BLE Peripheral

Contents

nRF5 SDK v11.0.0-2.alpha

BLE Peripheral

This information applies to the following SoftDevices: **S130, S132**

The following examples demonstrate the BLE Peripheral role:

Alert Notification Application

Apple Notification Center Service (ANCS) Client Application

Beacon Transmitter Sample Application

Blood Pressure Application

Current Time Application

Cycling Speed and Cadence Application

Glucose Application

Health Thermometer Application

Heart Rate Application

Heart Rate Application with FreeRTOS

Heart Rate Application with RTX

HID Keyboard Application

HID Mouse Application

Multiprotocol Application nRF51 only

Power Profiling Application

Proximity Application

Running Speed and Cadence Application

Template Application

UART/Serial Port Emulation over BLE

Experimental: BLE Blinky Application

Experimental: Bluetooth Developer Studio Example

Experimental: Eddystone Beacon Application

Experimental: Heart Rate Example with pairing over NFC nRF52 only

Experimental: Location and Navigation Application

Experimental: Multi Activity Example

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

1. nRF5 SDK v11.0.0-2.alpha

Alert Notification Application

This example requires one of the following SoftDevices: **S130, S132**

Important: Before you run this example, make sure to program a SoftDevice.

The Alert Notification Application is an example that implements the client role of the Alert Notification Profile using the hardware delivered in the nRF5 Development Kit.

The application includes one mandatory client needed for the Alert Notification Profile:

- Alert Notification Service Client

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and 30 seconds and go to system-off mode. Push the button 1 to wake it up and to make it restart advertising.

Setup

The name of the example is **ble_app_alert_notification_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\ble_app_alert_notification`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During Connection:
  - Button 1: Check the current state of new alert and enable, disable, or clear the alert depending on the state.
  - Button 2: Check the current state of unread alert and enable, disable, or clear the alert depending on the state.
  - Button 3: Request peer to notify all current alert statuses.

The Alert Notification Application does fast advertising for 30 seconds at power up and also after a disconnection. It will then switch to slow advertising mode for 180 seconds. After 180 seconds of slow advertising, the board will go to system-off.

When a new alert notification is received, this is indicated by the state BSP_INDICATE_ALERT_0. When an unread alert notification is received, this is indicated by the state BSP_INDICATE_ALERT_1. The alert state can be cleared by pressing the corresponding button.

Testing

Test the Alert Notification Application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. In the Master Control Panel, create the Alert Notification Service (ANS) server by performing the following steps:
   a. Click File -> Server Setup. In the window that opens up, click File -> Load setup.
   b. In the resulting window, navigate to the `<InstallFolder>\examples\ble_peripheral\ble_app_alert_notification` folder. If you are using Keil packs, this folder is located at `<KeilFolder>\ARM\Pack\NordicSemiconductor\nRF_Examples\<version>\ble_peripheral\ble_app_alert_notificatio` Load the file `AlertNotifServer.bin`.
   c. Click 'Stop server' and then 'Start server'.
   d. Set the values of characteristics with UUID 0x2A47 and 0x2A48 to 0xFF.
   e. You must keep the 'Server Setup' window opened.
3. Select the device from Master Control Panel (the device will be advertising as 'Nordic_Alert_Notif'), then connect and perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
4. Upon connection, verify in the 'Server Setup' that the Client Characteristic Configuration descriptors (CCCD) for 0x2A45 and 0x2A46 are set to 0001.
5. Upon connection, verify in the 'Server Setup' that the Alert Notification Control Point, 0x2A44, is written with the values 0x01 0x03.
6. Set the value of characteristic 0x2A46 to '01-02' in the 'Server Setup' and press 'update'. Verify that the BSP_INDICATE_ALERT_0 state is indicated.
7. Press button 1 one time. Verify that the BSP_INDICATE_ALERT_0 state is cleared.
8. Press button 1 again. Verify in 'Server Setup' that the CCCD for 0x2A46 is set to 0000.
9. Set the value of characteristic 0x2A46 to '01-03' in the 'Server Setup' and press 'update'. Verify that the BSP_INDICATE_ALERT_0 state is cleared.
10. Press button 1 again. Verify in 'Server Setup' that the CCCD for 0x2A46 is set to 0001. Verify that the BSP_INDICATE_ALERT_0 state is cleared.
11. Set the value of characteristic 0x2A46 to '01-03' in the 'Server Setup' and press 'update'. Verify that the BSP_INDICATE_ALERT_0 state is indicated.
12. Set the value of characteristic 0x2A45 to '01-02-41-42-43' in the 'Server Setup' and press 'update'. Verify that the BSP_INDICATE_ALERT_1 state is indicated.
13. Press button 2 one time. Verify that the BSP_INDICATE_ALERT_1 state is cleared.
14. Press button 2 again. Verify in 'Server Setup' that the CCCD for 0x2A45 is set to 0000.
15. Disconnect from the Master Control Panel and observe that the BSP_INDICATE_ADVERTISING state is indicated.
16. Wait until the application goes to system-off (BSP_INDICATE_IDLE).
17. Press button 1 and connect to the Master Control Panel. Observe that the BSP_INDICATE_CONNECTED state is indicated.
18. Disconnect from the Master Control Panel.

Apple Notification Center Service (ANCS) Client Application

*This example requires one of the following SoftDevices: **S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The ANCS Client Application is an example that implements an Apple Notification Center Service client. This client receives iOS notifications and is therefore a Notification Consumer. It can be connected with a Notification Provider, typically an iPhone or some other Apple device, which functions as ANCS server.

For detailed information about the Apple Notification Center Service, see Apple's iOS Developer Library.

When the application is connected to a Notification Provider, it receives and prints incoming notifications on the UART. Pressing Button 2 requests the attributes of the last received notification. For example, if the notification indicates a new email, notification attributes contain the title, the message, the date, and so on. If any attributes are received, they are printed to the UART.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push Button 1 to restart advertising.

Setup

The name of the example is **ble_app_ancs_c_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_ancs_c`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During connection:
  - Button 2: Send iOS notification message attributes (content) on UART.

Testing

The ANCS Client Application can be tested with an iOS device (for example, an iPhone) or by using Master Control Panel.

To test using an iPhone:

1. Start a COM listener like PUTTY and connect to the used COM port with the following UART settings:
   - Baud rate: **115.200** (Note that this value differs from the baud rate used in most other examples.)
   - 8 data bits
   - 1 stop bit
   - No parity
   - HW flow control: RTS/CTS
2. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
3. Select the device in the iOS settings -> Bluetooth menu and connect.
4. Observe that the BSP_INDICATE_CONNECTED state is indicated.
5. Observe that all notifications that are displayed in the iOS notification tab also show up on the UART from the device.

You can also test the application with the Master Control Panel by performing the following steps:

1. Start a COM listener like PUTTY and connect to the used COM port with the following UART settings:
   - Baud rate: **115.200** (Note that this value differs from the baud rate used in most other examples.)
   - 8 data bits
   - 1 stop bit
   - No parity
   - HW flow control: RTS/CTS
2. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
3. In the Master Control Panel, create the Apple Notification Center Service (ANCS) server by completing the following steps:
   - Click File -> Server setup. In the window that opens up, click File -> Load setup.
   - In the resulting window, navigate to the `<InstallFolder>\examples\ble_peripheral\ble_app_ancs_c` folder. If you are using Keil packs, this folder is located at `<KeilFolder>\ARM\Pack\NordicSemiconductor\nRF_Examples\<version>\ble_peripheral\ble_app_ancs_c`. Load the file `ANCS_central.bin`.
   - Stop the server, and then start it again.
   - You must keep the Server setup window open.
4. Select the device from Master Control Panel (the device will be advertising as 'ANCS').
5. Connect and bond.
6. Observe that the BSP_INDICATE_CONNECTED state is indicated.
7. Observe that the server is started.
8. After bonding, verify in the Server setup that the Client Characteristic Configuration descriptors (CCCD) for 0x120D and 0xC6E9 are set to 0001.
9. Send an iOS notification to the application: In the Server setup, set the value of Notification Source (characteristic 0x120D) to 00-01-06-02-01-02-03-04 and click Update.
10. Verify that the UART data is received as follows:

```
Notification
Event:       Added
Category ID: Email
Category Cnt:2
UID:         67305985
Flags:
 Silent
```

11. Press Button 2 to request notification attributes for the iOS notification received before.
12. In the Server setup, verify that the CCCD for 0xD8F3 is updated to 00-01-02-03-04-01-20-00-02-20-00-03-20-00-04-05-06-07.
13. Respond to the get attribute request by sending two notification attributes, Title and Message. To do this, set the Data Source (Characteristic 0xC6E9) in the Server setup to 00-01-02-03-04-01-03-00-6E-52-46-03-03-00-34-32-32.
14. Verify that the UART data is received as follows:

```
Title: nRF

Message: 422
```

15. Disconnect from the Master Control Panel and observe that the BSP_INDICATE_ADVERTISING_WHITELIST state is indicated.
16. Wait until the application goes to system-off (BSP_INDICATE_IDLE).
17. Press Button 1 and connect to the Master Control Panel. Observe that the BSP_INDICATE_CONNECTED state is indicated.
18. Disconnect from the Master Control Panel.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

Beacon Transmitter Sample Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Beacon Transmitter Sample Application is an example that implements a transmitter beacon using the hardware delivered in the nRF5 Development Kit.

