburger, 74638 Germany Attention: Gudrun Eckhardt, Manager | |
| 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: R7T1101102 Name of Grantee: Wuerth Elektronik eiSos GmbH & Co KG Equipment Class: Digital Transmission System Notes: Radio module HVIN 1101102 Modular Type: Single Modular | |
| Grant Notes | FCC Rule Parts 15C |
| CC | Frequency Range (MHz) 2402.0 - 2480.0 |
| | Output Watts 0.0042 |
| | Frequency Emission Tolerance Designator |
| Output power listed is conducted. Single modular approval for portable RF Exposure conditions. Approval is limited to OEM installations only. OEM integrators must be provided with antenna installation instructions. OEM integrators and end users must be provided with transmitter operating conditions for satisfying RF exposure compliance. This grant is valid only when the device is sold to OEM integrators and the OEM integrators are instructed to ensure that the end user has no manual instructions to remove or install the device. Only those antenna(s) tested with the device or similar antenna(s) with equal or lesser gain may be used with this transmitter. | |
| CC: This device is certified pursuant to two different Part 15 rules sections. | |
| Certificate No.: 192180963/AA/00 | Gürhan Vural Product Assessor |
|  | |

Figure 40: FCC certificate

22.8. IC Compliance Statement (Canada)

Certification Number: 5136A-1101102

HVIN: 1101102

This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

22.8.1. IC certificate

telefication bv
The Netherlands
Chamber of Commerce
5155536
www.telefication.com



telefication

TECHNICAL ACCEPTANCE
CERTIFICATE

CERTIFICATION No.
No. DE CERTIFICATION

5136A-1101102

TELEFICATION No.
No. DE TELEFICATION

192170345/AA/00

TEST SITE No.
No. DE LABORATOIRE

3699A

ISSUED TO
DÉLIVRÉ À

TYPE OF EQUIPMENT
GENRE DE MATÉRIEL

SPREAD SPECTRUM DEVICE (2400-2483.5 MHz)

TRADE NAME AND MODEL
MARQUE ET MODÈLE

Würth Elektronik eiSos GmbH & Co. KG / 1101102

CERTIFIED TO
CERTIFIÉ SELON LE

SPECIFICATION
CAHIER DES CHARGES

RSS-102
RSS-247

ISSUE
ÉDITION

5
2

Certification of equipment means only that the equipment has met the requirements of the above-noted specification. Licence applications, where applicable to use certified equipment, are acted on accordingly by theISED issuing office and will depend on the existing radio environment, service and location of operation. This certificate is issued on condition that the holder complies and will continue to comply with the requirements and procedures issued by ISED. The equipment for which this certificate is issued shall not be manufactured, imported, distributed, leased, offered for sale or sold unless the equipment complies with the applicable technical specifications and procedures issued by ISED.

La certification du matériel signifie seulement que le matériel a satisfait aux exigences de la norme indiquée ci-dessus. Les demandes de licences nécessaires pour l'utilisation du matériel certifié sont traitées en conséquence par le bureau de délivrance d'ISDE et dépendent des conditions radio ambiantes, du service et de l'emplacement d'exploitation. Le présent certificat est délivré à la condition que le titulaire satisfasse et continue de satisfaire aux exigences et aux procédures d'ISDE. Le matériel à l'égard duquel le présent certificat est délivré ne doit pas être fabriqué, importé, distribué, loué, mis en vente ou vendu à moins d'être conforme aux procédures et aux spécifications techniques applicables publiées par ISDE.

ISSUED BY TELEFICATION BV (NL0001), RECOGNIZED CERTIFICATION BODY BY INNOVATION, SCIENCE AND ECONOMIC DEVELOPMENT CANADA
DÉLIVRÉ PAR TELEFICATION BV (NL0001), ORGANISME DE CERTIFICATION RECONNU PAR INNOVATION, SCIENCES ET DÉVELOPPEMENT ÉCONOMIQUE CANADA

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 a été jugé conforme à la spécification ci-dessus

DATE 02 Dec 2019 BY

Gürhan Vural
Product Assessor

This certificate has one annex.











