nRF5 SDK v11.0.0-2.alpha

Extending your application

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

You can extend any BLE application to support the BLE DFU Service, so that you can easily enter bootloader mode to update the device firmware at any time.

The following changes are required to add BLE DFU Service support to an application:

- The application must use the Device Manager. If the application does not use the Device Manager yet, you must update the application. See BLE Device Manager for more information.
- You must add DFU related files to the BLE application project. See Including DFU files in application.
- You must implement a function that executes application-specific operations to ensure a graceful shutdown of the application. See Implementing graceful shutdown.
- The application must correctly set up the BLE services. See Setting up the BLE services.
- The application must propagate SoftDevice BLE events to the BLE DFU Service. See Propagating BLE stack events.

With these changes to the application, the application can restart the device in bootloader mode to accept firmware updates. To remotely trigger the application to restart the device in bootloader mode, you must write "DFU Start" to the DFU Service Control Point. See Switching to bootloader/DFU mode.

Including DFU files in application

Perform the following steps to include the files that are required to support the BLE DFU Service in an application:

1. Open the application project in Keil.
2. Click **Project** > **Manage** > **Components, Environment, Books** or the respective icon.
3. On the Project Items tab, add a group with the name "Dfu".
4. Select the new group and click **Add Files**.
5. Add the following files:
    - **bootloader_util_arm.c**
      (Located in `C:\Keil\ARM\Pack\NordicSemiconductor\nRF_Libraries\<version>\bootloader_dfu` if you are using Keil packs and `<InstallFolder>\components\libraries\bootloader_dfu` if you are using the repository distribution variant of the SDK.)
    - **dfu_app_handler.c**
      (Located in `C:\Keil\ARM\Pack\NordicSemiconductor\nRF_Libraries\<version>\bootloader_dfu` if you are using Keil packs and `<InstallFolder>\components\libraries\bootloader_dfu` if you are using

the repository distribution variant of the SDK.)

- **ble_dfu.c**
  (Located in `C:\Keil\ARM\Pack\NordicSemiconductor\nRF_BLE\<version>\ble_services\ble_dfu` if you are using Keil packs and `<InstallFolder>\components\ble\ble_services\ble_dfu` if you are using the repository distribution variant of the SDK.)

Implementing graceful shutdown

Before the application switches to bootloader mode, it should shut down gracefully. To achieve that, you must implement a function that is called before the shutdown.

`dfu_app_handler.c` supports a reset_prepare callback function, which will be executed before the reset and allows the application to perform any tasks that are considered important before restarting in bootloader mode. For example:

- Gracefully disconnect and close any active BLE links
- Wait for pending flash operation to finish to ensure known state
- Clean up registers
- Turn off LEDs

The reset will not occur until the reset_prepare function returns.

See the following example of a reset_prepare function from `main.c` of the Heart Rate Application with BLE DFU Service support:

```
/**@brief Function for preparing for system reset.
 *
 * @details This function implements @ref dfu_app_reset_prepare_t. It will be
       called by
 *          @ref dfu_app_handler.c before entering the bootloader/DFU.
 *          This allows the current running application to shut down gracefully.
 */
static void reset_prepare(void)
{
    uint32_t err_code;

    if (m_conn_handle != BLE_CONN_HANDLE_INVALID)
    {
        // Disconnect from peer.
        err_code = sd_ble_gap_disconnect(m_conn_handle,
       BLE_HCI_REMOTE_USER_TERMINATED_CONNECTION);
        APP_ERROR_CHECK(err_code);
        err_code = bsp_indication_set(BSP_INDICATE_IDLE);
        APP_ERROR_CHECK(err_code);
    }
    else
    {
        // If not connected, the device will be advertising. Hence stop the
       advertising.
```

```
        advertising_stop();
    }

    err_code = ble_conn_params_stop();
    APP_ERROR_CHECK(err_code);

    nrf_delay_ms(500);
}
```

Setting up the BLE services

To be able to trigger an update remotely, you must initialize the BLE DFU Service and enable the Service Changed characteristic. In addition, you must make sure that the Device Manager knows about the new Service Changed characteristics.

Initializing the DFU Service

The BLE DFU Service is required to connect to the application and trigger a restart into bootloader mode. To be able to write to the BLE DFU Service, you must initialize the service in the application.

See the following example code from `main.c` of the Heart Rate Application with BLE DFU Service support:

```
#include "ble_dfu.h"
#include "dfu_app_handler.h"
```

```
    ble_dfu_init_t   dfus_init;

    // Initialize the Device Firmware Update Service.
    memset(&dfus_init, 0, sizeof(dfus_init));

    dfus_init.evt_handler   = dfu_app_on_dfu_evt;
    dfus_init.error_handler = NULL;
    dfus_init.evt_handler   = dfu_app_on_dfu_evt;
    dfus_init.revision      = DFU_REVISION;

    err_code = ble_dfu_init(&m_dfus, &dfus_init);
    APP_ERROR_CHECK(err_code);

    dfu_app_reset_prepare_set(reset_prepare);
    dfu_app_dm_appl_instance_set(m_app_handle);
```

Enabling the Service Changed characteristic

You must enable the Service Changed characteristic so that changes in the application are indicated to other devices, most importantly the DFU controller.

To enable the characteristic, simply set IS_SRVC_CHANGED_CHARACT_PRESENT to 1 in the application:

```
#define IS_SRVC_CHANGED_CHARACT_PRESENT   0
```

Updating the Device Manager configuration

The Device Manager keeps track of the characteristics that are enabled for an application. Therefore, you must update the Device Manager configuration to account for the two new Service Changed characteristics (one for the application itself and one for the DFU Service).

Set the maximum number of characteristic client descriptors in the file `device_manager_cnfg.h` (located in `C:\Keil\ARM\Pack\NordicSemiconductor\nRF_BLE\<version>\device_manager\config` if you are using Keil packs and `<InstallFolder>\components\ble\device_manager\config` if you are using the repository distribution variant of the SDK):

```
#define DM_GATT_CCCD_COUNT                  4
```

Propagating BLE stack events

The BLE DFU Service relies on BLE stack events. Therefore, for the service to function properly, all BLE Stack events from the SoftDevice must be propagated to the DFU Service.

To propagate BLE stack events to the BLE DFU Service, call ble_dfu_on_ble_evt with the BLE stack event as a parameter. In most SDK examples, BLE stack events are propagated to submodules in the function ble_evt_dispatch(ble_evt_t * p_ble_evt) in `main.c`.

See the following example code from `main.c` of the Heart Rate Application with BLE DFU Service support:

```
    ble_dfu_on_ble_evt(&m_dfus, p_ble_evt);
```

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 <https://devzone.nordicsemi.com/questions/>.