The beacon broadcasts information to all compatible devices in its range as Manufacturer Specific Data in the advertisement packets.

This information includes:

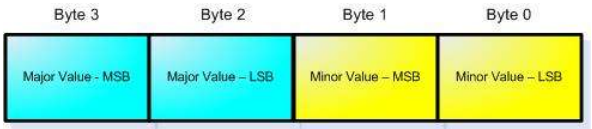
- A 128-bit UUID to identify the beacon's provider.
- An arbitrary Major value for coarse differentiation between beacons.
- An arbitrary Minor value for fine differentiation between beacons.
- The RSSI value of the beacon measured at 1 meter distance, which can be used for estimating the distance from the beacon.

Note
This application is not power optimized!

The name of the example is **ble_app_beacon_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_beacon`

Configuring Major and Minor values

This example implements an optional feature which allows the change of Major and Minor values used in the advertisement packets without the need to recompile the application each time. To use this feature, the compiler define USE_UICR_FOR_MAJ_MIN_VALUES should be defined during compilation. If this is done, the application uses the value of the UICR (User Information Configuration Register) located at address 0x10001080 to populate the major and minor values. Whenever the values need to be updated, the user must set the UICR to a desired value using the nrfjprog tool and restart the application to bring the changes into effect. The byte ordering used when decoding the UICR value is shown in the image below.



Byte ordering used while reading UICR

See Testing for more information about how to use this feature.

Note
This application can be used as a starting point to write an iBeacon application. The procedure for creating an iBeacon application and the specification should be available at mfi.apple.com. Once the specification is obtained, this application can be modified to fit the requirements of iBeacon.

Testing

Test the Beacon Transmitter Sample Application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. After starting discovery in the Master Control Panel main window, observe that the beacon is advertising with its *Bluetooth* device address without a Device Name.

3. Click the plus sign next to the beacon's address to view the full advertisement data.
4. Observe that the **Advertising Type** is 'Non-connectable' and the **Manufacturer Specific Data** field of the Advertising Data is as follows:

   59-00-02-15-01-12-23-34-45-56-67-78-89-9A-AB-BC-CD-DE-EF-F0-01-02-03-04-C3

1. Define the compiler define USE_UICR_FOR_MAJ_MIN_VALUES and recompile and flash.
2. Use nrfjprog tool to write the value 0xabcd0102 to the UICR register as follows:

   ```
   nrfjprog --snr <Segger-chip-Serial-Number> --memwr 0x10001080 --val 0xabcd0102
   ```

3. Reset the board and observe the bytes in bold below are seen in the **Manufacturer Specific Data** field of the Advertising Data. This indicates that the Major and Minor values have been picked up from the UICR register written in the above step.

   xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-xx-**AB-CD-01-02**-xx

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

Blood Pressure Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Blood Pressure Application is an example that implements the Blood Pressure profile using the hardware delivered in the nRF5 Development Kit.

The application includes the two services in the Blood Pressure profile:

- Blood Pressure Service
- Device Information Service

In addition, use of the Battery Service is also demonstrated. When the application starts, a timer for generating battery measurements is started.

When indication of the Blood Pressure Measurement characteristic is enabled, a Blood Pressure Measurement is transmitted. This also happens when reconnecting to a previously bonded central. A Blood Pressure Measurement is also transmitted each time button 1 is pushed.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push button 1 or button 2 to wake it up and restart advertising.

Setup

The name of the example is **ble_app_bps_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_bps`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During Connection:
  - Button 1: Simulate and send a Blood Pressure Measurement.

Testing

Test the Blood Pressure Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_BPS'), then bond and perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on the Master Control Panel. Observe that one Blood Pressure Measurement indication is received. Also, Battery Level notifications are received every two seconds.
4. Click the 'Disconnect' button and then the 'Connect' button on the Master Control Panel. Observe that one Blood Pressure Measurement indication is received.
5. Push Button 1. Observe that one Blood Pressure Measurement indication is received each time the button is pushed.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

5. nRF5 SDK v11.0.0-2.alpha

Current Time Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Current Time Application is an example that implements the client role of the Current Time Profile using the hardware delivered in the nRF5 Development Kit.

This application utilizes the client implementation of the Current Time Service:

- Current Time Service client

Note
This application is not power optimized!

The purpose of this example is to use the Current Time Service to read the current time. The time is printed on the UART in the following format:

```
Date:
  Day of week   Saturday
  Day of month  15
  Month of year November
  Year          1986

Time:
    Hours     13
    Minutes   37
    Seconds   42
    Fractions 254/256 of a second

Adjust Reason:
    Daylight savings 1
    Time zone        0
    External update  0
    Manual update    0
```

Bonding is usually initiated by the central device. If it is not initiated within a specific time-out period, the application initiates bonding.

The current implementation requires security to be established before service discovery. After service discovery has been completed, button 1 can be used to read the current time (if the Current Time Service and Characteristic are found on the central side).

Setup

The name of the example is **ble_app_cts_c_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_cts_c`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During Connection:
  - Button 1: Read the current time from the connected central.

The Current Time Application does fast advertising for 30 seconds at power up and also after a disconnection. It will then switch to slow advertising mode for 180 seconds. After 180 seconds of slow advertising, the board will go to system-off.

The application uses the following UART settings:

- Baud rate: 38.400
- 8 data bits
- 1 stop bit
- No parity
- HW flow control: RTS/CTS

Testing

Test the Current Time Application with the Master Control Panel by performing the following steps:

1. Compile and download the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. In the Master Control Panel, create the Current Time Service (CTS) server by performing the following steps:
   a. Click File -> Server Setup. In the window that opens up, click File -> Load setup.
   b. In the resulting window, navigate to the `<InstallFolder>\examples\ble_peripheral\ble_app_cts_c` folder. If you are using Keil packs, this folder is located at

`<KeilFolder>\ARM\Pack\NordicSemiconductor\nRF_Examples\<version>\ble_peripheral\ble_app_cts_c`.
Load the file `cts_central.bin`.
c. Click 'Stop server' and then 'Start server'.
d. You must keep the 'Server Setup' window opened.
3. Select the device from Master Control Panel (the device will be advertising as 'Nordic_CTS'), connect and bond. Observe that the BSP_INDICATE_CONNECTED state is indicated.
4. After the security procedure is complete, verify that the Current Time Service on the server has been discovered. The message "Current Time Service discovered on server." will be printed on the UART.
5. In the 'Server Setup' window, set the values of characteristics with UUID 0x2A2B to C2-07-0B-0F-0D-25-2A-06-FE-08 and press 'update'.
6. Press button 1. Verify that the current time printed on the UART matches the time that was input in the Current Time Characteristic UUID 0x2A2B (see the example UART output).
7. Disconnect from the Master Control Panel and observe that the BSP_INDICATE_ADVERTISING state is indicated.
8. Wait until the application goes to system-off (BSP_INDICATE_IDLE).
9. Press button 1 and connect to the Master Control Panel. Observe that the BSP_INDICATE_CONNECTED state is indicated.
10. Disconnect from the Master Control Panel.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

Cycling Speed and Cadence Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Cycling Speed and Cadence Application is an example that implements the Cycling Speed and Cadence profile using the hardware delivered in the nRF5 Development Kit.

The application includes the two services in the Cycling Speed and Cadence profile:

- Cycling Speed and Cadence Service
- Device Information Service

In addition, use of the Battery Service is also demonstrated.

When the application starts, it starts a timer to simulate Cycling Speed and Cadence Measurements.

Also, a timer for generating battery measurements is started.

The sensor measurements are simulated the following way:

- Instantenous speed and instantaneous cadence: see Sensor Data Simulator.
- Battery Level: see Sensor Data Simulator.

When notification of Cycling Speed and Cadence Measurement characteristic is enabled, the Cycling Speed and Cadence Measurement, containing the current value for all the components of the Cycling Speed and Cadence Measurement characteristic, is notified each time the Cycling Speed and Cadence measurement timer expires. When notification of Battery Level characteristic is enabled, the Battery Level is notified each time the Battery Level measurement timer expires.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push the button 1 to restart advertising.

Setup

The name of the example is **ble_app_cscs_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_cscs`

Button assignments: BSP BLE Button Assignments.

Testing

Test the Cycling Speed and Cadence Application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_CSC'), then perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on the Master Control Panel. Observe that Cycling Speed and Cadence notifications are received every second, and Battery Level notifications are received every two seconds.
4. Do a 'start auto calibration' speed and cadence control point operation:
   - Write value 02 (OpCode: BLE_SCPT_START_AUTOMATIC_CALIBRATION) to the SC Control Point Characteristic (UUID 0x2A55).
   - Observe that you receive one indication of the SC Control Point Characteristic indicating success (100201).
5. Do a 'set cumulative value' speed and cadence control point operation:

- Write value 0112345678 (OpCode: BLE_SCPT_SET_CUMULATIVE_VALUE, Operand: 0x012345678) to the SC Control Point Characteristic (UUID 0x2A55).
- Observe that you receive one indication of the SC Control Point Characteristic indicating success (100101) and that the bytes 1-4 of the next Csc Measurement nofitications match the new value (received notification should be yyxx345678, where yy being the byte 0 containing flags, xx being the cumulative value LSB which changes quickly, and should be bigger than 12).

