

nRF8001

Single-chip *Bluetooth*[®] low energy solution

Preliminary Product Specification 0.9.3

Key Features

- *Bluetooth* low energy peripheral device
- Stack features:
 - Low energy PHY layer
 - Low energy link layer slave
 - Low energy host for devices in the peripheral role
 - Proprietary Application Controller Interface (ACI)
- Hardware features:
 - 16 MHz crystal oscillator
 - Low power 32 kHz ± 250 ppm RC oscillator
 - 32.768 kHz crystal oscillator
 - DC/DC converter
 - Temperature sensor
 - Battery monitor
 - Direct Test Mode interface
- Ultra low power consumption
- Single 1.9-3.6V power supply
- Temperature range -40 – 85°C
- Compact 5×5 mm QFN32 package
- RoHS compliant

Applications

- Sport and fitness sensors
- Health care sensors
- Proximity
- Watches
- Personal User Interface Devices (PUID)
- Remote controls

Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

Life support applications

Nordic Semiconductor's products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

| Datasheet status | |
|-----------------------------------|---|
| Objective Product Specification | This product specification contains target specifications for product development. |
| Preliminary Product Specification | This product specification contains preliminary data; supplementary data may be published from Nordic Semiconductor ASA later. |
| Product Specification | This product specification contains final product specifications. Nordic Semiconductor ASA reserves the right to make changes at any time without notice in order to improve design and supply the best possible product. |

Contact details

For your nearest dealer, please see www.nordicsemi.com

Main office:

Otto Nielsens veg 12
 7004 Trondheim
 Phone: +47 72 89 89 00
 Fax: +47 72 89 89 89
www.nordicsemi.com



RoHS statement

Nordic Semiconductor's products meet the requirements of Directive 2002/95/EC of the European Parliament and of the Council on the Restriction of Hazardous Substances (RoHS). Complete hazardous substance reports as well as material composition reports for all active Nordic Semiconductor products can be found on our website www.nordicsemi.com.

Revision History

| Date | Version | Description |
|---------------|---------|--|
| May 27th 2011 | 0.9.3 | <ul style="list-style-type: none">Added section 12.4 on page 34Revised chapter 13 on page 35.Modified Figure 24. on page 55 through Figure 28. on page 57Modified Figure 32. on page 60 and Figure 33. on page 64Fixed minor issues throughout document This release of the document is accurate for build code C. |
| May 5th 2011 | 0.9.2 | Section 22.3.2.1 "UUID configuration format" added. This release of the document is accurate for build code CX. |
| February 2011 | 0.9.1 | Current consumption chapter (chapter 12) updated. nRF8001 configuration chapter (chapter 7) added. This release of the document is accurate for build code BX. |
| January 2011 | 0.9 | First release of the document. |

Contents

| | | |
|-----------|---|-----------|
| 1 | Introduction | 7 |
| 1.1 | Prerequisites | 7 |
| 1.2 | Writing conventions | 7 |
| 1.3 | Bluetooth specification releases | 7 |
| 2 | Bluetooth Qualification ID | 8 |
| | Part A: nRF8001 Physical description..... | 9 |
| 3 | Product overview | 10 |
| 4 | Bluetooth low energy features | 11 |
| 4.1 | Features..... | 12 |
| 5 | Physical product overview | 13 |
| 5.1 | Package and pin assignment..... | 13 |
| 5.2 | Pin functions | 14 |
| 6 | Analog features | 15 |
| 6.1 | RF transceiver | 15 |
| 6.2 | On-chip oscillators | 15 |
| 6.3 | DC/DC converter | 18 |
| 6.4 | Temperature sensor | 18 |
| 6.5 | Battery monitor | 18 |
| 7 | Interfaces | 19 |
| 7.1 | Application Controller Interface (ACI) | 19 |
| 7.2 | Active signal..... | 25 |
| 7.3 | Direct Test Mode interface..... | 25 |
| 8 | nRF8001 configuration | 27 |
| 9 | Data storage and memory retention..... | 29 |
| 9.1 | Permanent Storage..... | 29 |
| 9.2 | Volatile Storage | 29 |
| 10 | Absolute maximum ratings | 30 |
| 11 | Operating conditions | 31 |
| 12 | Electrical specifications | 32 |
| 12.1 | Digital I/O signal levels | 32 |
| 12.2 | Radio characteristics | 32 |
| 12.3 | Analog feature characteristics | 33 |
| 12.4 | Current consumption parameters | 34 |
| 13 | Dynamic current consumption | 35 |
| 13.1 | Current consumption - connection..... | 35 |
| 13.2 | Current consumption - advertising..... | 37 |
| 13.3 | Current consumption calculation examples | 39 |
| 13.4 | Recommendations for low power operation | 40 |
| 14 | External component requirements and recommendations..... | 41 |
| 14.1 | 16 MHz crystal specification requirements | 41 |
| 14.2 | 32.768 kHz crystal specification requirements | 41 |
| 14.3 | Antenna Matching and Balun..... | 42 |
| 14.4 | DC/DC Converter requirements..... | 42 |

| | | |
|-----------|---|-----------|
| 15 | Mechanical specifications | 43 |
| 16 | Ordering information | 44 |
| 16.1 | Package marking | 44 |
| 16.2 | Abbreviations | 44 |
| 16.3 | Product options | 44 |
| 17 | Reference circuitry | 45 |
| 17.1 | Schematic for nRF8001 with DC/DC converter enabled | 45 |
| 17.2 | Layout | 46 |
| 17.3 | Bill of Materials | 46 |
| 17.4 | Schematic for nRF8001 with DC/DC converter disabled | 47 |
| 17.5 | Layout | 48 |
| 17.6 | Bill of Materials | 48 |
| | Part B: The nRF8001 Application Controller Interface (ACI) | 49 |
| 18 | Operating principle | 50 |
| 18.1 | Packet structure | 51 |
| 19 | ACI packet types | 52 |
| 19.1 | System commands | 52 |
| 19.2 | Data commands | 52 |
| 19.3 | Events | 52 |
| 20 | Service pipes | 53 |
| 20.1 | Functional description | 53 |
| 20.2 | Defining Service pipes | 54 |
| 20.3 | Data transfer on a service pipe | 54 |
| 20.4 | Transmit pipes | 54 |
| 20.5 | Receive pipes | 58 |
| 20.6 | Service pipe availability | 60 |
| 21 | Flow control | 63 |
| 21.1 | System command buffering | 63 |
| 21.2 | Data command buffering | 63 |
| 21.3 | Flow control initialization | 65 |
| 22 | Operational modes | 66 |
| 22.1 | Overview of operational modes | 66 |
| 22.2 | Sleep mode | 68 |
| 22.3 | Setup mode | 68 |
| 22.4 | Active mode | 71 |
| 22.5 | Test mode | 74 |
| 22.6 | RF PHY testing | 75 |
| 23 | Protocol reference | 77 |
| 23.1 | Command and event overview | 77 |
| 24 | System commands | 82 |
| 24.1 | Test (0x01) | 82 |
| 24.2 | Sleep (0x04) | 83 |
| 24.3 | GetDeviceVersion (0x09) | 84 |
| 24.4 | Echo (0x02) | 85 |
| 24.5 | Wakeup (0x05) | 86 |

| | | |
|-----------|-------------------------------------|------------|
| 24.6 | GetBatteryLevel (0x0B) | 87 |
| 24.7 | GetTemperature (0x0C)..... | 88 |
| 24.8 | Setup (0x06) | 89 |
| 24.9 | SetTxPower (0x12) | 90 |
| 24.10 | GetDeviceAddress (0x0A) | 91 |
| 24.11 | Connect (0x0F) | 92 |
| 24.12 | RadioReset (0x0E) | 94 |
| 24.13 | Bond (0x10) | 95 |
| 24.14 | Disconnect (0x11)..... | 97 |
| 24.15 | ChangeTimingRequest (0x13)..... | 98 |
| 24.16 | OpenRemotePipe (0x14)..... | 100 |
| 24.17 | DtmCommand (0x03) | 102 |
| 24.18 | ReadDynamicData (0x07) | 103 |
| 24.19 | WriteDynamicData (0x08)..... | 104 |
| 25 | Data commands..... | 105 |
| 25.1 | SendData (0x15)..... | 105 |
| 25.2 | RequestData (0x17)..... | 106 |
| 25.3 | SetLocalData (0x0D) | 107 |
| 25.4 | SendDataAck (0x16)..... | 108 |
| 26 | System Events..... | 109 |
| 26.1 | DeviceStartedEvent (0x81)..... | 109 |
| 26.2 | EchoEvent (0x82) | 110 |
| 26.3 | HardwareErrorEvent (0x83)..... | 111 |
| 26.4 | CommandResponseEvent (0x84)..... | 112 |
| 26.5 | ConnectedEvent (0x85) | 113 |
| 26.6 | DisconnectedEvent (0x86)..... | 115 |
| 26.7 | BondStatusEvent (0x87)..... | 116 |
| 26.8 | PipeStatusEvent (0x88) | 118 |
| 26.9 | TimingEvent (0x89)..... | 121 |
| 27 | Data Events..... | 122 |
| 27.1 | DataCreditEvent (0x8A)..... | 122 |
| 27.2 | PipeErrorEvent (0x8D)..... | 123 |
| 27.3 | DataReceivedEvent (0x8C) | 124 |
| 27.4 | DataAckEvent (0x8B) | 125 |
| 28 | Appendix..... | 126 |
| 28.1 | ACI Status Codes | 126 |
| 28.2 | Bonding Status Codes | 127 |
| 28.3 | Error Codes | 128 |
| 28.4 | Setup and procedure selection | 129 |
| 29 | Glossary..... | 130 |

1 Introduction

nRF8001 is a *Bluetooth*[®] low energy solution designed for operation in the peripheral role. By integrating a *Bluetooth* low energy compliant radio (PHY), slave mode link controller and host, nRF8001 offers you an easy way to add *Bluetooth* low energy connectivity to your application.

nRF8001 offers a serial interface (ACI) for configuration and control from your microcontroller. This microcontroller will in the remainder of this document be referred to as the application processor.

This document is divided into two parts:

- **Part A** defines the nRF8001 hardware and electrical specifications as well as operating procedures.
- **Part B** describes the Application Controller Interface (ACI); the logical interface between the nRF8001 and your application

1.1 Prerequisites

To fully understand this document a good knowledge of electronic and software engineering is required.

Knowledge of *Bluetooth* specification, Ver. 4.0, Volumes 1, 3, 4 and 6 is required to operate nRF8001 correctly and understand the terminology used within this document.

1.2 Writing conventions

This product specification follows a set of typographic rules to ensure that the document is consistent and easy to read. The following writing conventions are used:

- Command and event names, bit state conditions, and register names are written in *Courier*.
- Pin names and pin signal conditions are written in *Courier New bold*.
- Placeholders for parameters are written in *italic regular text font*. For example, a syntax description of *Connect* will be written as: *Connect(TimeOut, AdvInterval)*.
- Fixed parameters are written in regular text font. For example, a syntax description of *Connect* will be written as: *Connect(0x00F0, Interval)*.
- Cross references are [underlined and highlighted in blue](#).
- Terms originating from *Bluetooth* specifications are written in **bold** upon first mention, and are written in normal text font in subsequent mentions.

1.3 *Bluetooth* specification releases

This document is valid based on *Bluetooth* specification core v4.0 for a low energy device operating in the peripheral role.

2 ***Bluetooth* Qualification ID**

nRF8001 is listed as an EP-QDL on the Qualified listings page of the *Bluetooth* Special Interest Group website (<https://www.bluetooth.org/tpg/listings.cfm>).

For details on the design qualifications, please refer to the following qualification IDs:

- B017595: EP-QDL containing core PICS
- B017566: RF PHY Component QDL containing layout and schematics for the EP-QDL reference design

Part A: nRF8001 Physical description

This section defines the physical features of nRF8001 and its electrical/mechanical specifications. It also defines the nRF8001 hardware, specifications and provides information on operating procedures.

3 Product overview

The main features of nRF8001 are the *Bluetooth* low energy PHY and the protocol engine handling the *Bluetooth* low energy link controller and host stack. Additional analog sub-systems needed for the *Bluetooth* low energy operation, such as power management and several oscillator options, are also included.

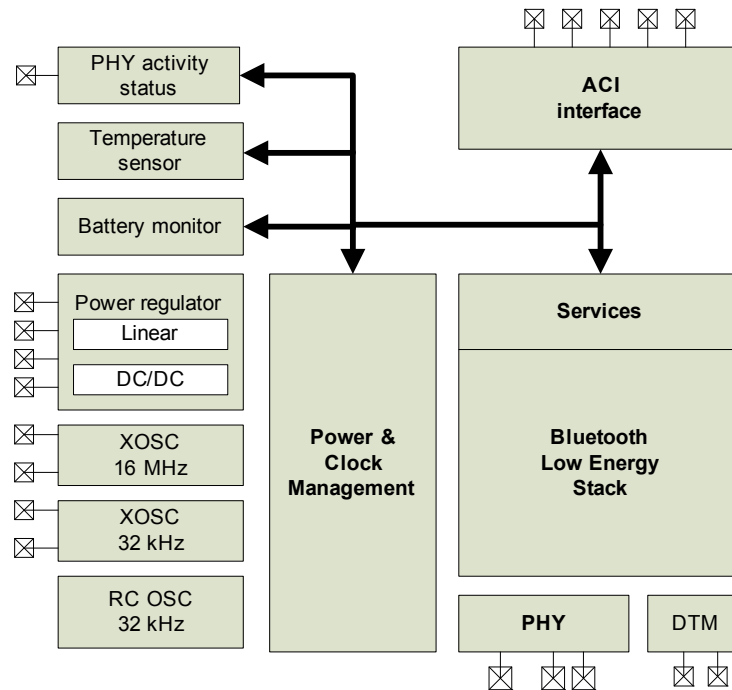


Figure 1. nRF8001 block diagram

nRF8001 offers on-chip non-volatile memory for storing service configurations. This on-chip storage lets you select and combine the necessary services for your application. It also relieves your application processor of the burden of handling all real-time operations in relation to the *Bluetooth* low energy communication protocol.

nRF8001 includes a power supply voltage monitor and a temperature sensor that further reduces the requirements to the application processor. These features are accessible through the serial interface (ACI).

nRF8001 also offers an optional output signal that is activated before the radio becomes active. This timing signal enables you to control the peak current drain of your application, avoiding overload of your power supply (for most applications this is usually a small battery). You can also use this timing signal to control the application circuitry, avoiding noise interference when the nRF8001 radio is operating.

A separate serial interface (UART) gives you access to the *Bluetooth* low energy Direct Test Mode (DTM). This interface is used to control the *Bluetooth* low energy radio (RF PHY) and is supported by commercially available *Bluetooth* test equipment used for *Bluetooth* qualification. This serial interface also enables you to test radio performance and to optimize your antenna.

4 Bluetooth low energy features

nRF8001 includes *Bluetooth* low energy protocols and profiles (see [Figure 2.](#)) that are defined in the *Bluetooth* specification 4.0 in the following volumes:

- Volume 2 Part D : Error Codes
- Volume 3: Core System Package [Host Volume]
 - Part A: Logical Link Control and Protocol
 - Part C: Generic Access Profile (GAP)
 - Part F: Attribute Protocol (ATT)
 - Part G: Generic Attribute Profile (GATT)
 - Part H: Security Manager (SM)
- Volume 6: Core System Package [Low Energy Controller Volume]

nRF8001 supports the peripheral role as defined in the *Bluetooth* low energy specification, Volume 3, Part C, 2.2.2.3 Peripheral Role. All mandatory features for a device operating in the peripheral role are supported. In addition to the mandatory features, a subset of optional features are available for use. Access to these features is specified in Part B of this document.

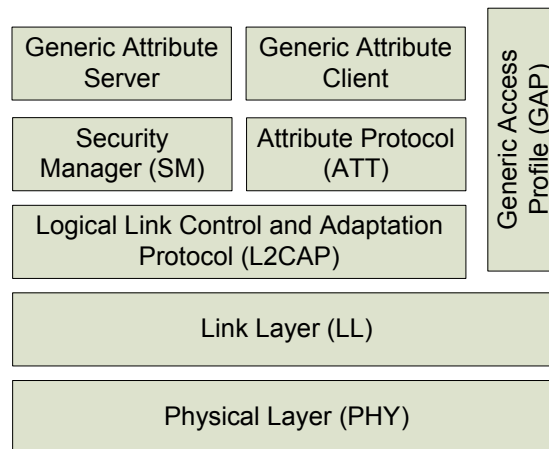


Figure 2. Bluetooth low energy layers implemented in nRF8001

4.1 Features

General nRF8001

- Radio features
 - *Bluetooth* low energy RF transceiver
 - Ultra low peak current consumption <14mA
 - Common Tx/Rx terminals
 - Low current for connection oriented profiles, typically 2 μ A
 - Ultra low current for connection-less oriented profiles, typically 500nA
- Auxiliary features
 - Integrated low frequency reference oscillator
 - Power management
 - Battery monitor
 - Temperature monitor
 - DC-DC converter to reduces current by up to 20% if enabled
 - Integrated 16 MHz crystal oscillator
 - OTP for customer configuration
- Interfaces
 - UART Test Interface for Direct Test Mode
 - Application Controller Interface (ACI)
 - Radio Active signal

nRF8001 *Bluetooth* low energy

- *Bluetooth* low energy stack
 - All layers up to GATT included in core software stack
- Link Layer Features
 - Slave role
 - Control PDUs in the slave role
 - 27 byte MTU
 - Encryption
- L2CAP
 - 27 byte MTU
 - Slave connection update
 - Attribute Channel
 - Security Channel
- General Access Profile (GAP) features
 - Discoverable modes
 - Dedicated bonding
 - GAP attributes
- Attribute Protocol
 - Mandatory client protocol
 - Mandatory server protocol
- Security Manager
 - Generation of keys for encryption
 - Just works security
- Generic Attribute Profile (GATT)
 - Mandatory client profile features
 - Mandatory server profile features
- Direct Test Mode (DTM)
 - DTM for RF qualification

5 Physical product overview

This section describes the physical properties of nRF8001.

5.1 Package and pin assignment

nRF8001 is available in a 5x5 mm QFN32 package. The backplate of the QFN32 capsule must be grounded to the application PCB in order to achieve optimal performance. The physical dimensions of the QFN32 are presented in [chapter 15 on page 43](#).

[Figure 3](#) shows the pin assignment for nRF8001 and [Table 1. on page 14](#) describes the pin functionality.

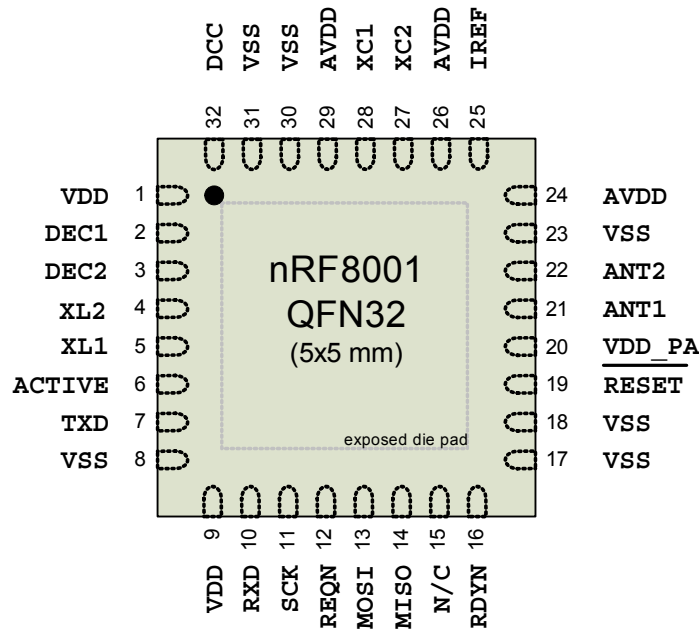


Figure 3. nRF8001 pin assignment (top view)

5.2 Pin functions

| Pin | Pin name | Pin function | Description |
|-----------------|----------|----------------|--|
| 1 | VDD | Power | Power supply (1.9 – 3.6V) |
| 2 | DEC1 | Power | Regulated power supply output for decoupling purposes only. Connect 100nF capacitor to ground |
| 3 | DEC2 | Power | Regulated power supply output for decoupling purposes only. Connect 33nF capacitor to ground |
| 4 | XL2 | Analog output | Connect to 32.768 kHz crystal oscillator. If internal RC oscillator is enabled, this pin shall not be connected. |
| 5 | XL1 | Analog input | Connect to 32.768 kHz crystal oscillator or external 32.768 kHz clock reference. If internal RC oscillator is enabled, this pin shall not be connected. |
| 6 | ACTIVE | Digital output | Device RF front end activity indicator |
| 7 | TXD | Digital output | UART (transmit) for <i>Bluetooth</i> low energy Direct Test Interface |
| 8 | VSS | Power | Ground (0V) |
| 9 | VDD | Power | Power supply (1.9 – 3.6V) |
| 10 | RXD | Digital input | UART (receive) for <i>Bluetooth</i> low energy Direct Test Interface |
| 11 | SCK | Digital input | ACI clock input |
| 12 | REQN | Digital input | ACI request pin (handshaking, active low) |
| 13 | MOSI | Digital input | ACI Master Out Slave In |
| 14 | MISO | Digital output | ACI Master In Slave Out |
| 15 | N/C | Digital input | Not connected |
| 16 | RDYN | Digital output | ACI device ready indication (handshaking), active low |
| 17 | VSS | Power | Ground (0V) |
| 18 | VSS | Power | Ground (0V) |
| 19 | RESET | Digital input | Reset (active low) |
| 20 | VDD_PA | Power output | Regulated power supply output for on-chip RF Power amplifier |
| 21 | ANT1 | RF | Differential antenna connection (TX and RX) |
| 22 | ANT2 | RF | Differential antenna connection (TX and RX) |
| 23 | VSS | Power | Ground (0V) |
| 24 | AVDD | Power | Analog power supply (1.9 – 3.6V DC) |
| 25 | IREF | Analog output | Current reference terminal. Connect 22 kΩ 1% resistor to ground |
| 26 | AVDD | Power | Analog power Supply (1.9 – 3.6V) |
| 27 | XC2 | Analog output | Connection for 16 MHz crystal oscillator. Leave unconnected if not in use. |
| 28 | XC1 | Analog input | Connection for 16 MHz crystal or external 16 MHz reference |
| 29 | AVDD | Power | Analog power supply (1.9 – 3.6V DC) |
| 30 | VSS | Power | Ground (0V) |
| 31 | VSS | Power | Ground (0V) |
| 32 | DCC | Power | Pulse Width Modulated (PWM) driver for the external LC filter if the DC/DC converter is enabled. If the DC/DC converter is disabled this pin shall be not connected. |
| Exposed die pad | vss | Power | Ground (0V) |

Table 1.nRF8001 pin functions

6 Analog features

The following analog features are included in nRF8001:

- *Bluetooth* low energy RF transceiver
- Three on-chip reference oscillators
- DC/DC converter for extended battery life with coin-cell batteries
- Temperature sensor
- Battery monitor

6.1 RF transceiver

nRF8001 includes an integrated RF transceiver which is compliant with the *Bluetooth* low energy specification Volume 6, Part A. The RF transceiver requires the following external components to operate:

- a 16 MHz crystal
- a resistor to set internal bias currents
- a balun to match an antenna to the receiver/ transmitter pins of nRF8001

6.1.1 Enabling the RF transceiver

All RF transceiver functionality and operation is controlled through the ACI. Configuring the GAP parameters and entering a mode of operation through the ACI enables the transceiver to send advertisement events and connect to a peer device. Data transfer is initiated after the negotiated *Bluetooth* low energy setup procedures have been completed.

6.2 On-chip oscillators

nRF8001 includes three integrated oscillators:

- Low power amplitude regulated 16 MHz crystal oscillator
- Ultra low power amplitude regulated 32.768 kHz crystal oscillator
- Ultra low power 32.768 kHz RC oscillator with ± 250 ppm frequency accuracy

The 16 MHz crystal oscillator provides the reference frequency for the RF transceiver. The two low frequency 32.768 kHz oscillators provide the protocol timing. Only one low frequency reference can be used at any one time. The choice of which reference to use depends on your application and will affect the design cost and current consumption. The low-frequency crystal oscillator clock can be driven by either a 32.768 kHz crystal oscillator or a 32.768 kHz external clock source.

6.2.1 Enabling the oscillators

The 16 MHz oscillator is automatically enabled when nRF8001 requires it. The 32.768 kHz oscillator is automatically enabled when nRF8001 is in a connection or advertising state. Both the 32.768 kHz and the 16 MHz reference sources and oscillator settings are set through the ACI, see [Part B, section 22.3 on page 68](#).

6.2.2 16 MHz oscillator

The 16 MHz crystal oscillator is designed for use with an AT-cut quartz crystal in parallel resonant mode. To achieve correct oscillation frequency, the load capacitance must match the specification in the crystal datasheet. [Figure 4. on page 16](#) shows how the crystal is connected to the 16 MHz crystal oscillator.

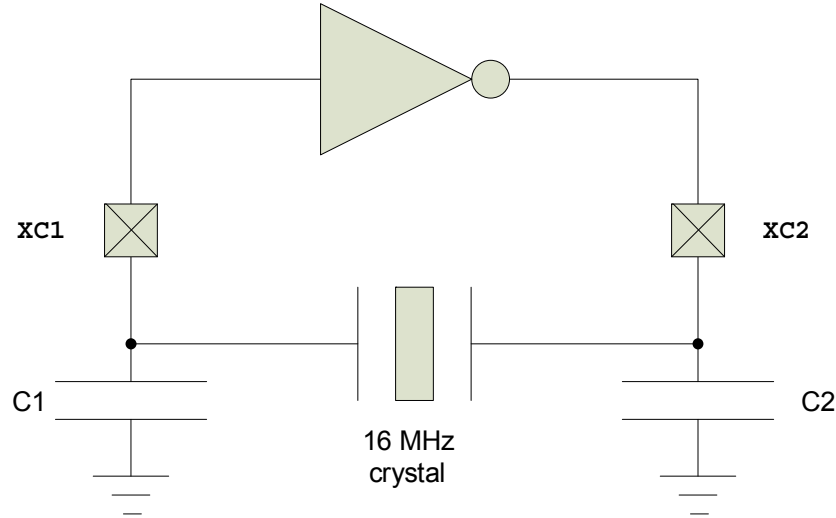


Figure 4. Circuit diagram of the nRF8001 16 MHz crystal oscillator

The load capacitance is the total capacitance seen by the crystal across its terminals and is given by:

$$C_{load} = \frac{(C1' \times C2')}{(C1' + C2')}$$

$$C1' = C1 + C_{pcb1} + C_{pin}$$

$$C2' = C2 + C_{pcb2} + C_{pin}$$

C1 and C2 are ceramic SMD capacitors connected between each crystal terminal and ground. C_{pcb1} and C_{pcb2} are stray capacitances on the PCB. C_{pin} is the pin input capacitance on the xc1 and xc2, typically 1pF. The load capacitance C1 and C2 should be of the same value.

6.2.3 External 16 MHz clock

nRF8001 may be used with an external 16 MHz reference applied to the xc1 pin instead of a 16 MHz crystal. An input amplitude of 0.8 V peak-to-peak or higher is recommended to achieve low current consumption. Keep the maximum voltage level so that all peak voltages are under the recommended maximum operating conditions as specified in [chapter 11 on page 31](#). The external signal must have an accuracy of 40 ppm or better. The xc1 pin loads the external applications crystal oscillator with approximately 1 pF in addition to PCB routing. Do not connect the xc2 pin.

6.2.4 32.768 kHz crystal oscillator

The 32.768 kHz crystal oscillator is designed for use with a quartz crystal in parallel resonant mode. To achieve correct oscillation frequency, the load capacitance must match the specification in the crystal datasheet. [Figure 5. on page 17](#) shows how the crystal is connected to the 32.768 kHz crystal oscillator.

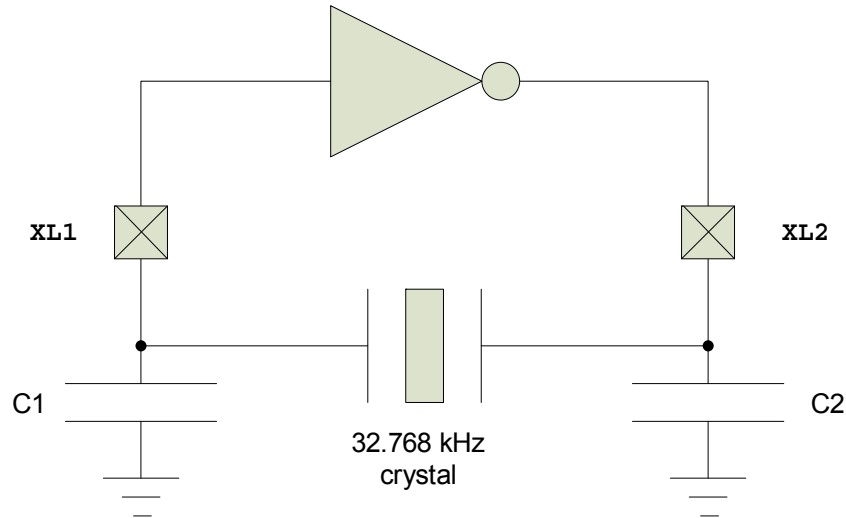


Figure 5. Circuit diagram of the nRF8001 32.768 kHz crystal oscillator

The load capacitance is the total capacitance seen by the crystal across its terminals and is given by:

$$C_{load} = \frac{(C1' \times C2')}{(C1' + C2')}$$

$$C1' = C1 + C_{pcb1} + C_{pin}$$

$$C2' = C2 + C_{pcb2} + C_{pin}$$

C1 and C2 are ceramic SMD capacitors connected between `xc1` and `xc2` and ground. `C_pcb1` and `C_pcb2` are stray capacitances on the PCB. `C_pin` is the input capacitance on the `xc1` and `xc2` pins, typically 1pF. C1 and C2 should be of the same value.

6.2.5 32.768 kHz RC oscillator

The nRF8001 32.768 kHz RC low frequency oscillator may be used as an alternative to the 32.768 kHz crystal oscillator. It has a frequency accuracy of ± 250 ppm in a stable temperature environment. The 32.768 kHz RC oscillator does not require external components.

6.2.6 External 32.768 kHz clock

nRF8001 may be used with an external 32.768 kHz clock applied to the `XL1` pin. The application processor sets the reference signal configuration. It can be a rail-to-rail signal or an analog signal. An analog input signal must have an amplitude of 0.2V peak-to-peak or greater. Keep the maximum and minimum voltage levels so that all peak voltages are under recommended maximum operating conditions as specified in [chapter 11 on page 31](#). If the external source is derived from the application processor's crystal oscillator, the `XL1` pin will load the applications crystal oscillator with approximately 3pF in addition to PCB routing.

6.3 DC/DC converter

nRF8001 incorporates linear supply voltage regulators and an optional step-down DC/DC converter. The internal linear regulators are always enabled. When enabled, the step-down DC/DC converter transforms the battery voltage to a lower internal voltage with minimal power loss. The converted voltage is then fed to the input of the linear regulators.

This feature is particularly useful for applications using battery technologies with higher nominal cell voltages. The reduction in supply voltage level from a high voltage to a low voltage reduces the peak power drain from the battery. Used with a 3V coin cell battery, the peak current drawn from the battery is reduced by approximately 20%.

Note: Three external discrete components are required in order to use the step-down converter. See [chapter 17 on page 45](#) for details on schematics, layout and BOM for the two power supply alternatives.

