BLE OTA External Memory Bootloader and Bootloadable Example Projects

Features

- Over The Air (OTA) firmware update
- Hidden BLE service to receive bootloadable images
- Storage of received images in external memory
- Device Information Service (DIS)
- LED status indication

General Description

This example project demonstrates an OTA firmware update using the BLE Bootloader Service that is hidden by default and can be activated by pressing button **SW2** while the device is running. By default, this is a regular bootloadable project that contains the BLE component with the Device Information Service. Once the bootloader mode is enabled, this example project is ready for receiving a new image of the bootloadable project and storing it to the external memory.

Development Kit Configuration

This example project is designed to run on the CY8CKIT-042-BLE from Cypress Semiconductor. A full description of the kit, along with more example programs and ordering information, can be found at http://www.cypress.com/go/cy8ckit-042-ble.

Make sure that the kit is powered by 3.3 V (J16 is set to 1 and 2). No connection on the kit board is required to use this example project.

Refer to Section "Functional Description" for instructions on this example project usage.

Bootloader Project Configuration

The Bootloader project consists of the Bootloader component and SCB (I^{2}C master).

The purpose of this project is to replace a bootloadable image stored in the internal memory with a bootloadable image that is stored in the external memory.

After a bootloadable image in the internal memory is replaced, the bootloader project invalidates the image in the external memory and resets the device with a scheduled bootloadable launch after a startup.

SCB

This project uses the SCB component in the I^{2}C Master mode for communication with the external memory located on the CY8CKIT-042-BLE kit board.

Bootloader

This project also uses the Bootloader component configured to work with the custom interface based on the SCB component.

Bootloadable Project Configuration

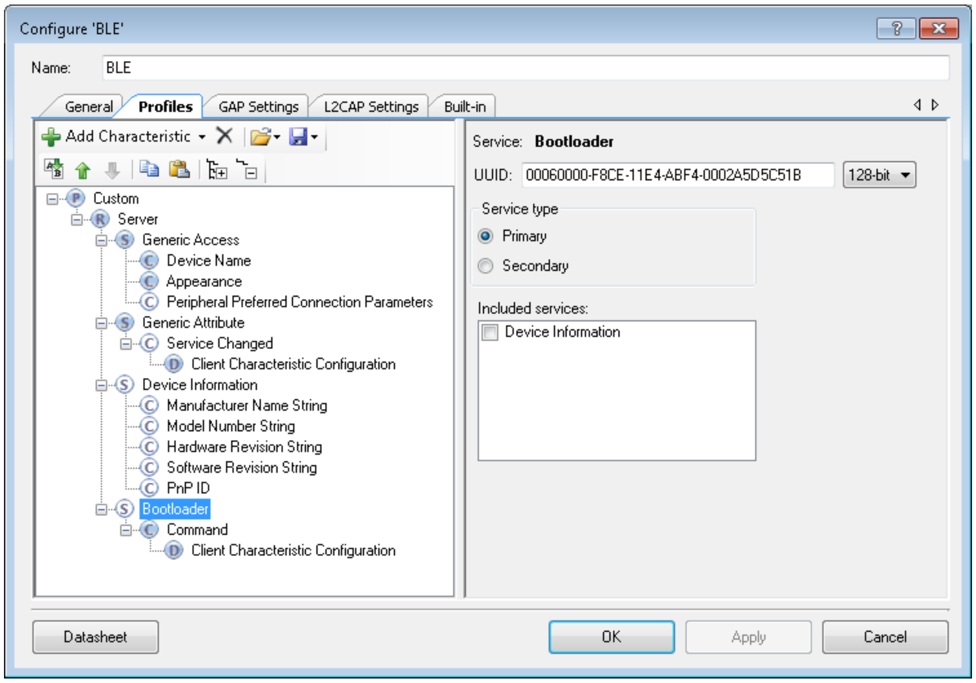
This project consists of the following components:

- Bootloadable
- SCB (I^{2}C in Master mode)
- BLE (central role, DIS and Bootloader services)
- Pins

BLE

The BLE component is used to implement the BLE Device Information Service (DIS) and a hidden Bootloader Service. The purpose of the Bootloader Service is to receive other bootloadable images for further storing in the external memory. The Bootloader service is enabled when **SW2** is pressed.