6. Do a 'request supported sensor locations' speed and cadence control point operation:
   - Write value 04 (OpCode: BLE_SCPT_REQUEST_SUPPORTED_SENSOR_LOCATIONS) to the SC Control Point Characteristic (UUID 0x2A55).
   - Observe that you receive one indication of the SC Control Point Characteristic indicating success and the list of supported locations (10040104050607080 90A0B0C0D).
7. Do an 'update sensor location' speed and cadence control point operation:
   - Write value 030A (OpCode: BLE_SCPT_UPDATE_SENSOR_LOCATION Operand: SENSOR_LOCATION_REAR_DROPOUT) to the SC Control Point Characteristic (UUID 0x2A55).
   - Observe that you receive one indication of the SC Control Point Characteristic indicating success (100301).
   - Read the Sensor Location Charactersitic (UUID 0x2A5D) and verify that the read value is 0x0A (SENSOR_LOCATION_REAR_DROPOUT).

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

7. nRF5 SDK v11.0.0-2.alpha

Glucose Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Glucose Application is an example that implements the Glucose Profile using the hardware delivered in the nRF5 Development Kit.

The application includes the two services in the Glucose Profile:

- Glucose Service
- Device Information Service

In addition, use of the Battery Service is also demonstrated.

When the application starts, a timer for generating battery measurements is started.

The actual measurements are hard-coded values. They get added when connected and bonded to a central, at each press of the button 2. The timestamps are simulated and it uses the Record Access Control point and implements the filtering on sequence number. It supports only glucose measurement characteristic, not glucose context.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push the button 1 to restart advertising.

Setup

The name of the example is **ble_app_gls_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_gls`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- When awake:
  - Button 2: Create a new record.

MITM protection

The Glucose Application secures the bond using Man-in-the-Middle protection (MITM). A passkey must be entered at the central to complete the bonding and ensure MITM protection. The Glucose Application will present the passkey during bonding. It is sent on the UART. It is therefore important that UART is set up correctly on the receiving side. The application uses the following UART settings:

- Baud rate: 38.400
- 8 data bits
- 1 stop bit
- No parity
- HW flow control: RTS/CTS

Demonstrating the Glucose Meter Profile

The Glucose Meter Profile uses the Record Access Control Point characteristic to access all measurements. This means that the sensor does not automatically send new measurements, but the collector has to fetch them writing request to the control point (see developer.bluetooth.org for more detail on the RACP characteristic).

Testing

Test the Glucose Meter Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect and to the device from Master Control Panel (the device will be advertising as 'Nordic_Glucose')
3. Bond to the device. A new window opens up and asks for a passkey. Enter the key received over the UART.
4. Perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
5. Click the 'Enable services' button on the Master Control Panel. Observe that Battery Level notifications are received every 10 seconds.
6. Press Button 2 five times.
7. Do a Read all records operation:
   - Write 0101 (OpCode : REPORT_STORED_RECORDS, Operator : ALL_STORED_RECORDS) to Record Access
   - Control Point Characteristic (UUID 0x2A52).
   - Observe that you receive five notification of the glucose measurements(UUID 0x2A18) and one indication of the RACP indicating success (06000101);
8. Read the number of stored records:
   - Write 0401 (OpCode: REPORT_NUM_STORED_RECORDS, Operator: ALL_STORED_RECORDS) to Record Access Control Point
   - Characteristic (UUID 0x2A52).
   - Observe that you get an indication of the RACP giving you the number of records, 5 (05000500).

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

Health Thermometer Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Health Thermometer Application is an example that implements the Health Thermometer profile using the hardware delivered in the nRF5 Development Kit.

The application includes the two services in the Health Thermometer profile:

- Health Thermometer Service
- Device Information Service

In addition, use of the Battery Service is also demonstrated. When the application starts, a timer for generating battery measurements is started.

When indication of the Temperature Measurement characteristic is enabled, a Temperature Measurement is transmitted. This also happens when reconnecting to a previously bonded central. A Temperature Measurement is also transmitted each time button 1 is pushed.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push button 1 to wake it up and restart advertising.

Setup

The name of the example is **ble_app_hts_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_hts`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During Connection:
  - Button 1: Simulate and send a Temperature Measurement.

Testing

Test the Health Thermometer Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_HTS'), then bond and perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on the Master Control Panel. Observe that one Temperature Measurement indication is received. Also, Battery Level notifications are received every two seconds.

4. Click the 'Disconnect' button and then the 'Connect' button on the Master Control Panel. Observe that one Temperature Measurement indication is received.
5. Push Button 1. Observe that one Temperature Measurement indication is received each time the button is pushed.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

Heart Rate Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Heart Rate Application is an example that implements the Heart Rate profile using the hardware delivered in the nRF5 Development Kit.

The application includes the three services in the Heart Rate profile:

- Heart Rate Service
- Battery Service
- Device Information Service

When the application starts, the Board Support Package is initialized. Next, three timers are started. These timers control the generation of various parts of the Heart Rate Measurement characteristic value:

- Heart Rate
- RR Interval
- Sensor Contact Detected

Also, a timer for generating battery measurements is started.

The sensor measurements are simulated the following way:

- Heart Rate: See Sensor Data Simulator.
- RR Interval: See Sensor Data Simulator.
- Sensor Contact: The state is toggled each time the timer expires.
- Battery Level: See Sensor Data Simulator.

When notification of Heart Rate Measurement characteristic is enabled, the Heart Rate Measurement, containing the current value for all the components of the Heart Rate Measurement characteristic, is notified each time the Heart Rate measurement timer expires. When notification of Battery Level characteristic is enabled, the Battery Level is notified each time the Battery Level measurement timer expires.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push the button 1 to restart advertising.

Setup

The name of the example is **ble_app_hrs_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_hrs`

Button assignments: BSP BLE Button Assignments.

Testing

Test the Heart Rate Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_HRM'), then perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on the Master Control Panel. Observe that Heart Rate notifications are received every second, and Battery Level notifications are received every two seconds.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

Heart Rate Application with FreeRTOS

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Heart Rate Application with FreeRTOS is a firmware example that shows the cooperation between FreeRTOS and the SoftDevice.

The name of the example is **ble_app_hrs_freertos_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\ble_app_hrs_freertos`

See Heart Rate Application for a detailed description of the example scenario.

Note
See FreeRTOS support for more information about FreeRTOS.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

11. nRF5 SDK v11.0.0-2.alpha

Heart Rate Application with RTX

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Heart Rate Application with RTX is a firmware example that shows the cooperation between RTX and the SoftDevice.

The name of the example is **ble_app_hrs_rtx_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_hrs_rtx`

See Heart Rate Application for a detailed description of the example scenario.

Note
See RTX support for more information about RTX.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

12. nRF5 SDK v11.0.0-2.alpha

HID Keyboard Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The HID Keyboard Application is an example that implements the HID over GATT profile for keyboard using the hardware delivered in the nRF5 Development Kit.

The application includes the three mandatory services needed for the HID over GATT profile:

- Human Interface Device Service
- Device Information Service
- Battery Service

Note
This application is not power optimized!

The application will stop advertising and go to system-off mode after 3 minutes and 30 seconds (if no advertising with whitelist was done) or after 4 minutes (if advertising with whitelist was done). Push the button 1 to restart advertising.

Setup

The name of the example is **ble_app_hids_keyboard_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\ble_app_hids_keyboard`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During Connection:
  - Button 1: Send sample text to the computer
  - Button 2: Simulates the Shift key

Testing

Test the HID Keyboard Application with a Microsoft Windows 8 computer with a *Bluetooth* radio:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. On the Windows 8 computer, search for *Bluetooth* devices, and connect to the device named 'Nordic_Keyboard'. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Open a text editing application (for example Notepad).
4. Press Button 1 on the board. This will send one character of the test message 'hello' (the test message includes a carriage return) to the computer, and this will be displayed in the text editor.
5. Press Button 1 while keeping the Button 2 pressed. Observe that the same 'hello' text will appear in capital letters. This is because Button 2 simulates the Shift key.
6. Turn Caps Lock on on the computer. Observe that:
   - BSP_INDICATE_ALERT_3 is indicated.
   - The text 'CAPSON' will be seen on the text editor window on the computer. This verifies that output report characteristic has been written successfully on the HID device.
7. Turn Caps Lock off on the computer. This should result in two events:
   - BSP_INDICATE_ALERT_OFF is indicated.
   - The text 'capsof' will be seen on the editor window.
8. Disconnect the computer from the device by removing the device from the computer's devices list. Observe that for the first period, the BSP_INDICATE_ADVERTISING_DIRECTED state is indicated. Then it switches to BSP_INDICATE_ADVERTISING_WHITELIST. Press and hold Button 2 to turn off the whitelist. Observe that BSP_INDICATE_ADVERTISING is indicated at first, then it switches to BSP_INDICATE_ADVERTISING_SLOW, and finally it switches to BSP_INDICATE_IDLE.
9. Reset the device while pressing Button 2 to erase bond info. Repeat step 2, and check if the device can send messages to the computer.
10. Disconnect the device from Windows 8, thereby removing the bond info from the computer. Start a search for *Bluetooth* devices and observe that while the device is advertising with Whitelist, Windows 8 is not able to discover the device.
11. The device should be discoverable again after it switches to advertising mode.