6.3.1 Enabling the DC/DC converter

The DC/DC converter is enabled through the ACI configuration, see [Part B, section 22.3 on page 68](#).

6.4 Temperature sensor

nRF8001 incorporates an integrated temperature sensor. The temperature sensor reports the silicon temperature. The electrical specification of the temperature sensor is in [chapter 12 on page 32](#).

6.4.1 Enabling the temperature sensor

The temperature sensor is enabled through the ACI protocol, see [Part B, section 24 on page 82](#). When nRF8001 receives an ACI command initiating the temperature reading it will enable the temperature sensor and start the internal measurement procedure. Upon completion, the nRF8001 returns an ACI event reporting the current temperature reading.

6.5 Battery monitor

nRF8001 incorporates an integrated battery monitor. The battery monitor reports the supply voltage (VDD) connected to nRF8001 supply pins. The electrical specification of the battery monitor is in [chapter 12 on page 32](#).

6.5.1 Enabling the battery monitor

The battery monitor sensor is enabled through the ACI protocol see [Part B, section 24 on page 82](#). When nRF8001 receives an ACI command initiating the battery reading it will enable the battery monitor and start the internal measurement procedure. Upon completion, the nRF8001 returns an ACI event reporting the current battery monitor reading.

7 Interfaces

This chapter defines the physical interfaces for nRF8001:

- Application Controller Interface, (ACI)
- Active signal
- *Bluetooth* low energy Direct Test Interface

7.1 Application Controller Interface (ACI)

The Application Controller Interface (ACI) enables an application processor to communicate with nRF8001. The ACI consists of a physical transport which is described in this chapter and a logical interface which is described in Part B.

7.1.1 Physical description

The physical ACI interface on nRF8001 consists of five pins. All ACI data exchanges use a standard SPI interface, with nRF8001 using a mode 0 slave interface to the application processor.

However, nRF8001 does not behave as a pure SPI slave device; nRF8001 can receive new data over-the-air at any time or be busy processing a connection event or new data. Consequently, the traditional CSN signal used to initiate a SPI transaction is replaced by two active low hand-shake signals; RDYN and REQN.

These hand shake signals allow nRF8001 to notify the application processor when it has received new data over-the-air and also to hold new data exchanges initiated by the application processor until it is ready to accept and process them. The ACI connections are shown in [Figure 6](#).

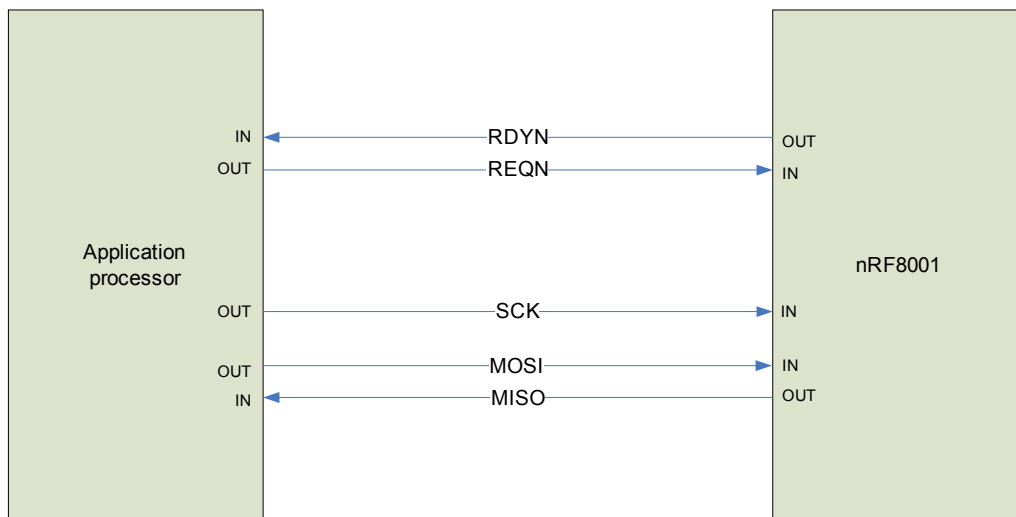


Figure 6. ACI interface between application processor and nRF8001

The data exchanges on the ACI interface are split in two types:

- Commands – Exchanges that are initiated by the application processor and data that is sent from the application processor to nRF8001.
- Event – Exchanges that are initiated by nRF8001 and data that is sent from nRF8001 to the application processor.

If nRF8001 has event data ready for the application processor when the processor requests a command exchange, the command and event will be combined in a full duplex exchange. nRF8001 sends out the event data at the same time as it receives command data. To accommodate this the application controller must always monitor the incoming data when issuing a command.

7.1.2 SPI mode

The ACI transport layer uses the SPI in the following mode (SPI mode 0):

| Type | Value |
|----------------|--|
| Data order | LSB first |
| Clock polarity | Zero (base value for the clock is zero) |
| Clock phase | Zero (data is read on the clock's rising edge) |

Table 2. SPI signal description

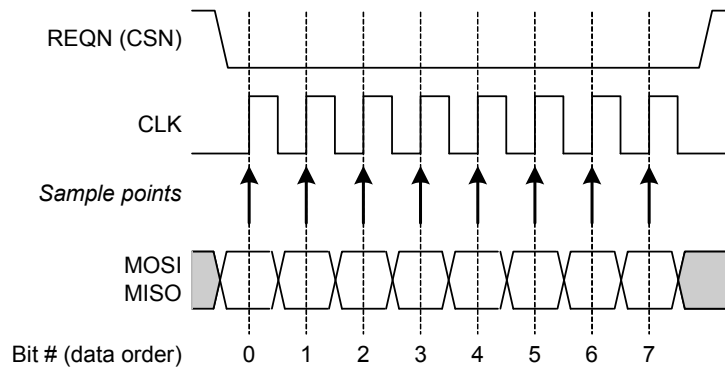


Figure 7. SPI mode 0 description

7.1.3 ACI connections

The required I/O pins needed on the application processor and nRF8001 for the ACI interface are listed in [Table 3](#).

| Signal | Application processor | nRF8001 | Description |
|--------|-----------------------|---------|---|
| MISO | Input | Output | SPI: Master In Slave Out |
| MOSI | Output | Input | SPI: Master Out Slave In |
| SCK | Output | Input | SPI: Serial data Clock |
| REQN | Output | Input | Application processor to nRF8001 handshake signal |
| RDYN | Input | Output | nRF8001 to application processor handshake signal |

Table 3. ACI I/O signals for an application processor and nRF8001

7.1.3.1 RDYN line

The application processor shall at all times have RDYN line configured as input with pull-up drivers. At power on reset and wake up from sleep scenarios, the RDYN level is valid after 62 ms from reset or wake up.

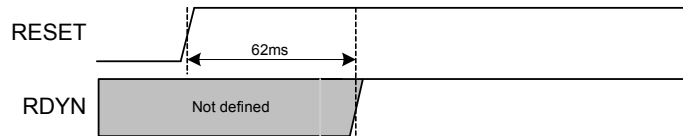


Figure 8. Power up sequence

7.1.4 ACI command exchange

Figure 9 shows the signaling timing of a command sent from the application processor to nRF8001.

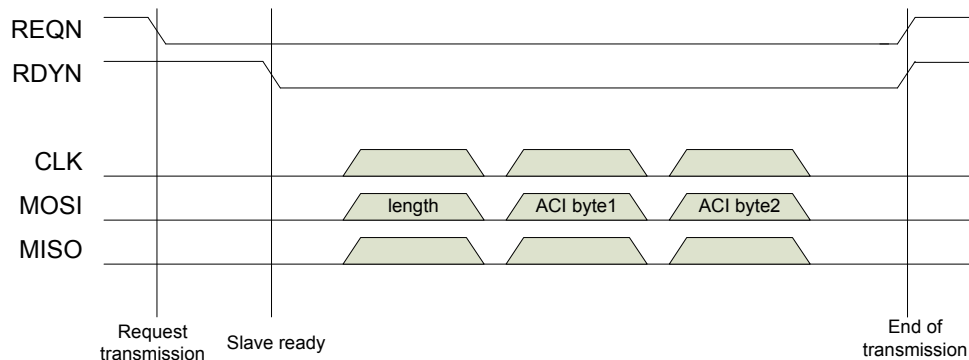


Figure 9. Data exchange from an application processor to nRF8001

The following procedure is performed when the application processor sends a command to nRF8001:

1. The application processor requests the right to send data by setting the **REQN** pin to ground.
2. nRF8001 sets the **RDYN** pin to ground when it is ready to receive data.
3. The application processor starts sending data on the **MOSI** pin:
 - Byte 1 (length byte) from the application processor defines the length of the message.
 - Byte 2 (ACI byte1) is the first byte of the ACI data.
 - Byte N is the last byte of the ACI data.
 - The application processor sets the **REQN** pin high to terminate the data transaction

Note: The maximum length of one packet is 32 bytes, including the length byte.

7.1.5 ACI event exchange

[Figure 10.](#) shows the signaling in an ACI event exchange from nRF8001 to the application processor.

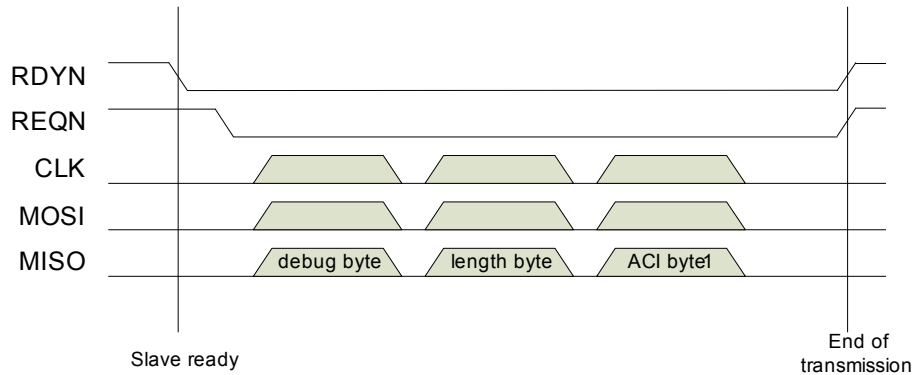


Figure 10. Receiving an ACI event from nRF8001

The application processor receives the ACI event by performing the following procedure:

1. nRF8001 sets the **RDYN** pin to ground.
2. The application processor sets the **REQN** pin to ground and starts sending the data from the **MISO** pin.
 - Byte 1 (debug byte) from nRF8001 is an internal debug byte and the application processor discards it.
 - Byte 2 (length byte) from nRF8001 defines the length of the message.
 - Byte 3 (ACI byte1) is the first byte of the ACI data.
 - Byte N is the last byte of the ACI data.
3. The application processor sets the **REQN** pin high to close the event.

The maximum length of a packet is 31 bytes, including the length byte.

7.1.6 ACI full-duplex transaction

nRF8001 is capable of receiving an ACI command simultaneously as it sends an ACI event to the application processor.

The application processor shall always read the length byte from nRF8001 and check if the length is greater than 0. If the length is greater than 0 the data on the MISO line shall be read as described in [section 7.1.5.](#)

An ACI event received from the nRF8001 processor is never a reply to a command being transmitted. For a given command the corresponding event will always be received in a subsequent ACI transaction.

7.1.7 SPI timing

The signaling and timing of each byte transaction for the nRF8001 SPI interface are shown in [Figure 11.](#) and [Figure 12. on page 24.](#) Critical timing parameters are listed in [Table 4. on page 24](#)

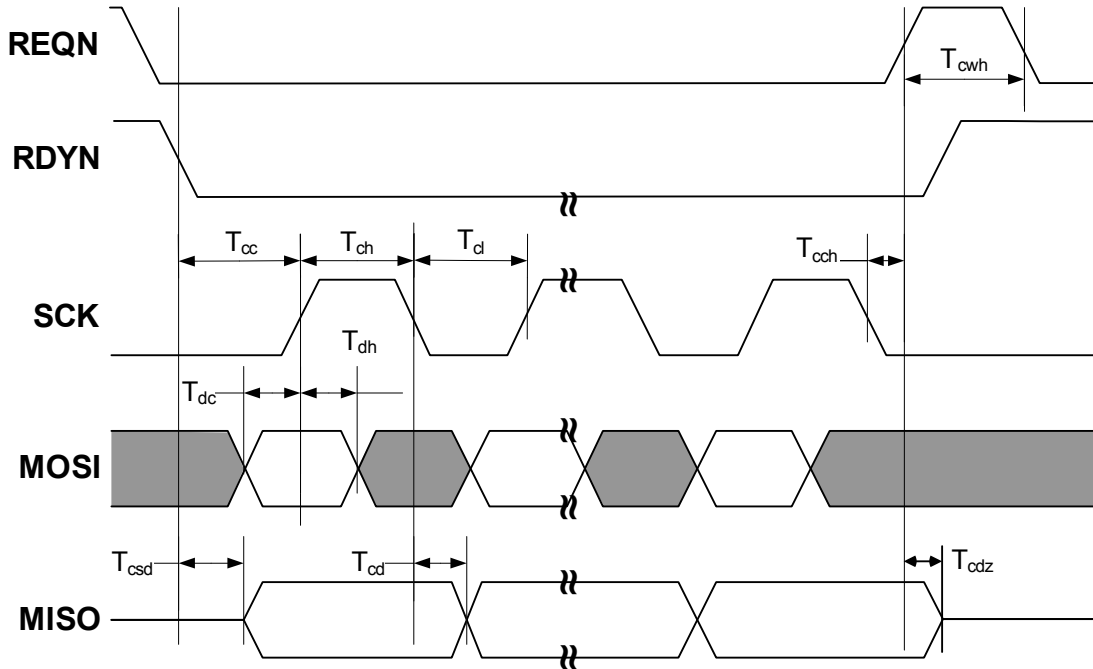


Figure 11. Application processor initiated packet SPI timing

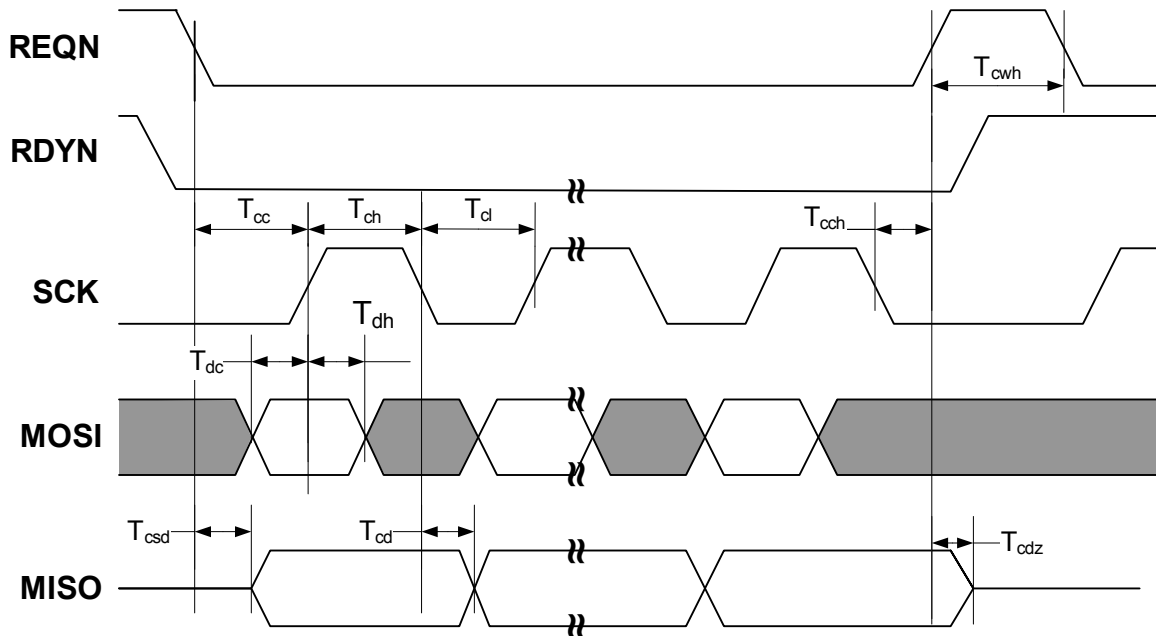


Figure 12. nRF8001 initiated packet SPI timing

| Symbol | Description | Min | Max | Unit |
|------------|--|-----|-------|------|
| T_{dc} | Data to SCK setup | 15 | | ns |
| T_{dh} | SCK to Data Hold | 5 | | ns |
| T_{csd} | REQN/RDYN to Data Valid | | 100 | ns |
| T_{cd} | SCK to Data Valid | 12 | 100 | ns |
| T_{cl} | SCK Low time | 40 | | ns |
| T_{ch} | SCK High time | 40 | 20000 | ns |
| F_{sck} | SCK frequency | 0 | 3 | MHz |
| T_r, T_f | REQN, SCK and MOSI rise time/fall time | | 15 | ns |
| T_{cc} | REQN/RDYN to SCK setup | 20 | | ns |
| T_{cch} | SCK to REQN hold | 10 | | ns |
| T_{cwh} | REQN inactive time | 250 | | ns |
| T_{cdz} | REQN to output high-Z | | 100 | ns |

Table 4. SPI timing parameters

Note: $C_{LOAD}=25$ pF, input transition to REQN, SCK and MOSI is in the range 5-15 ns. Rise and fall times are defined as the time when the signal is between 10-90 % of VDD.

7.2 Active signal

The active signal is an information signal provided by nRF8001. It indicates that the nRF8001 radio is active. Its polarity can be configured active high or active low. When the active signal is asserted, nRF8001 will drain peak currents as described in [chapter 13 on page 35](#). To maintain minimum peak current load on the battery, the active signal may be used to limit activity in the application processor during the connection event. The active signal may be configured to assert up to 20 ms before the radio in nRF8001 is switched on, see [Figure 13](#). Jitter on this signal is ± 312.5 us and the signal may occur early by up to 0.1% of the interval length, as a result of 32 KHz oscillator drift.

Note: When advertising, the random delay of 0 to 10 ms for advertising events (*Bluetooth Specification*, v. 4.0, Volume 6, Part B, Section 4.4.2.2 Advertising Interval) will shift the active signal in time in this range from event to event as the radio activity is shifted by this delay.

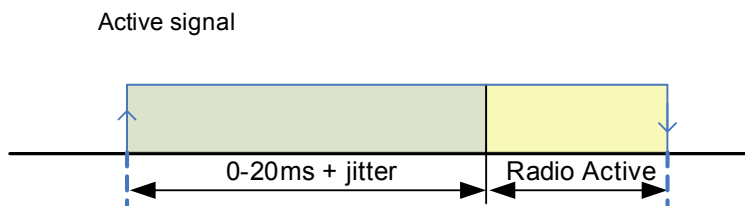


Figure 13. Active signal

Active signal is unavailable for advertising or connection intervals less than 30 ms regardless of prior configuration. If the active signal is enabled and the connection interval or advertising interval is below 30 ms, the active signal will automatically be disabled. If a subsequent connection update increases the interval ≥ 30 ms, the active signal will automatically be re-enabled without a need for reconfiguration.

7.3 Direct Test Mode interface

The Direct Test Mode (DTM) enables testing of the RF parameters of a *Bluetooth* low energy radio design. All *Bluetooth* low energy end products must include access to the DTM UART interface for end-product qualification testing of the RF transceiver layer.

The DTM has two modes of operation; transmit and receive test mode. In transmit test mode, nRF8001 generates a predefined set of test packets. In receive test mode, nRF8001 counts the number of test packets received from a dedicated RF transceiver tester.

The nRFGo Studio enables RF transceiver testing using the DTM. For more information, visit www.nordicsemi.com.

The *Bluetooth* low energy Direct Test Mode implemented in nRF8001 is described in the *Bluetooth Specification*, Ver. 4.0, Vol. 6, Part F.

7.3.1 Test interface characteristics

Direct test mode UART interface features:

- 2-wire UART interface (TXD/RXD)
- Baud rate: 19200
- Number of data bits: 8
- No parity
- 1 stop bit
- No flow control (meaning no RTS/CTS)

7.3.2 Functional description

The DTM is activated using the ACI. When active, the nRF8001 *Bluetooth* low energy radio is controlled by 2-byte commands on the 2-wire UART interface (pins TXD and RXD) or alternatively over the Application Controller Interface. Please See the *Bluetooth* Specification, Ver 4.0, Vol 6, part F, for command word format and options.

When the DTM is active, the nRF8001 stack features are disabled. To exit test mode and return to normal operation, the ACI command Test(ExitTestMode) or a device reset can be used.

8 nRF8001 configuration

nRF8001's hardware and protocol parameters are configured through the nRFgo Studio (**nRF8001 Configuration** menu option), see Part B, [Section 22.3 on page 68](#). These parameters may be written to non-volatile memory and are permanently stored through all power modes, see [chapter 9 on page 29](#). Once programmed, the parameters set the circuit to a default state on power up or reset. The available configuration parameters that may be set are listed in [Table 5](#). These parameters are available from the Hardware Settings and the GAP Settings tabs in the nRFgo Studio.

| Hardware settings | Description | Default |
|-----------------------|---|---|
| 32 KHz clock source | 32.768 kHz reference source: <ul style="list-style-type: none"> Internal RC oscillator External crystal External digital clock External analog source | Internal RC |
| 32 KHz clock accuracy | 32.768 kHz accuracy if using external source: <ul style="list-style-type: none"> 251 ppm to 500 ppm 151 ppm to 250 ppm 101 ppm to 150 ppm 76 ppm to 100 ppm 51 ppm to 75 ppm 31 ppm to 50 ppm 21 ppm to 30 ppm 0 ppm to 20 ppm If the default internal RC is used, this is set to 151 to 250 ppm. | 151 to 250 ppm |
| 16 MHz clock source | Sets the input reference source and the start time of the internal 16 MHz reference clock: <ul style="list-style-type: none"> Digital source (500 us) Crystal source (1.5 ms) | Crystal source |
| Initial TX power | Output power setting of PA: <ul style="list-style-type: none"> -18 dBm -12 dBm -6 dBm 0 dBm <p>Note: This parameter may also be changed in Run mode.</p> | 0 dBm |
| DC/DC converter | Enables the DC/DC converter | Disabled |
| Active signal | Set active signal timing requirements: <ul style="list-style-type: none"> Enable/Disable 0 to 20 ms before event start <ul style="list-style-type: none"> Resolution = 312.5 us Polarity <ul style="list-style-type: none"> 0 positive 1 negative | Disable |
| Timing parameters | Preferred slave connection parameters for L2CAP connection update command: <ul style="list-style-type: none"> Set maximum connection interval Set minimum connection interval Set slave latency Set connection supervision timeout | Min = User defined Max = User defined Latency = 0 Timeout = User defined |

| Hardware settings | Description | Default |
|-----------------------------|--|---------|
| Minimum encryption key size | Minimum size of encryption key length acceptable for the application. Range is between 7 and 16 bytes. | 7 |
| Maximum encryption key size | Maximum size of encryption key length acceptable for the application. Range is between 7 and 16 bytes. | 16 |

Table 5. Configurable parameters set through nRFgo Studio

9 Data storage and memory retention

Data stored in nRF8001 is stored in volatile or non-volatile memory, depending on the type of data. In this document, data is differentiated into two categories; static and dynamic data.

Static data:

nRF8001 can be configured through the ACI to hold hardware and protocol parameters, see [Part B, section 22.3 on page 68](#). Setup data can be written to non-volatile memory for permanent storage or to volatile memory during application development. Once programmed in non-volatile memory, the parameters set the circuit to the defined default state on power up or reset.

Dynamic data:

During normal runtime operation, your nRF8001 application will contain the Attributes and acquire information about peer devices and the services they offer. Your application may also establish a bonded relationship with a peer device. The information your application acquires as a result of normal runtime operation, is stored in nRF8001 volatile memory as dynamic data.

9.1 Permanent Storage

nRF8001 includes one time programmable Non-Volatile Memory (NVM). The hardware device setup and pipe setup as defined in [Part B, section 22.3 on page 68](#) are programmed into the NVM memory for permanent storage. Once information has been stored in NVM, it cannot be changed. The nRFgo Studio configuration tool offers creates two setup file alternatives. One file will store the setup in NVM, the other will store the setup in volatile memory. For application development, setup storage in volatile memory will allow adjustments without discarding the device.

Note: Setup storage in volatile memory will be lost when the device is reset or power cycled.

9.2 Volatile Storage

Dynamic data is stored in RAM and will be lost if nRF8001 is power cycled or reset. Typical data stored in RAM includes profile client information, bonding addresses and keys.

Dynamic data may be read out of nRF8001 and stored in the application processor. Upon power cycling and attempting to re-enter a connection with a previously established relationship to a peer device, the data is written back into nRF8001 from the external application processor. This procedure is defined in [Part B, section 22.4.6 on page 74](#).


10 Absolute maximum ratings

Maximum ratings are the extreme limits to which nRF8001 can be exposed without permanently damaging it. Exposure to absolute maximum ratings for prolonged periods of time may affect nRF8001's reliability. [Table 6](#) specifies the absolute maximum ratings for nRF8001.

| Parameter | Minimum | Maximum | Unit |
|------------------------|---------|----------|------|
| Supply voltages | | | |
| VDD | -0.3 | +3.6 | V |
| VSS | | 0 | V |
| I/O pin voltage | | | |
| V _{IO} | -0.3 | VDD+0.3V | V |
| Temperatures | | | |
| Storage temperature | -40 | +125 | °C |

Table 6. Absolute maximum ratings

Attention!

| | |
|---|---|
| <p>Observe precaution for handling Electrostatic Sensitive Device.</p> <p>HBM (Human Body Model): Class 2</p> |  |
|---|---|

11 Operating conditions

The operating conditions are the physical parameters that nRF8001 can operate within. The operating conditions for nRF8001 are defined in [Table 7](#).

| Symbol | Parameter (condition) | Notes | Min | Nominal | Max | Units |
|--------------------|---|-------|-----|---------|------|-----------|
| VDD | Supply voltage | | 1.9 | 3.0 | 3.6 | V |
| VDD _{DC} | Supply voltage with DC/DC converter enabled | | 2.1 | 3.0 | 3.6 | V |
| t _{R_VDD} | Supply rise time (0V to 1.9V) | | 1μs | | 50ms | μs and ms |
| T _A | Operating temperature | | -40 | | +85 | °C |

Table 7. Operating conditions

12 Electrical specifications

This section contains electrical specifications for signal levels, radio parameters and current consumption. The test levels referenced are defined in [Table 8](#).

| Test level | Description |
|------------|--|
| I | By design (simulation, calculation, specification limit) |
| II | Prototype verification @ EOC |
| III | Verified @ EOC (3 corner lots x 30 samples) |
| IV | 100% test @ NOC |

Table 8. Test level definitions

12.1 Digital I/O signal levels

The digital I/O signal levels are defined [Table 9](#). The operating conditions are: VDD = 3.0V, T_A = -40°C to +85°C (unless otherwise noted).

| Symbol | Parameter (condition) | Test level | Min | Nom | Max | Unit |
|-----------------|--|------------|---------|-----|---------|------|
| V _{IH} | Input high voltage | I | 0.7×VDD | | VDD | V |
| V _{IL} | Input low voltage | I | VSS | | 0.3×VDD | V |
| V _{OH} | Output high voltage (I _{OH} = 0.5 mA) | II | VDD-0.3 | | | V |
| V _{OL} | Output low voltage (I _{OL} = 0.5 mA) | II | | | 0.3 | V |

Table 9. Digital inputs/outputs

12.2 Radio characteristics

nRF8001 electrical characterization is defined in [Table 12](#). The operating conditions are: VDD = 3.0V, T_A = -40°C to +85°C (unless otherwise noted).

| Symbol | Parameter (condition) | Test level | Notes | Min | Nom | Max | Unit |
|--------------------|---------------------------|------------|-------|------|-----|------|------|
| f _{OP} | Frequency operating range | I | | 2402 | | 2480 | MHz |
| f _{XTAL} | Crystal frequency | I | | | 16 | | MHz |
| Δf | Frequency deviation | I | | | 250 | | kHz |
| R _{GFSK} | On air data rate | I | | | 1 | | Mbps |
| PLL _{RES} | RF channel spacing | I | | | 2 | | MHz |

Table 10. Radio general electrical characteristics

| Symbol | Parameter (condition) | Test level | Notes | Min | Nom | Max | Unit |
|--------------------|----------------------------|------------|-------|-----|-----|-----|------|
| P _{RF} | Maximum output power | I | 1 | | 0 | 4 | dBm |
| P ₋₆ | Output power setting | | | | -6 | | dBm |
| P ₋₁₂ | Output power setting | | | | -12 | | dBm |
| P ₋₁₈ | Output power setting | | | | -18 | | dBm |
| BW _{20dB} | 20dB signal bandwidth | I | | | 670 | | kHz |
| P _{RF1.1} | 1st adjacent channel power | I | | | TBD | | |
| P _{RF2.1} | 2nd adjacent channel power | I | | | TBD | | |

1. Antenna load impedance = $15\Omega + j88$

Table 11. Radio transmitter electrical characteristics

| Symbol | Parameter (condition) | Test level | Notes | Min | Nom | Max | Unit |
|----------------------|---|------------|-------|-----|-----|-----|------|
| P _{RX max} | Maximum input signal strength at PER $\leq 30.8\%$ | I | | | 0 | | dBm |
| P _{sens IT} | Receiver sensitivity: ideal transmitter | I | | | -87 | | dBm |
| P _{sens DT} | Receiver sensitivity: dirty transmitter | I | 1 | | -86 | | dBm |
| P _{sens DC} | Receiver Sensitivity DC/DC Converter Enabled: dirty transmitter | I | | | -85 | | dBm |
| C/I _{CO} | Co-channel rejection | I | | | -13 | | dB |
| C/I _{1st} | Adjacent channel selectivity: 1 MHz offset | I | 1. | | -7 | | dB |
| C/I _{2nd} | Adjacent channel selectivity: 2 MHz offset | I | 1. | | 23 | | dB |
| C/I _{3+n} | Adjacent channel selectivity: (3+n) MHz offset [n=0,1,2...] | I | 1. | | 51 | | dB |
| C/I _{Image} | Image frequency rejection | I | 1. | | 26 | | dB |
| P _{IM} | IMD performance (P _{in} =64 dBm) | I | 1. | | -38 | | dBm |

1. As defined in Bluetooth V4.0 Volume 6: Core System Package [Low Energy Controller Volume].

Table 12. Radio receiver electrical characteristics

12.3 Analog feature characteristics

| Symbol | Parameter (condition) | Test level | Notes | Min | Nom | Max | Unit |
|--------------------|-----------------------------|------------|-------|-------|-----|------|------|
| T _{range} | Temperature Sensor Range | I | | -40 | | 85 | C |
| T _{acc} | Temperature Sensor Accuracy | I | | -2 | | 2 | C |
| B _{range} | Battery Monitor Range | I | | 1.9 | | 3.6 | V |
| B _{acc} | Battery Monitor Accuracy | I | | -0.05 | | 0.05 | V |

Table 13. Analog feature electrical characteristics

12.4 Current consumption parameters

The nRF8001 static current consumption is defined in [Table 14.](#) and [Table 15.](#) The dynamic current consumption is defined in [Table 15. on page 34.](#)

The operating conditions are: VDD = 3.0V, T_A = -40°C to +85°C. The numbers in the **Reference to figures 14 and 15** (see [Figure 14. on page 35](#) and [Figure 15. on page 36](#)) column in [Table 14.](#) and [Table 15.](#) refer to the current drain profile for a connection or advertising event as defined above.

| Symbol | Parameter (condition) | Test level | Reference to figures 14 and 15 | Min | Nom | Max | Unit |
|-----------------------|--|------------|--------------------------------|-----|------|-----|------|
| I _{RX} | Peak current, receiver active | I | 2 | | 14.5 | | mA |
| I _{TX} | Peak current, transmitter active | I | 3 | | 13 | | mA |
| I _{TFS} | Peak current when switching between receive and transmit | I | 3 | | 7 | | mA |
| I _{MCU_HOST} | Peak current for host processing | I | 6 | | 5 | | mA |
| I _{MCU_LL} | Peak current for LL processing | I | 1 & 5 | | 3.5 | | mA |
| I _{Standby} | Standby current, | I | 1 | | 1.6 | | mA |
| I _{idle} | Current drain between connection/ advertising events ACI = run mode, 32 kHz Osc active | I | | | 2 | | µA |
| I _{sleep} | Current drain, connection-less state ACI = sleep mode | I | | | 0.5 | | µA |

Table 14. Current consumption for static values when DC/DC not active

| Symbol | Parameter (condition) | Test level | Reference to figures 14 and 15 | Min | Nom | Max | Unit |
|--------------------------|--|------------|--------------------------------|-----|------|-----|------|
| I _{RX_DC} | Peak current, receiver active | I | 2 | | 12.5 | | mA |
| I _{TX_DC} | Peak current, transmitter active | | 3 | | 11 | | mA |
| I _{TFS_DC} | Peak current when switching between receive and transmit | | 3 | | 6 | | mA |
| I _{MCU_HOST_DC} | Peak current for host processing | I | 6 | | 4 | | mA |
| I _{MCU_LL_DC} | Peak current for LL processing | I | 1 & 5 | | 2.5 | | mA |
| I _{Standby_DC} | Standby current | I | 1 | | 1.1 | | mA |
| I _{idle} | Current drain between connection/ advertising events ACI = run mode, 32 kHz Osc active | I | | | 2 | | µA |
| I _{sleep} | Current drain, connection-less state ACI = sleep mode, with memory retention | I | | | 0.5 | | µA |

Table 15. Current consumption for static values when DC/DC converter active

13 Dynamic current consumption

To predict battery lifetime, it is important to understand how the hardware and *Bluetooth* low energy protocol parameters influence the overall power consumption.