Figure 41: IC certificate

22.9. FCC and IC requirements to OEM integrators

This module has been granted modular approval. OEM integrators for host products may use the module in their final products without additional FCC/IC (Industry Canada) certification if they meet the following conditions. Otherwise, additional FCC/IC approvals must be obtained. The host product with the module installed must be evaluated for simultaneous transmission requirements.

- The users manual for the host product must clearly indicate the operating requirements and conditions that must be observed to ensure compliance with current FCC/IC RF exposure guidelines.
- To comply with FCC/IC regulations limiting both maximum RF output power and human exposure to RF radiation, the maximum antenna gain including cable loss in a mobile-only exposure condition must not exceed 6dBi.
- A label must be affixed to the outside of the host product with the following statements:
This device contains FCCID: R7T1101102
This equipment contains equipment certified under ICID: 5136A-1101102
- The final host / module combination may also need to be evaluated against the FCC Part 15B criteria for unintentional radiators in order to be properly authorized for operation as a Part 15 digital device.
- If the final host / module combination is intended for use as a portable device (see classifications below) the host manufacturer is responsible for separate approvals for the SAR requirements from FCC Part 2.1093 and RSS-102.

OEM requirements:

The OEM must ensure that the following conditions are met.

- The Proteus-III-SPI will be used at a distance of at least 10 mm.
- End users of products, which contain the module, must not have the ability to alter the firmware that governs the operation of the module. The agency grant is valid only when the module is incorporated into a final product by OEM integrators.
- The end-user must not be provided with instructions to remove, adjust or install the module.
- The Original Equipment Manufacturer (OEM) must ensure that FCC labeling requirements are met. This includes a clearly visible label on the outside of the final product. Attaching a label to a removable portion of the final product, such as a battery cover, is not permitted.
- The label must include the following text:
Contains FCC ID: R7T1101102
The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(i.) this device may not cause harmful interference and
(ii.) this device must accept any interference received, including interference that may

cause undesired operation.

When the device is so small or for such use that it is not practicable to place the statement above on it, the information required by this paragraph shall be placed in a prominent location in the instruction manual or pamphlet supplied to the user or, alternatively, shall be placed on the container in which the device is marketed. However, the FCC identifier or the unique identifier, as appropriate, must be displayed on the device.

- The user manual for the end product must also contain the text given above.
 - Changes or modifications not expressly approved could void the user’s authority to operate the equipment.
 - The OEM must ensure that timing requirements according to 47 CFR 15.231(a-c) are met.
 - The OEM must sign the OEM Modular Approval Agreement.
 - The module must be used with only the following approved antenna(s).

22.9.1. Pre-certified antennas

The Proteus-III-SPI is pre-certified with the following antennas.

| Product | Certified antenna |
|------------------------------------|---|
| Proteus-III-SPI (2611011024010) | PCB antenna included in the Proteus-III-SPI |

Registration No: ETA-SD-20230302018 Date: 06-04-2023

The Proteus-III-SPI complies with the provisions on the Equipment Type Approval WPC Wing for India.

22.10.1. ETA-WPC certificate

**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-III-SPI

Registration No: ETA-SD-20230302018

Date: 06-04-2023

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-III-SPI |
| 5. | Frequency range(s) of Equipment | 1. 2402-2480 MHz |
| 6. | Max output power/Field strength/PSD | 1. E.I.R.P. (dBm). 4 |

1 / 2

Figure 42: ETA-WPC certificate page 1

| | | | |
|----|---|--|--------------------------------------|
| 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 |
| | 7layers GmbH & Borsigstrasse 11 40880 Ratingen Germany | D-PL-12140-01-03 | MDE_WUERTH_1901_RADIO01 & 15-10-2019 |

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>.

Figure 43: ETA-WPC certificate page 2


22.11. NCC Compliance Statement (Taiwan)

NCC Certification ID: CCAF23Y10120T9

The Proteus-III-SPI complies with the provisions on the National Communications Commission of Taiwan.

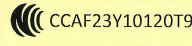
Due to the small size of the Proteus-III-SPI label the NCC Certification ID is placed on the packaging label.