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_Keyboard'), then bond and perform service discovery. Observe that BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on Master Control Panel, and observe that Battery Level notifications are being received every two seconds.
4. Push Button 1 on the board. Observe that 2 notifications on one of the HID Report characteristics are received; two notifications (press and release) each for one character of the test message. The first one has the value '00000B0000000000', the second has the value '0000000000000000'. This corresponds to press and release of character 'h'. Similarly, further press and release events will result in press and release notifications of subsequent characters of the test string. Therefore, pushing Button 1 again will result in notification of press and release reports for character 'e'.
5. Simulate Shift key:
   - Keeping Button 2 pressed, push the Button 1 on the board.
   - Observe that 2 notifications on one of the HID Report characteristics are received; two notifications (press and release) each for one character of the test message. The first one has the value '02000F0000000000', the second has the value '0200000000000000'. This corresponds to press and release of character 'l' with the Shift key pressed.
6. Simulate Caps Lock ON:
   - On the Master Control Panel, select the HID Report, which has the properties Read, WriteWithoutResponse, and Write (the Output Report) and UUID: 0x2A4D.
   - Click the 'hex' radio button.
   - Enter '02' in the text box.
   - Click the 'Write' button.
   - Observe that BSP_INDICATE_ALERT_3 is indicated and that 7 notifications on the other HID Report characteristic (the Input Report) are received. The first one has the value '0000060413161211', the second has the value '0000000413161211', third has the value '0000000013161211', and so on until release event for all characters is sent as the 7th one as '0000000000000000'.
7. Simulate Caps Lock OFF:
   - On the Master Control Panel, select the same HID Report as in the previous step (the Output Report).
   - Click the 'hex' radio button.
   - Enter '00' in the text box.
   - Click the 'Write' button.
   - Observe that BSP_INDICATE_ALERT_OFF is indicated and that 7 notifications on the other HID Report characteristic are received. The first one has the value '0000060413161209', the second has the value '0000000413161209', third has the value '0000000013161209', and so on until release event for all characters is sent as the 7th one as '0000000000000000'.
8. Click on the Disconnect button in Master Control Panel. Observe that no new notifications are received and the device is advertising with Whitelist.
9. As bond info is preserved by Master Control Panel, it should be possible to immediately reconnect to the device by clicking on the Connect button again. Observe that Battery Level notifications are being received every two seconds.

HID Mouse Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The HID Mouse Application is an example that implements the HID over GATT profile for mouse using the hardware delivered in the nRF5 Development Kit.

The application includes the three mandatory services needed for the HID over GATT profile:

- Human Interface Device Service
- Device Information Service
- Battery Service

Note

This application is not power optimized!

The application will stop advertising and go to system-off mode after 3 minutes and 30 seconds(if no advertising with whitelist was done) or after 4 minutes (if advertising with whitelist was done).

Push the button 1 to restart advertising.

Setup

The name of the example is **ble_app_hids_mouse_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_hids_mouse`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During Connection:
  - Button 1: Move mouse pointer 5 pixels to the left
  - Button 2: Move mouse pointer 5 pixels to the right
  - Button 3: Move mouse pointer 5 pixels upward
  - Button 4: Move mouse pointer 5 pixels downward

Testing

The HID Mouse Application should be connected to a Microsoft Windows 8 computer with a *Bluetooth* radio. When connected, the application will act as a mouse, enabling the user to move the computer's mouse pointer using four push buttons on the board:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. On the Windows 8 computer, search for *Bluetooth* devices, and connect to the device named 'Nordic_Mouse'. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Push Button 1 on the board. Observe that the mouse pointer on the Windows 8 computer moves to the left.
4. Push Button 2. Observe that the mouse pointer on the Windows 8 computer moves to the right.
5. Push Button 3. Observe that the mouse pointer on the Windows 8 computer moves up.
6. Push Button 4. Observe that the mouse pointer on the Windows 8 computer moves down.
7. Disconnect the computer from the device by removing the device from the computer's devices list. Observe that for the first period, the BSP_INDICATE_ADVERTISING_DIRECTED state is indicated, then it switches to BSP_INDICATE_ADVERTISING_WHITELIST, then to BSP_INDICATE_ADVERTISING, then to BSP_INDICATE_ADVERTISING_SLOW, and then all off.
8. Reset the device while pressing Button 2 to erase bond info. Repeat step 2, and check if the device can control the mouse pointer on the computer.
9. Disconnect the device from Windows 8, thereby removing the bond info from the computer. Start a search for *Bluetooth* devices and observe that while the device is advertising with Whitelist, Windows 8 is not able to discover the device.
10. The device should be discoverable again after it switches to advertising mode.

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_Mouse'), then bond and perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on Master Control Panel, and observe that Battery Level notifications are being received every two seconds.
4. Push Button 1 on the board. Observe that a notification on one of the HID Report characteristics is received, containing the value 'FB0F00'.
5. Push Button 2. Observe that a notification on the same HID Report characteristic is received, containing the value '00B0FF'.
6. Push Button 3. Observe that a notification on the same HID Report characteristic is received, containing the value '050000'.
7. Push Button 4. Observe that a notification on the same HID Report characteristic is received, containing the value '005000'.
8. Click on the Disconnect button in Master Control Panel. Observe that no new notifications are received, and the device is advertising with Whitelist.
9. As bond info is preserved by Master Control Panel, it should be possible to immediately reconnect to the device by clicking on the Connect button again. Observe that Battery Level notifications are being received every two seconds.

Multiprotocol Application

*This information applies to the **nRF51 Series** only.*

*This example requires one of the following SoftDevices: **S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Multiprotocol Application is an example that implements the Heart Rate profile in *Bluetooth* mode and the Gazell device mode.

When the application starts, it is in *Bluetooth* mode and behaves as the Heart Rate Application application. If button 2 is pressed, the application switches to Gazell device mode. In this mode, it attempts to send packets to a host. If the packets are successfully sent, the BSP_INDICATE_SENT_OK state is indicated. If it fails, the BSP_INDICATE_SEND_ERROR state is indicated. If button 1 is pressed, the application switches back to *Bluetooth* mode.

Note

This application is not power optimized!

In *Bluetooth* mode, the application goes to system-off after 3 minutes of advertising if no connection is initiated. Press button 1 to restart in *Bluetooth* mode or button 2 to restart in Gazell mode.

This application uses a local copy of Application Timer using SWI1 instead of SWI0.

Setup

The name of the example is **ble_app_gzll_*board***, where *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\multiprotocol\ble_app_gzll`

Button assignments:

- During system-off:
  - Button 1: Wake up and start advertising in *Bluetooth* mode.
  - Button 2: Wake up and start advertising in *Bluetooth* mode.
- During Advertising and Connection:
  - Button 1: Switch to *Bluetooth* mode.
  - Button 2: Switch to Gazell device mode.

Testing

Test the *Bluetooth* part of the Multiprotocol Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_BLE_GZL'), then perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on the Master Control Panel. Observe that Heart Rate notifications are received every second, and Battery Level notifications are received every two seconds.
4. Click on the Disconnect button on the Master Control Panel. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
5. Press the Gazell button (Button 2) and observe that the BSP_INDICATE_SEND_ERROR state is indicated (if a Gazell host device is running, you should observe that the BSP_INDICATE_SENT_OK state is indicated and that the LEDs on the host device are lit in a running sequence).

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

15. nRF5 SDK v11.0.0-2.alpha

Power Profiling Application

*This example requires one of the following SoftDevices: **S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Power Profiling Applicationis designed to be used to measure the current consumption of the nRF5 IC, loaded with the SoftDevice, during advertisement and data transmission. The user can change some of the parameters (listed below) of this application and measure current consumption.

This application also demonstrates the use of Vendor Specific (128 bit) UUIDs.

This application can be used in two different modes:

1. Connectable mode:

In this mode, the application advertises in connectable mode. It sets itself up with a service and a characteristic with 'Notify' property. This characteristic has a CCCD associated with it.

Once the central enables notification of this characteristic (writes 0x0001 to this CCCD), the application starts sending notifications of this characteristic for a time interval equal to the value of APP_CFG_CHAR_NOTIF_TIMEOUT. The application will attempt to send one notification per connection interval, but this is not guaranteed. There is a possibility that in one connection interval there are no notifications but in the next there are two notifications. This depends on when exactly the repeating application timer expires. The total number of notifications sent by the application will be close to (APP_CFG_CHAR_NOTIF_TIMEOUT / (APP_CFG_CONNECTION_INTERVAL)). After sending these notifications, the application puts the chip in 'System Off' mode.