The connection and advertising events consist of a sequence of radio transmissions, each of which has individual current drain. The average power consumption of the event is calculated by integrating the current drain over the duration of the event.

Peak current consumption data is found in [section 12.4 on page 34](#).

13.1 Current consumption - connection

[Figure 14](#), illustrates the principle of current drain over time for a typical *Bluetooth* low energy device that is connected. The maximum peak-current drain occurs when the receiver is active ($I_{\text{peak_RX}}$).

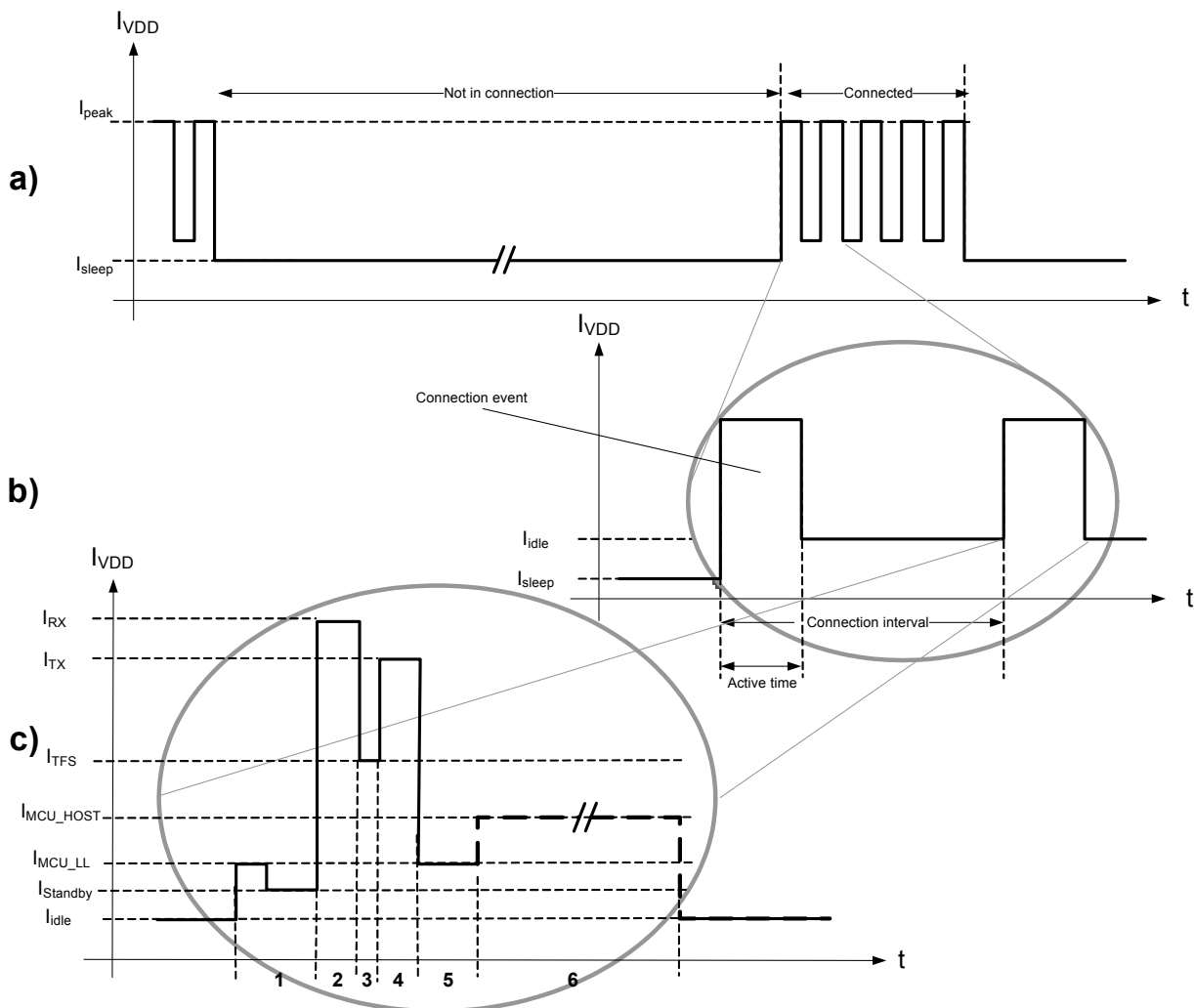


Figure 14. Current consumption over time for a typical nRF8001 connection event

Segment a) of [Figure 14](#), shows a typical scenario. A connection is defined as a physical radio connection between two *Bluetooth* low energy devices. This may consist of several connection events at a given

connection interval. The communication time is defined as the time that the *Bluetooth* low energy devices maintain the physical radio connection. It consists of one or more connection events separated in time by the connection interval. I_{sleep} is defined as the current consumption in between communication intervals.

Segment b) of [Figure 14. on page 35](#) illustrates the periodicity of the connection interval. The average current drain of each connection event depends on the link parameter settings and ACI activity. I_{idle} is defined as the current drain between connection events.

Segment c) of [Figure 14. on page 35](#) illustrates a typical current drain profile for a connection event. Each connection event consists of the following states and operations (the numbers below correspond to the numbers displayed in segment c) of [Figure 14. on page 35](#)):

1. Radio pre-processing period
2. Active radio receive time
3. Radio Inter Frame Space (T_{IFS})
4. Active transmit time
5. Link layer post processing period
6. Data post processing period, enabled only if data has been received

[Figure 15.](#) shows the average current consumption (with and without DC/DC converter enabled) as a function of the length of the connection interval.

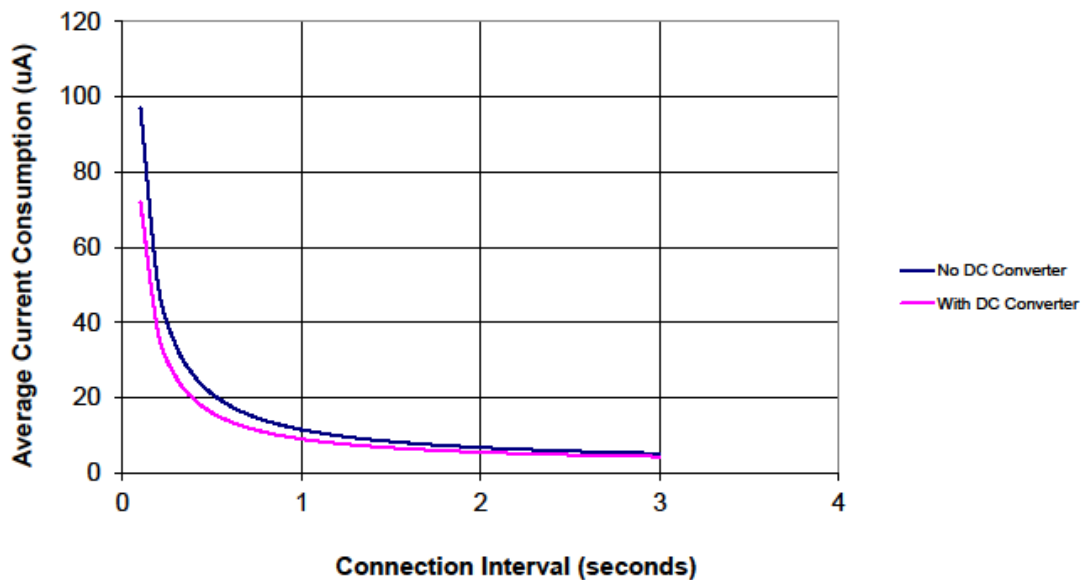


Figure 15. Graph showing average current consumption as a function of connection interval length

13.2 Current consumption - advertising

Figure 16 illustrates the principle of current drain over time for a typical *Bluetooth* low energy device that is advertising.

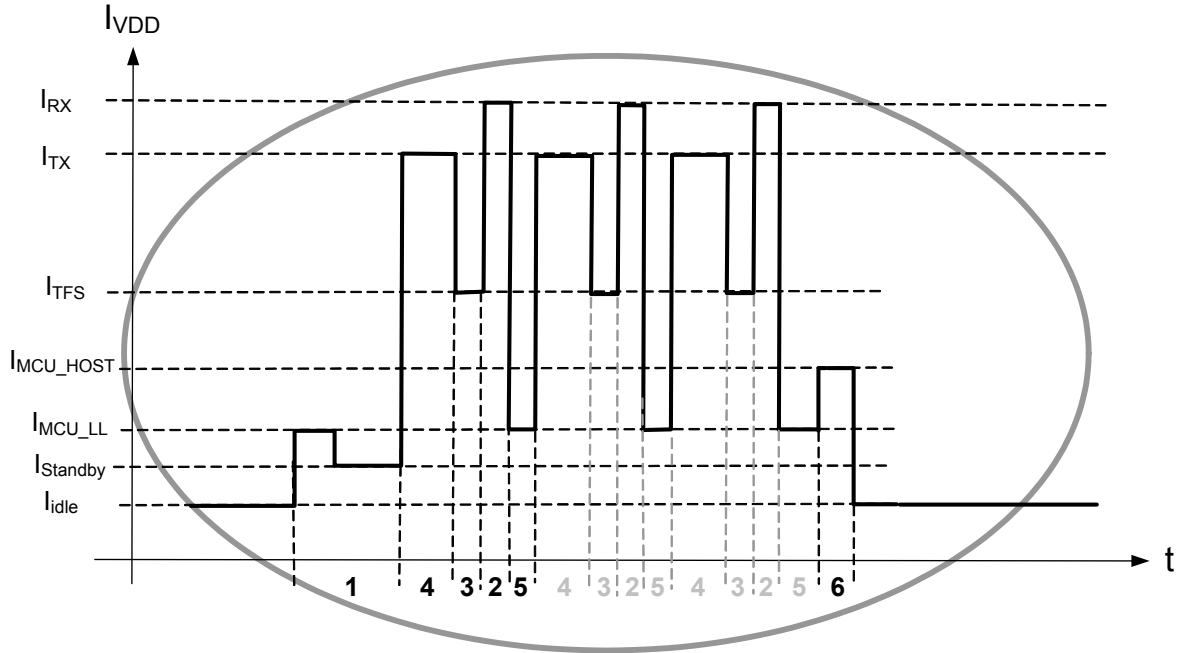


Figure 16. Current consumption over time for a typical nRF8001 advertising event

Each advertising event consists of the following states and operations (the numbers below correspond to the numbers displayed in [Figure 16. on page 37](#)):

1. Radio pre-processing period
2. Active radio receive time
3. Radio Inter Frame Space (T_{IFS})
4. Active transmit time
5. Link layer post processing period
6. Data post processing period, enabled only if data has been received

[Figure 17.](#) shows the average current consumption (with and without DC/DC converter enabled) as a function of the length of the advertisement interval.

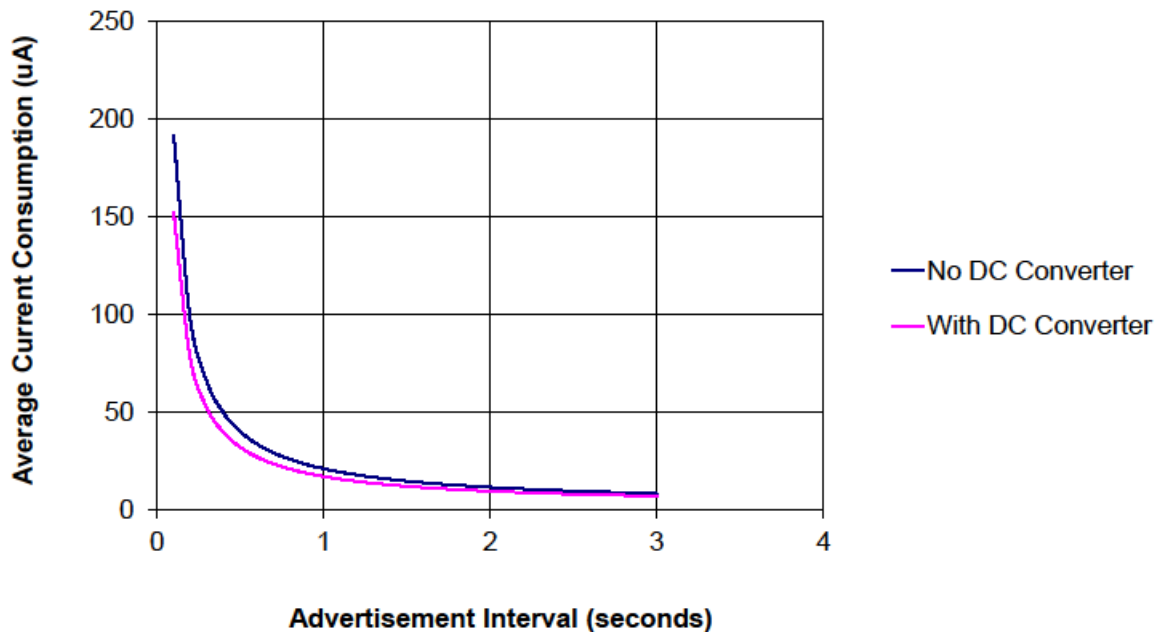


Figure 17. Graph showing average current consumption as a function of advertisement interval length

13.3 Current consumption calculation examples

You can calculate the average current consumption using the nRFgo Studio Current Consumption Calculator. For more information on nRFgo Studio, visit www.nordicsemi.com.

A set of typical profile scenario examples are presented below for different use cases based on typical profile scenarios. The current consumption values are calculated by the nRFgo Studio Current Consumption Calculator.

13.3.1 Estimated lifetime for proximity profile

The proximity profile typically has the following connection parameters:

- A 100% connection model, that is, the proximity tag is nearly always connected.
- 1 second connection interval.
- Zero data for 99% of the time, data is only sent if a condition is met, 4 bytes.
- Peripheral device is a server and initiates alerts upon receiving write commands.
- Pipe used: RX pipe.
- No encryption used.
- Total system sleep clock accuracy +/- 300ppm.
- 220 mAh coin cell battery.

DC/DC converter disabled:

Average current consumption = 16.5 μ A

Calculated battery lifetime = 18 months

DC/DC converter enabled:

Average current consumption = 13.5 μ A

Calculated battery lifetime = 22 months

13.3.2 Estimated lifetime for heart rate profile

The heart rate profile typically has the following connection parameters:

- 1 hour per day connected.
- 1 second connection interval when connected.
- Data indicated every second, 4 bytes.
- Peripheral device is a server and indicates data.
- Pipe used: TX-Ack pipe.
- No encryption used.
- Total system sleep clock accuracy +/- 300 ppm.
- 220 mAh coin cell battery.

DC/DC converter disabled:

Average current consumption = 3.3 μ A

Calculated battery lifetime = 7.5 years

DC/DC converter enabled:

Average current consumption = 2.7 μ A

Calculated battery lifetime = 9 years

13.4 Recommendations for low power operation

Obtaining low power operation and long battery lifetime is a compromise between cost and performance. The following recommendations are given to obtain suitable battery lifetimes:

- Use a reference source derived from an external high accuracy 32 kHz crystal source instead of the internal 32 kHz RC oscillator. Improving the total timing accuracy within the system can significantly reduce power consumption. This is a direct trade off of the cost of the 32.768 KHz crystal.
- Set Preferred Peripheral Connection Parameters for the application. The peer device operating in the central role defines the connection parameters to use for that connection. However, the peripheral device may request the peer device to change these to battery favorable parameters. Connection Interval and Slave Latency are important parameters in achieving suitable battery lifetime. These parameters are set by the ACI when nRF8001 is in the Setup mode as defined in [Part B section 22.3 on page 68](#)
- Using the DC/DC converter reduces the active peak currents as defined in [Table 15. on page 34](#) by approximately 20%. Additional external components are required when the DC/DC converter is enabled.

14 External component requirements and recommendations

The tables in this chapter specify the required crystal parameters that are required for nRF8001 to function and meet the Bluetooth low energy specification. [Table 16](#) specifies the requirements for the 16 MHz crystal. [Table 17. on page 42](#) specifies the requirements for the 32.768 kHz crystal.

14.1 16 MHz crystal specification requirements

To ensure a functional *Bluetooth* low energy radio link the frequency accuracy must be ± 40 ppm or better.

To maintain correct operation, the tolerance of the crystal, drift over temperature, aging and frequency pulling due to incorrect load capacitance must be taken into account.

The crystal load capacitance, shunt capacitance, Equivalent Series Resistance (ESR) and drive level must comply with the specifications given in [Table 16](#). It is recommended to use a crystal with lower than maximum ESR if the load capacitance and/or shunt capacitance is high. This will result in a faster start-up time and lower the current consumption.

The start-up time is typically less than 1ms for a crystal with 12 pF load capacitance, 3 pF shunt capacitance and an ESR of 50 Ω .

Note: The use of a crystal is optional, if external clock sources exist they can be used.

| Symbol | Parameter (condition) | Notes | Min | Nom | Max | Units |
|------------|---------------------------------|-------|-----|-----|----------|----------|
| F_{XO16} | 16 MHz crystal frequency | | | 16 | | MHz |
| Δf | Tolerance | 1 | | | ± 40 | ppm |
| C_L | Load capacitance | | 8 | 12 | 16 | pF |
| C_0 | Equivalent parallel capacitance | | | 3 | 7 | pF |
| L_S | Equivalent series inductance | 2 | | 30 | 50 | mH |
| ESR | Equivalent series resistance | | | 50 | 100 | Ω |
| P_D | Drive level | | | | 100 | μW |

1. Frequency offset at 25 °C, temperature drift, aging and crystal loading

2. Startup time from power down is dependant on the L_S parameter

Table 16. 16 MHz crystal specifications

14.2 32.768 kHz crystal specification requirements

The crystal load capacitance, shunt capacitance, Equivalent Series Resistance (ESR) and drive level must comply with the specifications listed in [Table 17. on page 42](#). It is recommended to use a crystal with an ESR lower than maximum if the load capacitance and/or shunt capacitance is high. This will provide a faster start-up time and lower the current consumption.

The start-up time is typically < 0.5 s for a crystal with 9 pF load capacitance, 1 pF shunt capacitance and an ESR of 50 kΩ.

| Symbol | Parameter (condition) | Notes | Min | Nom | Max | Units |
|-------------------|---------------------------------|-------|-----|--------|------|-------|
| F _{XO32} | 32.768 kHz crystal frequency | | | 32.768 | | kHz |
| Δf | Tolerance | 1 | 0 | | 500 | ppm |
| C _L | Load capacitance | | | 9 | 12.5 | pF |
| C ₀ | Equivalent parallel capacitance | | | 1 | 2 | pF |
| ESR | Equivalent series resistance | | | 50 | 80 | kΩ |
| P _D | Drive level | | | | 1 | μW |

1. Frequency accuracy including tolerance at 25 °C, temperature drift, aging and crystal loading

Table 17. 32.768 kHz crystal specification

14.3 Antenna Matching and Balun

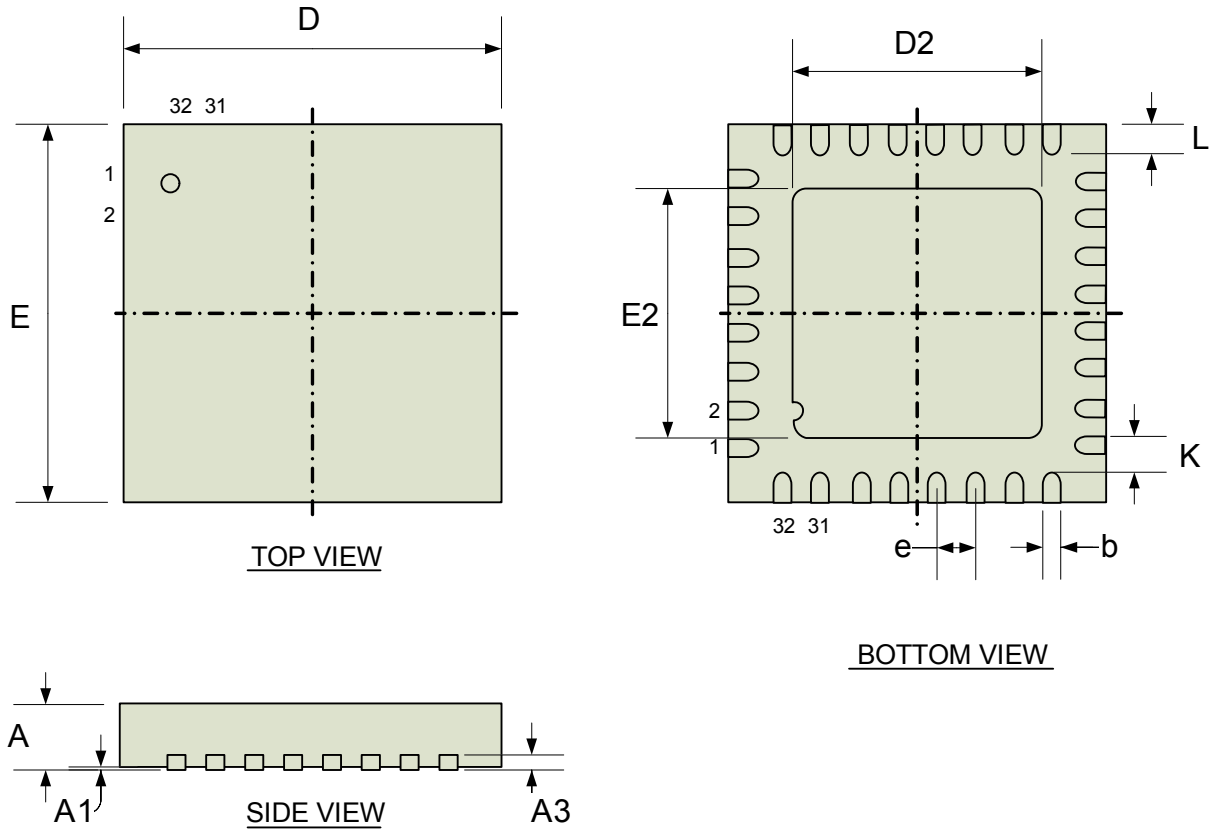
The **ANT1** and **ANT2** pins provide a balanced RF connection to the antenna. The pins must have a DC to **VDD_PA**, either through an RF choke or through the center point in a balanced dipole antenna. A load impedance at **ANT1** and **ANT2** of 15 Ω + j88 Ω is recommended for maximum output. A load impedance of 50 Ω can be obtained by fitting a simple matching network between the load and the **ANT1** and **ANT2** pins. A recommended matching network for 50 Ω load impedance is described in [chapter 17 on page 45](#).

14.4 DC/DC Converter requirements

The DC/DC converter requires three external components, two inductors and one decoupling capacitor, see [Figure 18. on page 45](#). The inductors should feature low, serial resistance (<1.0 Ω) and must have a maximum DC current rating of at least 50 mA. The capacitors should have low serial resistance.

15 Mechanical specifications

nRF8001 is packaged in a QFN32 5×5×0.85 mm, 0.5 mm pitch.



| Package | A | A1 | A3 | b | D, E | D2, E2 | e | K | L | |
|---------|------|------|------|------|------|--------|-----|------|------|-----|
| QFN32 | 0.80 | 0.00 | | 0.18 | 4.9 | 3.50 | | 0.20 | 0.35 | Min |
| | 0.85 | 0.02 | 0.20 | 0.25 | 5.0 | 3.60 | 0.5 | | 0.40 | Nom |
| | 0.90 | 0.05 | | 0.30 | 5.1 | 3.70 | | | 0.45 | Max |

Table 18. QFN32 dimensions in mm

16 Ordering information

16.1 Package marking

| | | | | | |
|---|---|---|---|---|---|
| N | R | F | | B | X |
| 8 | 0 | 0 | 1 | | |
| Y | Y | W | W | L | L |

16.2 Abbreviations

| Abbreviation | Definition |
|--------------|---|
| 8001 | Product number |
| B | Build Code, that is, unique code for production sites, package type and test platform |
| X | "X" grade, that is, Engineering Samples (optional) |
| YY | Two-digit year number |
| WW | Two-digit week number |
| LL | Two-letter wafer-lot number code |

Table 19. Abbreviations

16.3 Product options

16.3.1 RF silicon

| Ordering code | Package | Container | MOQ ¹ |
|------------------|-------------------------------------|-----------|------------------|
| nRF8001-R2Q32-R | 5×5mm 32-pin QFN, lead free (green) | 13" Reel | 4000 |
| nRF8001-R2Q32-R7 | 5×5mm 32-pin QFN, lead free (green) | 7" Reel | 1500 |
| nRF8001-R2Q32-T | 5×5mm 32-pin QFN, lead free (green) | Tray | 490 |

1. Minimum Order Quantity

Table 20. nRF8001 RF silicon options

16.3.2 Development tools

| Type Number | Description |
|-------------|-------------------------|
| nRF8001-DK | nRF8001 Development Kit |
| nRF6700 | nRFgo Starter Kit |

Table 21. nRF8001 solution options

17 Reference circuitry

17.1 Schematic for nRF8001 with DC/DC converter enabled

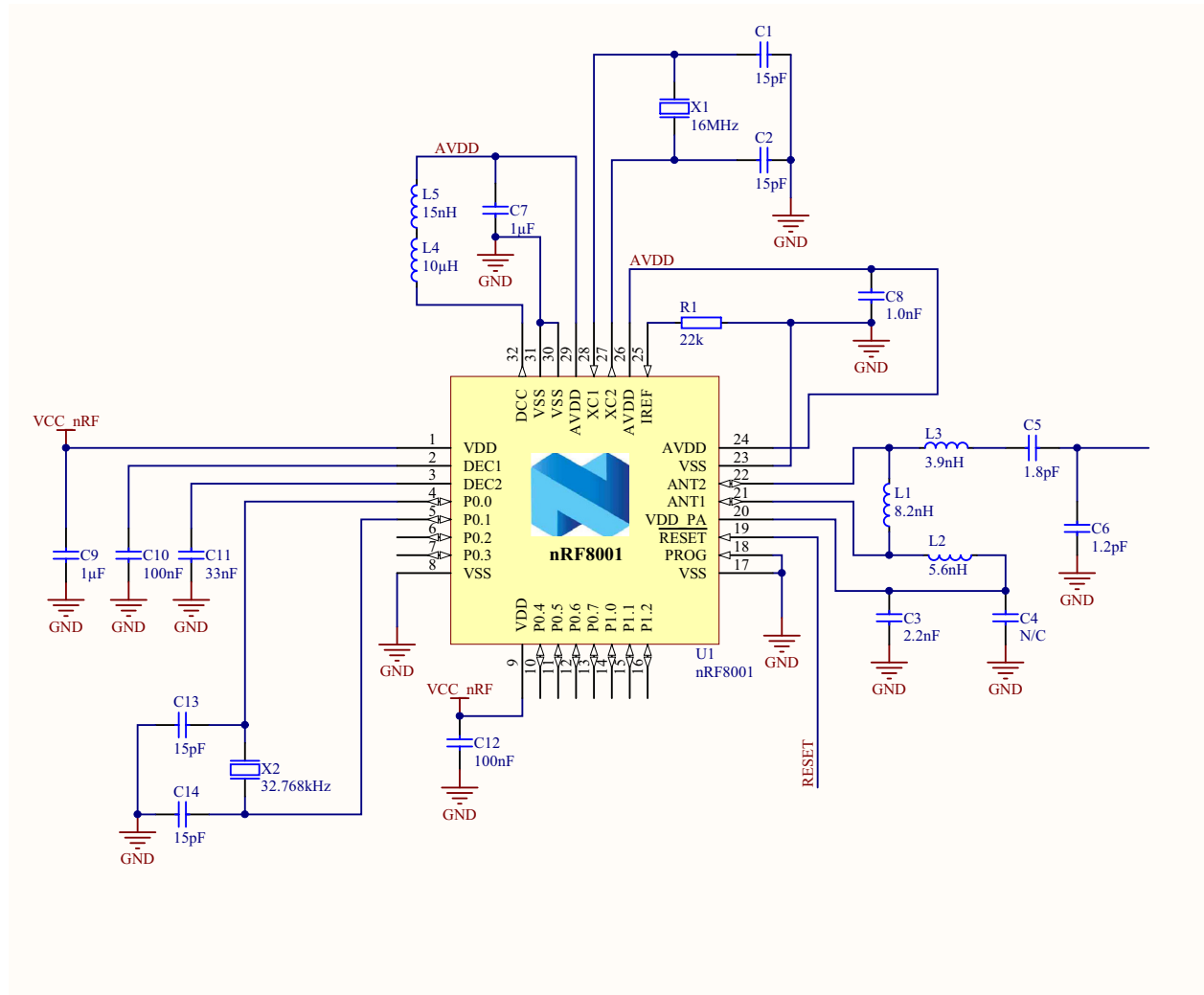
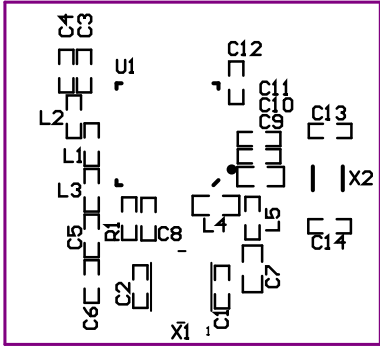
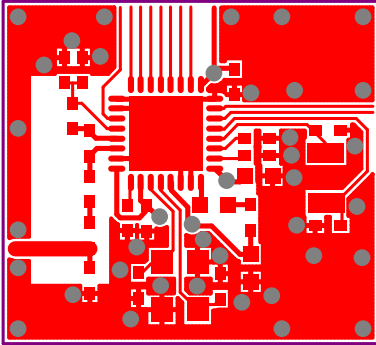
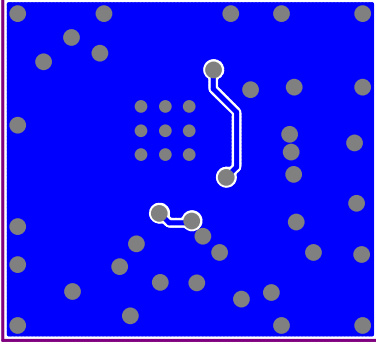