Figure 1. BLE GATT Settings



Name: BLE

Service: **Bootloader**

UUID: 00060000-F8CE-11E4-ABF4-0002A5D5C51B 128-bit

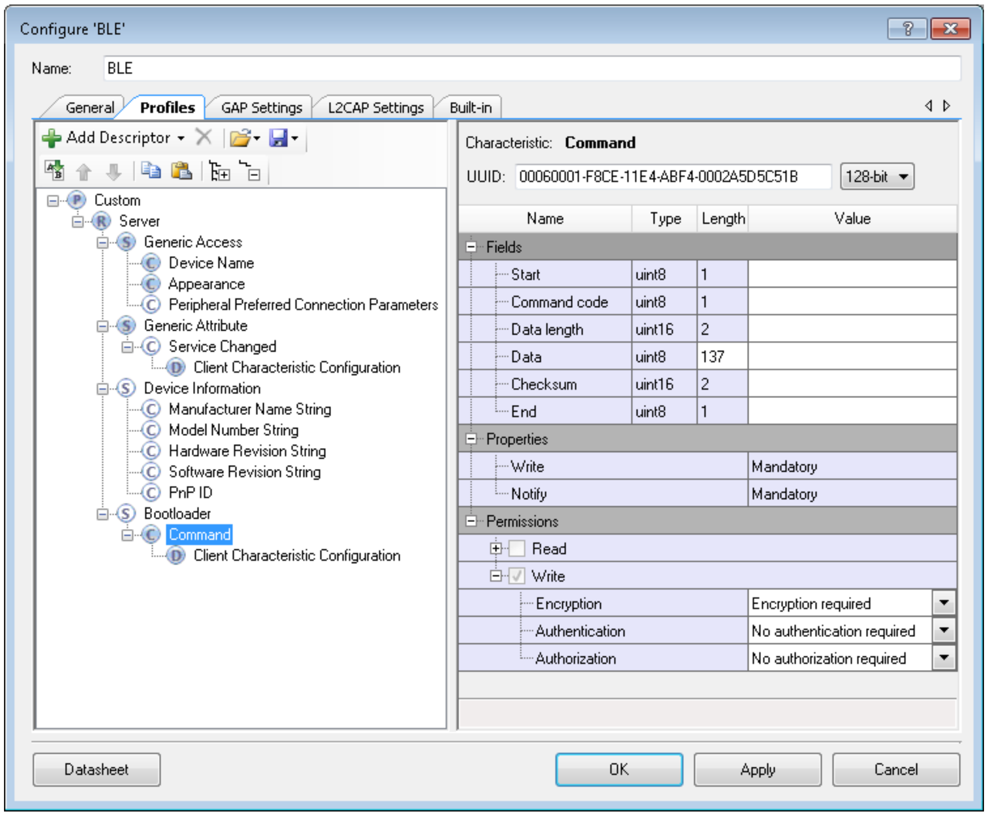
Service type

[● Primary] Secondary

Included services:

