BLE Central

Contents

nRF5 SDK v11.0.0-2.alpha

BLE Central

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

The following examples demonstrate the BLE Central role:

BLE Heart Rate Collector Example

BLE Multi-link Example

BLE Running Speed and Cadence Collector Example

Nordic UART Service Client

1. nRF5 SDK v11.0.0-2.alpha

BLE Heart Rate Collector Example

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

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

The BLE Heart Rate Collector application implements the Heart Rate Collector Role.

GAP role implemented is the Central role. GATT role implemented is the Client role.

The Heart Rate Collector application implements the Collector role for the Heart Rate Profile. It scans peripheral devices, connects to the device advertising with Heart Rate Service UUID in its advertisement report, and discovers and configures the **Heart Rate Service** to start sending Notifications of Heart Rate Measurement. The Heart Rate Measurement received is logged on UART interface. This application will also discover and configure **Battery Service** at the peer.

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

- Heart Rate Service Client
- Battery Service Client

Note

The application currently does not support connecting to multiple peripherals. Handling Connection Parameters Update and Pairing/Bonding are not implemented. Once connected, the application does not initiate disconnection but is capable of handling disconnection event initiated by peer or triggered unexpectedly.

Setup

The name of the example is **ble_app_hrs_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_central\ble_app_hrs_c`

Button assignments: BSP BLE Button Assignments.

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

Two boards are needed to perform this test:

- **Collector Board**: nRF5 development board containing the S13x SoftDevice.
- **Sensor Board**: nRF5 development board containing the S13x SoftDevice and the Heart Rate Application provided with the nRF5 SDK.

The application running on the Sensor Board is intended to serve as a peer (i.e. Heart Rate Sensor role) to this Heart Rate Collector application.

Test the BLE Heart Rate Collector Example application by performing the following steps:

1. Compile the BLE Heart Rate Collector Example application and program both SoftDevice and application on the Collector Board.
2. On the Collector board, observe that the BSP_INDICATE_ADVERTISING state is indicated. This shows that the application is scanning for Heart Rate Sensors.
3. Compile the Heart Rate Sensor application and program both SoftDevice and application on the Sensor Board.
4. Observe that the Heart Rate Sensor is advertising.
5. Once the connection is established, the BSP_INDICATE_CONNECTED state is indicated on the Collector Board.
6. After a few seconds, observe Heart Rate measurements displayed on the UART.

2. nRF5 SDK v11.0.0-2.alpha

BLE Multi-link Example

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

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

The BLE Multi-link Central application together with the BLE Multi-link Peripheral application tests the multi-link functionality of the S13x SoftDevice. The BLE Multi-link Example demonstrates how one central device can connect to multiple peripheral devices.

On startup, the peripheral devices will start advertising. Once connected to the central, pressing button 1 on the peripheral changes the state of the peripheral and a corresponding LED on the central. When multiple peripherals are connected, each peripheral is assigned one LED by the central depending on the order in which the peripherals got connected. For example, LED 1 will be assigned to the peripheral that connects first. LED 2 will be assigned to the peripheral that connects second and so on.

Setup

BLE Multi-link Central: The name of the example is **ble_app_multilink_central_*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_central\ble_app_multilink_central`

BLE Multi-link Peripheral: The name of the example is **ble_app_multilink_peripheral_*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_central\ble_app_multilink_peripheral`

Note

The BLE Multi-link Peripheral application needs the S13x SoftDevice to work. Its primary use is to act as helper application to test the BLE Multi-link Central application.

The BLE Multi-link Example requires at least three boards to verify that more than one link is active. The central device can have a maximum of 8 concurrent connections to peripheral devices.

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

- During Connection:
    - Button 1: Change state on peripheral side (BSP_INDICATE_ALERT_3/BSP_INDICATE_ALERT_OFF) and toggle assigned LED on central side

UART logging can be used instead of the LEDs.

Testing

Test the BLE Multi-link Example application by performing the following steps:

1. Compile and program the peripheral application to two boards. The BSP_INDICATE_ADVERTISING state is indicated on both boards. This means they are advertising and are waiting for the central device to connect.
2. Compile and program the central application onto another board.
3. As soon as the central application is programmed, it starts scanning and attempts to connect to the peripherals and enable notifications. Hence on the peripherals, observe that the BSP_INDICATE_CONNECTED state is indicated after a few seconds.
4. Press button 1 on one of the peripherals. Observe that BSP_INDICATE_ALERT_3 is indicated on that peripheral and one of the LEDs on the central turns ON.
5. Press button 1 on the same peripheral again. Observe that BSP_INDICATE_ALERT_OFF is indicated and the corresponding LED on the central is turned off.
6. Repeat the last two steps with other connected peripherals and observe the same behavior.

3. nRF5 SDK v11.0.0-2.alpha

BLE Running Speed and Cadence Collector Example

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

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

The BLE Running Speed and Cadence Collector application implements the Running Speed and Cadence Collector Role using the S13x SoftDevice.

GAP role implemented is the Central role. GATT role implemented is the Client role.

The Running Speed and Cadence Collector application implements the Collector role for the Running Speed and Cadence Profile. It scans peripheral devices, connects to the device advertising with Running Speed and Cadence Service UUID in its advertisement report, and discovers and configures the **Running Speed and Cadence Service** to start sending Notifications of Running Speed and Cadence Measurements. The received measurements are logged on UART interface.

The application includes one service from the Running Speed and Cadence profile:

- Running Speed and Cadence Service Client

Note

The application currently does not support connecting to multiple peripherals. Handling Connection Parameters Update and Pairing/Bonding are not implemented. Once connected, the application does not initiate disconnection but is capable of handling disconnection event initiated by peer or triggered unexpectedly.