Figure 18. nRF8001 schematic, with DC/DC converter enabled

17.2 Layout

| | |
|--|---|
|  | <p>No components on bottom layer</p> |
| <p>Top silk screen</p> | |
|  |  |
| <p>Top view</p> | <p>Bottom view</p> |

17.3 Bill of Materials

| Designator | Value | Footprint | Comment |
|---------------|--------------|----------------|---|
| C1,C2,C13,C14 | 15 pF | 0402 | NP0 +/- 2% |
| C3 | 2.2 nF | 0402 | X7R +/- 10% |
| C4 | NA | 0402 | Not mounted |
| C5 | 1.8 pF | 0402 | NP0 +/- 0.1 pF |
| C6 | 1.2 pF | 0402 | NP0 +/- 0.1 pF |
| C7, C9 | 1.0 μF | 0603 | X7R +/- 10% |
| C8 | 1.0 nF | 0402 | X7R +/- 10% |
| C10, C12 | 100 nF | 0402 | X7R +/- 10% |
| C11 | 33 nF | 0402 | X7R +/- 10% |
| L1 | 8.2 nH | 0402 | High frequency chip inductor +/- 5% |
| L2 | 5.6 nH | 0402 | High frequency chip inductor +/- 5% |
| L3 | 3.9 nH | 0402 | High frequency chip inductor +/- 5% |
| L4 | 10 μF | 0603 | Chip inductor, I _{DC,min} = 50mA, +/-20% |
| L5 | 15 nH | 0402 | High frequency chip inductor +/- 10% |
| R1 | 22 k | 0402 | +/- 1% |
| U1 | nRF8001 | QFN32 | QFN32 5x5 mm package |
| X1 | 16 MHz | 3.2 × 2.5 mm | SMD-3225, 16 MHz, CL=9pF, +/-50 ppm |
| X2 | 32.768 kHz | 3.2 × 1.5 mm | SMD-3215, 32.768kHz, Cl=9pF, ±20 ppm |
| PCB substrate | FR4 laminate | 18.4 × 16.7 mm | 2 layer, thickness 1.6 mm |

Table 22. Bill of materials, with DC/DC converter enabled

17.4 Schematic for nRF8001 with DC/DC converter disabled

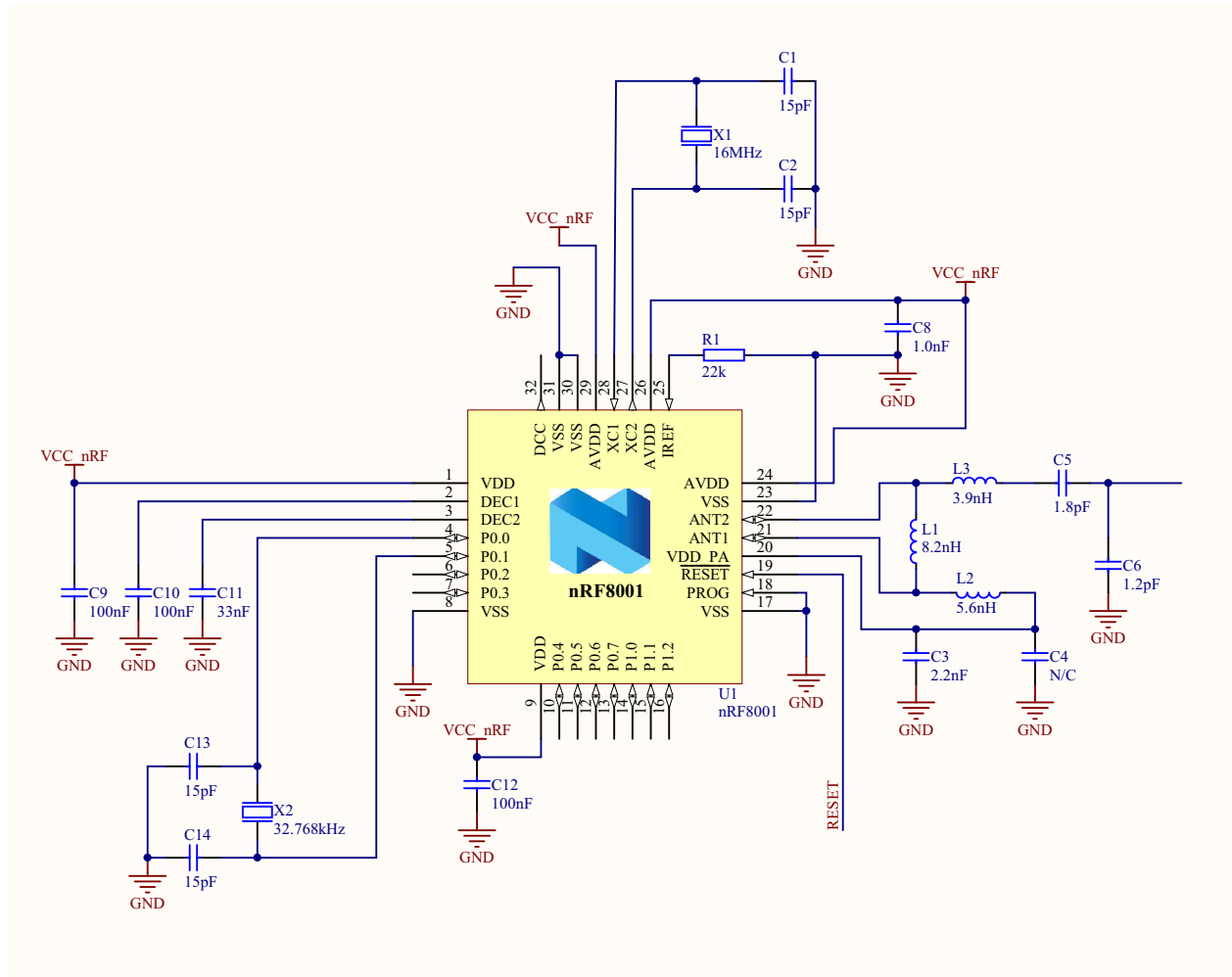
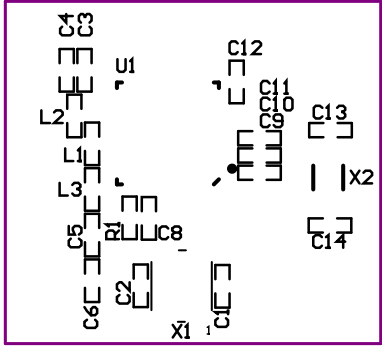
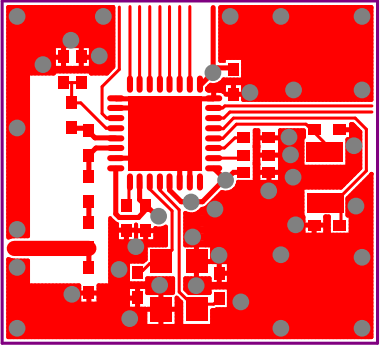
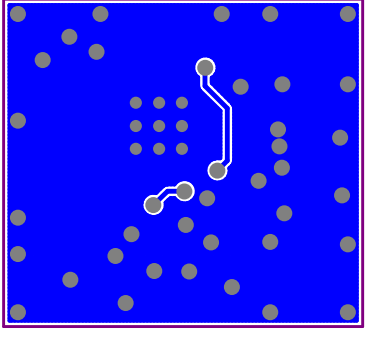


Figure 19. nRF8001 schematic, without DC/DC converter enabled

17.5 Layout

| | |
|--|---|
|  | <p>No components on bottom layer</p> |
| <p>Top silk screen</p> | |
|  |  |
| <p>Top view</p> | <p>Bottom view</p> |

17.6 Bill of Materials

| Designator | Value | Footprint | Comment |
|---------------|--------------|----------------|--------------------------------------|
| C1,C2,C13,C14 | 15 pF | 0402 | NP0 +/- 2% |
| C3 | 2.2 nF | 0402 | X7R +/- 10% |
| C4 | NA | 0402 | Not mounted |
| C5 | 1.8 pF | 0402 | NP0 ±0.1 pF |
| C6 | 1.2 pF | 0402 | NP0 ±0.1 pF |
| C8 | 1.0 nF | 0402 | X7R +/- 10% |
| C9, C10, C12 | 100 nF | 0402 | X7R +/- 10% |
| C11 | 33 nF | 0402 | X7R +/- 10% |
| L1 | 8.2 nH | 0402 | High frequency chip inductor +/- 5% |
| L2 | 5.6 nH | 0402 | High frequency chip inductor +/- 5% |
| L3 | 3.9 nH | 0402 | High frequency chip inductor +/- 5% |
| R1 | 22 k | 0402 | +/- 1% |
| U1 | nRF8001 | QFN32 | QFN32 5x5 mm package |
| X1 | 16 MHz | 3.2 × 2.5 mm | SMD-3225, 16 MHz, CL=9pF, +/-50 ppm |
| X2 | 32.768 kHz | 3.2 × 1.5 mm | SMD-3215, 32.768kHz, CI=9pF, ±20 ppm |
| PCB substrate | FR4 laminate | 18.4 × 16.7 mm | 2 layer, thickness 1.6 mm |

Table 23. Bill of materials, without DC/DC converter enabled

Part B: The nRF8001 Application Controller Interface (ACI)

The Application Controller Interface (ACI) is a bidirectional serial interface that enables generic application processors to set up and operate nRF8001. [Figure 20](#) illustrates how nRF8001 uses the ACI to logically connect to the application processor.

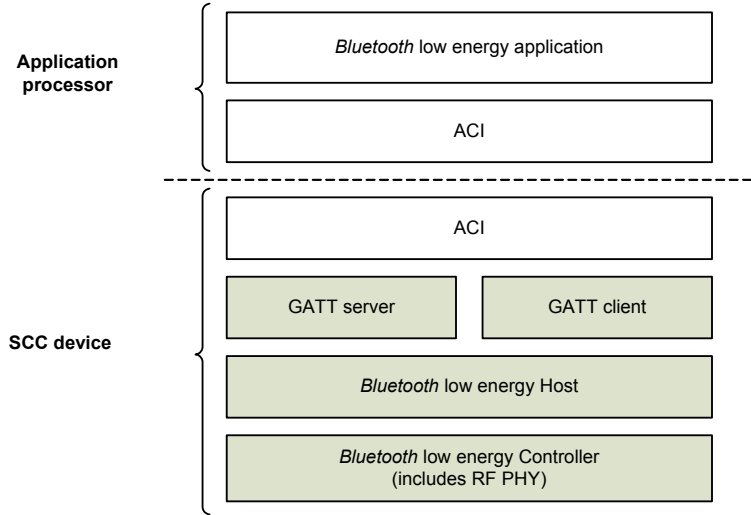


Figure 20. nRF8001 ACI connectivity

18 Operating principle

Figure 21. illustrates the operating principle of the ACI in a typical application.

ACI information traffic is bidirectional; control is exerted by the application processor and nRF8001 responds to ACI commands. nRF8001 may also independently send information to the application processor and all information between the two devices is structured in variable length packets.

Information packets sent from the application processor to nRF8001 are called commands. Commands are classified into two categories; system commands and data commands:

- System commands are commands used for nRF8001 configuration and for operation mode control.
- Data commands are commands that either aim to transfer, or receive, application data when nRF8001 is in a connected state with a peer device.

Information packets sent from nRF8001 to the application processor are called events. Events are classified into two categories; system events and data events.

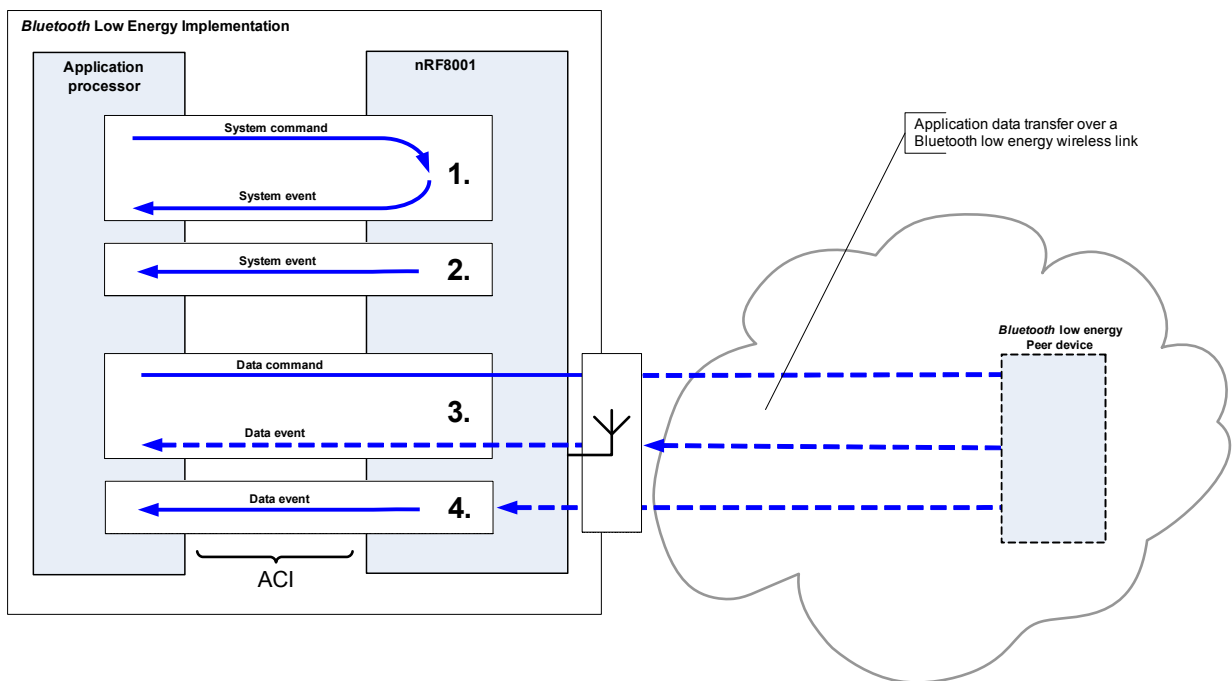


Figure 21. ACI operating principle

Information exchange can be divided into four typical scenarios as illustrated in [Figure 21. on page 50](#)

1. System command – System event
The application processor sends a system command and receives acknowledgment from nRF8001 in the form of an event.
2. System event
nRF8001 sends an event to the application processor triggered by a predefined condition.
3. Data command – Data event
The application processor sends a data command requesting application data transfer or reception. Data is returned in the form of a Data event if the transaction is successful.
4. Data event
nRF8001 sends an event packet to the application processor. This is triggered by a data packet transfer by the peer device or a predefined condition related to application data transfer.

18.1 Packet structure

ACI information traffic is organized in packets. Each packet consists of a 2 byte header field followed by a variable length packet payload.

The byte ordering follows the Little Endian format (Least significant byte transmitted first). In this part of the document capitalized letters as in MSB (Most Significant Byte) indicate bytes.

Text data is transmitted leftmost character first.

For information on bit ordering and description of the ACI physical transport, see [Part A, section 7.1 on page 19](#)

[Figure 22.](#) illustrates the ACI packet.

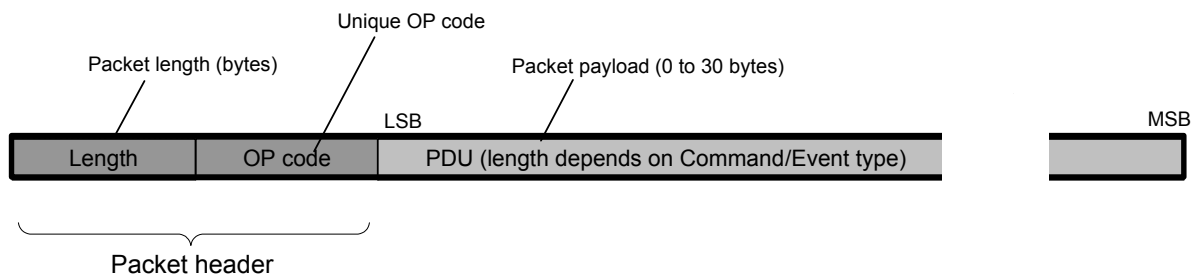


Figure 22. ACI packet structure

The packet header consists of two bytes. The first byte represents the total packet length of the packet (excluding the length field) in bytes. The opcode field contains the unique OP code for the specific command/event.

The PDU length is determined by the ACI packet type. Some ACI packets may have a variable length PDU depending on the parameter options for the specific ACI packet.

19 ACI packet types

This chapter defines the packet types sent or received on the ACI.

19.1 System commands

System commands are commands sent by the application processor to nRF8001. These commands control nRF8001 configuration, operating mode and behavior.

19.2 Data commands

Data commands are commands sent by the application processor to nRF8001. Data commands are used when application data traffic exchange between nRF8001 and a peer device is required. Application data is stored in the peer device (remote GATT server), or in nRF8001 (local GATT server).

Data commands initiate data transfer between the nRF8001 and the peer device:

- When nRF8001 is acting as a GATT server it can either:
 - initiate transfer of local application data to the peer device.
 - Receive application data sent from the peer device.
- When nRF8001 is acting as a GATT client it can either:
 - Send application data to the peer device.
 - Request application data to be transmitted from the peer device.
 - Receive application data from the peer device in the form of a **Handle Value Indication** or **Handle Value Notification** message.

Note: The timing of radio traffic is controlled by the *Bluetooth* low energy radio and is thus only indirectly tied to the timing of data commands. For example; data transmission will only occur during the short periodic time slots in a connection interval when the radio is active. Likewise, reception of data will only occur during a connection event if the peer device is able to respond.

19.3 Events

Events are messages sent from nRF8001 to the application processor. Events can be either responses or indications depending on the triggering factor for the event:

- Response events are direct acknowledgment responses to a command (for example, `CommandResponseEvent`).
- Indication events are messages to the application processor indicating that a condition or instance has occurred. For example, a `DisconnectedEvent` is generated when the RF link has been lost. This may be the result of the peer device moving out of range or having lost power. Similarly receiving data from a peer device will generate a `DataReceivedEvent`. This means that these events have no regular or predictable time relationship to an ACI command.

20 Service pipes

A key activity in a *Bluetooth* low energy application is accessing and exchanging specific application data contained in the Server setup and/or the Client. Application data can be stored locally (stored in the nRF8001) and remotely (stored in the peer device).

20.1 Functional description

In nRF8001, the concept of *service pipes*¹ is used to simplify access to service characteristics in a Client and/or Server. A service pipe may be considered a data transfer pipe to (or from) a specific characteristic in a Server or Client.

Service pipes point to a specific Characteristic declaration in a Service, for example, the Temperature Characteristic declaration in a Thermometer Service. The value of this Characteristic is transmitted (or received) through the Pipe. Once you have programmed the service pipes configuration into nRF8001, they are static for the lifetime of the application. The type and number of service pipes you need to define is dependant on your application profile requirements. When the application is active, only the application data of interest is sent through the defined service pipes.

The service pipe setup also defines the following:

- Direction of data transfer
 - This is either transmit or receive. A transmit pipe carries data from the application processor to a peer device. A receive pipe carries data from a peer device to the application processor.
- Server location
 - The characteristics value may either be located on the nRF8001 server or on the peer device.
- Acknowledgment requirements
 - The service pipes can be set to require acknowledgment from the peer device that transfers are successful and data is correctly received. A peer device may also require an acknowledgment to be sent from the nRF8001 application processor.
- Request initiation
 - There are two alternatives for transferring data between nRF8001 and a peer device. Data may either be transmitted from the peer device, or be received by nRF8001 upon it requesting the peer device to send data.

Acknowledgment and Request are service pipe features that you can enable once the characteristics value location and direction of data transfer are defined.

Each service pipe is assigned its unique service pipe number. When application data is sent to (or received from) a service server, the application software uses the service pipe number to map a pipe to a GATT Characteristic UUID and the service pipe features. A Characteristic is always associated with a specific Service UUID.

Multiple service pipes can be assigned to a specific characteristic value depending on the application. The transmit and receive service pipes are described in [section 20.4 on page 54](#) and [section 20.5 on page 58](#).

[Figure 23. on page 54](#) illustrates how different service pipes can be assigned to two separate characteristics. Both service pipe 1 and 2 will access the same data, but with different pipe features. Service pipe 3 is linked to a separate characteristics value and has a different feature.

1. Service pipe is a concept specific for the nRF8001 ACI - it is not a *Bluetooth* concept.

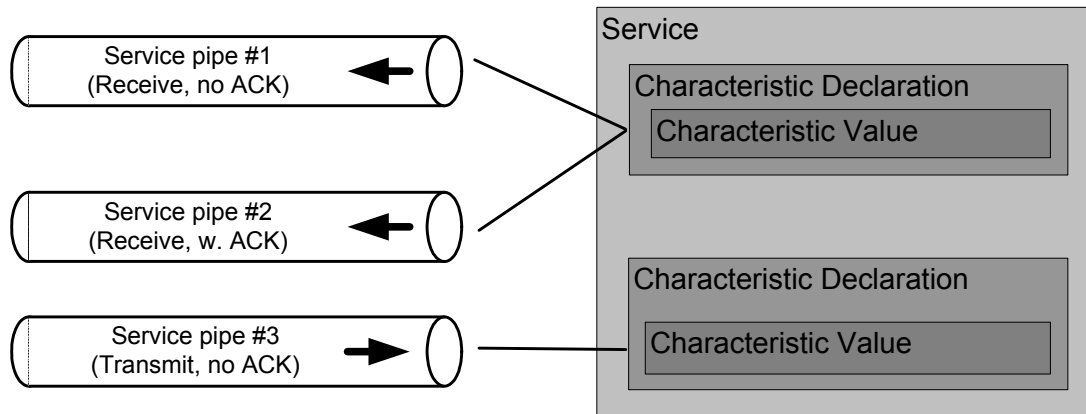


Figure 23. Service pipes assigned to characteristics under a service

20.2 Defining Service pipes

You need to define the services in the local GATT server and the service pipes associated to a remote or local GATT server according to your application requirements and then program them into nRF8001.

nRFGo Studio is a software program that lets you construct the nRF8001 GATT server and define the peer GATT server characteristics required for your application. This program is available on the Nordic Semiconductor website (www.nordicsemi.com) and is used for nRF8001 configuration. nRFGo Studio features predefined services that automatically define and set up the required service pipes.

Once you have defined the pipes and device setup in nRFGo Studio, the service pipe configuration can be exported and downloaded into nRF8001 using the ACI command `Setup`.

20.3 Data transfer on a service pipe

In normal operating mode, application data transfer is controlled by data commands. The data commands reference the defined service pipes when sending and receiving data. For example, the command `SendData(4, 0xF5)` will send 0xF5 through service pipe number 4. The service pipe number is used to identify the specific Characteristic declaration associated with the transmitted/received data.

To transmit or receive data on a service pipe, the service pipe must be open and the nRF8001 in the connected state. For service pipes associated with a remote GATT server, Service Discovery is automatically executed in order to connect the pipes.

20.4 Transmit pipes

Transmit pipes enable the transfer of application data to a peer device. A transmit pipe is associated with a Characteristic declaration and a Characteristic Property defined in the service pipe configuration.

When acknowledgment is enabled, a transmit is acknowledged by the peer device. A `DataAckEvent` is generated by nRF8001 upon receiving the acknowledgment.

Table 24. lists the available transmit pipe configurations and their functional description

| # | Characteristic location | Transmit pipe feature | | Functional description | Characteristic Property ¹ |
|---|-------------------------|-----------------------|---------|--|--------------------------------------|
| | | Ack | Request | | |
| 1 | Local | No | No | <ul style="list-style-type: none"> Update Server and notify Client (Peer device). Notification contains the new Characteristics Value. | Notify |
| 2 | Local | Yes | No | <ul style="list-style-type: none"> Update Server and send an indication of the update to the Client (Peer device). Peer device acknowledges a successful reception of the indication. nRF8001 generates <code>DataAckEvent</code> | Indicate |
| 3 | Remote | No | No | <ul style="list-style-type: none"> Transmit data to Server (Peer device). Peer device does not acknowledge data reception. | Write without Response |
| 4 | Remote | Yes | No | <ul style="list-style-type: none"> Transmit data to Server (Peer device). Peer device acknowledges a successful data reception. nRF8001 generates <code>DataAckEvent</code> | Write |

1. Bluetooth specification version 4.0, Vol. 3, Part G, Chapter 4.2

Table 24. Transmit pipe feature combinations

20.4.1 Opening a transmit pipe

Transmit pipes and transmit pipes with acknowledge on the local GATT server require opening prior to use. Opening is initiated by the peer device. To open and close the pipes, the peer device writes to the Client Configuration Characteristic Descriptor. For a Transmit Pipe, the peer device sets the Notification bit to open. For a Transmit Pipe with acknowledge, the peer device uses the Indication bit to open and close it, see table 3.11 in the Bluetooth specification version 4.0, Vol. 3, Part G, section 3.3.3. Once opened by the peer, the pipe(s) will be listed in the `OpenPipes` bitmap returned by the `PipesStatusEvent`.

Figure 24. on page 55 shows the sequence of events required to open local transmit pipes and local transmit pipes with acknowledge.

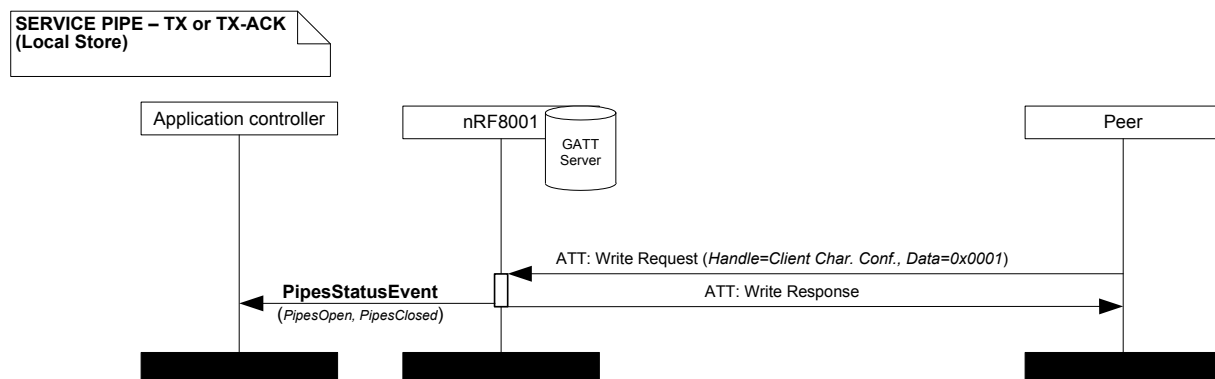


Figure 24. Transmit pipe opening; local data storage

20.4.2 Data transfer on a transmit pipe

Figure 25. through Figure 28. on page 57 shows the successful transfer of data on transmit pipes with and without the acknowledgment feature enabled and with data stored locally and remotely.

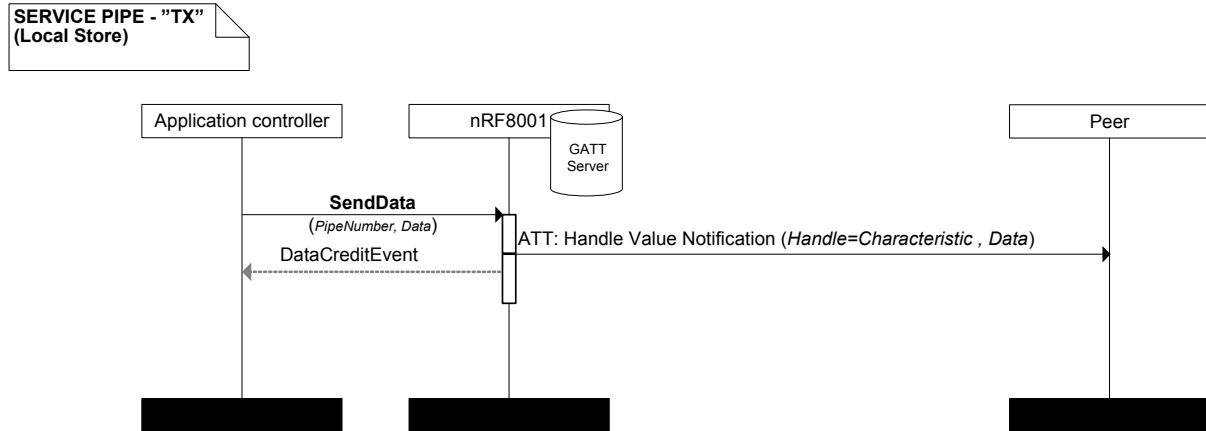


Figure 25. Data transfer on a transmit pipe; data stored locally

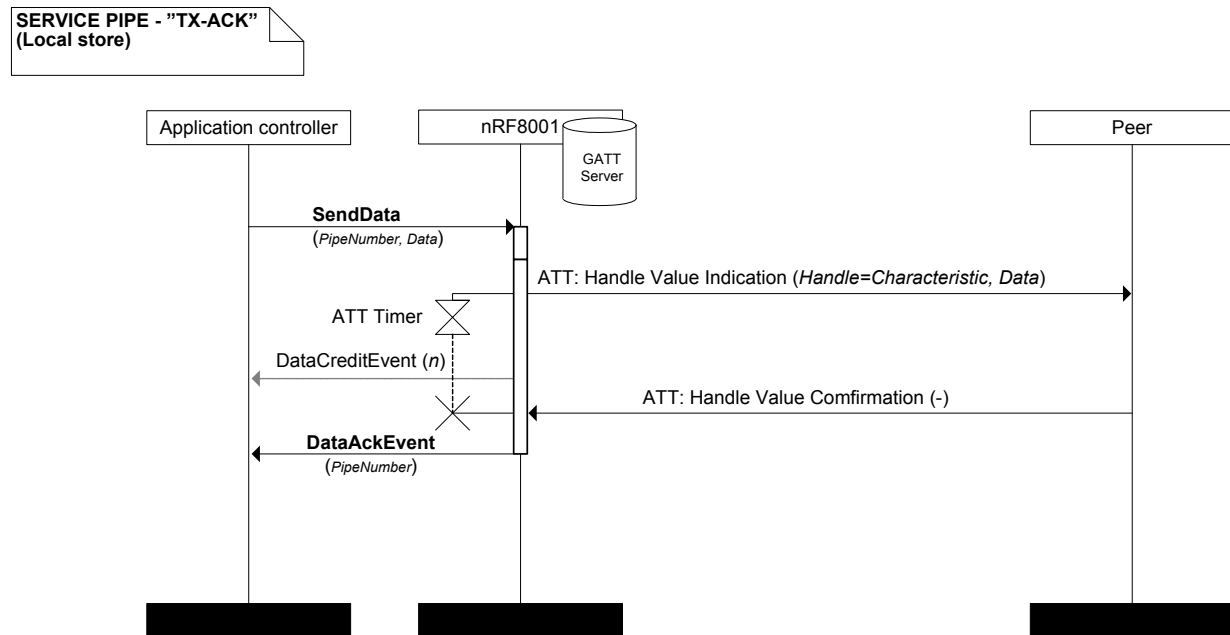


Figure 26. Data transfer on a transmit pipe; acknowledgment enabled and data stored locally

SERVICE PIPE - "TX"
(Remote Store)

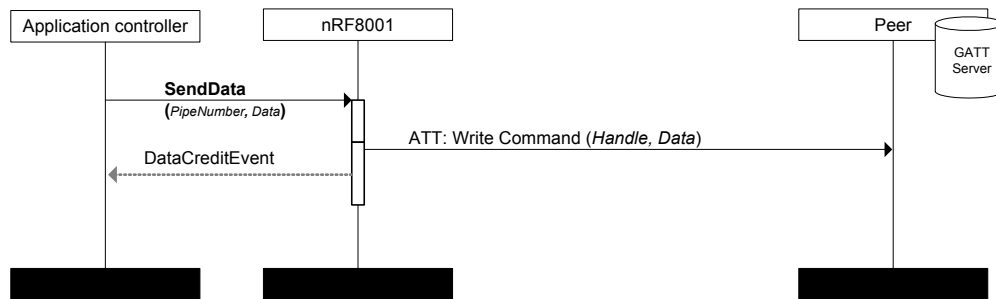


Figure 27. Data transfer on a transmit pipe; data stored remotely

SERVICE PIPE - "TX-ACK"
(Remote store)

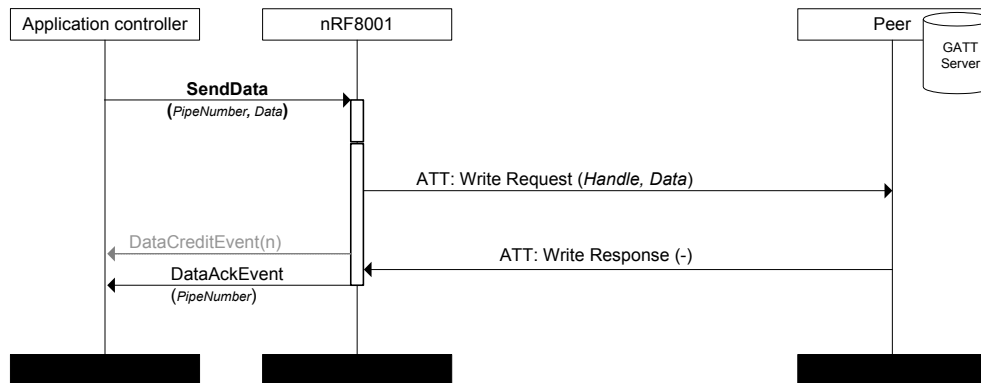


Figure 28. Data transfer on a transmit pipe; acknowledgment enabled and data stored remotely

20.4.3 Error events

Two events can be issued and returned to the application controller in the case of data transfer failure:

- If the failure is caused by loss of connection, a `DisconnectedEvent` is issued and returned to the application processor.
- If the failure is related to the data transfer, a `PipeErrorEvent` is issued and returned to the application processor.