☐ Device Information

Figure 2. BLE GATT Settings



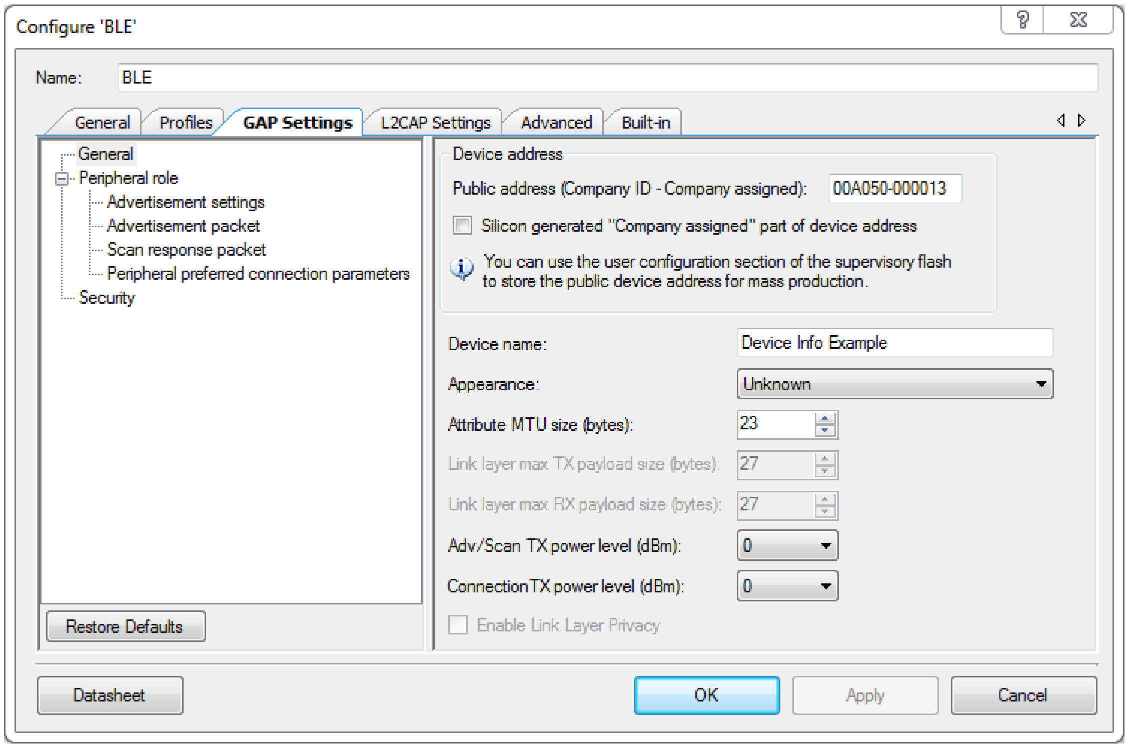
Name: BLE

Characteristic: **Command**

UUID: 00060001-F8CE-11E4-ABF4-0002A5D5C51B 128-bit

Name	Type	Length	Value
Fields			
Start	uint8	1	
Command code	uint8	1	
Data length	uint16	2	
Data	uint8	137	
Checksum	uint16	2	
End	uint8	1	
Properties			
Write			Mandatory
Notify			Mandatory
Permissions			
☐ Read			
✓ Write			
Encryption			Encryption required
Authentication			No authentication required
Authorization			No authorization required

Figure 3. BLE GAP Settings


Configure 'BLE'

Name: BLE

General | Profiles | GAP Settings | L2CAP Settings | Advanced | Built-in

- General
- Peripheral role
  - Advertisement settings
  - Advertisement packet
  - Scan response packet
  - Peripheral preferred connection parameters
- Security

Restore Defaults

Device address

Public address (Company ID - Company assigned): 00A050-000013

☐ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name:	Device Info Example
Appearance:	Unknown
Attribute MTU size (bytes):	23
Link layer max TX payload size (bytes):	27
Link layer max RX payload size (bytes):	27
Adv/Scan TX power level (dBm):	0
Connection TX power level (dBm):	0

☐ Enable Link Layer Privacy

Datasheet | OK | Apply | Cancel

The BLE component has two services: the Device Information Service that is always available, and the Bootloader Service that is hidden by default. Operation of the DIS service is well described in a separate dedicated example project.

The Bootloader Service can be enabled if button **SW2** is pressed for longer than 0.1 second. The purpose for this service is to emulate a bootloader component interface during communication with the Bootloader Host.

The Bootloader Host is a software application used to send firmware images to a device. For BLE OTA bootloading, Cypress provides an example of the Bootloader Host - CySmart. See Section Upgrade Project Images with CySmart for more details.

The Bootloader Service has one characteristic that supports the “Write” procedure and notifications and it also has a characteristic descriptor – the Client Characteristic Configuration.

For communication, the Bootloader Host writes a command to the Command Characteristic and enables notifications in the characteristic descriptor.

The bootloader emulator reads the command from the characteristic, processes it, and if notifications are enabled in the characteristic descriptor, writes a response to the notification that is sent to Bootloader Host.