22.11.1. NCC certificate



耕興股份有限公司
電信管制射頻器材型式認證證明

證照字號：型式字第 AF 號

| | |
|---------------|---|
| 一、申請者： | 伍仕電子股份有限公司 |
| 二、地址： | 臺北市內湖區民權東路6段18之1號4樓 |
| 三、製造廠商： | WE, Würth Elektronik or Würth Elektronik eiSos |
| 四、器材名稱： | Bluetooth 5.1 module T&R |
| 五、廠牌： | Würth Electronics Co., Ltd. |
| 六、型號： | 2611011024010 |
| 七、發射功率（電場強度）： | 詳如備註表格 |
| 八、工作頻率： | 詳如備註表格 |
| 九、審驗日期： | 112年8月3日 |
| 十、審驗合格標籤式樣： |  |

十一、警語或標示要求：（器材本體、使用手冊、外包裝盒等應遵守下列標示要求）

- 請依上列型號、標籤式樣於電信管制射頻器材本體明顯處標示其型號及審驗合格標籤，並於包裝盒標示主管機關印章。最終產品應於本體明顯處標示非隨插即用射頻模組（組件）之審驗合格標籤及最終產品型號，並於包裝盒標示主管機關印章。
- 電信管制射頻器材應依本會或相關技術規範規定於指定位置標示中文警語。
- 經授權使用射頻模組（組件）之審驗合格標籤者，應於最終產品說明書及包裝盒提供充分與正確之資訊。
- 於國際網路販賣取得審驗證明之電信管制射頻器材者，應於該國際網路網頁提供審驗合格標籤或符合性聲明標籤資訊。
- 使用手冊應標示下列資訊：
 - 取得審驗證明之低功率射頻器材，非經核准，公司、商號或使用者均不得擅自變更頻率、加大功率或變更原設計之特性及功能。低功率射頻器材之使用不得影響飛航安全及干擾合法通信；經發現有干擾現象時，應立即停用，並改善至無干擾時方得繼續使用。
 - 前述合法通信，指依電信管理法規定作業之無線電通信。低功率射頻器材須忍受合法通信或工業、科學及醫療用電波輻射性電機設備之干擾。

十二、特殊記載事項：

- 經取得審驗證明之電信管制射頻器材，如變更其廠牌、型號、技術規格或射頻性能時，除電信管制射頻器材審驗管理办法另有規定外，應重新申請審驗。
- 經取得審驗證明之審驗證明之電信管制射頻器材或非隨插即用射頻模組（組件），於電信管制射頻器材相關技術規範修訂審驗相關章節時，應依技術規範明定之實施期限，申請重新審驗辦理，未依規定重新審驗者，原證機關（構）得廢止其審驗證明。
- 取得型式認證證明、符合性聲明證明或簡易符合性聲明證明者，應妥為保管申請審驗之電信管制射頻器材或非隨插即用射頻模組（組件）、外接電源、配件、外接天線、與檢驗報告或測試報告相符之測試治具及與檢驗報告或測試報告使用相同版本之測試軟體至該器材停止生產或停止輸入後五年。
- 取得型式認證證明或符合性聲明證明者得授權他人於同廠牌同型號之電信管制射頻器材或非隨插即用射頻模組（組件）使用審驗合格標籤或符合性聲明標籤，授權他人使用審驗合格標籤或符合性聲明標籤以取得審驗證明者於主管機關指定網站登錄或委託原證機關（構）登錄。
- 以取得審驗證明之完全射頻模組（組件）組裝成完全最終產品後，取得該完全射頻模組（組件）之審驗證明者，應於該完全最終產品販賣前，檢附標註完全最終產品廠牌、型號及外觀照片之電子檔案並應提供組裝該完全射頻模組（組件）之完全最終產品其內部及電路板4x6吋以上具尺規之至少二面彩色照片，電路板主要元件須清晰可辨識。向原證機關（構）登錄。