20.5 Receive pipes

Receive pipes enable the reception of application data from a peer device. A receive pipe is associated with a Characteristic declaration defined in the service pipe setup. When application data is received from the peer, nRF8001 sends a `DataReceivedEvent` to the application processor.

Acknowledgment may be enabled for receive pipes. When acknowledgment is enabled, the application processor should send an acknowledgment for the `DataReceivedEvent` with the `SendDataAck`.

The Request feature must be enabled if nRF8001 is to initiate the data transfer through a Read Request to the remote device.

[Table 25](#) lists the available receive pipe configurations and their functional description. [Figure 29 on page 59](#) and [Figure 30 on page 59](#) show the MSCs applicable for the available feature settings.

| # | Data location | Receive pipe feature | | Functional description | Characteristic Property ¹ |
|---|---------------|----------------------|---------|---|--------------------------------------|
| | | Ack | Request | | |
| 1 | Local | No | No | <ul style="list-style-type: none"> Receive updates to Server from peer device nRF8001 generates <code>DataReceivedEvent</code> containing the received data | Write without response |
| 2 | Remote | No | No | <ul style="list-style-type: none"> Receive notification from Server (peer device) Upon receiving the notification, nRF8001 generates a <code>DataReceivedEvent</code> containing the received data | Notify |
| 3 | Remote | Yes | No | <ul style="list-style-type: none"> Receive indication from Server (peer device) Upon receiving the indication, nRF8001 generates a <code>DataReceivedEvent</code> containing the updated value The application processor acknowledges the indication by sending a <code>SendDataAck</code> to nRF8001. | Indicate |
| 4 | Remote | Yes | Yes | <ul style="list-style-type: none"> nRF8001 sends a Read Request to the Server (peer device) Peer device responds, returning the requested data Upon receiving the data, nRF8001 generates a <code>DataReceivedEvent</code> | Read |

1. *Bluetooth* specification Ver. 4.0, Vol. 3, Part G, Chapter 4.2

Table 25. Receive pipe feature combinations

20.5.1 Opening a receive pipe

Remote receive pipes and remote receive pipes with acknowledge require opening prior to use. Opening is initiated by the application processor. Once opened by application processor, the pipe(s) will be listed in the OpenPipes bitmap returned by the `PipesConnectedEvent`.

Figure 29. shows the sequence of events required to open a remote receive pipe.

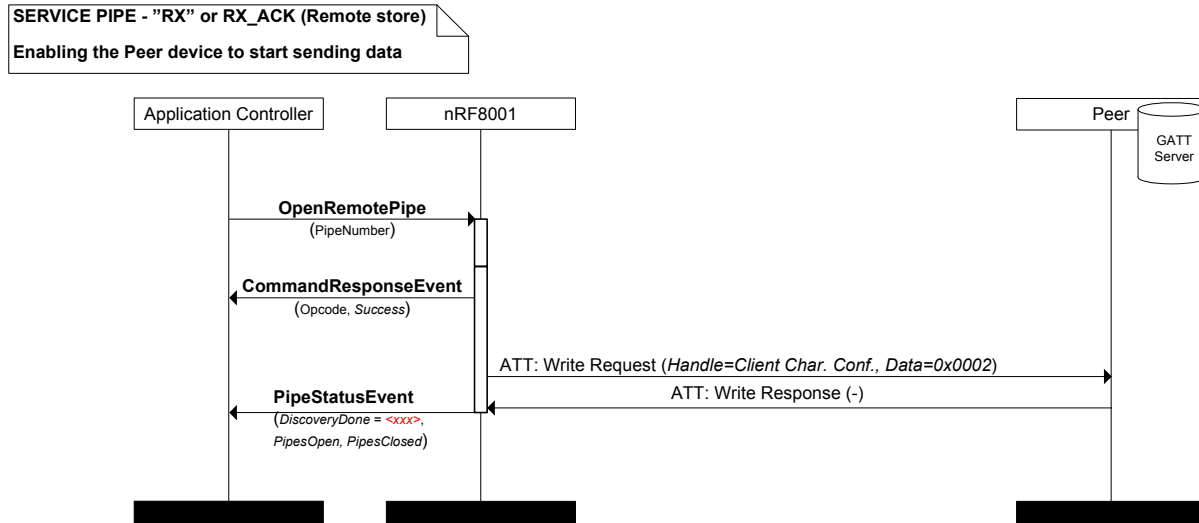


Figure 29. Receive pipe opening; data stored remotely

20.5.2 Data transfer on a receive pipe

The figures in this section show the successful transfer of data on receive pipes with and without the acknowledgment feature enabled and with data stored locally and remotely.

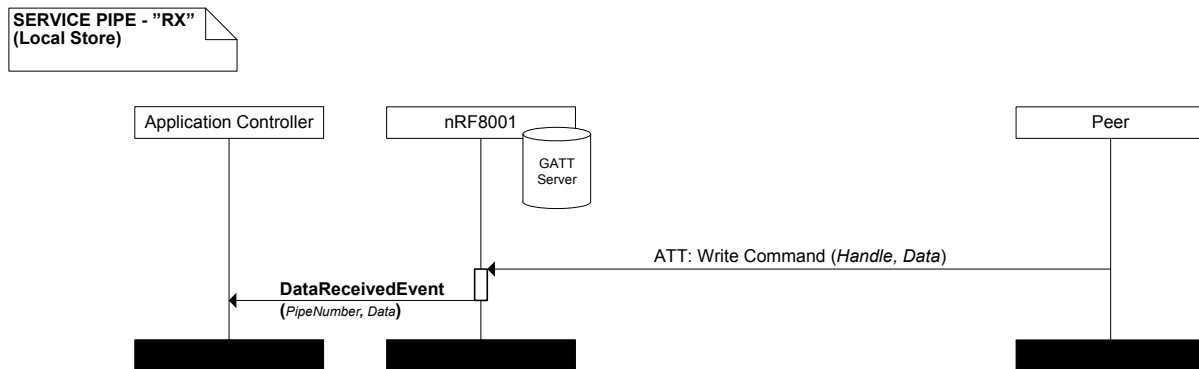


Figure 30. Data transfer on a receive pipe; data stored locally

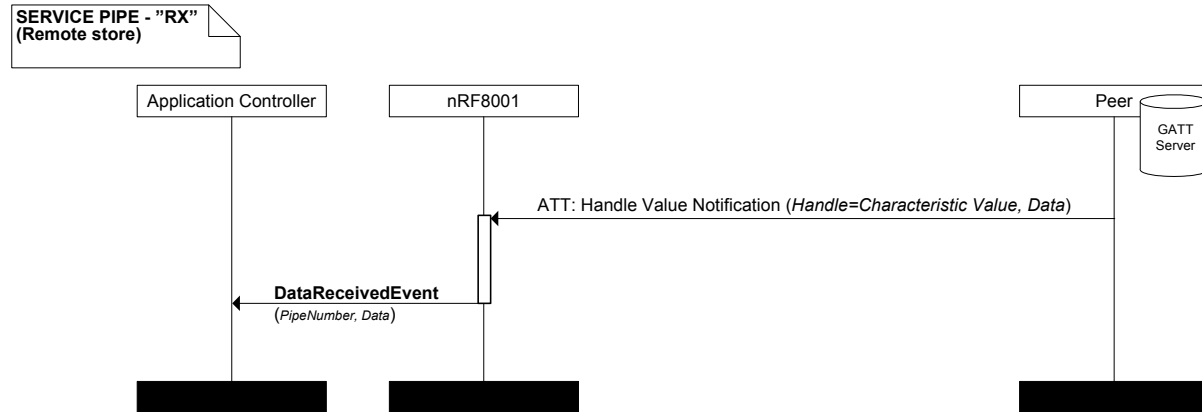


Figure 31. Data transfer on a receive pipe; data stored remotely

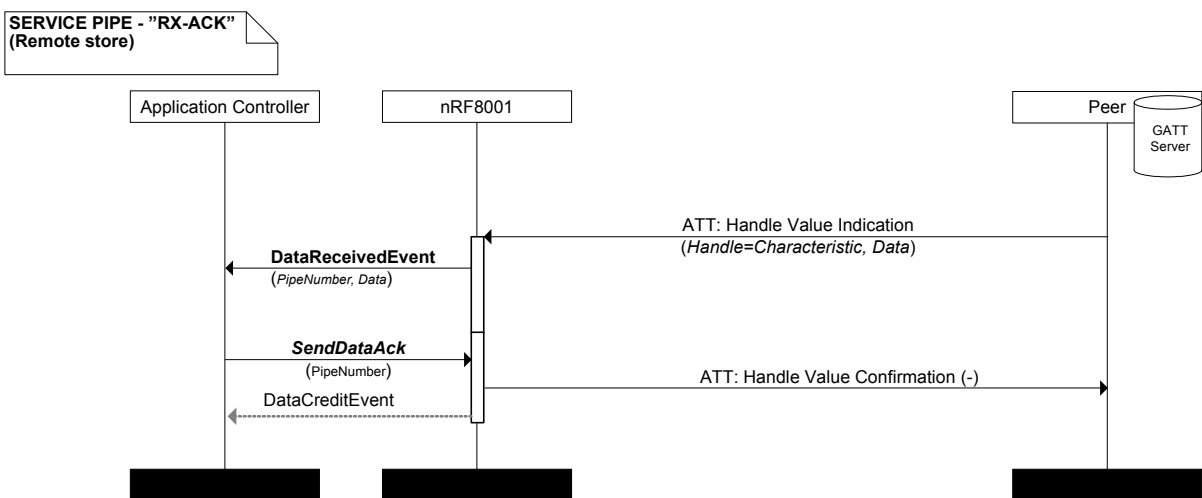


Figure 32. Data transfer on a receive pipe; acknowledgment enabled and data stored remotely

20.5.3 Error events

No error event is ever generated on a receive pipe. In the event of connection loss, a `DisconnectedEvent` is issued and returned to the application processor.

20.6 Service pipe availability

A service pipe to a remote server needs to associate the pipe number to the UUID of a Characteristic, Handle of a Characteristic, and the property of the same Characteristic. This operation is performed in the Service Discovery procedure (see [Section 22.4.2 on page 72](#)). Once the Service Discovery procedure is successfully completed, nRF8001 maps the relationship.

Service pipes need to be listed as available by nRF8001 before data transfer can take place. The pipe availability status is reported to the application processor in the `PipeStatusEvent`.

The `PipeStatusEvent` returns two pipe lists in the form of bitmaps:

- Pipes Open Bitmap: Open pipes where data can be received (or transmitted) without further action.
- Pipes Closed Bitmap: Closed pipes where data can be received only after nRF8001 application processor has instructed the GATT server (peer device) to send data (See [Table 26. on page 62](#)).

The receive pipes identified in the Pipes Closed Bitmap require opening by the application processor using the `OpenRemotePipe` command. The `OpenRemotePipe` command configures the peer device to transmit data on the receive pipe. The opened pipe is then listed in the Pipes Open Bitmap in the following `PipeStatusEvent`.

Transmit pipes that require opening by the peer device (see [Table 26. on page 62](#)) will appear in the Pipes Open Bitmap when nRF8001 has received the instruction from the peer device. A new `PipeStatusEvent` will occur whenever there is a change in the pipe availability status.

20.6.1 Availability timeline

A `PipeStatusEvent` will occur whenever there is a change in pipe availability status.

The time of which the pipe is listed as available depends on the pipe type and configuration. [Table 26](#) lists the available pipe configurations and the conditions required for being listed in the Pipes Open Bitmap.

| Direction | Pipe configuration | | | Conditions required to be listed in PipesOpen bitmap |
|-----------|--------------------|-----|---------|--|
| | Data location | Ack | Request | |
| TX | Local | No | No | 1. Connection established |
| TX | Local | Yes | No | 2. (Peer) Discovery procedure completed 3. Opened by peer device |
| TX | Remote | No | No | 1. Connection established |
| TX | Remote | Yes | No | 2. nRF8001 Discovery procedure completed |
| RX | Local | No | No | 1. Connection established 2. (Peer) Discovery procedure completed |
| RX | Remote | No | No | 1. Connection established |
| RX | Remote | Yes | No | 2. nRF8001 Discovery procedure completed 3. Opened by the application processor using <code>OpenRemotePipe</code> command |
| RX | Remote | No | Yes | 1. Connection established 2. nRF8001 Discovery procedure completed |

Table 26. Pipe availability; conditions required

21 Flow control

ACI commands received by nRF8001 are executed using the First In, First Out (FIFO) principle. System and Data commands differ in the flow control scheme they enforce:

- System commands must be confirmed as executed by nRF8001 before a new system command can be issued by the application processor. This implies that no system command can be sent before receiving an event from nRF8001 confirming the execution of the previous command.
- Data commands can be queued in a data command buffer pending execution. The application processor must ensure that the number of issued data commands does not overflow the data command buffer.

21.1 System command buffering

Only one System Command can be outstanding at any moment in time before the application processor is allowed to send another one. When the pending system command has executed, nRF8001 issues an event or a sequence of events. See the command descriptions in [chapter 23 on page 77](#) for details.

The application processor must receive an event confirming the execution of the command before sending the next system command.

The response event for a command is sent from nRF8001 after receiving the command from the application processor.

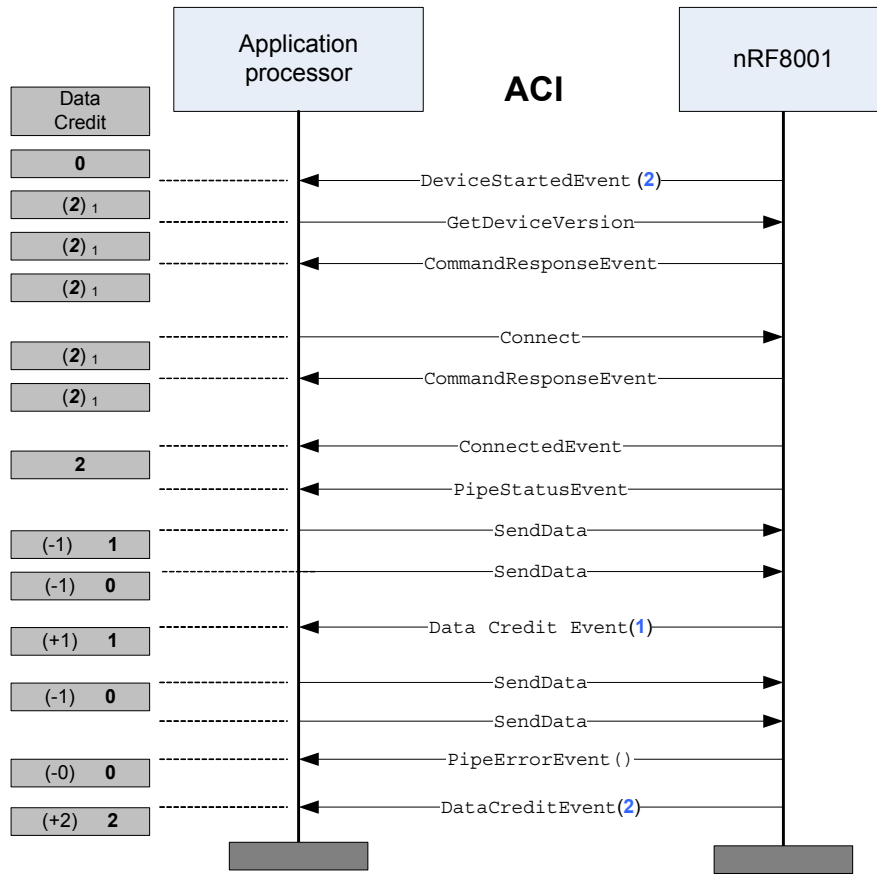
21.2 Data command buffering

The data command buffer size of nRF8001 is returned in the `DeviceStartedEvent`. The buffer size value represents the maximum number of data commands that may be queued by nRF8001, and is represented by a number of credits granted by nRF8001 to the application processor.

When a data command has been executed, a buffer location is released. Upon release of one or more buffer locations, nRF8001 sends a `DataCreditEvent` to the application processor. The `DataCreditEvent` contains the number of freed buffer locations, called credits. The application processor must keep track of the number of available credits at any time. No assumptions can be made by the application controller as to the timing of credit allowance, as a single `DataCreditEvent` may grant more than one credit. Submitting a command will subtract one credit from the number of available credits. [Figure 33. on page 64](#) illustrates the flow control and the data credit principle.

If the application processor tries to send a data command when no credit is available, nRF8001 will respond with a `PipeErrorEvent` with its status code set to `ACI_STATUS_ERROR_CREDIT_NOT_AVAILABLE`.

Please note that further restrictions apply related to ATT (Attribute Protocol) flow control, please refer to the *Bluetooth* Specification, Ver. 4.0, Vol. 3, Part F, section 3.3 for details.



1 : Credits may only be used while connected (after the Connected event)

Figure 33. Flow control example

If no credit event is received within 180 seconds after issuing a data command, then the application processor should issue the disconnect command to recover from this error condition². [Table 27](#) lists relevant ACI commands/events and lists their effect on the data command buffer memory.

| ACI Command/Event | Effect on command buffer memory |
|---------------------|--|
| DeviceStartedEvent | Data pipes disconnected: No credits can be used |
| PipeStatusEvent | Data pipes are connected and in open state: Credits can be used for the pipes identified as open |
| DisconnectedEvent | Data pipes disconnected: No credits can be used |
| DataAckEvent | No effect on buffer memory status |
| DataCreditEvent (n) | Returns <i>n</i> buffer memory credits to the application processor |
| DataReceivedEvent | No effect on buffer memory status |
| SendData | Uses ONE available credit |
| SendDataAck | Uses ONE available credit |
| RequestData | Uses ONE available credit |
| OpenRemotePipe | No effect on buffer memory status |

Table 27. ACI commands and events affecting command buffer memory credits

2. Data sent over a *Bluetooth* LE link can be No Acknowledged by the peer *Bluetooth* radio infinitely, that is, when data is sent using SendData/RequestData/SendDataAck. This is seen on the ACI as a missing Data Credit Event for a SendData/RequestData/SendDataAck. The application can disconnect to recover from this condition.

21.3 Flow control initialization

Before ACI commands are issued to nRF8001, the following conditions apply:

- No commands must be sent before the `DeviceStartedEvent` has been received by the application processor.
- Service pipes must be confirmed as open before data commands are issued. No data command shall be sent from the application processor before receiving the first `PipeStatusEvent` containing at least one open pipe.

22 Operational modes

nRF8001 has four modes of operation; Sleep, Setup, Active and Test. The application processor controls the nRF8001 operating modes by means of the ACI commands: *Sleep*, *Wakeup*, *Setup* and *Test*.

Discovered Services and bonding information are retained in all modes except Setup and Test since entering these mode will clear all dynamic data. To flush dynamic data requires a power reset of nRF8001.

22.1 Overview of operational modes

The following is a description of the nRF8001 operational modes:

- **Sleep mode**
 - Power saving mode; all functionality is stopped
 - Stored configuration settings are retained in memory
 - Dynamic data (like bonding information) is stored in memory
- **Setup mode**
 - nRF8001 configuration and setup:
 - GAP configuration
 - GATT service and GATT client configuration
 - Hardware configuration
 - Default operating mode entered upon reset unless setup is stored in NVM
 - All dynamic data is cleared
- **Active mode**
 - Mode used for runtime operation
 - Active mode controls three levels of activity:
 - Connected; nRF8001 is connected to a peer device, data transfer
 - Advertising; nRF8001 is advertising/trying to connect
 - Standby; No radio activity, Idle state
 - Completing the setup sequence puts nRF8001 in active mode
 - Establish a connection with a *Bluetooth* low energy central device
 - Establish a bonded relationship with a *Bluetooth* low energy central device
 - Send and receive data using service pipes
 - Save or restore dynamic data like bonding and pipe status
- **Test mode**
 - Two test features are available: RF PHY and ACI physical connection
 - Direct RF PHY Direct Test Mode (DTM)³ for qualification, test and evaluation of RF PHY layer performance
 - ACI physical connectivity test
 - All dynamic data is cleared

3. *Bluetooth* Specification, Ver. 4.0, Vol. 3, Part F, 'Direct Test Mode'

nRF8001 mode dependencies are illustrated in the state machine chart in [Figure 34](#).

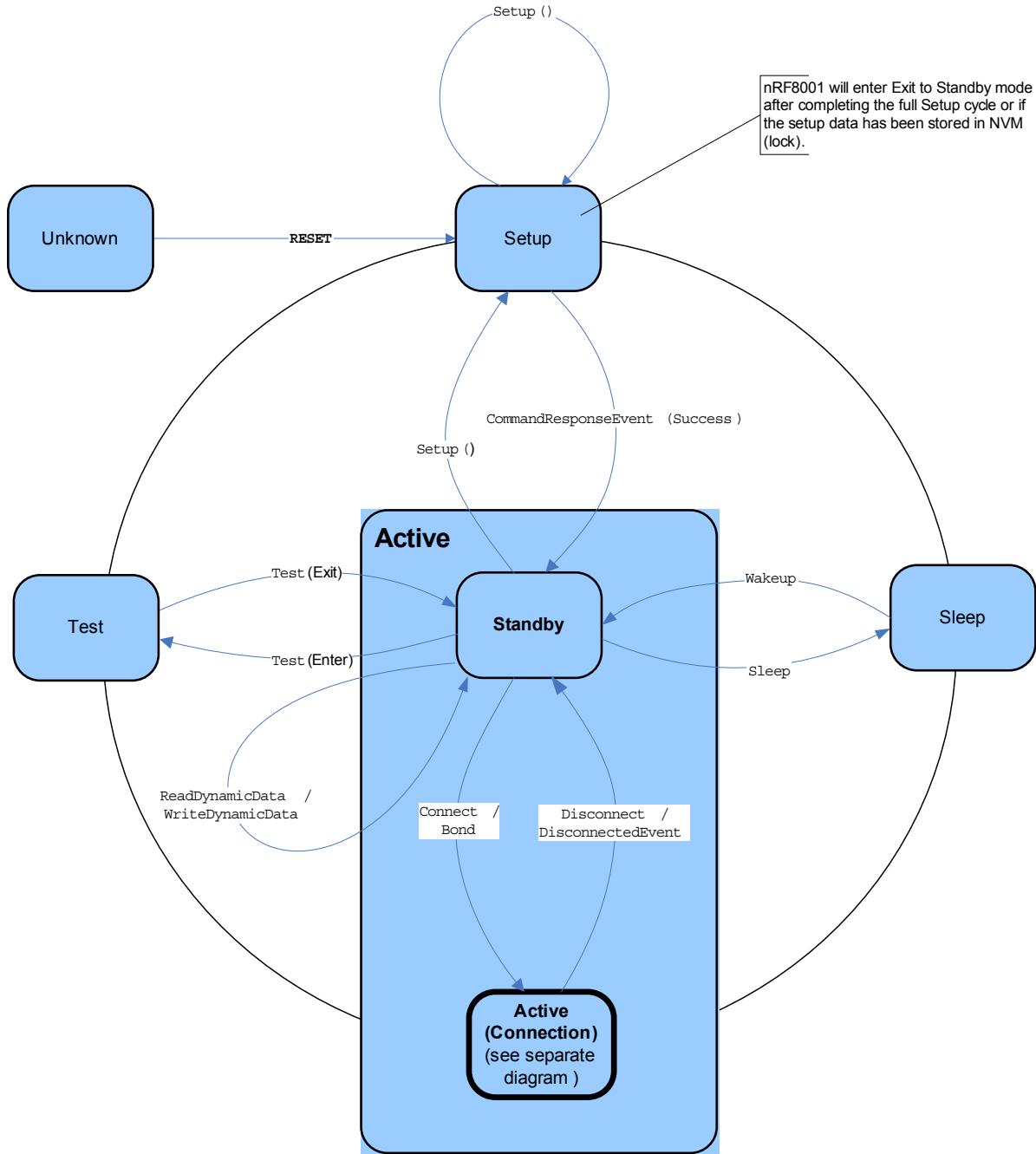


Figure 34. State Machine: Transition between operational modes⁴

Each mode has an associated set of ACI commands and events. An overview of the ACI commands and events applicable to the operational modes is listed in [Table 28. on page 78](#).

4. The state diagram illustrates the normal mode transitions for nRF8001. Note that all possible transitions are not depicted in the figure. For example; it is possible to enter Setup mode from Test mode. See [section 23.1 on page 77](#) and the protocol reference description of the ACI commands `Setup`, `Sleep` and `Test` for details.

22.2 Sleep mode

Sleep mode is used to preserve battery power when nRF8001 is not in a connection or actively broadcasting. Before entering Sleep mode, all connections must be terminated. When in sleep mode, all active connections are disconnected and no features are available. All configuration settings are retained in memory while in Sleep mode. No reconfiguration is required in order to resume normal operation.

The ACI command `Sleep` initiates Sleep mode. Upon receiving the ACI command `WakeUp`, nRF8001 is brought back to Standby mode.

22.3 Setup mode

Setup mode is used for uploading configuration and setup data generated in nRFgo Studio into nRF8001 volatile or non-volatile memory. Once written into non-volatile memory, the configuration is locked and can not be reprogrammed.

nRFgo Studio gives you the option of writing configuration and setup data to volatile memory. This option is intended for use in the application development phase and enables multiple re-writes without having to discard a device upon configuration data updates. Configuration data written to volatile memory will be lost upon reset or power cycling.

nRF8001 setup involves configuration of the following:

- GAP settings
- GATT services
- Hardware settings

Use nRFgo Studio to set up configuration settings. Once you have completed the configuration setup, the configuration can be exported from nRFgo Studio in the form of a set of ACI `Setup` commands. The setup procedure of nRF8001 must be completed before nRF8001 can be used to send or receive data.

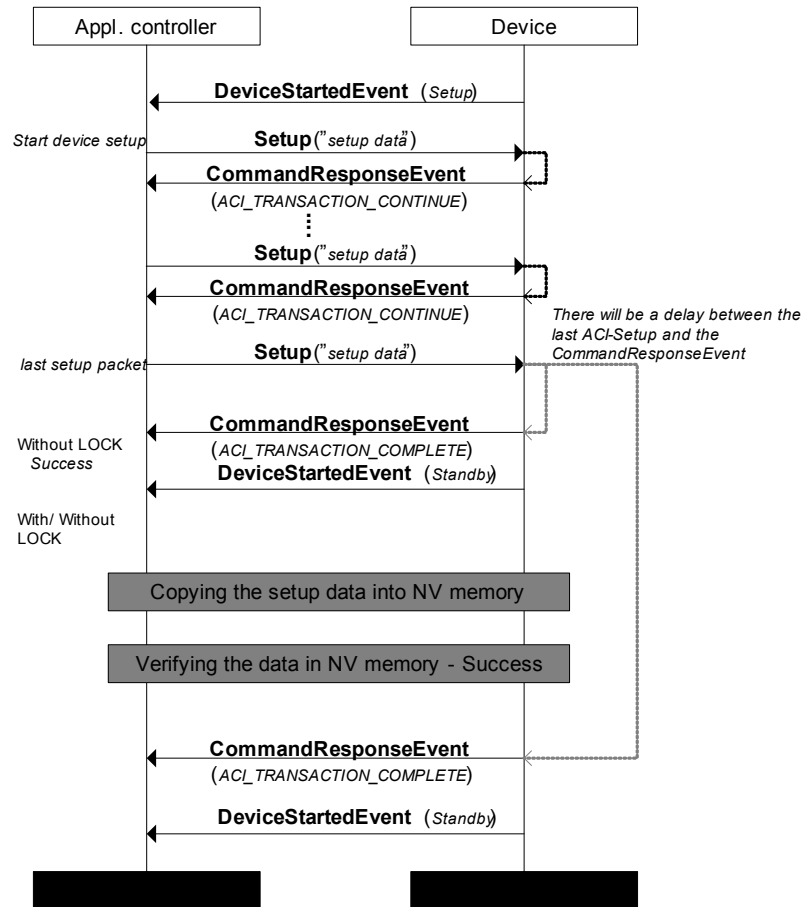


Figure 35. Setup procedure MSC

Note that the write time to non volatile memory is significantly longer than the write time to volatile memory. While typical write time of 1616 bytes of configuration data to non volatile memory is approx. 50 ms, the corresponding procedure execution time to non volatile memory is in the order of 1.6 sec.