The Bootloader Host receives the notification and depending on its content (a bootloader emulator response) either sends the next command or reports an error if any.

If a command contains a valid flash row, the bootloader emulator writes this row to the external memory. After the transfer of a bootloadable image is completed, the bootloader emulator writes metadata to the external flash as well as a flag for the bootloader to replace the bootloadable image.

At the next device reset, the bootloader is expected to replace the bootloadable image with an image from the external flash if a flag in the external flash is set and if the bootloadable image from the external flash is valid.

The BLE component is configured to have an MTU of 23 bytes and the Bootloader Service UUID 00060000-F8CE-11E4-ABF4-0002A5D5C51B.

After pressing **SW2** onboard, the LED changes its indication from green to red, this can be applied to both “advertising” and “connected” modes.

Note The bootloadable project is performing FRAM writes. Due to this maximum connection, the interval value is set to 1000 ms. Changing this parameter value to a smaller value might result in disconnection during bootloading if an external memory write takes longer.

SCB

This project uses the SCB component in the I^2C Master mode for communication with the external memory located on the CY8CKIT-042-BLE kit board. More detail on the external FRAM memory is available in the datasheet – http://www.cypress.com/?mpn=FM24V10-G.

Bootloadable

The Bootloadable component is used to create an image with bootloadable firmware that can be updated without affecting the bootloader.

External Memory

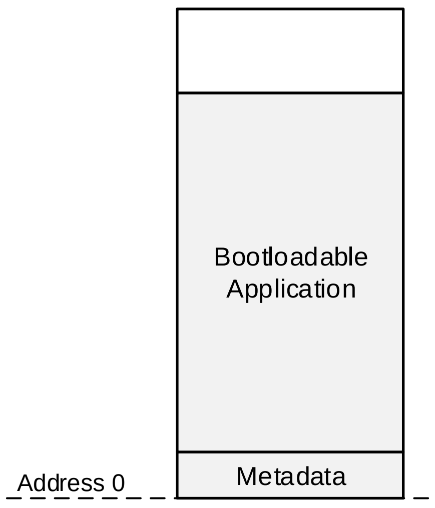
External Memory Interface

The SCB component in the I^2C Master mode is used for communication with the external memory. This project is designed to work with 1-Mbit ferroelectric random access memory (F-RAM) present on the CY8CKIT-042-BLE kit board. More details on the used memory can be found in the device datasheet http://www.cypress.com/?mpn=FM24V10-G.

The external memory interface API designed in a generic way, so the used F-RAM device can be easily adopted for other external memory devices (I2C EEPROM, SPI EEPROM, etc). The external memory API is available in the *ota_mandatory.c* file.

Memory Map

The external memory contains “Metadata” and “Application” sections.



Metadata

The metadata section is a 128-byte block of memory used as a common area for both bootloader and bootloadable applications. The metadata section is placed at the beginning of the external memory.

Address	Size, Bytes	Description
EMI_MD_APP_STATUS_ADDR	1	The status of the application image located in the external memory. The following values are recognized: • EMI_MD_APP_STATUS_VALID – The application is valid. • EMI_MD_APP_STATUS_LOADED – The application is valid and copied to the internal flash. • EMI_MD_APP_STATUS_INVALID – The application is invalid (it was not copied there or its checksum does not match the checksum value stored in the external memory metadata section).
EMI_MD_ENCRYPTION_STATUS_ADDR	1	Stores an encryption status.
EMI_MD_APP_EM_CHECKSUM_ADDR	2	Stores the application image checksum.
EMI_MD_APP_SIZE_IN_ROWS_ADDR	2	Stores the application image size in flash rows.
EMI_MD_APP_FIRST_ROW_NUM_ADDR	2	Stores the number of the first flash row of the application.
EMI_MD_EXTERNAL_MEMORY_PAGE_SIZE_ADDR	1	The external memory page size.

Application

The “Application” section starts immediately after the “Metadata” section. The size of the ”Application” section is stored in the ”Metadata” section (EMI_MD_APP_SIZE_IN_ROWS_ADDR).