2. Non-connectable mode:

In this mode, the application advertises the given length of data (APP_CFG_ADV_DATA_LEN) for a given duration of time (APP_CFG_NON_CONN_ADV_TIMEOUT) in non-connectable mode and then puts the chip in System Off mode.

Note

In both modes, the device name does not show up in the advertisement packet as per design. The advertisement packet has only 'Flags' and 'Manufacturer specific data' fields. This is done to make the least possible size of an advertisement packet as 8 bytes.

Power Management

The application demonstrates the use of two different power saving modes, namely CPU Sleep and System-Off mode. All sample applications delivered with the SDK use these two modes.

Low Power Mode:

The application uses the sd_app_evt_wait function to enter CPU sleep mode. This will place the chip in 'Low Power' mode. The chip will wake up from this mode on application interrupts. Typically, any application should call sd_app_evt_wait in an infinite loop in main. It will return when an application interrupt has occurred, thereby allowing the main thread to process it if needed.

System OFF Mode:

This is the deepest power saving mode the system can enter. In this mode, all core functionality is powered down, and most peripherals are therefore non-functional or non-responsive. The only mechanisms that are functional and responsive in this mode are the reset mechanism and the wakeup mechanism.

This sample application starts up, configures wakeup buttons, and enters the Szstem OFF mode. On button presses, the application wakes up and goes into either 'Connectable mode' or 'Non-connectable mode' based on which button is pressed.

For more information on both modes, see sd_app_evt_wait and the nRF5 Series User Specification (Section: POWER - Power control).

Compiler Optimization:

This example is optimized for reduced power consumption. Therefore, the default optimization level for this example application is 'Level 3' (`-O3`) and 'Optimize for Time' (`-Otime`). If the Keil debugger is needed for debugging purposes, the optimization level should be changed to 'Level 0' (`-O0`), and 'Optimize for Time' should be removed to ease the use of the debugger.

Setup

The name of the example is **ble_app_pwr_profiling*_SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\ble_app_pwr_profiling`

The application does not use any LEDs.

Button assignments:

- Button 1: Place the application in connectable mode.
- Button 2: Place the application in non-connectable mode.

Testing

Test the Power Profiling Application with the Master Control Panel by performing the following steps:

1. Compile and program the application.
2. Start Master Control Panel on the computer.
3. Click File -> Options. Verify that the connection interval value displayed in the 'Options' window is the same as APP_CFG_CONNECTION_INTERVAL.
4. Close the Options window and click 'Start discovery' in the Master Control Panel main window.
5. Push Button 1 on the board. Observe that the application is advertising in connectable mode.
6. Connect to the device from Master Control Panel, click 'Service discovery', then click 'Enable Services'.
7. Observe notifications of the characteristic being received by the central.
8. After APP_CFG_CHAR_NOTIF_TIMEOUT, the Master control panel will lose connection with the chip because it enters System Off mode.
9. Verify that the size of the characteristic value received in all the notifications was equal to APP_CFG_CHAR_LEN.
10. Click the 'Back' button in the Master control panel, right click in the 'Discovered Devices' section in Master Control Panel, and click 'Clear list' to clear the list of devices.
11. Verify that the device is not advertising.
12. Push Button 2 on the board. Observe the application is advertising in non-connectable mode (the Adv. type field should be Non-Connectable).
13. Verify that the number of bytes in the Manufacturer Specific Data in Adv. data is APP_CFG_ADV_DATA_LEN - 5.
14. After APP_CFG_NON_CONN_ADV_TIMEOUT, clear the list of 'Discovered Devices'. Verify that the device is not advertising.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

16. nRF5 SDK v11.0.0-2.alpha

Proximity Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Proximity Application is an example that implements the Proximity Profile and the Find Me Profile using the hardware delivered in the nRF5 Development Kit. The Proximity Profile alerts the user when connected devices are too far apart, while the Find Me Profile is used to alert the remote device (for example, a phone).

The application includes the following services:

- For the Proximity Profile:
  - Link Loss Service
  - Immediate Alert Service
  - TX Power Service
- For the Find Me Profile:
  - Immediate Alert Service Client

In addition, use of the Battery Service and Device Manager are also demonstrated.

At each power up and disconnect, the Proximity Application will do 20 seconds of fast advertising with whitelist followed by 40 seconds of fast advertising without whitelist. It will then switch to slow advertising mode, and the state BSP_INDICATE_ADVERTISING_WHITELIST is indicated. When the link is lost, or when an alert is signaled using the Immediate Alert Service, an alert is indicated using the state BSP_INDICATE_ALERT_0 or BSP_INDICATE_ALERT_1, depending on the value of the Alert Level characteristic.

The example uses the ADC peripheral to measure the battery level every 2 minutes. If there is a change in the battery level, the application sends the current level as notification.

Note

This application is not power optimized!

- The device must be in bonded state to change any characteristic in this application.
- The TX Power Level characteristic is not writable.
- The application will stop advertising and go to system-off mode after 3 minutes and 30 seconds (if no advertising with whitelist was done) or after 4 minutes (if advertising with whitelist was done). Push button 1 to restart advertising, or push and hold button 1 for 5 seconds to erase all bonds and restart advertising.
- Upon disconnecting a bonded connection, the application will advertise with whitelist for ADV_INTERVAL_FAST_PERIOD seconds. After this time-out, the application will be able to bond to a new central.

Setup

The name of the example is **ble_app_proximity_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_proximity`

Button assignments - in addition to those defined in BSP BLE Button Assignments:

- During Connection:
  - Button 1: Raise/clear High Alert to the central with an Immediate Alert Service instance.

Testing

Test the Proximity Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_Prox'), then bond and perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Set the AlertLevel characteristic in Immediate Alert Service to 1 (click the 'hex' radio button, enter '01' in the text box, and click the 'Write' button), and observe the Mild alert (BSP_INDICATE_ALERT_0).
4. Change the AlertLevel value to 2 and observe the High alert (BSP_INDICATE_ALERT_3).
5. Change the AlertLevel value to 0 and observe BSP_INDICATE_ALERT_OFF.
6. Set the AlertLevel characteristic in Link Loss Service to 1.
7. Simulate a link loss situation (for example by shielding the Master Emulator) and observe that both BSP_INDICATE_ALERT_0 and BSP_INDICATE_ADVERTISING are indicated.
8. Reconnect and observe that only the BSP_INDICATE_CONNECTED state is indicated.
9. Set the AlertLevel characteristic in Link Loss Service to 2.
10. Simulate a link loss situation, for example by disconnecting the Master Emulator, and observe that the BSP_INDICATE_ALERT_3 and BSP_INDICATE_ADVERTISING are indicated.
11. Reconnect and observe that only the BSP_INDICATE_CONNECTED state is indicated.
12. Select the TxPowerLevel characteristic in TX Power Service, click the 'Read' button, and verify that it was successfully read.
13. Click the 'Enable services' button to start notification of Battery Level and observe that the battery level is notified periodically (if there is a change).
14. Disconnect from the Master Control Panel and observe that BSP_INDICATE_ADVERTISING is indicated.
15. Wait until the application goes to system-off (BSP_INDICATE_IDLE).

16. Press button 1 and connect to the Master Control Panel. Observe that BSP_INDICATE_CONNECTED is indicated and that the battery level is notified periodically (if there is a change).
17. Disconnect from the Master Control Panel.
18. Connect with a different central and bond and disconnect.
19. Wait until the application goes to system-off (BSP_INDICATE_IDLE).
20. Press button 1 and reconnect to the first Master Control Panel. Observe that BSP_INDICATE_CONNECTED is indicated and that the battery level is notified periodically (if there is a change).
21. Disconnect from the Master Control Panel.
22. Connect with the second central and observe that the link encryption is on and that the battery level is not notified.
23. Disconnect from the second central and wait for the application to go to system-off.
24. Press and hold button 2 (erase bonds) for 5 seconds and connect with the first Master Control Panel. Observe that the link encryption is OFF and that no notifications are received.
25. Disconnect from Master Control Panel.

Test the Find Me locater (Immediate Alert Service Client) functionality by performing the following steps:

1. In the Master Control Panel, create the Immediate Alert Service (IAS) server by performing the following steps:
   - Click File -> Server Setup. In the window that is displayed, click File -> Load setup.
   - In the resulting window, navigate to the `<InstallFolder>\examples\ble_peripheral\ble_app_proximity` folder. If you are using Keil packs, this folder is located at `<KeilFolder>\ARM\Pack\NordicSemiconductor\nRF_Examples\<version>\ble_peripheral\ble_app_proximity`. Load the file `findMeMasterServer.bin`.
   - Click the 'Stop Server' button and then 'Start Server'. This restarts the server with Immediate Alert Service configured.
   - You must keep the 'Server Setup' window opened.
2. Go to the Master Control Panel main window and connect to the device.
3. Press button 1 and observe that the AlertLevel value at the 'Server setup' window of the Master Control Panel changes to 02. This indicates that the device was able to signal a High Alert to the central.
4. Press button 1 again and observe that the AlertLevel value changes to 00. This indicates that the device was able to clear the High Alert previously signalled to the central.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

Running Speed and Cadence Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Running Speed and Cadence Application is an example that implements the Running Speed and Cadence profile using the hardware delivered in the nRF5 Development Kit.