22.3.1 GAP settings

The GAP settings defines the nRF8001 behaviour in normal operating mode (Active mode) and defines *Bluetooth* low energy specific parameters, such as (but not limited to):

- Device name
- Advertisement packet format and content
- Encryption requirement and key size

22.3.2 GATT services

GATT services represent the configuration of services (and the characteristics grouped under them)⁵ supported by nRF8001 when it acts as a server, and which services to create pipes to on a peer device, when acting as a client. When implementing a *Bluetooth* low energy profile, the required services are specified in the profile specification.

5. *Bluetooth* specification, Ver. 4.0, Vol. 3, Part G (GATT), Sect. 2.6 and Sect. 3

Setup of GATT services involves configuration of the following:

- Local Services (Server), relevant remote services (Client)
- Applicable service pipes

22.3.2.1 UUID configuration and format

All services and characteristics are identified by a 128 bit Universally Unique Identifier (UUID). Service and Characteristic UUIDs are either defined by the *Bluetooth* SIG or you may define your own.

The UUID's associated with the adopted *Bluetooth* services are listed in the Assigned Numbers document. This document can be downloaded from the *Bluetooth* SIG website: https://www.bluetooth.org/Technical/AssignedNumbers/service_discovery.htm. The format of the *Bluetooth* SIG UUIDs is illustrated in [Figure 36](#).

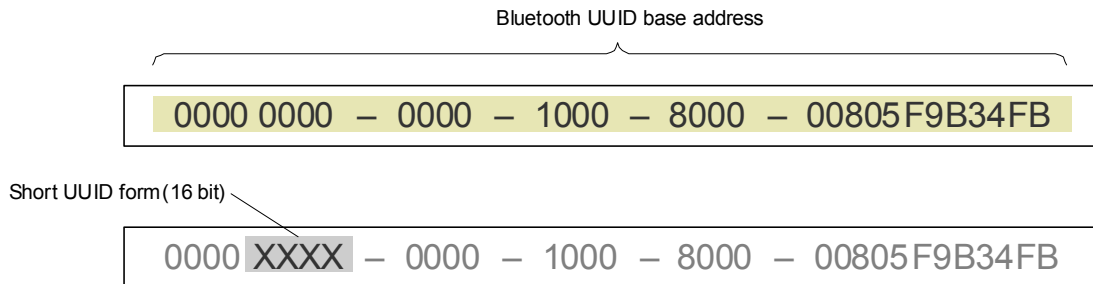


Figure 36. *Bluetooth* UUID format and organisation (Big Endian format)

The characters represented by bytes 13 and 14 are the short form UUID (16 bits rather than the full 128 bit version) which is used to identify the various *Bluetooth* services and characteristics.

If your application requires proprietary services or characteristics, it will use UUIDs that are outside the *Bluetooth* UUID address space.

It is your responsibility to ensure that any proprietary UUIDs you have defined are unique. Visit the International Telecommunication Union (ITU) website for details on the procedure for how to register your own UUIDs: <http://www.itu.int/ITU-T/asn1/uuid.html>.

nRF8001 supports storage of 5 vendor specific 128-bit base UUID that you can specify to any value. Each of the 5 base UUIDs can be further expanded to 65536 UUIDs by changing the 16 bits of the short form UUID, see [Figure 37](#).

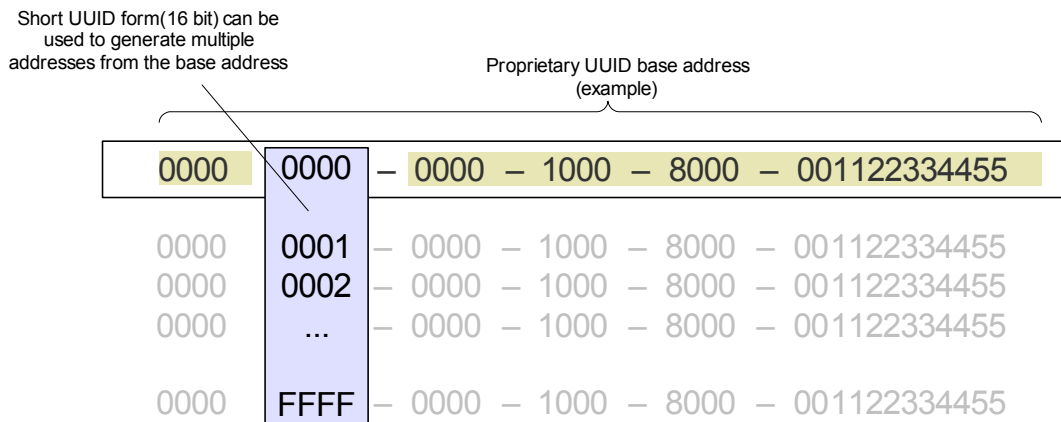


Figure 37. *nRF8001* UUID format and organization

22.3.3 Hardware settings

Hardware settings represent the configuration of proprietary hardware specific parameters, such as:

- Clock sources and settings
- Radio settings
- nRF8001 hardware feature activation and settings (Active signal, antenna EIRP, DC/DC converter and so on)

22.4 Active mode

Active mode handles run time operation and application data exchange. Completing nRF8001 setup is required prior to entering Active mode.

nRF8001 enters Active mode upon completion of the setup procedure, wakeup from sleep mode or directly from reset if the device configuration is stored in Non-Volatile Memory (NVM). nRF8001 exits Active mode upon receiving the ACL commands *Test*, *Setup* or *Sleep*.

Active mode enables the following operations and procedures to be initiated:

- Advertisement including service discovery upon connection establishment
- Bonding
- Sending application data through a transmit service pipe
- Receiving application data through a receive service pipe
- Saving and restoring dynamic data like bonding and pipe connection status

The Active mode state machine diagram is shown in [Figure 38. on page 71](#).

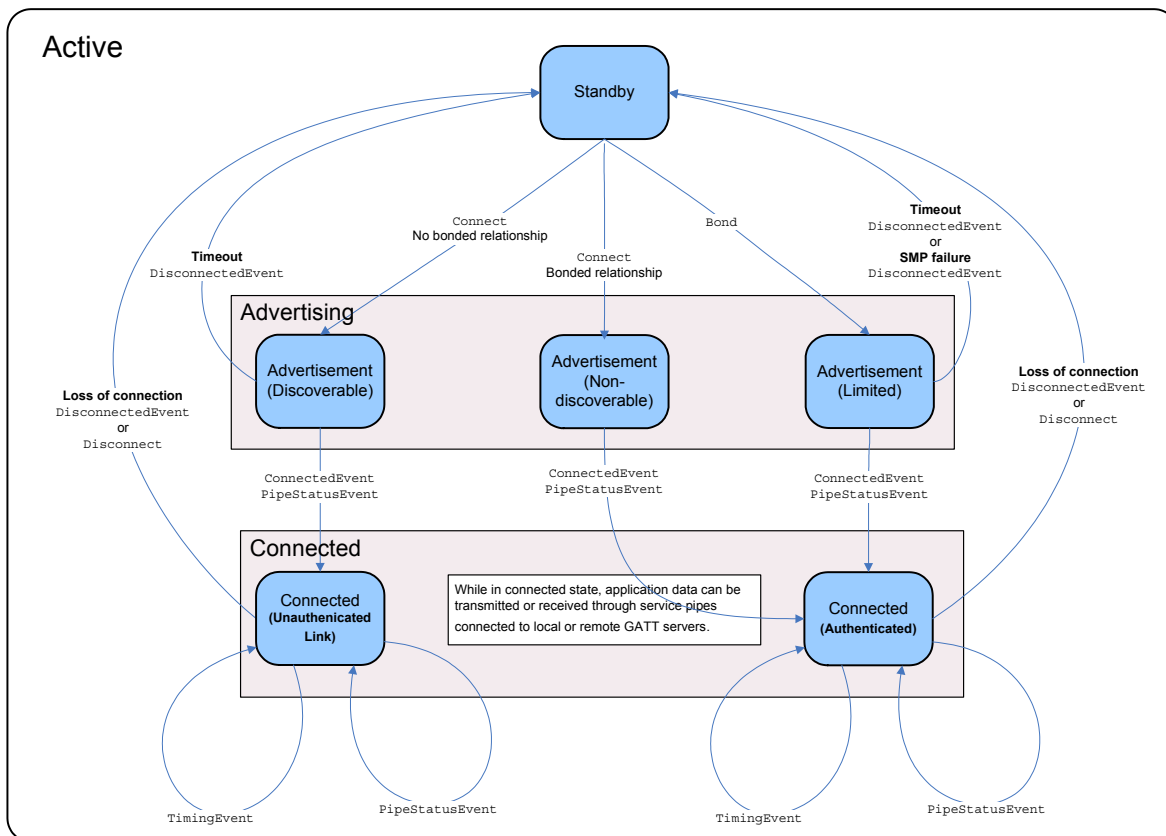


Figure 38. Active mode operation state machine

When in active mode, nRF8001 will start advertising in order to establish a connection to a peer device in the central role upon receiving the ACI commands Connect or Bond. A successful connection results in a transition to the connected state.

Once connected, the nRF8001 initiates the service discovery procedure when required, see [section 22.4.2 on page 72](#) for details.

A timeout value is set for advertising. nRF8001 advertises until a connection has been established, or until the timeout value is exceeded. If nRF8001 successfully connects to a peer device, nRF8001 will remain connected even after the timeout expires.

22.4.1 Advertising and Connection Establishment

While in Active mode, nRF8001 may be set to the following advertising modes;

- General Discoverable mode⁶ (Connect when not bonding)
- Limited Discoverable mode⁷ (Bond)
- Non-Discoverable mode⁸ (Connect when bonded)

In the case of a successful connection establishment, nRF8001 generates a `ConnectedEvent` followed by one or more `PipeStatusEvents`. nRF8001 will remain connected unless disconnected by the peer or by the application processor. The connection may also be lost as a result of the nRF8001 or peer device moving out of range or due to a protocol timeout or failure.

Upon advertisement timeout or connection loss, nRF8001 will return to Standby mode until a new `Bond` or `Connect` command is issued by the application processor.

22.4.2 Service Discovery

The service discovery procedure is initiated automatically upon connection establishment. The discovery procedure discovers the remote services on the peer device that have been defined through the setup procedure, see [section 22.3 on page 68](#). This procedure is required in order to establish the mapping between the pipe number and the remote characteristic attribute. The nRF8001 service discovery procedure activates the following GATT procedures:

- Discover Primary Services by Service UUID⁹
- Discover All Characteristics of a Service¹⁰
- Discover All Characteristic Descriptors¹¹
- Enabling the Service Changed characteristic¹²

Upon execution of the service discovery procedure, a `ConnectedEvent` is generated followed by a `PipeStatusEvent`, returning the service pipe availability status. Multiple `PipeStatusEvents` may result as the pipe availability status is updated during the service discovery procedure.

6. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.2.4., General Discoverable Mode

7. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.2.3., Limited Discoverable Mode

8. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.2.2., Non-Discoverable Mode

9. *Bluetooth* specification, Ver. 4, Vol. 3, Part G (GATT), Sect. 4.4.2., Discover Primary Service by Service UUID

10. *Bluetooth* specification, Ver. 4, Vol. 3, Part G (GATT), Sect. 4.6.1., Discover All Characteristics of a Service

11. *Bluetooth* specification, Ver. 4, Vol. 3, Part G (GATT), Sect. 4.7.1, Discover All Characteristics Descriptors.

12. *Bluetooth* specification, Ver. 4, Vol. 3, Part G (GATT), Sect. 7.1, Service Changed

To minimize power consumption, the nRF8001 service discovery procedure is only executed when the existing pipe mapping is outdated or non-existent. The following conditions will initiate the nRF8001 service discovery procedure:

- the services on the peer device are unknown (that is, when connecting to a new device)
- the nRF8001 is re-connecting to a non-bonded device that contains the **Service Changed** characteristic¹³
- services change while in a connection
- services have changed since last connection (only applicable for a bonded device)

The following applies to the lifetime of service discovery association:

- When bonded, the service discovery information is stored in nRF8001 until the bond is deleted.
- Any existing bond can be deleted by issuing a new `Bond` command or by the peer device.
- If the peer device is not bonded and contains the Service Changed characteristic, the existing service discovery association is lost upon connection loss.
- Upon power loss, Reset or Setup, the service discovery association is lost.

The service discovery association, bonding status and other dynamic data stored in volatile memory can be extracted from nRF8001 using the ACI command `ReadDynamicData`. When stored in the application processor, the same data can be reinstated at any time using the `WriteDynamicData` command.

22.4.3 Sending application data to a peer device

You can send application data using transmit service pipes as defined in [section 20.4 on page 54](#). The application data is sent to a peer device using the command `SendData(Data, ServicePipeNo)`. The application data is transmitted to the peer device at the next available connection event.

If the service pipe is set with acknowledgment, the application processor receives a `DataAckEvent` after a successful data transmission.

22.4.4 Receiving application data from a peer device

You can receive application data using receive service pipes as defined in [section 20.5 on page 58](#). Data is sent on the peer devices initiative. When nRF8001 receives data from a peer device, it will generate a `DataReceivedEvent(Data, ServicePipeNo)`.

You may request the transfer of data stored on the peer device by sending a `RequestData(ServicePipeNo)` command. Upon receiving the requested data from the peer device, nRF8001 generates a `DataReceivedEvent(Data, ServicePipeNo)`.

22.4.5 Bonding

Bonding is the process of exchanging and storing security keys and identity information with a peer device. Bonding is required if the application requires authentication.

The ACI command `Bond` initiates the **Bonding** procedure¹³ with a peer device as described in the *Bluetooth* low energy GAP specification. Once bonded, nRF8001 will generate a `ConnectedEvent` followed by a `BondStatusEvent` and one or more `PipeStatusEvent(s)`.

13. *Bluetooth* specification, Ver. 4.0, Vol. 3, Part C, Sect. 9.4, Bondable Mode

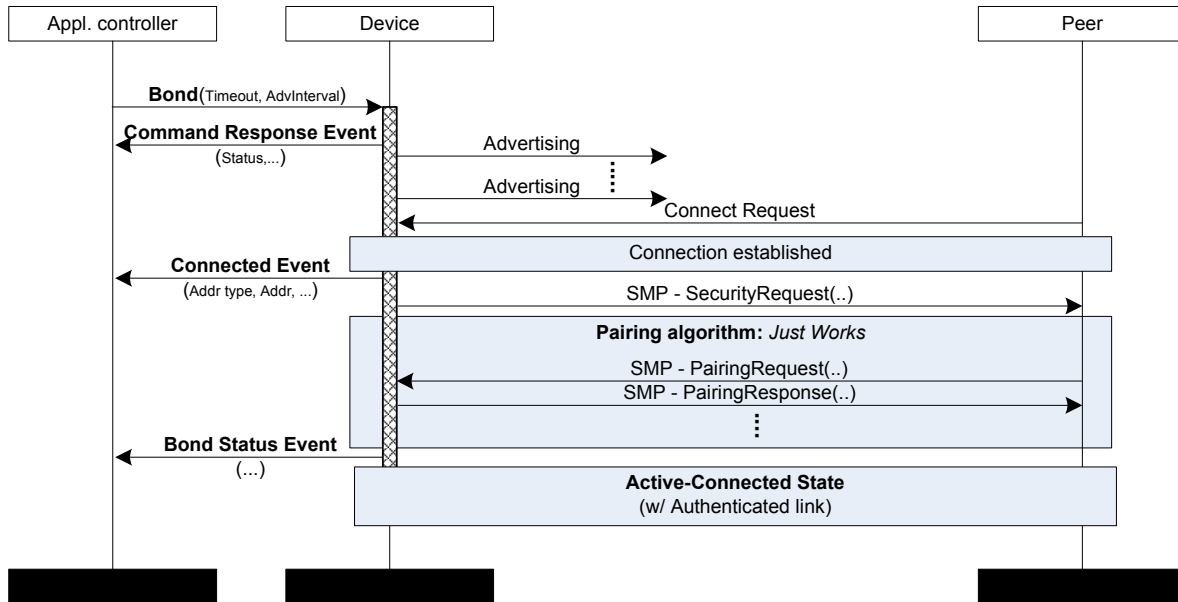


Figure 39. Bonding procedure MSC

22.4.6 Saving and restoring dynamic data

During normal runtime operation, your nRF8001 application will contain the Attributes and acquire information about peer devices and the services they offer. Your application may also establish a bonded relationship with a peer device. The information your application acquires as a result of normal runtime operation, is stored in nRF8001 as dynamic data. This information is stored in volatile memory and will be lost if the device is reset or disconnected from the supply voltage, or if the data is overwritten with new dynamic data using the `WriteDynamicData` procedure.

For applications that disconnect the power supply between periods of activity, dynamic data may be stored in the application processor and retrieved when the power supply is restored. This will enable the application to remember the bonding relationship and the pipe availability status valid at the time when the dynamic data was stored.

The ACI command `ReadDynamicData` will extract the dynamic data from nRF8001 for storage in the application processor. After power cycling, the ACI command `WriteDynamicData` can be used to reinstate the stored dynamic data in nRF8001. Note that nRF8001 must contain a valid setup before you read- or write dynamic data to volatile memory.

22.5 Test mode

Test mode is used to test the ACI physical connection and the RF PHY layer of nRF8001.

The ACI command `Test` is used to initiate and exit Test mode.

In test mode, the following test features can be enabled:

- RF PHY testing (*Bluetooth* low energy Direct Test Mode)
- ACI loopback test

While in Test mode, all active connections are disconnected and no device features are available other than the specified test functionality. All nRF8001 dynamic data is lost when entering Test mode.

Figure 40. on page 75 illustrates the test related interfaces and data exchange packets applicable for test mode.

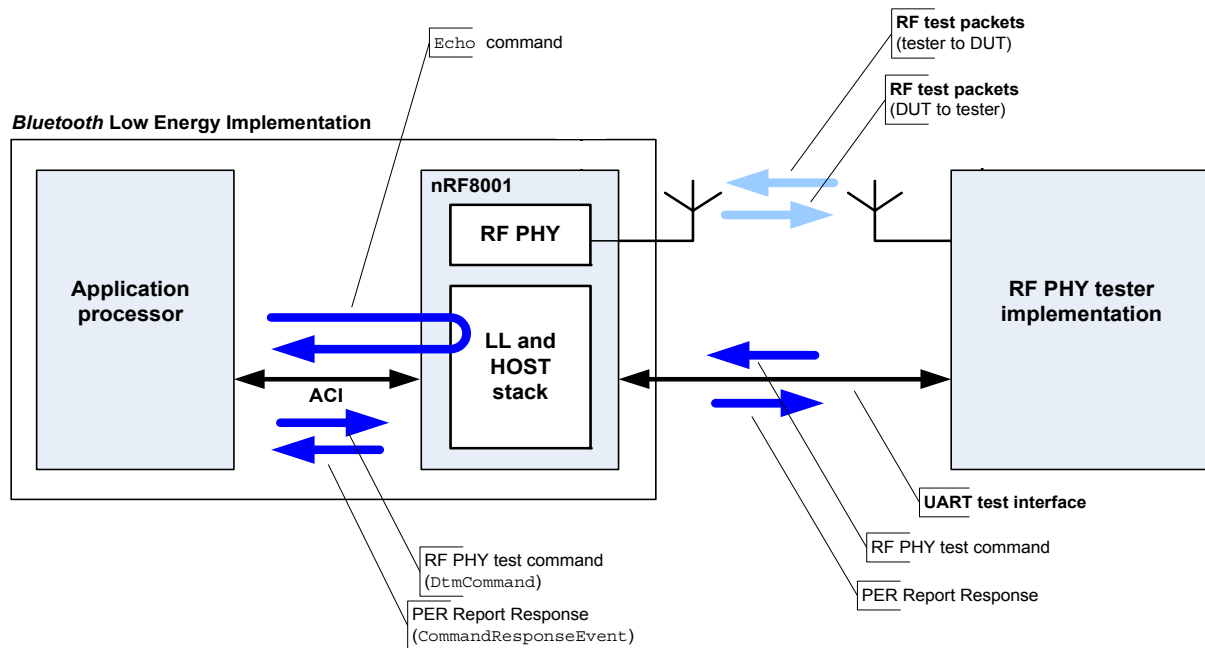


Figure 40. Test interfaces and data exchange in test mode

22.6 RF PHY testing

RF PHY testing can be performed using the UART interface command format as specified in Bluetooth Specification, v. 4.0, Volume 6, Part F, Direct Test Mode. Alternatively, DTM commands can be sent over the ACI using the command DtmCommand.

When in Test mode, the RF PHY Direct test mode UART interface is active and can be connected to a validated *Bluetooth* low energy RF PHY tester or to a proprietary RF test system. A proprietary RF tester must implement the Direct Test Mode commands and responses in the defined format¹⁴.

The UART test interface pins and electrical characteristics are described in [Part A, section 5.2 on page 14](#) and [section 7.3 on page 25](#).

22.6.1 Transmitter constant carrier operation

The DTM can also be used to initiate a constant unmodulated carrier mode on the specified RF channel. When initiated, this mode enables easy antenna and matching network tuning.

To initiate the constant carrier mode, the PKT¹⁴ field in the LSByte of the DTM command must be set to binary value 11.

14. *Bluetooth* specification, Ver. 4.0, Vol. 6, Part F, Section 3.3, 'Commands and Events'

22.6.2 ACI loopback test

The ACI command `Echo` is used to test the physical connection of the ACI. Upon receiving the `Echo` command, nRF8001 returns an `EchoEvent` containing the identical content of the `Echo` command to the application processor.

23 Protocol reference

23.1 Command and event overview

| Packet | Link to relevant section | OP code | Operating mode | | | | |
|------------------------|---|---------|----------------|---------|----------------|-------|------|
| | | | Setup | Standby | Active | Sleep | Test |
| System commands | | | | | | | |
| Test | Section 24.1 on page 82 | 0x01 | X | X | | | X |
| Sleep | Section 24.2 on page 83 | 0x04 | | X | | | |
| GetDeviceVersion | Section 24.3 on page 84 | 0x09 | X | X | X | | X |
| Echo | Section 24.4 on page 85 | 0x02 | | | | | X |
| Wakeup | Section 24.5 on page 86 | 0x05 | | | | X | |
| GetBatteryLevel | Section 24.6 on page 87 | 0x0B | X | X | X | | X |
| GetTemperature | Section 24.7 on page 88 | 0x0C | X | X | X | | X |
| Setup | Section 24.8 on page 89 | 0x06 | X | X | | | |
| SetTxPower | Section 24.9 on page 90 | 0x12 | | X | | | |
| GetDeviceAddress | Section 24.10 on page 91 | 0x0A | X | X | X | | X |
| Connect | Section 24.11 on page 92 | 0x0F | | X | | | |
| Bond | Section 24.13 on page 95 | 0x10 | | X | | | |
| Disconnect | Section 24.14 on page 97 | 0x11 | | | X | | |
| ChangeTimingRequest | Section 24.15 on page 98 | 0x13 | | | X ¹ | | |
| OpenRemotePipe | Section 24.16 on page 100 | 0x14 | | | X ² | | |
| DtmCommand | Section 24.17 on page 102 | 0x03 | | | | | X |
| ReadDynamicData | Section 24.18 on page 103 | 0x07 | | X | | | |
| WriteDynamicData | Section 24.19 on page 104 | 0x08 | | X | | | |
| RadioReset | Section 24.12 on page 94 | 0x0E | | X | X | | |
| System events | | | | | | | |
| DeviceStartedEvent | Section 26.1 on page 109 | 0x81 | X | X | X | | |
| EchoEvent | Section 26.2 on page 110 | 0x82 | | | | | |
| HardwareErrorEvent | Section 26.3 on page 111 | 0x83 | X | | | | |

| Packet | Link to relevant section | OP code | Operating mode | | | | |
|----------------------|--|---------|----------------|---------|--------|-------|------|
| | | | Setup | Standby | Active | Sleep | Test |
| CommandResponseEvent | Section 26.4 on page 112 | 0x84 | X | X | | | |
| ConnectedEvent | Section 26.5 on page 113 | 0x85 | | | X | | |
| DisconnectedEvent | Section 26.6 on page 115 | 0x86 | | | X | | |
| BondStatusEvent | Section 26.7 on page 116 | 0x87 | | | X | | |
| PipeStatusEvent | Section 26.8 on page 118 | 0x88 | | | X | | |
| TimingEvent | Section 26.9 on page 121 | 0x89 | | | X | | |

1. Only available after a `ConnectedEvent`
2. Only available after a `ConnectedEvent` and the applicable pipe is listed in the `PipesClosed` bitmap returned in the `PipeStatusEvent`

Table 28. System command/System event operating mode dependency

| Packet | Link to relevant section | OP code | Operating mode | | | | |
|----------------------|--|---------|----------------|---------|-----------------|-------|------|
| | | | Setup | Standby | Active | Sleep | Test |
| Data commands | | | | | | | |
| SendData | Section 25.1 on page 105 | 0x15 | | | X ¹ | | |
| RequestData | Section 25.2 on page 106 | 0x17 | | | X ²⁸ | | |
| SetLocalData | Section 25.3 on page 107 | 0x0D | | X | X | | |
| SendDataAck | Section 25.4 on page 108 | 0x16 | | | X ² | | |
| Data events | | | | | | | |
| DataReceivedEvent | Section 27.3 on page 124 | 0x8C | | | X | | |
| PipeErrorEvent | Section 27.2 on page 123 | 0x8D | | X | X | | |
| DataCreditEvent | Section 27.1 on page 122 | 0x8A | | | X | | |
| DataAckEvent | Section 27.4 on page 125 | 0x8B | | | X | | |

1. Only available after a ConnectedEvent and the applicable pipe is listed in the PipesOpen bitmap returned in the PipeStatusEvent
2. Only available after a ConnectedEvent and as a response to a DataReceivedEvent

Table 29. Data command/Data event operating mode dependency

| Command | Header | | Parameter | Relevant Events |
|------------------|---------|--------|------------------------------------|--|
| | OP code | length | | |
| Test | 0x01 | 2 | • TestFeature (1) | DeviceStartedEvent |
| Sleep | 0x04 | 1 | | |
| GetDeviceVersion | 0x09 | 1 | | CommandResponseEvent |
| Echo | 0x02 | 1..30 | • Data (0..29) | EchoEvent |
| Wakeup | 0x05 | 1 | | DeviceStartedEvent CommandResponseEvent |
| GetBatteryLevel | 0x0B | 1 | | CommandResponseEvent |
| GetTemperature | 0x0C | 1 | | CommandResponseEvent |
| Setup | 0x06 | 2..31 | • SetupData (1..30) | CommandResponseEvent DeviceStartedEvent |
| SetTxPower | 0x12 | 2 | • RadioTransmitPowerLevel (1) | CommandResponseEvent |
| GetDeviceAddress | 0x0A | 1 | | CommandResponseEvent |
| Connect | 0x0F | 5 | • Timeout (2) • AdvInterval (2) | Successful connection: CommandResponseEvent ConnectedEvent PipeStatusEvent (s) Failed Connection: CommandResponseEvent DisconnectedEvent |

| Command | Header | | Parameter | Relevant Events |
|---------------------|---------|--------|---|---|
| | OP code | length | | |
| Bond | 0x10 | 5 | <ul style="list-style-type: none"> Timeout (2) AdvInterval (2) | <p>Successful connection: CommandResponseEvent ConnectedEvent BondStatusEvent</p> <p>Failed Connection: CommandResponseEvent ConnectedEvent (opt) DisconnectedEvent</p> |
| Disconnect | 0x11 | 2 | <ul style="list-style-type: none"> Reason (1) | CommandResponseEvent DisconnectedEvent |
| ChangeTimingRequest | 0x13 | 1/9 | <ul style="list-style-type: none"> IntervalMin (2) IntervalMax (2) SlaveLatency (2) Timeput (2) | <p>Successful timing change: CommandResponseEvent TimingEvent</p> <p>Failed Connection: CommandResponseEvent DisconnectedEvent</p> |
| OpenRemotePipe | 0x14 | 2 | <ul style="list-style-type: none"> ServicePipeNumber (1) | <p>Successful opening: CommandResponseEvent PipeStatusEvent</p> <p>Failed opening: CommandResponseEvent PipeErrorEvent</p> |
| DtmCommand | 0x03 | 3 | <ul style="list-style-type: none"> DtmCommand (2) | CommandResponseEvent |
| ReadDynamicData | 0x07 | 1 | <ul style="list-style-type: none"> | CommandResponseEvent |
| WriteDynamicData | 0x08 | 3..29 | <ul style="list-style-type: none"> Sequence Number (1) SetupData (1..27) | CommandResponseEvent |
| RadioReset | 0x0E | 1 | | CommandResponseEvent |

Table 30. System commands format overview

| Command | Header | | Parameter | Relevant Events |
|--------------|---------|--------|---|---|
| | OP code | length | | |
| SendData | 0x15 | 2..22 | <ul style="list-style-type: none"> ServicePipeNumber (1) Data (1..20) | Successful transfer: DataCreditEvent (DataAckEvent) ¹ Failed transfer: DataCreditEvent PipeErrorEvent |
| RequestData | 0x17 | 2 | <ul style="list-style-type: none"> ServicePipeNumber (1) | Successful reception: DataReceivedEvent Failed transfer: PipeErrorEvent |
| SetLocalData | 0x0D | 3..22 | <ul style="list-style-type: none"> ServicePipeNumber (1) Data (1..20) | CommandResponseEvent |
| SendDataAck | 0x16 | 2 | <ul style="list-style-type: none"> ServicePipeNumber (1) | |

1. In case of the acknowledge feature being enabled on the service pipe

Table 31. Data commands format overview

24 System commands

System commands are commands used for nRF8001 configuration, operation mode control and runtime operations.

24.1 Test (0x01)

Test enables (or disables) the nRF8001 test mode.

24.1.1 Functional description

When in test mode, Direct Test Mode is enabled and ready to receive test commands as specified in the *Bluetooth* Specification, Ver. 4.0, Vol. 6, Part F, 'Direct Test Mode'. The physical test interface can be UART or ACI. Refer to [section 22.5 on page 74](#) and [section 24.17 on page 102](#) for details. The ACI physical interface may be tested using the command `Echo` (see [section 24.4 on page 85](#)). Upon entering and exiting Test mode, nRF8001 is reset.

24.1.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--------------------------|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Command | 1 | 0x01 | Test |
| Content | | | |
| <i>TestFeature</i> | 1 | | Test feature to activate |

Table 32. ACI message structure for Test

24.1.3 Accepted values

| Parameter | Data value | Description |
|--------------------|------------|--------------------------------|
| <i>TestFeature</i> | 0x01 | Enable DTM over UART interface |
| | 0x02 | Enable DTM over ACI |
| | 0xFF | Exit test mode |

Table 33. Accepted values for parameters, Test

24.1.4 Returned events

A `DeviceStartedEvent` is returned indicating that nRF8001 has entered or exited test mode. A `CommandResponseEvent` will result in case of failing to enter or exit Test mode.

24.1.5 Bluetooth low energy procedures used

This command invokes the *Bluetooth* low energy direct test mode functionality for further details, see *Bluetooth* Specification, v4.0, Volume 6, Part F, Direct Test Mode.

24.2 Sleep (0x04)

Sleep activates the nRF8001 Sleep mode.

24.2.1 Functional description

When in Sleep mode, all active connections are disconnected and no device features are available. Sleep mode should be used whenever possible in order to preserve battery power.

nRF8001 will remain in Sleep mode until receiving the Wakeup command.

24.2.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---------------|
| Header | | | |
| Length | 1 | 1 | Packet length |
| Command | 1 | 0x04 | Sleep |

Table 34. ACI message structure for Sleep

24.2.3 Accepted values

None

24.2.4 Returned events

None

24.2.5 Bluetooth low energy procedures used

None

24.3 GetDeviceVersion (0x09)

GetDeviceVersion requests nRF8001 version information.