Functional Description

Encryption

This example project contains encryption APIs. If the image stored on the external FRAM memory must be encrypted, the user can enable the encryption algorithm. The encryption APIs use the same algorithms as the BLE stack. In this case, all the information is encrypted before it is stored to the external FRAM and decrypted for while writing to the internal memory. The metadata part is not encrypted.

The APIs are located in the *ota_optional.c* files for both bootloader and bootloadable projects.

The encryption algorithm is enabled by setting the define value ENCRYPT_ENABLED to “YES” in files *options.h* for bootloadable example projects. The ENCRYPT_ENABLED value must be always set to “YES” for a bootloader project. Enabling of encryption in a bootloader project is controlled by EMI_MD_ENCRYPTION_STATUS_ADDR in metadata.

Using UART for Debugging

In these example projects, the UART component is used for printing various debug information (disabled by default).

File *options.h* contains the define “DEBUG_UART_ENABLED” set to “NO”. If extra debugging information must be provided, this define should be set to “YES” in each: the bootloader or bootloadable or both. This will decrease the project’s performance, but it will provide extra debugging output to the UART.

A HyperTerminal program is required in the PC to receive debugging information. If you don’t have a HyperTerminal program installed, download and install any serial port communication program. Freeware such as HyperTerminal, Bray’s Terminal, Putty etc. is available on the web.

1. Connect the PC and kit with a USB cable.
2. Open the device manager program in your PC, find the COM port in which the kit is connected, and note the port number.
3. Open the HyperTerminal program and select the COM port in which the kit is connected.
4. Configure the Baud rate, Parity, Stop bits and Flow control information in the HyperTerminal configuration window. By default, the settings are: Baud rate – 57600, Parity – None, Stop bits – 1 and Flow control – XON/XOFF. These settings have to match the configuration of the PSoC Creator UART component in the project
5. Start communicating with the device as explained in the project description.

File *debug.h* contains macros used for printing various types of data:

- DBG_PRINT_TEXT(a) - Prints a text string.
- DBG_PRINT_DEC(a) – Prints a decimal number.
- DBG_PRINT_HEX(a) – Prints a hexadecimal number.
- DBG_PRINT_ARRAY(a,b) – Prints ***b*** first elements of array ***a***.
- DBG_PRINTF(...) – Prints the function macro.

These macros are used for printing information to UART only if DEBUG_UART_ENABLED define is set to “YES”.

Checksum type option

The bootloader component allows choosing the type of a checksum. The type of a checksum in the bootloader project is chosen by the CI_PACKET_CHECKSUM_CRC value. If CI_PACKET_CHECKSUM_CRC is set to “YES”, the CRC-16 CCITT checksum algorithm will be used for the checksum calculation. If CI_PACKET_CHECKSUM_CRC is set to “NO”, the basic summation checksum algorithm will be used for the checksum calculation. Choose the same checksum type in the bootloader component customizer and in bootloadable project options (Option.h) to make the projects work.

Project options summary

Option	Value	Explanation
ENCRYPT_ENABLED	YES	Enable encryption of the external memory bootloader image. Always set “YES” to the bootloader project.
	NO	Disable encryption of the external memory bootloader image (Default).
DEBUG_UART_ENABLED	YES	Enable the output of debug messages to UART.
	NO	Disable the output of debug messages to UART (Default).
CI_PACKET_CHECKSUM_CRC	YES	Set a checksum type to the CRC-16 CCITT checksum algorithm.
	NO	Set a checksum type to the basic summation checksum algorithm (Default).