The application includes the two services in the Running Speed and Cadence profile:

- Running Speed and Cadence Service
- Device Information Service

In addition, use of the Battery Service is also demonstrated.

When the application starts, it starts a timer to simulate Running Speed and Cadence Measurements.

Also, a timer for generating battery measurements is started.

The sensor measurements are simulated the following way:

- Instantaneous speed and instantaneous cadence: see Sensor Data Simulator.
- Battery Level: see Sensor Data Simulator.

When notification of Running Speed and Cadence Measurement characteristic is enabled, the Running Speed and Cadence Measurement, containing the current value for all the components of the Running Speed and Cadence Measurement characteristic, is notified each time the Running Speed and Cadence measurement timer expires. When notification of Battery Level characteristic is enabled, the Battery Level is notified each time the Battery Level measurement timer expires.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push the button 1 restart advertising.

Setup

The name of the example is **ble_app_rscs_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_rscs`

Button assignments: BSP BLE Button Assignments.

Testing

Test the Running Speed and Cadence Application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_RSC'), then perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Click the 'Enable services' button on the Master Control Panel. Observe that Running Speed and Cadence notifications are received every second, and Battery Level notifications are received every two seconds.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

18. nRF5 SDK v11.0.0-2.alpha

Template Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Template Application is an application that you can use as a starting point for developing your own application, using the hardware delivered in the nRF5 Development Kit.

Note
The application will stop advertising after 3 minutes, and then go to system-off mode. To wake it up, press the button 1, and it will start advertising again.

Setup

The name of the example is **ble_app_template_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_template`

Buttons assignments:

- Button 1: Wake up and start advertising.

Testing

Test the Template Application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_Template'), then connect and perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Disconnect, observe that the BSP_INDICATE_ADVERTISING state is indicated. You can wait for 3 minutes and observe that the BSP_INDICATE_IDLE is indicated.
4. In system-off, press button 1, the application starts advertising again.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

19. nRF5 SDK v11.0.0-2.alpha

UART/Serial Port Emulation over BLE

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Nordic UART Service (NUS) Application is an example that emulates a serial port over BLE. In the example, Nordic Semiconductor's development board serves as a peer to the phone application "nRF UART", which is available for iOS from Apple Store and for Android from Google Play. In addition, the example demonstrates how to use a proprietary (vendor-specific) service and characteristics with the SoftDevice.

The application includes one service: the Nordic UART Service. The 128-bit vendor-specific UUID of the Nordic UART Service is 6E400001-B5A3-F393-E0A9-E50E24DCCA9E (16-bit offset: 0x0001).

This service exposes two characteristics: one for receiving and one for transmitting (as seen from the nRF5 application).

- **TX Characteristic (UUID: 6E400002-B5A3-F393-E0A9-E50E24DCCA9E)**
  If the peer has enabled notifications for the TX Characteristic, the application can send data to the peer as notifications. The application will transmit all data received over UART as notifications.
- **RX Characteristic (UUID: 6E400003-B5A3-F393-E0A9-E50E24DCCA9E)**
  The peer can send data to the device by writing to the RX Characteristic of the service. ATT Write Request or ATT Write Command can be used. The received data is sent on the UART interface.

Important code lines

The following sections describe some important parts of the example code.

Adding proprietary service and characteristics

The initialization of the proprietary service and its characteristics is done in `ble_nus.c`.

The Nordic UART Service is added to the SoftDevice as follows:

```
    // Add a custom base UUID.
    err_code = sd_ble_uuid_vs_add(&nus_base_uuid, &p_nus->uuid_type);
    VERIFY_SUCCESS(err_code);

    ble_uuid.type = p_nus->uuid_type;
    ble_uuid.uuid = BLE_UUID_NUS_SERVICE;

    // Add the service.
    err_code = sd_ble_gatts_service_add(BLE_GATTS_SRVC_TYPE_PRIMARY,
                                        &ble_uuid,
                                        &p_nus->service_handle);
```

The RX characteristic is added to the SoftDevice as shown in the code below. The read and write permissions of the characteristic and its CCCD are set as open, which means that there are no security restrictions on this characteristic. The type of the UUID (`ble_uuid.type`) is the value that was returned in the call to sd_ble_uuid_vs_add(). The TX characteristic is added in a similar way.

```
    ble_gatts_char_md_t char_md;
    ble_gatts_attr_md_t cccd_md;
    ble_gatts_attr_t    attr_char_value;
    ble_uuid_t          ble_uuid;
    ble_gatts_attr_md_t attr_md;

    memset(&cccd_md, 0, sizeof(cccd_md));

    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.read_perm);
    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.write_perm);

    cccd_md.vloc = BLE_GATTS_VLOC_STACK;

    memset(&char_md, 0, sizeof(char_md));

    char_md.char_props.notify = 1;
    char_md.p_char_user_desc  = NULL;
    char_md.p_char_pf         = NULL;
    char_md.p_user_desc_md    = NULL;
    char_md.p_cccd_md         = &cccd_md;
    char_md.p_sccd_md         = NULL;

    ble_uuid.type = p_nus->uuid_type;
    ble_uuid.uuid = BLE_UUID_NUS_RX_CHARACTERISTIC;

    memset(&attr_md, 0, sizeof(attr_md));

    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.read_perm);
    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.write_perm);

    attr_md.vloc    = BLE_GATTS_VLOC_STACK;
    attr_md.rd_auth = 0;
    attr_md.wr_auth = 0;
    attr_md.vlen    = 1;

    memset(&attr_char_value, 0, sizeof(attr_char_value));

    attr_char_value.p_uuid    = &ble_uuid;
    attr_char_value.p_attr_md = &attr_md;
    attr_char_value.init_len  = sizeof(uint8_t);
    attr_char_value.init_offs = 0;
    attr_char_value.max_len   = BLE_NUS_MAX_RX_CHAR_LEN;

    return sd_ble_gatts_characteristic_add(p_nus->service_handle,
                                           &char_md,
                                           &attr_char_value,
                                           &p_nus->rx_handles);
```

Initializing UART

The initialization of the application and the handling of data sent and received through BLE and UART are done in `main.c`.

The UART initialization is done as shown in the code below. The code segment uses the UART module that is provided in the SDK to perform the UART configuration. Note that app_uart_comm_params_t configures the application to use hardware flow control. Therefore, RTS_PIN_NUMBER and CTS_PIN_NUMBER are used as Ready-to-Send and Clear-to-Send pins, respectively.

```
static void uart_init(void)
{
    uint32_t                     err_code;
    const app_uart_comm_params_t comm_params =
```

```
    {
        RX_PIN_NUMBER,
        TX_PIN_NUMBER,
        RTS_PIN_NUMBER,
        CTS_PIN_NUMBER,
        APP_UART_FLOW_CONTROL_ENABLED,
        false,
        UART_BAUDRATE_BAUDRATE_Baud38400
    };

    APP_UART_FIFO_INIT( &comm_params,
                        UART_RX_BUF_SIZE,
                        UART_TX_BUF_SIZE,
                        uart_event_handle,
                        APP_IRQ_PRIORITY_LOW,
                        err_code);
    APP_ERROR_CHECK(err_code);
}
```

Handling data received over BLE

When initializing the service in the services_init() function, the application passes the function nus_data_handler, which is used for handling the received data. When the Nordic UART Service indicates that data has been received over BLE from the peer, the same data is relayed to the UART. The nus_data_handler function is implemented as follows:

```
static void nus_data_handler(ble_nus_t * p_nus, uint8_t * p_data, uint16_t length)
{
    for (uint32_t i = 0; i < length; i++)
    {
        while(app_uart_put(p_data[i]) != NRF_SUCCESS);
    }
    while(app_uart_put('\n') != NRF_SUCCESS);
}
```

Handling data received over UART

Data that is received from the UART undergoes certain checks before it is relayed to the BLE peer using the Nordic UART Service. The code shown below is part of the app_uart event handler, which is called every time a character is received over the UART. Received characters are buffered into a string until a new line character is received or the size of the string exceeds the limit defined by NUS_MAX_DATA_LENGTH. When one of these two conditions is met, the string is sent over BLE using the ble_nus_send_string function.

Note

By default, the macro NUS_MAX_DATA_LENGTH is set to the maximum size of a notification packet (BLE_ATT_MTU - 3). This is the maximum value for NUS_MAX_DATA_LENGTH, which should not be exceeded.

```
void uart_event_handle(app_uart_evt_t * p_event)
{
    static uint8_t data_array[BLE_NUS_MAX_DATA_LEN];
    static uint8_t index = 0;
    uint32_t       err_code;

    switch (p_event->evt_type)
    {
        case APP_UART_DATA_READY:
            UNUSED_VARIABLE(app_uart_get(&data_array[index]));
            index++;

            if ((data_array[index - 1] == '\n') || (index >= (BLE_NUS_MAX_DATA_LEN)))
            {
                err_code = ble_nus_string_send(&m_nus, data_array, index);
                if (err_code != NRF_ERROR_INVALID_STATE)
                {
                    APP_ERROR_CHECK(err_code);
                }

                index = 0;
            }
            break;

        case APP_UART_COMMUNICATION_ERROR:
            APP_ERROR_HANDLER(p_event->data.error_communication);
            break;

        case APP_UART_FIFO_ERROR:
            APP_ERROR_HANDLER(p_event->data.error_code);
            break;

        default:
            break;
    }
}
```

Setup

The name of the example is **ble_app_uart_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder: `<InstallFolder>\examples\ble_peripheral\ble_app_uart`

LED assignments:

- Advertising mode: LED 1 is blinking (period 2 seconds, duty cycle 10%).
- Connected mode: LED 1 is on.