24.3.1 Functional description

This command returns the nRF8001 version and configuration information. The information returned in the `CommandResponseEvent` may be requested by Nordic Semiconductor technical support when this is required.

24.3.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|------------------|
| Header | | | |
| Length | 1 | 1 | Packet length |
| Command | 1 | 0x09 | GetDeviceVersion |

Table 35. ACI message structure for GetDeviceVersion

24.3.3 Accepted values

None

24.3.4 Returned events

This command returns a `CommandResponseEvent`. Data returned in the event is:

- Command code: `GetDeviceVersion`
- Status: Success
- Response data:
 - Configuration ID (2 bytes); nRF8001 configuration identifier (LSB/MSB). This number can be used to trace the nRF8001 HW and FW versions.
 - ACI protocol version (1 byte); nRF8001 ACI version¹⁵
 - Current setup format (1 bytes); Format identifier of the nRF8001 configuration setup data
 - Setup ID¹⁶ (4 bytes); Setup ID for the application configuration
 - Configuration status (1 byte); bit 0: 1=Setup locked (NVM); 0=Setup open (VM)

24.3.5 Bluetooth low energy procedures used

None

15. The ACI protocol version is mapped to a specific and documented set of ACI commands, ACI events and ACI error and status codes. The version is incremented in the event of additional commands, events and codes being added to the nRF8001 design. The ACI version is backwards compatible with earlier versions.

16. You can set the Setup ID upon creating a configuration setup in nRFgo Studio. The Setup ID can then be used to identify a specific configuration setup and provide traceability for your design.

24.4 Echo (0x02)

Echo (0x0E) tests the nRF8001 ACI transport layer.

24.4.1 Functional description

Upon receiving an `Echo` command, nRF8001 returns an `EchoEvent` containing the identical command packet data to the application processor.

The reception of a loopback packet confirms a working ACI transport layer.

24.4.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---------------|
| Header | | | |
| Length | 1 | 1..30 | Packet length |
| Command | 1 | 0x02 | Echo |
| Content | | | |
| Data | 0..29 | | Any data |

Table 36.ACI message format for Echo

24.4.3 Accepted values

Any value from 0..29 bytes is accepted for this command.

24.4.4 Returned events

This command returns an `EchoEvent`.

24.4.5 Bluetooth low energy procedures used

None

24.5 Wakeup (0x05)

Wakeup wakes up nRF8001 from Sleep mode.

24.5.1 Functional description

Upon receiving the `Wakeup` command, nRF8001 is set to Standby mode.

24.5.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---------------|
| Header | | | |
| Length | 1 | 1 | Packet length |
| Command | 1 | 0x05 | Wakeup |

Table 37. ACI message structure for Wakeup

24.5.3 Accepted values

None

24.5.4 Returned events

This command returns a `DeviceStartedEvent`. It is then followed by a `CommandResponseEvent`.

Data returned in the `DeviceStartedEvent` is:

- Operating mode: Standby

Data returned in the `CommandResponseEvent` is:

- Command code: Wakeup
- Status: Success
- Response data: None

24.5.5 *Bluetooth* low energy procedures used

None

24.6 GetBatteryLevel (0x0B)

GetBatteryLevel measures the battery supply voltage level.

24.6.1 Functional description

Upon receiving the `GetBatteryLevel` command, the supply voltage level is sampled and reported as a 2 byte number.

24.6.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|-----------------|
| Header | | | |
| Length | 1 | 1 | Packet length |
| Command | 1 | 0x0B | GetBatteryLevel |

Table 38. ACI message structure for GetBatteryLevel

24.6.3 Accepted values

None

24.6.4 Returned events

This command returns a `CommandResponseEvent`.

Data returned in the `CommandResponseEvent` is:

- Command code: `GetBatteryLevel`
- Status: Success
- Response data: Supply voltage level (2 bytes, LSB/MSB). Analog voltage is calculated by multiplying the binary number by 3.52mV .

24.6.5 Bluetooth low energy procedures used

None

24.7 GetTemperature (0x0C)

GetTemperature (0x13) measures the ambient temperature.

24.7.1 Functional description

Upon receiving the `GetTemperature` command, the temperature is measured and reported as a 2 byte number.

24.7.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|----------------|
| Header | | | |
| Length | 1 | 1 | Packet length |
| Command | 1 | 0x0C | GetTemperature |

Table 39. ACI message structure for GetTemperature

24.7.3 Accepted values

None

24.7.4 Returned events

This command returns a `CommandResponseEvent`.

Data returned in the `CommandResponseEvent` is:

- Command code: `GetTemperature`
- Status: Success
- Response data: Temperature level (2 bytes, 2's complement format, LSB/MSB). Ambient temperature in degrees Celcius is calculated by multiplying the binary number by 4. For example, the value 0x000A represents 2.5 °C.

24.7.5 Bluetooth low energy procedures used

None

24.8 Setup (0x06)

Setup uploads the configuration bit pattern generated by nRFGo Studio.

24.8.1 Functional description

Setup is performed by issuing a consecutive series of `Setup` commands. The number and contents of the `Setup` commands required are defined by the nRFGo Studio output.

24.8.2 Message format

| Message field/ parameter | Value size (bytes) | Data value | Description |
|-----------------------------|-----------------------|------------|---|
| Header | | | |
| Length | 1 | 1..31 | |
| Command | 1 | 0x06 | Setup |
| Content | | | |
| SetupData | 1..30 | | nRF8001 configuration data exported from nRFGo Studio |

Table 40. ACI message structure for Setup

24.8.3 Accepted values

The `Setup` command accepts a configuration bit pattern generated by nRFGo Studio.

24.8.4 Returned events

This command returns a `CommandResponseEvent` followed by a `DeviceStartedEvent` upon completion of the complete `Setup` sequence. Data returned in the `CommandResponseEvent` is:

- Command code: `Setup`
- Status:
 - Status code
 - Transaction continue
 - Transaction complete
 - (Error)
- Response data: None

After the final `Setup` command has been received, nRF8001 will switch to Standby state and execute the new device settings. Upon switching to Standby state, a `DeviceStartedEvent` is generated.

24.8.5 Bluetooth low energy procedures used

None

24.9 SetTxPower (0x12)

SetTxPower sets the output power level of the *Bluetooth* low energy radio.

24.9.1 Functional description

This command is used to change the radio transmitter output power setting in runtime operation and overwrites the the transmit power setting set in the configuration settings. The transmit power setting set by the SetTxPower command will be used for all radio transmissions until set to a different value. In the event of device reset or power cycling, nRF8001 will reset the transmit power to the original configuration data setting. The ReadDynamicData command will extract the the transmit power setting set by the SetTxPower command as part of the dynamic data.

24.9.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Command | 1 | 0x12 | SetTxPower |
| Content | | | |
| RadioTransmitPowerLevel | 1 | | Device output power setting. Radio output power is set to the default value if SetTxPower command is not issued. |

Table 41. ACI message structure for SetTxPower

24.9.3 Accepted values

| Parameter | Data value | Description |
|-------------------------|------------|-----------------------|
| RadioTransmitPowerLevel | 0x00 | -18 dBm |
| | 0x01 | -12dBm |
| | 0x02 | -6 dBm |
| | 0x03 | 0 dBm (Default value) |

Table 42. Accepted values for parameters, SetTxPower

24.9.4 Returned events

This command returns a `CommandResponseEvent`. Data returned in the event is:

- Command code: `SetTxPower`
- Status: Success / Error code
- Response data: None

24.9.5 *Bluetooth* low energy procedures used

None

24.10 GetDeviceAddress (0x0A)

GetDeviceAddress returns the address of the nRF8001 device.

24.10.1 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|------------------|
| Header | | | |
| Length | 1 | 1 | Packet length |
| Command | 1 | 0x0A | GetDeviceAddress |

Table 43. ACL message structure for GetDeviceAddress

24.10.2 Accepted values

None

24.10.3 Returned events

This command returns a `CommandResponseEvent`. Data returned in the event is:

- Command code: `GetDeviceAddress`
- Status: Success / status code
- Response data:
 - Device address (6 byte) : Device address (Byte order LSB to MSB)
 - Address type (1 byte) :
 - 0x01 : Public address
 - 0x02 : Random static address
 - 0x03 : Random Private - Resolvable
 - 0x04 : Random Private - Unresolvable

24.10.4 Bluetooth low energy procedures used

None

24.11 Connect (0x0F)

Connect starts advertising and establishes a connection with a peer device

24.11.1 Functional description

nRF8001 configuration must be completed before issuing this command.

24.11.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 5 | Packet length |
| Command | 1 | 0x0F | Connect |
| Content | | | |
| Timeout | 2 | | Advertisement time duration in seconds. Upon timeout, nRF8001 stops advertising and exits to Standby mode. |
| AdvInterval | 2 | | Advertisement interval setting |

Table 44. ACI message structure for Connect

24.11.3 Accepted values

| Parameter | Data value | Description |
|--------------------|-------------------------------|---|
| <i>Timeout</i> | 0x0000 | Infinite advertisement, no timeout If required, the <code>RadioReset</code> command will abort the continuous advertisement and return nRF8001 to Standby mode |
| | 1-16383 (0x3FFF) | Valid timeout range in seconds |
| <i>AdvInterval</i> | 32 - 16384 (0x0020 to 0x4000) | Advertising interval set in periods of 0.625 msec |

Table 45. Accepted values for parameters, Connect

24.11.4 Returned events

This command returns a series of events in a specific order. The order depends on the outcome of the connection establishment procedure.

In the case of a successful connection establishment, the event order is:

1. `CommandResponseEvent`
2. `ConnectedEvent`
3. `PipeStatusEvent(s)`¹⁷

In the case of a failed connection establishment, the event order is:

1. `CommandResponseEvent`
2. `DisconnectedEvent`

17. Multiple `PipeStatusEvents` may result depending on the pipe characteristics and the service discovery activity initiated by the peer device.

Data returned in the `CommandResponseEvent` is:

- Command code: `Connect`
- Status: `Success` / Status code
- Response data: `None`

24.11.5 ***Bluetooth low energy procedures used***

This command starts the following GAP procedures:

- General Discoverable Mode¹⁸
- Non-Discoverable Mode¹⁹
- Undirected Connectable Mode²⁰
- Non-Bondable Mode²¹

18. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.2.4

19. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.2.2

20. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.3.4

21. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.4.2

24.12 RadioReset (0x0E)

RadioReset resets the radio transceiver and forcibly terminates any active connection or advertisement.

24.12.1 Functional description

This command resets the radio transceiver and returns nRF8001 to Standby mode. All dynamic data is retained in memory after execution of the `RadioReset` command. Executing `RadioReset` while in a connection forcibly terminates the connection and returns nRF8001 to Standby mode without generating a `DisconnectedEvent`.

24.12.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---------------|
| Header | | | |
| Length | 1 | 5 | Packet length |
| Command | 1 | 0x0E | RadioReset |

Table 46. ACI message structure for RadioReset

24.12.3 Accepted values

None

24.12.4 Returned events

This command returns a `CommandResponseEvent`.

Data returned in the `CommandResponseEvent` is:

- Command code: `RadioReset`
- Status: Success / Status code
- Response data: None

24.12.5 Bluetooth low energy procedures used

None

24.13 Bond (0x10)

Bond starts advertising with the intent of setting up a trusted relationship with a peer device

24.13.1 Functional description

nRF8001 configuration must be completed before this command is issued.

24.13.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---|
| Header | | | |
| Length | 1 | 5 | Packet length |
| Command | 1 | 0x10 | Bond |
| Content | | | |
| Timeout | 2 | | Advertisement time duration. Upon timeout, nRF8001 stops advertising and exits to Standby mode. |
| AdvInterval | 2 | | Advertisement interval setting |

Table 47. ACI message structure for Bond

24.13.3 Accepted values

| Parameter | Data value | Description |
|--------------------|---------------------------|---|
| <i>Timeout</i> | 1-30 (0x0001 – 0x001E) | Valid advertisement timeout range in seconds |
| <i>AdvInterval</i> | 0x0020 to 0x4000 | Advertising interval set in periods of 0.625 msec (LSB/MSB) |

Table 48. Accepted values for parameters, Bond

24.13.4 Returned events

This command returns a series of events in a specific order. The order depends on the outcome of the connection establishment and bonding procedure.

In the case of a successful bonding, the event order is:

- `CommandResponseEvent`
- `ConnectedEvent`
- `BondStatusEvent`

In the case of a failed connection establishment or bonding procedure, the event order is:

- `CommandResponseEvent`
- `ConnectedEvent` (Optional)
- `DisconnectedEvent`

Data returned in the `CommandResponseEvent` is:

- Command code: `Bond`
- Status: Success / Error code
- Response data: None

24.13.5 *Bluetooth* low energy procedures used

This command starts the following GAP procedures:

- Limited Discoverable Mode²²
- Bondable Mode²³

22. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.2.3

23. *Bluetooth* specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.4.3

24.14 Disconnect (0x11)

Disconnect terminates the connection with the peer device

24.14.1 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Command | 1 | 0x11 | Disconnect |
| Content | | | |
| Reason | 1 | | Reason for connection termination request |

Table 49. ACI message structure for Disconnect

24.14.2 Accepted values

| Parameter | Data value | Description |
|---------------|------------|--|
| <i>Reason</i> | 0x01 | Request termination of the connection with the peer device with the reason "Remote user terminated connection" |
| | 0x02 | Request termination of the link with the peer device with the reason "Unacceptable connection timing" |

Table 50. Accepted values for parameters, Disconnect

24.14.3 Returned events

This command returns a `CommandResponseEvent`. It is then followed by a `DisconnectedEvent`.

Data returned in the `CommandResponseEvent` is:

- Command code: `Disconnect`
- Status: Success / Error code
- Response data: None

24.14.4 Bluetooth low energy procedures used

This command starts the following GAP procedures:

- Terminate Connection procedure²⁴

24. Bluetooth specification, Ver. 4, Vol. 3, Part C (GAP), Sect. 9.3.10

24.15 ChangeTimingRequest (0x13)

ChangeTimingRequest initiates the connection parameter update procedure.

24.15.1 Functional description

This command is used to request the peer device to change the connection timing. The command can be given both with or without the timing parameters specified.

If the command is given without any timing parameters included, the timing request will use the timing values specified in nRFgo Studio as part of the device configuration setup. If the command is sent with timing parameters included, the timing request will use the timing parameters specified in the ChangeTimingRequest command when it sends the request to the peer device. All 4 timing parameters must be specified in the command. That is, the length field can only be 1 or 9.

24.15.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|---------------|---|
| Header | | | |
| Length | 1 | 1 or 9 | |
| Command | 1 | 0x13 | ChangeTimingRequest |
| Content | | | |
| Interval Min | 2 | Interval Min | Minimum value for the connection event interval (LSB/MSB) |
| Interval Max | 2 | Interval Max | Maximum value for the connection event interval (LSB/MSB) |
| Slave latency | 2 | Slave latency | Slave latency setting (LSB/MSB) |
| Timeout | 2 | Timeout | Timeout value for the connection (LSB/MSB) |

Table 51. ACI message structure for ChangeTimingRequest

24.15.3 Accepted values

| Parameter | Data value | Description |
|---------------|------------------------------|---|
| Interval Min | 6..3200 | Minimum interval = data value multiplied by 1,25ms |
| Interval Max | 6..3200 | Maximum interval = data value multiplied by 1,25ms |
| Slave latency | 0..1000 (0x0000 - 0x03E8) | The number of consecutive connection events that the slave is not required to respond |
| Timeout | 10..3200 | Timeout = data value multiplied by 10ms |

Table 52. Accepted values for parameters, ChangeTimingRequest

24.15.4 Returned values

Events are returned in the following order:

1. Command response event.
2. Timing event.

The application processor should examine the Timing Event against the requested timing to verify that the link timing was changed successfully.

Data returned in the `CommandResponseEvent` is:

- Command code: `ChangeTimingRequest`
- Status: Success / Status code
- Response data: None

Figure 41. illustrates the communication scenarios for the `ChangeTimeRequest` command.

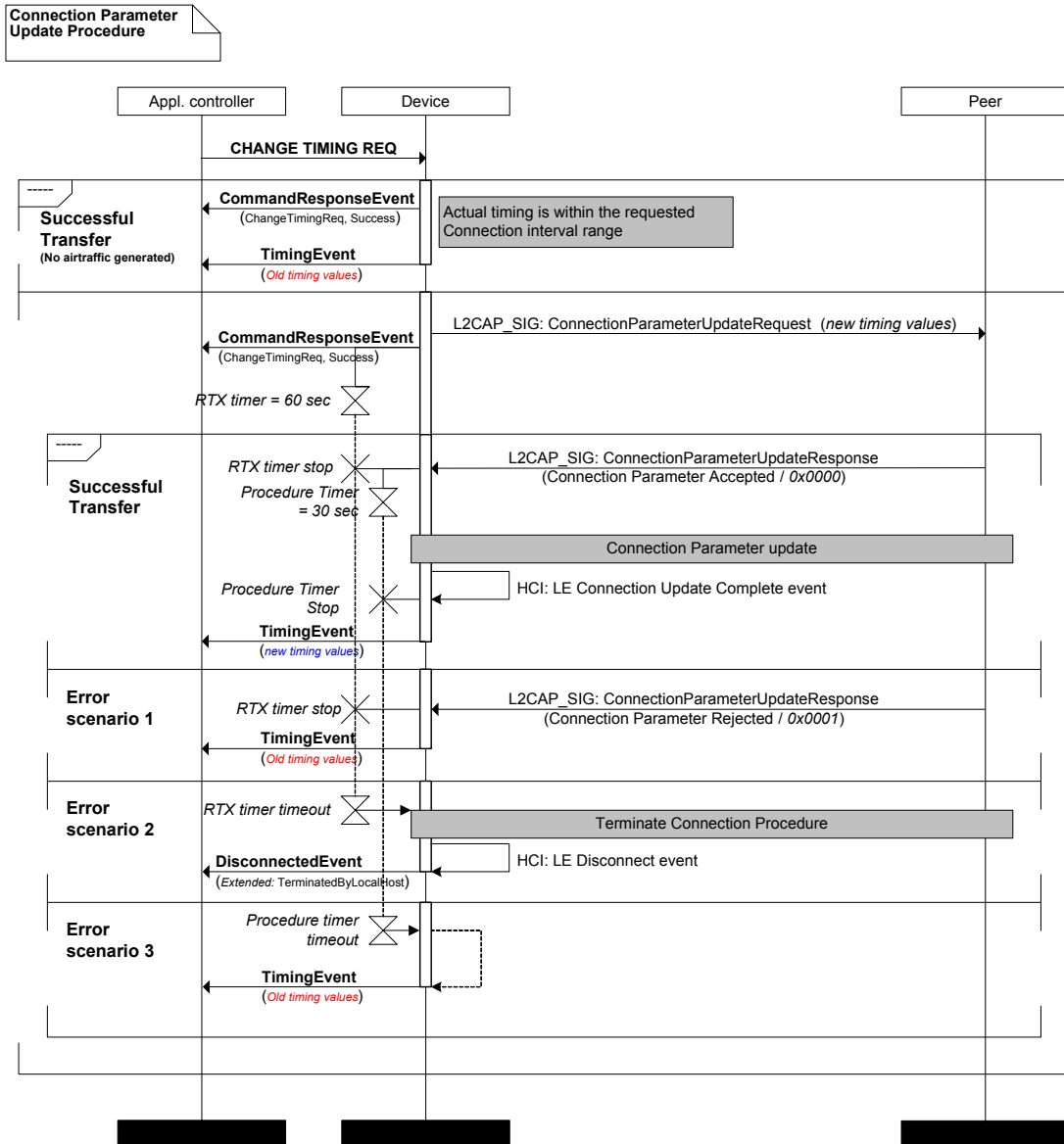


Figure 41. `ChangeTimingRequest` MSC

24.15.5 Bluetooth low energy procedures used

The following GAP procedures are used to change connection timing:

- Connection Parameter Update Procedure²⁵

²⁵ Bluetooth specification Ver 4.0, Vol. 3, Part C (GAP), Sect. 9.3.9

24.16 OpenRemotePipe (0x14)

OpenRemotePipe opens a receive pipe from a peer device for data transfer.

24.16.1 Functional description

This command is used to open pipes which are closed by default.

The Receive (Remote) pipe and the Receive with acknowledgment (Remote) pipe types are closed by default. Data cannot be received from the peer device on these pipes unless the pipes are opened using the OpenRemotePipe command.

This command can be used only after the service discovery procedure has successfully completed and resulted in a ConnectedEvent and PipeStatusEvent(s). The PipeStatusEvent will return a bitmap identifying the service pipes that needs opening before data transfer can take place. Only pipes identified in the PipesClosed can be opened using this command.

24.16.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---------------------------------|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Command | 1 | 0x14 | OpenRemotePipe |
| Content | | | |
| ServicePipeNumber | 1 | | ID of the service pipe to open. |

Table 53. ACL message structure for OpenRemotePipe

24.16.3 Accepted values

| Parameter | Data value | Description |
|-------------------|------------|---|
| ServicePipeNumber | 1..62 | Must be one of the pipes listed in the ClosedPipes bitmap returned in the PipeStatusEvent |

Table 54. Accepted values for parameters, OpenRemotePipe

24.16.4 Returned events

This command returns a CommandResponseEvent. It is then followed by a PipeStatusEvent reporting the result of the procedure and an updated pipe bitmap identifying the pipes available for data transfer.

In case of a failed procedure execution, a PipeErrorEvent is returned instead of the PipeStatusEvent.

Data returned in the CommandResponseEvent is:

- Command code: OpenRemotePipe
- Status: Success / Status code
- Response data:

See [Figure 29. on page 59](#) for an MSC illustrating the use of the OpenRemotePipe command.

24.16.5 Bluetooth low energy procedures used

This command uses the following GATT procedures²⁶:

- Write Characteristic Value (CCCD configuration)
- Characteristic Value Notification (pipe without acknowledgment feature)
- Characteristic Value Indication (pipe with acknowledgment feature)

²⁶ *Bluetooth* specification Ver 4.0, Vol. 3, Part G, Chapter 4.2

24.17 DtmCommand (0x03)

DtmCommand sends a Direct Test Mode command to the radio module through the ACI interface.

24.17.1 Functional description

This command allows DTM control through the ACI, as an alternative to the UART interface. The specified DTM operation is invoked and DTM events are returned in the format of a `CommandResponseEvent`.

24.17.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|------------------------------------|
| Header | | | |
| Length | 1 | 3 | Packet length |
| Command | 1 | 0x03 | DtmCommand |
| Content | | | |
| DtmCommand | 2 | | Direct Test Mode command (MSB/LSB) |

Table 55. ACI message structure for DtmCommand

24.17.3 Accepted values

| Parameter | Data value | Description |
|-------------------|---|--|
| <i>DtmCommand</i> | Refer to <i>Bluetooth</i> Specification, v. 4.0, Volume 6, Part F, Direct Test Mode, Sect.3.3.2 for a comprehensive list of valid DTM commands. | 2 byte DTM command (MSB/LSB) specifying the radio test operation. For transmitter tests, the vendor specific payload (PKT = 11) is implemented as a continuous unmodulated carrier signal at the specified frequency |

Table 56. Accepted values for parameters, DtmCommand

24.17.4 Returned events

This command returns a `CommandResponseEvent`.

Data returned in the `CommandResponseEvent` are:

- Command code: `DtmCommand`
- Status:
 - Success / Status code
- Response data (provided Status = Success):
 - DTM Event (2 bytes, MSB/LSB)²⁷

24.17.5 *Bluetooth* low energy procedures used

This command invokes the following *Bluetooth* low energy functionality:

- *Bluetooth* Specification, v. 4.0, Volume 6, Part F, Direct Test

²⁷. Refer to *Bluetooth* Specification, v. 4.0, Volume 6, Part F, Direct Test Mode, Sect.3.4, 'Events', for description of DTM event format and content

24.18 ReadDynamicData (0x07)

ReadDynamicData extracts nRF8001 dynamic data for storage in the application processor.

24.18.1 Functional description

This command reads the dynamic data from the nRF8001 volatile memory. The retrieved data can be stored in the application processor while power is disconnected from the nRF8001. The dynamic data is read out as a consecutive series of read dynamic data packets. The read cycle is repeated until all dynamic data has been retrieved.

When power is re-applied to nRF8001, the dynamic data can be reinstated by using the WriteDynamicData command. Once the dynamic data has been reinstated, the device status is restored to the same status valid at the time of performing the ReadDynamicData commands.

24.18.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|-----------------|
| Header | | | |
| Length | 1 | 1 | Packet length |
| Command | 1 | 0x07 | ReadDynamicData |

Table 57. ACL message structure for ReadDynamicData

24.18.3 Accepted values

None

24.18.4 Returned events

This command returns a `CommandResponseEvent`.

Data returned in the `CommandResponseEvent` is:

- Command code: `ReadDynamicData`
- Status:
 - Continue transaction / Transaction complete / Failure (See [section 28.1 on page 126](#) for details)
- Response data:
 - Sequence number (1 byte): Sequence number of the dynamic data packet. Dynamic data must be restored in the order set by the sequence number.
 - Dynamic data (1..27 bytes): Dynamic data

24.18.5 Bluetooth low energy procedures used

None

24.19 WriteDynamicData (0x08)

WriteDynamicData restores dynamic data to nRF8001 volatile memory.

24.19.1 Functional description

This command writes previously saved dynamic data back to the nRF8001 volatile memory. The dynamic data is written in a consecutive series of WriteDynamicData commands. The write cycle must be repeated until all dynamic data has been written to the nRF8001 volatile memory.

Once the dynamic data has been reinstated, the device status is restored to the same status valid at the time of performing the ReadDynamicData commands. For the device to be functional after restoring dynamic data, device setup²⁸ must have been performed before restoring unless static data is stored in non-volatile memory²⁹.

24.19.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 2..30 | Packet length |
| Command | 1 | 0x08 | WriteDynamicData |
| Content | | | |
| SequenceNumber | 1 | | Data packet sequence number. Data must be written in the same sequence as they were read using the ReadDynamicData command |
| SetupData | 1..27 | | 1..27 bytes of dynamic data extracted using the ReadDynamicData command |

Table 58.ACI message structure for WriteDynamicData

24.19.3 Accepted values

None

24.19.4 Returned events

This command returns a CommandResponseEvent.

Data returned in the CommandResponseEvent is:

- Command code: WriteDynamicData
- Status:
 - Transaction continue
 - Transaction complete
 - (Error; see [section 28.1 on page 126](#) for details)
- Response data: None

24.19.5 Bluetooth low energy procedures used

None

28. Refer to [section 22.3 on page 68](#)

29. Refer to nRFGo Studio; Setup-lock enabled

25 Data commands

Data commands are commands that either aim to transfer, or receive, application data when nRF8001 is in a connected state with a peer device.

25.1 SendData (0x15)

SendData sends data to a peer device through a transmit service pipe.

25.1.1 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 2..22 | Packet length |
| Command | 1 | 0x15 | SendData |
| Content | | | |
| ServicePipeNumber | 1 | | ID of the service transmit pipe through which data is sent |
| Data | 1..20 | | Application data |

Table 59. ACI message structure for SendData

25.1.2 Accepted values

| Parameter | Data value | Description |
|-------------------|------------|---|
| ServicePipeNumber | 1..62 | Must be a pipe listed in the OpenPipes bitmap returned in the PipeStatusEvent |
| Data | - | The data payload size must not exceed the maximum length defined in nRFGo Studio in the local service configuration (1..20 bytes) |

Table 60. Accepted values for parameters, SendData

25.1.3 Returned events

This command returns the DataCreditEvent. When using a transmit pipe with acknowledgment, this command returns a DataAckEvent and a DataCreditEvent. A PipeErrorEvent will occur if the transmission fails.

25.1.4 Bluetooth low energy procedures used

This command uses the following GATT procedures³⁰:

- Notifications (TX pipe, Local)
- Indications (TX pipe, Local, Ack)
- Read Characteristic Value / Read Using Characteristic UUID (TX pipe, Local, Ack)
- Write Without Response (Tx pipe, Remote)
- Write Characteristic Value (Tx pipe, Remote, Ack)
- Read Characteristic Value (RX pipe, Remote, Req)

30. Bluetooth specification Ver. 4.0, Vol. 3, Part G, Chapter 4.2

25.2 RequestData (0x17)

RequestData requests data from a peer device through a service receive pipe.

25.2.1 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Command | 1 | 0x17 | RequestData |
| Content | | | |
| ServicePipeNumber | 1 | | Service Pipe Number of the service receive pipe to request data from |

Table 61. ACI message structure for RequestData (0xA4)

25.2.2 Accepted values

| Parameter | Data value | Description |
|-------------------|------------|---|
| ServicePipeNumber | 1..62 | Must be a pipe listed in the OpenPipes bitmap returned in the PipeStatusEvent |

Table 62. Accepted values for parameters, RequestData

25.2.3 Returned events

This command returns `DataReceivedEvent` upon reception of the requested data. Alternatively a `PipeErrorEvent` is returned in case of transmission failure.

25.2.4 Bluetooth low energy procedures used

The following GATT procedures are used to receive data from a remote device³¹:

- Read Characteristic Value

31. Bluetooth specification Ver. 4.0, Vol. 3, Part G, Chapter 4.2

25.3 SetLocalData (0x0D)

SetLocalData sets a local Characteristic Value or Characteristic Descriptor.

25.3.1 Functional description

The SetLocalData command is used to set data stored in the local server (Server) through the associated service pipe. For local transmit pipes, this command does not trigger a Handle Value Notification or Handle Value Indication to the peer device.

25.3.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---|
| Header | | | |
| Length | 1 | 2..22 | Packet length |
| Command | 1 | 0x0D | SetLocalData |
| Content | | | |
| ServicePipeNumber | 1 | | ID of the service transmit pipe through which the data is set |
| Data | 0..20 | | Application data |

Table 63. ACI message structure for SetLocalData

25.3.3 Accepted values

| Parameter | Data value | Description |
|-------------------|------------|--|
| ServicePipeNumber | 1..62 | One of the available ServicePipeNumbers identified in the PipeStatusEvent |
| Data | | The data payload size must not exceed the maximum length defined in the local server (Server) (Limited to a maximum of 20 bytes in length) |

Table 64. Accepted values for parameters, SetLocalData

25.3.4 Returned events

This command returns a CommandResponseEvent. Data returned in the event is:

- Command code: SetLocalData
- Status: Success / Status code
- Response data: None

25.3.5 Bluetooth low energy procedures used

None

25.4 SendDataAck (0x16)

SendDataAck confirms reception of data from a peer device.

25.4.1 Functional description

This command is used in conjunction with the `DataReceivedEvent`. When data is received on a receive pipe with the acknowledgment feature enabled, the application processor shall respond with a `SendDataAck` command. When nRF8001 receives the `SendDataAck` command, a confirmation is sent to the peer device. This confirmation will be either `Handle Value Confirmation` if the data is stored locally or `Write Response` if data is stored remotely. See [section 20.4 on page 54](#) and [section 20.5 on page 58](#) for MSCs illustrating data transfer with acknowledgment.