說明：

- 本公司係依電信管理法第87條第2項及第3項規定訂定之電信管制射頻器材測試機構及驗證機構管理辦法第4條第1項規定，經認證組員（財團法人全國認證基金會）認可產品驗證制度符合 CNS 17065 或 ISO/IEC 17065 標準（TAF 標準編號：PC006），並經國家通訊傳播委員會委託辦理電信管制射頻器材之審驗工作。（機構地址：桃園市龜山區華亞一路52號、電話：03-3273456），核發本型式認證證明。
- 本設備之製造或輸入須遵守電信管理法相關規定。

(續下頁，續頁分開使用無效)

第 1 頁 共 2 頁

Figure 44: NCC certificate page 1

(三) 配件資訊:

| 名稱 | 廠牌/型號 | 備註 |
|-----|-------|----|
| 充電器 | 無此配件 | - |
| 煙電池 | 無此配件 | - |
| 充電線 | 無此配件 | - |
| 其 他 | 無此配件 | - |

備註:

1. 依「商品標示法」及「電器及電子商品標示基準」規定，標示事項貼於商品或內外包裝上，以免違法而受處分。

2. 本型式認證證明僅代表完成電信管制射頻器材審驗管理辦法所訂之審驗程序其審驗範圍僅限本器材之無線射頻硬體功能，不作為其他(如器材之資安安全檢測、產地、等)認可證明。

3. 本器材電信介面使用天線資訊如下:

| Ant. | 廠牌 | 型號 | 天線型式 | 天線增益 |
|---------|------|---------------|-------------|----------|
| Antenna | 伍仕電子 | 2611011024010 | PCB Antenna | -1.6 dBi |

4. 本設備之電信介面符合技術規範及工作頻段資訊如下。

| 電信介面 | 技術規範(年份) | 章節 | 工作頻率 | 功率 or 場強 | 備註 |
|-----------|----------------|------|--------------|----------|---------------|
| Bluetooth | LP0002(109 年版) | 4.10 | 2402~2480MHz | 2.10 dBm | BT LE GFSK |

BW=Bandwidth

中 華 民 國 1 1 2 年 8 月 3 日

以下空白

第 2 頁 共 2 頁

Figure 45: NCC certificate page 2

23. References

- [1] Bluetooth®. Bluetooth® Core Specification, version 5.1. <https://www.bluetooth.com/specifications/specs/core-specification-5-1/>.
- [2] Würth Elektronik. Application note 6 - Proteus high throughput mode. <http://www.we-online.com/ANR006>.
- [3] Würth Elektronik. Application note 9 - Proteus-III(-SPI) advanced developer guide. <http://www.we-online.com/ANR009>.
- [4] Würth Elektronik. Smart Commander PC tool. <https://www.we-online.com/SmartCommander>.
- [5] IEEE Registration Authority. <https://standards.ieee.org/products-programs/regauth/>.
- [6] Bluetooth®. Bluetooth® Assigned numbers. <https://www.bluetooth.com/specifications/assigned-numbers/>.
- [7] Nordic Semiconductor. nRF Device Firmware Update app for Android. <https://play.google.com/store/apps/details?id=no.nordicsemi.android.dfu>.
- [8] Nordic Semiconductor. nRF Device Firmware Update app for iOS. <https://apps.apple.com/us/app/device-firmware-update/id1624454660>.
- [9] Würth Elektronik. Application note 27 - Bluetooth listing guide. <http://www.we-online.com/ANR027>.

24. Important notes

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

24.1. General customer responsibility

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

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

It has to be clearly pointed out that the possibility of a malfunction of electronic components or failure before the end of the usual lifetime cannot be completely eliminated in the current state of the art, even if the products are operated within the range of the specifications. The same statement is valid for all software 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.

24.3. Best care and attention

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

24.4. Customer support for product specifications

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

24.5. Product improvements

Due to constant product improvement, product specifications may change from time to time. As a standard reporting procedure of the Product Change Notification (PCN) according to the JEDEC-Standard, we inform about major changes. In case of further queries regarding the PCN, the field sales engineer, the internal sales person or the technical support team in charge should be contacted. The basic responsibility of the customer as per section 24.1 and 24.2 remains unaffected. All wireless connectivity module driver software "wireless connectivity SDK" and its source codes as well as all PC software tools are not subject to the Product Change Notification information process.