Setup

The name of the example is **ble_app_rscs_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_central\experimental\ble_app_rscs_c`

Button assignments: BSP BLE Button Assignments.

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

Two boards are needed to perform this test:

- **Collector Board**: nRF5 Development board containing an S13x SoftDevice.

- **Sensor Board**: nRF5 Development board containing an S13x SoftDevice and the Running Speed and Cadence Application provided with the nRF5 SDK.

The application running on the Sensor Board is intended to serve as a peer (i.e. Running Speed and Cadence Sensor role) to this Running Speed and Cadence Collector application.

Test the BLE Running Speed and Cadence Collector Example application by performing the following steps:

1. Compile the BLE Running Speed and Cadence Collector Example application and program both the SoftDevice and the application on the Collector Board.
2. On the Collector board, observe that the BSP_INDICATE_ADVERTISING state is indicated. This shows that the application is scanning for Heart Rate Sensors.
3. Compile the Running Speed and Cadence Sensor application and program both the SoftDevice and the application on the Sensor Board.
4. Observe that the Running Speed and Cadence Sensor is advertising.
5. Once the connection is established, the BSP_INDICATE_CONNECTED state is indicated on the Collector Board.
6. After a few seconds, observe Running Speed and Cadence measurements displayed on the UART.

4. nRF5 SDK v11.0.0-2.alpha

Nordic UART Service Client

*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) Client Application is an example that implements the Nordic UART Service Client over BLE. In the example, the development board serves as a GAP central and a GATT client.

The application scans peripheral devices and connects to a device that advertises with the NUS UUID in its advertisement report. After connecting, the application enables notifications on the device that delivers the Nordic UART Service.

Note

This application will connect to only one device that delivers the NUS. If the application cannot find the NUS in the device discovery, it will not disconnect. This makes it possible to use more GATT clients at the same time.

Setup

The name of the example is **ble_app_uart_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_central\ble_app_uart_c`

Button assignments: BSP BLE Button Assignments.

The application uses the following UART settings:

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

Design overview

The responsibility of connecting to the NUS server is shared between the main program and the ble_nus_c module. The main program scans and reads the advertisement packets and connects to a device that delivers the NUS. After connecting, it starts service discovery. In this way, it is possible to scan for different services without needing to start unnecessary service discoveries.

The ble_nus_c module is responsible for parsing the service scan response and passing notifications back to the main program.

Design overview

Scanning and reading advertisement packets

The scanning parameters are set in the **m_scan_params** structure. Active scanning must be enabled, because the advertised service list is sent as scan response data.

The advertisement packet is parsed using the function **is_uuid_present**. If this function returns true, the application will connect to the device.

Handling events from ble_nus_c

The following code example shows how to handle events from the ble_nus_c module:

```
static void ble_nus_c_evt_handler(ble_nus_c_t * p_ble_nus_c, const
      ble_nus_c_evt_t * p_ble_nus_evt)
{
    uint32_t err_code;
    switch (p_ble_nus_evt->evt_type)
    {
        case BLE_NUS_C_EVT_FOUND_NUS_TX_CHARACTERISTIC:
            APPL_LOG("The device has the device TX
      characteristic\r\n");
            break;
        case BLE_NUS_C_EVT_FOUND_NUS_RX_CHARACTERISTIC:
            err_code = ble_nus_c_rx_notif_enable(p_ble_nus_c);
            APP_ERROR_CHECK(err_code);
            APPL_LOG("The device has the device RX
    characteristic\r\n");
            break;

        case BLE_NUS_C_EVT_NUS_RX_EVT:
            for (uint32_t i = 0; i < p_ble_nus_evt->data_len; i++)
            {
                while(app_uart_put( p_ble_nus_evt->p_data[i]) !=
    NRF_SUCCESS);
            }
            break;

        case BLE_NUS_C_EVT_DISCONNECTED:
            APPL_LOG("NUS device disconnected\r\n");
            scan_start();
            break;
    }
}
```

If the RX characteristic is found, notifications are enabled. When an RX notification is received, the data is printed out to UART.

Handling data received from UART

When data is received from UART, the data is split up in packets and sent using NUS. The following code example shows how to handle UART data:

```
        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)))
            {
                while (ble_nus_c_string_send(&m_ble_nus_c, data_array,
    index) != NRF_SUCCESS)
                {
                    // repeat until sent
                }
                index = 0;
            }
            break;
```

Testing

Two boards are needed to perform this test:

- **Central Board**: nRF5 development board containing an S13x SoftDevice.
- **Peripheral Board**: nRF5 development board containing an S13x SoftDevice and the UART peripheral application provided with the nRF5 SDK.

The application running on the Peripheral Board is intended to serve as a peer (for example, in the Nordic UART Service (NUS) Application role) to this NUS Client application.

Test the BLE NUS Client Example application by performing the following steps:

1. Compile the BLE NUS Client Example application and program both the SoftDevice and the application on the Central Board.
2. On the Central board, observe that the BSP_INDICATE_SCANNING state is indicated. This shows that the application is scanning for a NUS service Peripheral.
3. Compile the NUS application and program both the SoftDevice and the application on the Peripheral Board.
4. Observe that the NUS Peripheral is advertising.
5. Once the connection is established, the BSP_INDICATE_CONNECTED state is indicated on both boards.
6. Now it is possible to send data between the two boards. To do so, send data to one of the boards using the serial port. Observe that the data is displayed on the UART on the other board.