25.4.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Command | 1 | 0x16 | SendDataAck |
| Content | | | |
| ServicePipeNumber | 1 | | Number of the service pipe on which the acknowledge is to be sent |

Table 65. ACI message structure for SendDataAck

25.4.3 Accepted values

| Parameter | Data value | Description |
|--------------------------------|------------|--|
| <code>ServicePipeNumber</code> | 1..62 | Must a pipe listed in the <code>OpenPipes</code> bitmap returned in the <code>PipeStatusEvent</code> |

Table 66. Accepted values for parameters, SendDataAck

25.4.4 Returned events

None

25.4.5 Bluetooth low energy procedures used

This command uses the following GATT procedures³²:

- Characteristic Value Indication
- Write Response
- Handle Value Confirmation

³². Bluetooth specification Ver. 4.0, Vol. 3, Part G, Chapter 4.2

26 System Events

System events are event packets that have been triggered by a predefined condition and are sent by nRF8001 to the application processor.

26.1 DeviceStartedEvent (0x81)

DeviceStartedEvent indicates reset recovery or a state change.

26.1.1 Functional description

This event is sent from nRF8001 to the external application processor when nRF8001 is reset or changing operating mode.

26.1.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|----------------------|
| Header | | | |
| Length | 1 | 4 | Packet length |
| Event | 1 | 0x81 | DeviceStartedEvent |
| Content | | | |
| OperatingMode | 1 | | Current device mode |
| HWEError | 1 | | Cause of the restart |
| DataCreditAvailable | 1 | | Available buffers |

Table 67. ACI message structure for DeviceStartedEvent

26.1.3 Returned values

| Parameter | Data value | Description |
|----------------------------|------------|---|
| <i>OperatingMode</i> | 0x01 | Test |
| | 0x02 | Setup |
| | 0x03 | Standby |
| <i>HWEError</i> | 0x00 | No error |
| | 0x01 | Fatal error |
| <i>DataCreditAvailable</i> | 00 | Number of DataCommand buffers available |

Table 68. Accepted values for parameters, DeviceStartedEvent

26.1.4 Bluetooth low energy procedures used

None

26.2 EchoEvent (0x82)

EchoEvent returns a copy of the Echo ACI message.

26.2.1 Functional description

This event returns an identical copy of the PDU sent using the `Echo` command in Test mode.

26.2.2 Message format

| Message field/ parameter | Value size (bytes) | Data value | Description |
|-----------------------------|-----------------------|------------|---------------|
| Header | | | |
| Length | 1 | 1..30 | Packet length |
| Event | 1 | 0x82 | EchoEvent |
| Content | | | |
| EchoMessage | 0..29 | | Echo data |

Table 69. ACI message structure for EchoEvent

26.2.3 Returned values

| Parameter | Data value | Description |
|--------------------|------------|--|
| <i>EchoMessage</i> | - | Message identical to the <i>Data</i> parameter content of the last <code>Echo</code> command |

Table 70. Accepted values for parameters, EchoEvent

26.2.4 Bluetooth low energy procedures used

None

26.3 HardwareErrorEvent (0x83)

DebugInfoEvent returns hardware error debug information.

26.3.1 Functional description

This event is sent from nRF8001 to the external application processor to provide debug information. In case of firmware failure this event follows the DeviceStartedEvent .

26.3.2 Message format

| Message field/ parameter | Value size (bytes) | Data value | Description |
|-----------------------------|-----------------------|------------|---|
| Header | | | |
| Length | 1 | 25 | Packet length |
| Event | 1 | 0x83 | HardwareErrorEvent |
| Content | | | |
| LineNumber | 2 | | Code line where firmware failed |
| FileName | 22 | | Name of the firmware file where the error occurred. |

Table 71. ACI message structure for HardwareError Event

26.3.3 Returned values

| Parameter | Data value | Description |
|------------|------------|-------------------------------------|
| LineNumber | | Code line where the firmware failed |
| FileName | | Zero-terminated string |

Table 72. Accepted values for parameters, HardwareErrorEvent

26.3.4 Bluetooth low energy procedures used

None

26.4 CommandResponseEvent (0x84)

CommandResponseEvent confirms reception or execution of an ACI command.

26.4.1 Functional description

This event is sent from nRF8001 to the external application processor in response to ACI commands.

26.4.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 3..30 | Packet length |
| Event | 1 | 0x84 | CommandResponseEvent |
| Content | | | |
| CommandOpCode | 1 | | OP code of the command to which the event responds |
| Status | 1 | | Status of the command execution |
| ResponseData | 0..27 | | Command-specific data |

Table 73. ACI message structure for CommandResponseEvent

26.4.3 Returned values

| Parameter | Data value | Description |
|----------------------|-------------|---|
| <i>CommandOpCode</i> | | Command OP code |
| <i>Status</i> | 0x00 | Success |
| | 0x01 - 0xFF | Status code. See section 28.1 on page 126 for a comprehensive list of status codes. |
| <i>ResponseData</i> | | See the specific ACI command description for a list of return parameters associated with the command. |

Table 74. Returned values for CommandResponseEvent

26.4.4 Bluetooth low energy procedures used

None

26.5 ConnectedEvent (0x85)

ConnectedEvent indicates that a connection has been established with a peer device

26.5.1 Functional description

This event is sent from nRF8001 to the external application processor upon connection establishment with a peer device.

26.5.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|----------------------------|--------------------|------------|---------------------------------------|
| Header | | | |
| Length | 1 | 15 | Packet length |
| Event | 1 | 0x85 | ConnectedEvent |
| Content | | | |
| <i>AddressType</i> | 1 | | Peer Address Type |
| <i>PeerAddress</i> | 6 | | Peer Device Address |
| <i>ConnectionInterval</i> | 2 | | Connection Interval setting (LSB/MSB) |
| <i>SlaveLatency</i> | 2 | | Slave latency setting (LSB/MSB) |
| <i>SupervisionTimeout</i> | 2 | | Supervision timeout period (LSB/MSB) |
| <i>MasterClockAccuracy</i> | 1 | | Master (peer device) clock accuracy |

Table 75. ACI message structure for ConnectedEvent (0x20)

26.5.3 Returned values

| Parameter | Data value | Description |
|----------------------------|------------------------------|---|
| <i>AddressType</i> | 0x01 | Public address |
| | 0x02 | Random Static Address |
| | 0x03 | Random Private Address (Resolvable) |
| | 0x04 | Random Private Address (Un-resolvable) |
| <i>ConnectionInterval</i> | - | Connection interval in seconds when multiplied with 1.25ms |
| <i>SlaveLatency</i> | 0..1000 (0x0000 - 0x03E8) | The number of consecutive connection events that the slave is not required to respond |
| <i>SupervisionTimeout</i> | - | Supervision timeout in seconds when multiplied with 10ms |
| <i>MasterClockAccuracy</i> | 0x00 | 500 ppm |
| | 0x01 | 250 ppm |
| | 0x02 | 150 ppm |
| | 0x03 | 100 ppm |
| | 0x04 | 75 ppm |
| | 0x05 | 50 ppm |
| | 0x06 | 30 ppm |
| | 0x07 | 20 ppm |

Table 76. Returned values for ConnectedEvent

26.5.4 Bluetooth low energy procedures used

None

26.6 DisconnectedEvent (0x86)

DisconnectedEvent indicates the loss of a connection.

26.6.1 Functional description

This event is sent from nRF8001 to the external application processor to notify the application processor that connection with the peer device has been lost.

26.6.2 Message format

| Message field/ parameter | Value size (bytes) | Data value | Description |
|-----------------------------|-----------------------|------------|--|
| Header | | | |
| Length | 1 | 3 | Packet length |
| Event | 1 | 0x86 | DisconnectedEvent |
| Content | | | |
| <i>AciStatus</i> | 1 | | Reason for disconnection (Local Host origin) |
| <i>BtLeStatus</i> | 1 | | Reason for disconnection, <i>Bluetooth</i> low energy status code (Origin not related to local Host) |

Table 77. ACI message structure for DisconnectedEvent

26.6.3 Returned values

| Parameter | Data value | Description |
|-------------------|------------|---|
| <i>AciStatus</i> | 0x03 | Check the <i>Bluetooth</i> low energy status code; <i>BtLeStatus</i> |
| | 0x93 | Timeout while advertising, unable to establish connection |
| | 0x8D | Bond required to proceed with connection |
| <i>BtLeStatus</i> | 0x00 | n/a |
| | 0x01..0xFF | See the <i>Bluetooth</i> specification, v. 4.0, Volume 6, Part D, Error Code Description for a complete list of error codes |

Table 78. Returned values for DisconnectedEvent

26.6.4 Bluetooth low energy procedures used

None

26.7 BondStatusEvent (0x87)

BondStatusEvent returns the bonding procedure execution status

26.7.1 Functional description

This event is sent from nRF8001 to the application processor upon successful execution of the bonding procedure. In case of a failed bonding procedure, a `DisconnectedEvent` will result instead of a `BondStatusEvent`.

26.7.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|---------------------------------|--------------------|------------|-------------------------|
| Header | | | |
| Length | 1 | 7 | Packet length |
| Event | 1 | 0x87 | BondStatusEvent |
| Content | | | |
| <i>BondStatusCode</i> | 1 | | Bond Status code |
| <i>BondStatusSource</i> | 1 | | Bond Status source |
| <i>BondStatus-SecMode1</i> | 1 | | LE security mode 1 |
| <i>BondStatus-SecMode2</i> | 1 | | LE security mode 2 |
| <i>BondStatus-KeyExchSlave</i> | 1 | | Keys exchanged (slave) |
| <i>BondStatus-KeyExchMaster</i> | 1 | | Keys exchanged (master) |

Table 79. ACI message structure for BondStatusEvent

26.7.3 Returned values

| Parameter | Data value | Description |
|---------------------------------|------------|---|
| <i>BondStatusCode</i> | 0x00 | Bond succeeded |
| | 0x01..0xFF | Bond Failed, see section 28.2 on page 127 for more information |
| <i>BondStatusSource</i> | 0x01 | Status code generated locally |
| | 0x02 | Status code generated by the remote peer |
| <i>BondStatus-SecMode1</i> | - | Levels supported in LE Security Mode 1 <ul style="list-style-type: none"> • bit0: level 1 • bit1: level 2 • bit2: level 3 • bit3..7: <i>reserved</i> |
| <i>BondStatus-SecMode2</i> | - | Levels supported in LE Security Mode 2 <ul style="list-style-type: none"> • bit0: level 1 • bit1: level 2 • bit2..7: <i>reserved</i> |
| <i>BondStatus-KeyExchSlave</i> | - | Keys exchanged (slave distributed keys) <ul style="list-style-type: none"> • bit0: LTK using Encryption Information command • bit1: EDIV and Rand using Master Identification command • bit2: IRK using Identity Information command • bit3: Public device or static random address using Identity Address Information command • bit4: CSRK using Signing Information command • bit5..7: <i>reserved</i> |
| <i>BondStatus-KeyExchMaster</i> | - | Keys exchanged (master distributed keys) <ul style="list-style-type: none"> • bit0: LTK using Encryption Information command • bit1: EDIV and Rand using Master Identification command • bit2: IRK using Identity Information command • bit3: Public device or static random address using Identity Address Information command • bit4: CSRK using Signing Information command • bit5..7: <i>reserved</i> |

Table 80. Returned values for *BondStatusEvent*

26.7.4 Bluetooth low energy procedures used

None

26.8 PipeStatusEvent (0x88)

PipeStatusEvent lists the pipe connection and availability status

26.8.1 Functional description

This event is sent from nRF8001 to the external application processor whenever there is a change in service pipe availability status.

The `PipeStatusEvent` returns two pipe lists in the form of two 64-bit bitmaps:

`PipesOpen`: Available pipes on which data can be received (or transmitted) without further action

`PipesClosed`: Pipes on which data can be received only after nRF8001 has instructed the Client (peer device) to send data. These pipes are opened using the `OpenRemotePipe` command.

See [section 20.6 on page 60](#) for a functional description on pipe availability.

Each bit in the bitmaps corresponds to a service pipe. For the `PipesOpen` bitmap, a bit is set to '1' indicates that the service pipe is available, when set to '0' it is unavailable. For the `PipesClosed` bitmap, a bit is set to '1' indicates that the service pipe requires opening, when set to '0' no action is required.

The service pipes are counted from the first byte starting with the least significant bit.

Example:

PipesOpen bitmap = 0xFEFF0D0000000800

The nRF8001 service discovery procedure execution has not completed.
The following service pipes are open; Pipes 1 through 16, 18, 19 and 51.

PipesClosed bitmap = 0x0000821100100000

Service pipes 17, 23, 24, 28 and 44 require opening.

Table 81. illustrates the bitmap to service pipe mapping for the example.

| Bitmap bytes | Byte 1 | | | | | | | | Byte 2 | | | | | | | |
|---------------------------|--------|----|----|----|----|----|----|----|--------|----|----|----|----|----|----|----|
| Service pipe number | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PipesOpen (bits) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PipesOpen (byte values) | 0xFE | | | | | | | | 0xFF | | | | | | | |
| PipesClosed (bits) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PipesClosed (byte values) | 0x00 | | | | | | | | 0x00 | | | | | | | |
| Bitmap bytes | Byte 3 | | | | | | | | Byte 4 | | | | | | | |
| Service pipe number | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| PipesOpen (bits) | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PipesOpen (byte values) | 0x0D | | | | | | | | 0x00 | | | | | | | |
| PipesClosed (bits) | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| PipesClosed (byte values) | 0x82 | | | | | | | | 0x11 | | | | | | | |
| Bitmap bytes | Byte 5 | | | | | | | | Byte 6 | | | | | | | |
| Service pipe number | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| PipesOpen (bits) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PipesOpen (byte values) | 0x00 | | | | | | | | 0x00 | | | | | | | |
| PipesClosed (bits) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PipesClosed (byte values) | 0x00 | | | | | | | | 0x10 | | | | | | | |
| Bitmap bytes | Byte 7 | | | | | | | | Byte 8 | | | | | | | |
| Service pipe number | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 |
| PipesOpen (bits) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PipesOpen (byte values) | 0x08 | | | | | | | | 0x00 | | | | | | | |
| PipesClosed (bits) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PipesClosed (byte values) | 0x00 | | | | | | | | 0x00 | | | | | | | |

Table 81. Bitmap returned by PipeStatusEvent (Example)

26.8.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 17 | Packet length |
| Event | 1 | 0x88 | PipeStatusEvent |
| Content | | | |
| PipesOpen | 8 | | Bitmap of open pipes, 1...62 |
| PipesClosed | 8 | | Bitmap of pipes that will require opening (OpenRemotePipe) before they are operational, 1...62 |

Table 82. ACI message structure for PipeStatusEvent

26.8.3 Returned values

| Parameter | Data value | Description |
|--------------------|------------|--|
| <i>PipesOpen</i> | - | <p>Bitmap where each of the bits 1 to 62 represents the pipes with the number 1 to 31. Bit 63 is not in use. A "1" means that the corresponding pipe is open, while a "0" means that the pipe is unavailable.</p> <p>Bit 0 in the first byte contains the nRF8001 service discovery procedure execution status:</p> <ul style="list-style-type: none"> • When set to 1, the nRF8001 initiated service discovery procedure has completed. • When set to 0, the nRF8001 initiated service discovery has not yet completed¹. |
| <i>PipesClosed</i> | - | <p>Bitmap where each of the bits 1 to 62 represents the pipes with the number 1 to 62. Bit 63 is not in use. A "1" means that the corresponding pipe requires opening, while a "0" means that no action is required. Bit 0 in the first byte contains is always set to "0"</p> |

1. If service discovery is not required for nRF8001 (based on the existing service configuration), Bit 0 in the first bitmap byte is set to 1.

Table 83. Returned values for PipeStatusEvent

26.8.4 Bluetooth low energy procedures used

None

26.9 TimingEvent (0x89)

TimingEvent returns the current connection timing information upon change of parameters.

26.9.1 Functional description

This event is sent from nRF8001 to the external application processor when nRF8001 connects to a peer device or when the connection parameters are updated by a device in the central role.

26.9.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|---------------------------|--------------------|------------|--|
| Header | | | |
| Length | 1 | 7 | Packet length |
| Event | 1 | 0x89 | TimingEvent |
| Content | | | |
| <i>ConnectionInterval</i> | 2 | | Connection interval for the actual connection (MSB first) |
| <i>SlaveLatency</i> | 2 | | Slave latency setting (LSB/MSB) |
| <i>SupervisionTimeout</i> | 2 | | Supervision timeout for the connection (multiple of 10 ms) |

Table 84. ACI message structure for TimingEvent

26.9.3 Returned values

| Parameter | Data value | Description |
|---------------------------|------------------------------|---|
| <i>ConnectionInterval</i> | 6..3200 | Connection interval = data value multiplied by 1,25ms |
| <i>SlaveLatency</i> | 0..1000 (0x0000 - 0x03E8) | The number of consecutive connection events that the slave is not required to respond |
| <i>SupervisionTimeout</i> | 10..3200 | Timeout = data value multiplied by 10ms |

Table 85. Returned values for TimingEvent

26.9.4 Bluetooth low energy procedures used

None

27 Data Events

Data events are information packets related to application data transfer sent from nRF8001 to the application processor.

27.1 DataCreditEvent (0x8A)

DataCreditEvent returns data command buffer credits.

27.1.1 Functional description

This returns the number of data command buffer locations (credits) freed as a result of successful data command execution(s).

27.1.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|------------------|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Event | 1 | 0x8A | DataCreditEvent |
| Content | | | |
| DataCredits | 1 | | Returned credits |

Table 86. ACI message structure for DataCreditEvent

27.1.3 Returned values

| Parameter | Data value | Description |
|--------------------|------------|---|
| <i>DataCredits</i> | | The number of data credits returned to the application processor. |

Table 87. Returned values for DataCreditEvent

27.1.4 Bluetooth low energy procedures used

None

27.2 PipeErrorEvent (0x8D)

PipeErrorEvent reports a pipe transmission failure/error.

27.2.1 Functional description

This event is sent from nRF8001 to the external application processor in the case of transmission failure.

27.2.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---|
| Header | | | |
| Length | 1 | 3..30 | Packet length |
| Event | 1 | 0x8D | PipeErrorEvent |
| Content | | | |
| <i>ServicePipeNo</i> | 1 | | Pipe number of the pipe of which the error occurred |
| <i>ErrorCode</i> | 1 | | Status error code |
| <i>ErrorData</i> | 0..27 | | Optional error data |

Table 88. ACI message structure for PipeErrorEvent

27.2.3 Returned values

| Parameter | Data value | Description |
|----------------------|-------------|---|
| <i>ServicePipeNo</i> | 1..62 | Refers to a valid pipe number listed in the OpenPipes/ClosedPipes bitmaps returned in the PipeStatusEvent |
| <i>ErrorCode</i> | 0x01 – 0xFF | Status error code. See the Appendix, Sect. 11.1, 'ACI error codes' for a comprehensive list of error codes. |
| <i>ErrorData</i> | - | Optional error data, 0 to 27 bytes depending on error code. |

Table 89. Returned values for parameters, PipeErrorEvent

27.2.4 Bluetooth low energy procedures used

None

27.3 DataReceivedEvent (0x8C)

DataReceivedEvent indicates that data has been received from the peer device

27.3.1 Functional description

This event is sent from nRF8001 to the external application processor when new data is received on a receive service pipe.

27.3.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---|
| Header | | | |
| Length | 1 | 2..22 | Packet length |
| Event | 1 | 0x8C | DataReceivedEvent |
| Content | | | |
| ServicePipeNo | 1 | | ID of the service pipe through which data is received |
| Data | 0..20 | | Application data |

Table 90. ACI message structure for DataReceivedEvent

27.3.3 Returned values

| Parameter | Data value | Description |
|----------------------|------------|--|
| <i>ServicePipeNo</i> | 1..62 | Refers to a valid pipe number listed in the OpenPipes bitmap returned in the PipeStatusEvent |
| <i>Data</i> | n | The data payload size must not exceed the maximum length defined in the local server |

Table 91. Returned values for DataReceivedEvent

27.3.4 Bluetooth low energy procedures used

None

27.4 DataAckEvent (0x8B)

DataAckEvent indicates reception of data by the peer device

27.4.1 Functional description

This event is sent from nRF8001 to the application processor when an acknowledgment is received from the peer device in response to data sent to it.

27.4.2 Message format

| Message field/parameter | Value size (bytes) | Data value | Description |
|-------------------------|--------------------|------------|---|
| Header | | | |
| Length | 1 | 2 | Packet length |
| Event | 1 | 0x8B | DataAckEvent |
| Content | | | |
| ServicePipeNumber | 1 | | Pipe number of the pipe on which the Ack was received |

Table 92. ACI message structure for DataAckEvent

27.4.3 Returned values

| Parameter | Data value | Description |
|--------------------------|------------|--|
| <i>ServicePipeNumber</i> | 1..62 | Refers to a valid pipe number listed in the OpenPipes bitmap returned in the PipeStatusEvent |

Table 93. Returned values for DataAckEvent

27.4.4 Bluetooth low energy procedures used

None

28 Appendix

28.1 ACI Status Codes

[Table 94](#) lists the generic ACI status codes applicable for nRF8001. The status code is used to indicate the general command execution status or to identify the cause of an error.

| Status code | Name | Description |
|-------------|---|--|
| 0x00 | ACI_STATUS_SUCCESS | Success |
| 0x01 | ACI_STATUS_TRANSACTION_CONTINUE | Transaction continuation status |
| 0x02 | ACI_STATUS_TRANSACTION_COMPLETE | Transaction completed |
| 0x03 | ACI_STATUS_EXTENDED | Extended status, further checks needed |
| 0x80 | ACI_STATUS_ERROR_UNKNOWN | Unknown error |
| 0x81 | ACI_STATUS_ERROR_INTERNAL | Internal error |
| 0x82 | ACI_STATUS_ERROR_CMD_UNKNOWN | Unknown command |
| 0x83 | ACI_STATUS_ERROR_DEVICE_STATE_INVALID | Command invalid in the current device state |
| 0x84 | ACI_STATUS_ERROR_INVALID_LENGTH | Invalid length |
| 0x85 | ACI_STATUS_ERROR_INVALID_PARAMETER | Invalid input parameters |
| 0x86 | ACI_STATUS_ERROR_BUSY | Busy |
| 0x87 | ACI_STATUS_ERROR_INVALID_DATA | Invalid data format or contents |
| 0x88 | ACI_STATUS_ERROR_CRC_MISMATCH | CRC mismatch |
| 0x89 | ACI_STATUS_ERROR_UNSUPPORTED_SETUP_FORMAT | Unsupported setup format |
| 0x8A | ACI_STATUS_ERROR_INVALID_SEQ_NO | Invalid sequence number during a write dynamic data sequence |
| 0x8B | ACI_STATUS_ERROR_SETUP_LOCKED | Setup data is locked and cannot be modified |
| 0x8C | ACI_STATUS_ERROR_LOCK_FAILED | Setup error due to lock verification failure |
| 0x8D | ACI_STATUS_ERROR_BOND_REQUIRED | Bond required: Local Pipes need bonded/trusted peer |
| 0x8E | ACI_STATUS_ERROR_REJECTED | Command rejected as a transaction is still pending |
| 0x8F | ACI_STATUS_ERROR_DATA_SIZE | Pipe Error Event : Data size exceeds size specified for pipe, Transmit failed |
| 0x90 | ACI_STATUS_ERROR_PIPE_INVALID | Pipe Error Event : Transmit failed, Invalid or unavailable Pipe number or unknown pipe type |
| 0x91 | ACI_STATUS_ERROR_CREDIT_NOT_AVAILABLE | Pipe Error Event : Credit not available |
| 0x92 | ACI_STATUS_ERROR_PEER_ATT_ERROR | Pipe Error Event : Peer device has sent an error on an pipe operation on the remote characteristic |
| 0x93 | ACI_STATUS_ERROR_ADVT_TIMEOUT | Connection was not established before the BTLE advertising was stopped |
| 0x94 | ACI_STATUS_ERROR_PEER_SMP_ERROR | Remote device triggered a Security Manager Protocol error |

Table 94. nRF8001 ACI Status codes

28.2 Bonding Status Codes

[Table 95](#) lists the status codes applicable for the `BondStatusEvent`. The status code is used to report the bonding procedure execution status.

| Bond status code | Name | Description |
|------------------|---|--|
| 0x00 | ACI_BOND_STATUS_SUCCESS | Bonding succeeded |
| 0x01 | ACI_BOND_STATUS_FAILED | Bonding failed |
| 0x02 | ACI_BOND_STATUS_FAILED_TIMED_OUT | Bonding error: Timeout while bonding |
| 0x81 | ACI_BOND_STATUS_FAILED_PASSKEY_ENTRY_FAILED | Bonding error: Passkey entry failed |
| 0x82 | ACI_BOND_STATUS_FAILED_OOB_UNAVAILABLE | Bonding error: OOB unavailable |
| 0x83 | ACI_BOND_STATUS_FAILED_AUTHENTICATION_REQ | Bonding error: Authentication request failed |
| 0x84 | ACI_BOND_STATUS_FAILED_CONFIRM_VALUE | Bonding error: Confirm value failed |
| 0x85 | ACI_BOND_STATUS_FAILED_PAIRING_UNSUPPORTED | Bonding error: Pairing unsupported |
| 0x86 | ACI_BOND_STATUS_FAILED_ENCRYPTION_KEY_SIZE | Bonding error: Invalid encryption key size |
| 0x87 | ACI_BOND_STATUS_FAILED_SMP_CMD_UNSUPPORTED | Bonding error: Unsupported SMP command |
| 0x88 | ACI_BOND_STATUS_FAILED_UNSPECIFIED_REASON | Bonding error: Unspecified reason |
| 0x89 | ACI_BOND_STATUS_FAILED_REPEATED_ATTEMPTS | Bonding error: Too many attempts |
| 0x8A | ACI_BOND_STATUS_FAILED_INVALID_PARAMETERS | Bonding error: Invalid parameters |

Table 95. Bonding status codes

28.3 Error Codes

[Table 96](#) lists the status codes applicable for the `PipeErrorEvent`. The error code is used to report an error related to data transfer or service pipe association.

Refer to the *Bluetooth* specification for the latest set of error codes.³³

| Bond status code | Name | Description |
|----------------------------------|-----------|---|
| Invalid Handle | 0x01 | The attribute handle given was not valid on this server |
| Read Not Permitted | 0x02 | The attribute cannot be read. |
| Write Not Permitted | 0x03 | The attribute cannot be written. |
| Invalid PDU | 0x04 | The attribute PDU was invalid. |
| Insufficient Authentication | 0x05 | The attribute requires authentication before it can be read or written. |
| Request Not Supported | 0x06 | Attribute server does not support the request received from the client. |
| Invalid Offset | 0x07 | Offset specified was past the end of the attribute. |
| Insufficient Authorization | 0x08 | The attribute requires authorization before it can be read or written. |
| Prepare Queue Full | 0x09 | Too many prepare writes have been queued. |
| Attribute Not Found | 0x0A | No attribute found within the given attribute handle range. |
| Attribute Not Long | 0x0B | The attribute cannot be read or written using the Read Blob Request |
| Insufficient Encryption Key Size | 0x0C | The Encryption Key Size used for encrypting this link is insufficient. |
| Invalid Attribute Value Length | 0x0D | The attribute value length is invalid for the operation. |
| Unlikely Error | 0x0E | The attribute request that was requested has encountered an error that was unlikely, and therefore could not be completed as requested. |
| Insufficient Encryption | 0x0F | The attribute requires encryption before it can be read or written. |
| Unsupported Group Type | 0x10 | The attribute type is not a supported grouping attribute as defined by a higher layer specification. |
| Insufficient Resources | 0x11 | Insufficient Resources to complete the request |
| Application Error | 0x80-0xFF | Application error code defined by a higher layer specification. |

Table 96. Error codes

33. *Bluetooth* specification Ver. 4.0, Vol. 3, Part F, Chapter 3.4, Table 3.3, 'Error codes'

28.4 Setup and procedure selection

Figure 42. illustrates the required setup and decision process required prior to application data transfer.

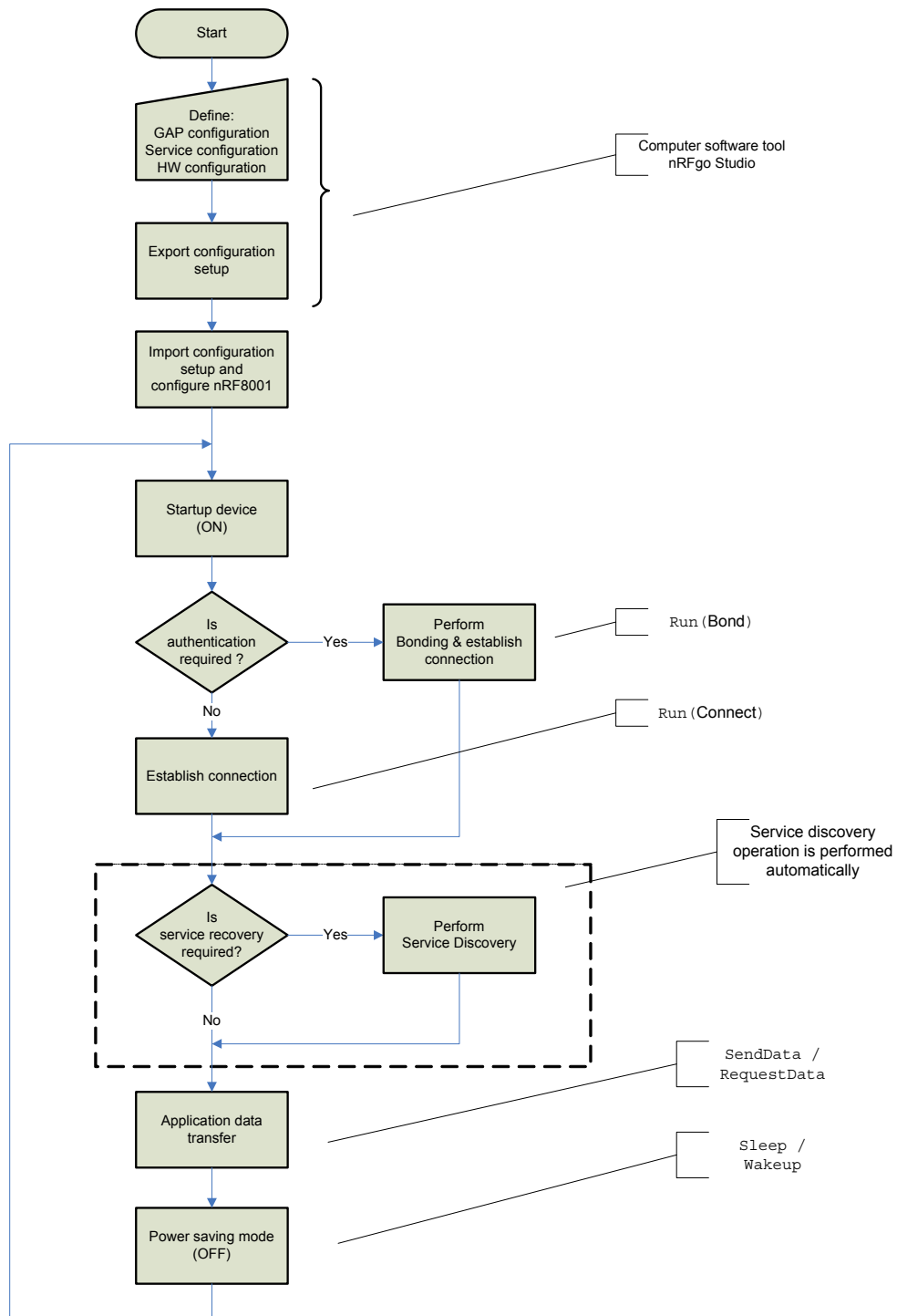


Figure 42. Normal configuration and connection establishment procedure (example)