24.6. Product life cycle

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

24.7. Property rights

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

24.8. General terms and conditions

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

25. Legal notice

25.1. Exclusion of liability

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

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

25.2. Suitability in customer applications

The customer bears the responsibility for compliance of systems or units, in which Würth Elektronik eiSos GmbH & Co. KG products are integrated, with applicable legal regulations. Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of Würth Elektronik eiSos GmbH & Co. KG components in its applications, notwithstanding any applications-related 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.

25.3. Trademarks

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

25.4. Usage restriction

Würth Elektronik eiSos GmbH & Co. KG products have been designed and developed for usage in general electronic equipment only. This product is not authorized for use in equipment

where a higher safety standard and reliability standard is especially required or where a failure of the product is reasonably expected to cause severe personal injury or death, unless the parties have executed an agreement specifically governing such use. Moreover, Würth Elektronik eiSos GmbH & Co. KG products are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. Würth Elektronik eiSos GmbH & Co. KG must be informed about the intent of such usage before the design-in stage. In addition, sufficient reliability evaluation checks for safety must be performed on every electronic component, which is used in electrical circuits that require high safety and reliability function or performance. By using Würth Elektronik eiSos GmbH & Co. KG products, the customer agrees to these terms and conditions.

26. 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.

26.1. Limited license

Würth Elektronik eiSos hereby grants you a limited, non-exclusive, non-transferable and royalty-free license to use the software and under the conditions that will be set forth in 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.

26.2. Usage and obligations

The responsibility for the applicability and use of the Würth Elektronik eiSos wireless connectivity product with the incorporated firmware in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to you to evaluate and investigate, where appropriate, and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for your respective application or not.

You are responsible for using the Würth Elektronik eiSos wireless connectivity product with the incorporated firmware in compliance with all applicable product liability and product safety laws. You acknowledge to minimize the risk of loss and harm to individuals and bear the risk for failure leading to personal injury or death due to your usage of the product.

Würth Elektronik eiSos' products with the incorporated firmware are not authorized for use in safety-critical applications, or where a failure of the product is reasonably expected to cause severe personal injury or death. Moreover, Würth Elektronik eiSos' products with the incorporated firmware are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. You

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

26.3. Ownership

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

26.4. Firmware update(s)

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

26.5. Disclaimer of warranty

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

26.6. Limitation of liability

Any liability not expressly provided by Würth Elektronik eiSos shall be disclaimed.

You agree to hold us harmless from any third-party claims related to your usage of the Würth Elektronik eiSos' products with the incorporated firmware, software and source code. Würth Elektronik eiSos disclaims any liability for any alteration, development created by you or your customers as well as for any combination with other products.

26.7. 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.

26.8. 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.

26.9. Miscellaneous

Würth Elektronik eiSos reserves the right at any time to change this terms at its own discretion. It is your responsibility to check at Würth Elektronik eiSos homepage for any updates. Your continued usage of the products will be deemed as the acceptance of the change.

We recommend you to be updated about the status of new firmware and software, which is available on our website or in our data sheet and manual, and to implement new software in your device where appropriate.

By ordering a product, you accept these license terms in all terms.