Button assignments: BSP BLE Button Assignments.

Testing

Test the UART Application with the nRF UART app, which is available for iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Connect the board to the computer using a USB cable. The board is assigned a COM port, which is visible in the Device Manager.
2. Start a terminal emulator like PuTTY and connect to the used COM port with the following UART settings:
    - Baud rate: 38.400
    - 8 data bits
    - 1 stop bit
    - No parity
    - HW flow control: RTS/CTS
3. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated, and that the device is advertising with the device name "Nordic_UART".
4. Observe that the text "Start..." is printed on the COM listener running on the computer.
5. Connect to the device from Master Control Panel, then perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
6. Click the "Enable services" button in the Master Control Panel.
7. Select the UART RX characteristic value in the Master Control Panel.
8. Choose the option "text" in the attribute value section.
9. Write "123456789" and click "Write". Verify that the text "123456789" is displayed on the COM listener.
10. For sending data from the device to the Master Control Panel, enter any text, for example, "Hello", on the COM listener. Observe that a notification with the corresponding ASCII values is sent to the peer on handle 0x0B. For the string "Hello", the notification is "48656C6C6F0D".
11. Press the "Disconnect" button in the Master Control Panel. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
12. If no peer connects to the application within 180 seconds (APP_ADV_TIMEOUT_IN_SECONDS), the application will put the chip into system-off mode.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

20. nRF5 SDK v11.0.0-2.alpha

Experimental: BLE Blinky Application

*This example requires one of the following SoftDevices: **S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The BLE Blinky Application uses the LED Button Service. This is a small custom service that is used to toggle LEDs and receive button statuses from the nRF5 Development board.

Setup

The name of the example is **experimental_ble_app_blinky_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\experimental_ble_app_blinky`

Button assignments: BSP BLE Button Assignments.

Testing

Test the BLE Blinky Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that LED 1 is on. This indicates that the application is advertising.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_Blinky').
3. Observe that LED 2 is on and LED 1 is off. This indicates that the connections is established.
4. In Master Control Panel, perform service discovery and enable services.
5. Observe that notifications are received on the Button Characteristic (0x1524) when pressing or releasing button 1.
6. Write 01 to the LED Characteristic (0x1525) and observe that LED 1 is turned on.
7. Write 00 to the LED Characteristic (0x1525) and observe that LED 1 is turned off.

Experimental: Bluetooth Developer Studio Example

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The bluetoothds_template application is an example that you can use as a starting point for developing your own application using services generated by Bluetooth Developer Studio.

Note
The application will stop advertising after 3 minutes and go to system-off mode. Push button 1 to wake it up and restart advertising.

Implementation

To implement your own application, you must first generate a custom profile in Bluetooth Developer Studio. You can then use this custom profile in the example application.

Generating a custom profile

1. Create a custom profile:
   a. Click **File** > **New**.
   b. Rename the Project to, for example, "Test project".
   c. Enter a string for NAMESPACE, for example "com.MyCompany".
   d. Rename the Profile to, for example, "Test Profile"
2. Create a custom service:
   a. Click **CUSTOM SERVICE**.
   b. Rename the new service to, for example, "Test profile test service".
3. Add a new characteristic:
   a. Click **+**.
   b. Rename the new characteristic to, for example, "RX".
   c. Set the Read/Write/Notify/Indicate properties. For example, set "Write Without Response" to "Mandatory".
   d. Add fields to the characteristic. For example, add a "value" field with format "_VARIABLE".
4. Add further characteristics, if required. For example, add a "TX" characteristic with the Notify property set to "Mandatory" and a "_VARIABLE" field named "value".
5. Generate the code:
   a. Click **Tools** > **Generate Code**.
   b. Select "Server" and "NORDIC SEMICONDUCTOR NRF5X V1.1.x".

Including the custom profile

1. If you want to use tracing, define the APP_LOG macro:
   ```
   #define APPL_LOG app_trace_log
   ```
2. In the Keil project, add the source file of the service by adding `ble_test_profile_test_service.c` to the group nRF_BLE.

Setup

The name of the example is **bluetoothds_template_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\experimental_bluetoothds_template`

Buttons assignments:

- Button 1: Wake up and start advertising.

Testing

Test the Template Application with the Master Control Panel by performing the following steps:

1. Compile and program the application. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
2. Connect to the device from Master Control Panel (the device will be advertising as 'Nordic_BDS'), then connect and perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
3. Disconnect and observe that the BSP_INDICATE_ADVERTISING state is indicated. You can wait for 3 minutes and observe that the BSP_INDICATE_IDLE is indicated.
4. In system-off mode, press button 1. The application starts advertising again.

Experimental: Eddystone Beacon Application

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The Eddystone Beacon Application is an example that demonstrates advertising with the Eddystone protocol. See the Eddystone GitHub repository for the protocol specification and further information.

The Eddystone protocol describes different formats for advertising packets, called *frame types*, that can be used to create beacons. The following three frame types are supported by this example application:

- **Eddystone-UID** for broadcasting unique 16-byte beacon IDs
- **Eddystone-URL** for broadcasting a URL in a compressed encoding format
- **Eddystone-TLM** for broadcasting telemetry information about the beacon

Currently, the example supports using only one frame type at a time. By default, the active frame type is the URL frame type. To switch to the UID frame type, do the following changes to the example code:

1. Uncomment the eddystone_uid_data array:

```
//static uint8_t eddystone_uid_data[] =   /**< Information advertised by the Eddystone UID
      frame type. */
//{
//    APP_EDDYSTONE_UID_FRAME_TYPE,   // Eddystone UID frame type.
//    APP_EDDYSTONE_RSSI,             // RSSI value at 0 m.
//    APP_EDDYSTONE_UID_NAMESPACE,    // 10-byte namespace value. Similar to Beacon Major.
//    APP_EDDYSTONE_UID_ID,           // 6-byte ID value. Similar to Beacon Minor.
//    APP_EDDYSTONE_UID_RFU           // Reserved for future use.
//};
```

2. In the advertising_init function, replace `eddystone_url_data` with `eddystone_uid_data`:

```
    eddystone_data_array.p_data = (uint8_t *) eddystone_url_data;   // Pointer to the data to
      advertise.
    eddystone_data_array.size = sizeof(eddystone_url_data);         // Size of the data to
      advertise.
```

To switch to the TLM frame type, do the same changes for `eddystone_tlm_data` instead of `eddystone_url_data`.

The name of the example is **experimental_ble_app_eddystone_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\experimental_ble_app_eddystone`

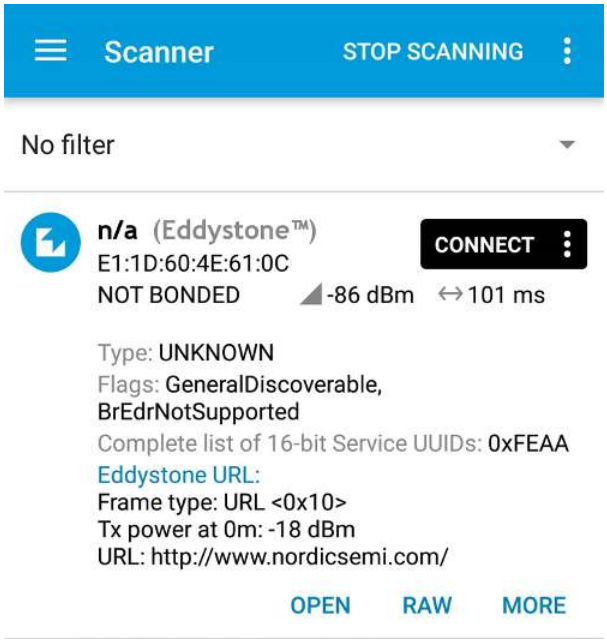
Testing

Test the Eddystone Beacon Application with the Master Control Panel by performing the following steps:

1. Compile and program the application.
2. Start discovery in Master Control Panel.
3. Observe that the beacon application is recognized:
   - In the Master Control Panel PC application, the advertising data contains the following information:

Field	Value
Flags	GeneralDiscoverable, BrEdrNotSupported
ServicesCompleteListUUID16	0xFEAA
ServiceData	Uuid:0xFEAA Data: 10-EE-00-6E-6F-72-64-69-63-73-65-6D-69-00

   - In the Master Control Panel Android app, the beacon is recognized as Eddystone:



Eddystone beacon in nRF Master Control Panel

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

23. nRF5 SDK v11.0.0-2.alpha

Experimental: Heart Rate Example with pairing over NFC

*This information applies to the **nRF52 Series** only.*

*This example requires one of the following SoftDevices: **S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

This example is based on the BLE Peripheral Heart Rate Application example. It implements the Heart Rate profile and enables pairing from a BLE Central device using the NFC interface. To do the pairing, it uses the NFC tag module that is one of the nRF5 peripherals and the NFC antenna delivered with the nRF5 Development Kit.

The application includes the same services that are present in the original Heart Rate Application example. See the documentation for that example for details. The following description presents only the differences from the original example.

When the application starts, it initializes and starts the NFCT peripheral, which is used for pairing. The application does not start advertising immediately, but only when the NFC tag is read by an NFC device, for example a smartphone or a tablet with NFC support. The message that the tag sends to the NFC device contains data required to initiate pairing. To start the NFC data transfer, the NFC device must touch the NFC antenna that is connected to the nRF5 Development Kit.

After reading the tag, the device can pair with the nRF5 device, which is advertising. After connecting, the example application behaves the same way as the original HRS example. Reading the NFC tag again when the application is in a connected state does not start advertising. When the connection is lost, advertising does not restart automatically. The NFC tag must be read again to restart advertising.

The behavior of this example depends on the operating system of the device that is used for pairing. The operation systems that implement BLE pairing over NFC behave differently after reading the NFC tag. The example is prepared for Android and Windows Phone devices, using a BLE Handover Select NDEF message for BLE pairing (see NFC_BLE_PAIR_MSG_FULL). This message contains three records:

- A Handover Select record (see Hs (Handover Select) records), which contains two Alternative Carrier records (see ac (Alternative carrier) records) that point to the two other records
- A Bluetooth LE OOB record (see LE OOB records), which is used for pairing with Android devices
- A Bluetooth EP OOB record (see EP OOB records), which is used for pairing with Windows Phone devices

With this Handover Select Message, only one NFC transfer is required to connect to the nRF5 device and get the heart rate measurements using the Heart Rate Monitor application for Android or Windows Phone.

Note
This application is not power optimized!

Alternative configuration

By default, this application uses a BLE Handover Select NDEF message, which is compatible with both Android and Windows Phone devices. However, you can use a simplified LE OOB NDEF message (see NFC_BLE_PAIR_MSG_BLUETOOTH_LE_SHORT, compatible only with Android devices) or a simplified EP OOB NDEF message (see NFC_BLE_PAIR_MSG_BLUETOOTH_EP_SHORT, compatible only with Windows Phone devices) instead.

This application uses Just Works pairing by default, because it is prepared for testing with Android and Windows Phone devices. However, you can configure the application to use Out-Of-Band pairing. In this case, you must define the required TK value in main.c using NFC_BLE_PAIRING_TYPE and OOB_AUTH_KEY.

Setup

The name of the example is **ble_app_hrs_pairing_nfc_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\experimental_ble_app_hrs_pairing_nfc`

Button assignments: BSP BLE Button Assignments.

Testing

Test the Heart Rate Application with BLE pairing over NFC with an Android smartphone or tablet that supports NFC by performing the following steps:

1. Compile and program the application.
2. Touch the NFC antenna with the smartphone or tablet and observe that the BSP_INDICATE_ADVERTISING state is indicated. LED 4 should be lit when the NFC tag detects the NFC field.
3. Confirm pairing with 'Nordic_HRM' in a pop-up window on the smartphone/tablet and observe that the BSP_INDICATE_CONNECTED state is indicated.
4. Test the Heart Rate Application using the nRF Toolbox app.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

24. nRF5 SDK v11.0.0-2.alpha

Experimental: Location and Navigation Application

This information applies to the following SoftDevices: ***S130, S132***

The Location and Navigation Service (LNS) Application is an example that implements the Location and Navigation Profile.

The application includes the three services in Location and Navigation Profile:

- Location and Navigation Service
- Battery Service
- Device Information Service

When the application starts, the Board Support Package is initialized. Timers are started to control the generation of various parts of the Location and Navigation Profile. Each time-out handler updates a simulation of sensor values. This includes updating

- The battery level.
- All fields of the Location and Navigation Service. The values are updated for demonstration purposes only and do not represent realistic measurements.

Note

This application is not power optimized!

The application will stop advertising after 3 minutes and go to system-off mode. Push Button 1 to restart advertising.

Setup

The name of the example is **experimental_ble_app_lns_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\experimental_ble_app_lns`

Button assignments: BSP BLE Button Assignments.

Testing

Test the Location and Navigation Application with the nRF Toolbox app, which is available on both iOS (Apple Store) and Android (Google Play).

You can also test the application with the Master Control Panel by performing the following steps:

1. Connect the board to the computer using a USB cable. The board is assigned a COM port, which is visible in the Device Manager.
2. Start a terminal emulator like PuTTY and connect to the used COM port with the following UART settings:
   - Baud rate: 38.400
   - 8 data bits

- 1 stop bit
- No parity
- HW flow control: RTS/CTS

3. Compile and program the application. To be able to debug the application, make sure to enable the ENABLE_DEBUG_LOG_SUPPORT and DM_DISABLE_LOGS compile flags. Observe that the BSP_INDICATE_ADVERTISING state is indicated and that the device is advertising with the device name "Nordic_LNS".
4. Observe that the text "[ADV] Starting fast advertisement." is printed on the COM listener running on the computer.
5. Connect to the device from Master Control Panel, then perform service discovery. Observe that the BSP_INDICATE_CONNECTED state is indicated.
6. Observe:
   - Primary service 0x1819 is the Location and Navigation Service.
     - Characteristic 0x2A6A is the feature characteristic UUID.
     - Characteristic 0x2A69 is the position quality characteristic UUID.
     - Characteristic 0x2A67 is the location and speed characteristic UUID.
     - Characteristic 0x2A68 is the navigation characteristic UUID.
     - Characteristic 0x2A6B is the control point characteristic UUID.
   - Primary service 0x180F is the Battery Service.
   - Primary service 0x180A is the Device Information Service.
7. Click the "Enable services" button in the Master Control Panel.
8. Observe that the text "LOC_SPEED_EVT: Notification enabled", "CTRL_PNT_EVT: Indication enabled", and "NAV_EVT: Notification enabled" is printed on the COM listener running on the computer.
9. In the Master Control Panel, observe that notifications are received from the location and speed characteristic. The control point is configured for indication. The navigation characteristic is configured for notification, but does not yet send notifications.
10. Select the LNS control point characteristic in the Master Control Panel.
11. Write 0301 and click "Write". This command corresponds to "navigation control" - "start". Verify that "NAV_EVT: Navigation state changed to 1" is printed on the COM listener.
12. In the Master Control Panel, observe that the navigation characteristic sends notifications.
13. Write 0300 and click "Write". This command represents "navigation control" - "stop". Verify that "NAV_EVT: Navigation state changed to 0" is printed on the COM listener.
14. In the Master Control Panel, observe that the navigation characteristic stops sending notifications.
15. Press the "Disconnect" button in the Master Control Panel. Observe that the BSP_INDICATE_ADVERTISING state is indicated.
16. If no peer connects to the application within 180 seconds (APP_ADV_TIMEOUT_IN_SECONDS), the application will put the chip into system-off mode.

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.

25. nRF5 SDK v11.0.0-2.alpha

Experimental: Multi Activity Example

This example requires one of the following SoftDevices: ***S130, S132***

Important: Before you run this example, make sure to program a SoftDevice.

The two examples included here (Advertisement Beacon and Scanner Beacon) illustrate how to use the Radio Timeslot API of the SoftDevice.

This folder contains an advertiser module that is shown in a heart rate sensor example and that advertises beacon data while in a connection.

Note
This application is not power optimized!

Advertisement Beacon

The advertiser beacon module illustrates using periodic timeslots on the radio when the SoftDevice is running. The example using the advertiser beacon is based on Heart Rate service example and starts advertising as a beacon when in a connection.

The name of the example is **experimental_ble_app_hrs_advertiser_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\experimental_ble_app_multiactivity_beacon\hrs_advertiser`

Scanner Beacon

The scanner beacon module illustrates using as much as possible of available radio time when the SoftDevice is running. The example using the advertiser beacon is based on Heart Rate service example and scans for advertising beacon at all times. If a beacon is found, indicated by the call to beacon_evt_handler, and if the major, minor and rssi values matches the expected one, the BSP_INDICATE_ALERT_3 is indicated (the alert indication is cleared when the device enters sleep mode, or leaves a connection).

The name of the example is **experimental_ble_app_hrs_scanner_*SoftDevice_board***, where *SoftDevice* is the SoftDevice that you are using and *board* is the supported development board. If you are not using the Keil Pack Installer, you can find the source code and project file of the example in the following folder:
`<InstallFolder>\examples\ble_peripheral\experimental_ble_app_multiactivity_beacon\hrs_scanner`

This document was last updated on Fri Dec 18 2015.
Please send us your feedback about the documentation! For technical questions, visit the Nordic Developer Zone.