List of Figures

| | | |
|-----|--|-----|
| 1. | Proteus-III-SPI | 14 |
| 2. | Block diagram of the module | 17 |
| 3. | Radio transmitting @ 8 dBm output power, 1 Mbps Bluetooth® LE mode, Clock = HFXO, Regulator = DC/DC (typical values) | 20 |
| 4. | Current consumption calculation in advertising mode with 40 ms advertising interval with 8 dBm output power, UART/SPI disabled | 21 |
| 5. | Pinout (top view) | 24 |
| 6. | Minimal pin connections | 27 |
| 7. | State overview | 35 |
| 8. | DTM flow chart | 56 |
| 9. | SPI packet timing, example <i>SPI_INT</i> "active high" | 64 |
| 10. | SPI master to slave communication, example <i>SPI_INT</i> "active high" | 65 |
| 11. | SPI slave to master communication, example <i>SPI_INT</i> "active high" | 66 |
| 12. | Command sequence when transmitting data | 175 |
| 13. | Configure the local GPIOs via local host | 178 |
| 14. | Configure the local GPIOs via remote device host | 178 |
| 15. | Read the configuration of the local GPIOs via local host | 179 |
| 16. | Read the configuration of the local GPIOs via remote device host | 179 |
| 17. | Set the output value of a GPIO via host controller | 180 |
| 18. | Read the input value of a GPIO via host controller | 180 |
| 19. | Set the output value of a GPIO via remote device | 181 |
| 20. | Read the input value of a GPIO via remote device | 181 |
| 21. | PWM behaviour | 182 |
| 22. | nRF Device Firmware Update app | 190 |
| a. | Select file and device | 190 |
| b. | Scan list | 190 |
| c. | FOTA running | 190 |
| 23. | Layout | 194 |
| 24. | Placement of the module with integrated antenna | 195 |
| 25. | Dimensioning the antenna feed line as micro strip | 196 |
| 26. | 2.4 GHz dipole-antenna | 198 |
| 27. | Reference design: Schematic page | 200 |
| 28. | Reference design: Layout | 201 |
| 29. | Antenna characteristic of the module with its integrated antenna measured on the official evaluation board (2611149024011) | 202 |
| 30. | Trace design: Layout | 203 |
| 31. | Reference design: Stack-up | 203 |
| 32. | Trace design: Schematic | 204 |
| 33. | On-board PCB antenna | 205 |
| 34. | External antenna connection | 205 |
| 35. | Reflow soldering profile | 208 |
| 36. | Module dimensions [mm] | 213 |
| 37. | Footprint WE-FP-4+ [mm] | 214 |
| 38. | Lot number structure | 215 |
| 39. | Label of the Proteus-III-SPI | 216 |
| 40. | FCC certificate | 222 |

| | | |
|-----|--------------------------------------|-----|
| 41. | IC certificate | 223 |
| 42. | ETA-WPC certificate page 1 | 226 |
| 43. | ETA-WPC certificate page 2 | 227 |
| 44. | NCC certificate page 1 | 228 |
| 45. | NCC certificate page 2 | 229 |

List of Tables

| | | |
|-----|--|-----|
| 1. | Ordering information | 17 |
| 2. | Recommended operating conditions | 18 |
| 3. | Absolute maximum ratings | 18 |
| 4. | Power consumption for 100% transmission/reception | 19 |
| 5. | Timing and RSSI | 22 |
| 6. | Transmit and receive power | 22 |
| 7. | Sensitivity at different data rates | 22 |
| 8. | Pin characteristics | 23 |
| 9. | Pinout, first part | 25 |
| 10. | Pinout, second part | 26 |
| 17. | LED behavior of the Proteus-III-SPI | 36 |
| 68. | Message overview: Requests | 121 |
| 69. | Message overview: Confirmations | 122 |
| 70. | Message overview: Indications | 123 |
| 71. | nRF52840 IC revision overview | 125 |
| 72. | Security configuration flags | 135 |
| 73. | Scan configuration flags | 137 |
| 74. | Beacon configuration flags | 139 |
| 75. | Advertising packet configuration flags | 142 |
| 76. | Table of settings | 172 |
| 77. | Maximum throughput timings, packet error rate = 0% | 176 |
| 78. | Supported GPIO_IDs | 183 |
| 79. | UUID default values | 185 |
| 80. | Compatibility matrix | 189 |
| 82. | Classification reflow soldering profile, Note: refer to IPC/JEDEC J-STD-020E | 207 |
| 83. | Package classification reflow temperature, PB-free assembly, Note: refer to IPC/JEDEC J-STD-020E | 208 |
| 84. | Dimensions | 212 |
| 85. | Weight | 212 |
| 86. | Lot number details | 215 |
| 87. | CRC8 Test Vectors | 240 |

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 87: CRC8 Test Vectors

B. Example codes for host integration

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

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

#define CMD_OFFSET_TO_DATAFIELD 4
#define CMD_OVERHEAD (CMD_OFFSET_TO_DATAFIELD+1)

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

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

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

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

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

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

**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

WÜRTH ELEKTRONIK MORE THAN YOU EXPECT