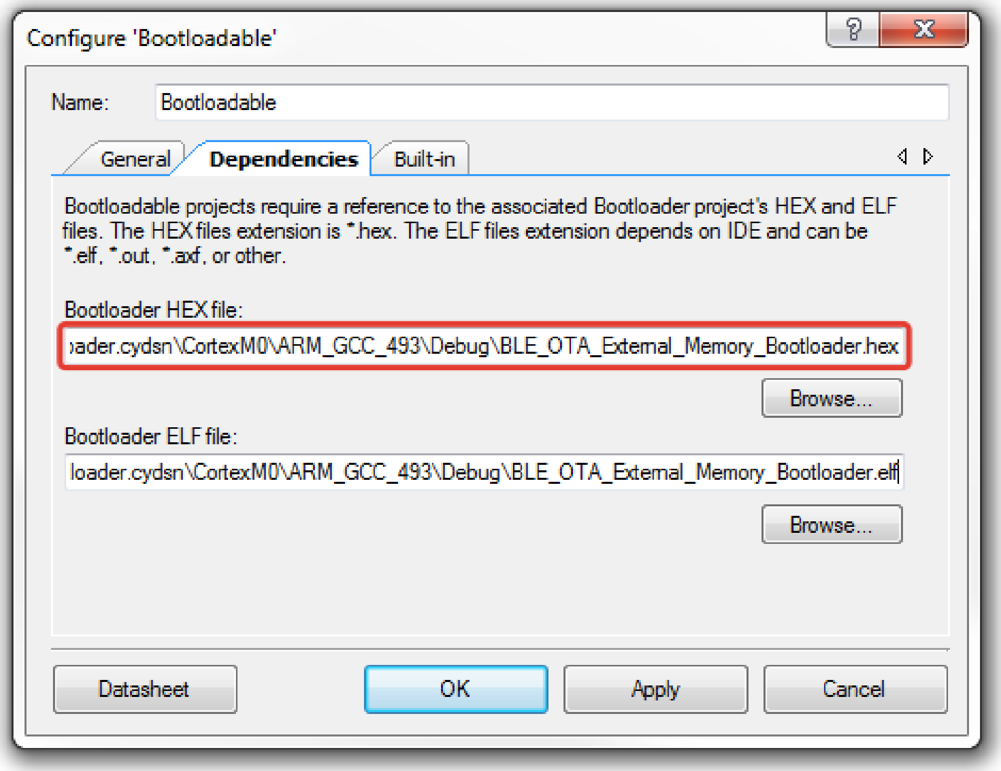
Setup and Run Example Project

Building Example Project

1. Build the *BLE_OTA_External_Memory_Bootloader* example project.
2. Add the *BLE_OTA_External_Memory_Bootloadable* example project to the workspace.

3. Open the top design schematic of the *BLE_OTA_External_Memory_Bootloadable* project. Specify the path to the bootloader project HEX and ELF files by double-clicking the Bootloadable component and going to the **Dependencies** tab and link Bootloadable to the BLE_OTA_External_Memory_Bootloader.hex file, as **Figure 4** shows.

Figure 4. Bootloadable Component Configuration


Configure 'Bootloadable'

Name: Bootloadable

Bootloadable projects require a reference to the associated Bootloader project's HEX and ELF files. The HEX files extension is *.hex. The ELF files extension depends on IDE and can be *.elf, *.out, *.axf, or other.

Bootloader HEX file: ›ader.cydsn\CortexM0\ARM_GCC_493\Debug\BLE_OTA_External_Memory_Bootloader.hex

Bootloader ELF file: loader.cydsn\CortexM0\ARM_GCC_493\Debug\BLE_OTA_External_Memory_Bootloader.elf

4. Build and Program the *BLE_OTA_External_Memory_Bootloadable* project.

   At this point, the CYC8CKIT-042-BLE contains firmware that can receive new updates over-the-air. The LED3 flashes green.

   In the *BLE_OTA_External_Memory_Bootloadable* example project's file *main.h*, there is #define LED_ADV_COLOR with value LED_GREEN. Change its value to LED_BLUE and build the project without flashing it.

Upgrading Project images

To perform any application update, enable the Bootloader service, see Section Bootloader for more information.

There is one software tool provided by Cypress that can send Stack or Application project image updates to device - CySmart. It is external software to be downloaded and installed separately.

There are two hardware Dongles that communicate with the BLE devices and can be used to update firmware:

- CY5670, with 128KB Flash chip, supports BLE 4.1.
- CY5677, with 256KB Flash chip, supports BLE 4.2. It supports higher transfer rates, up to 1Mbps.

Refer to the documentation that came with the dongle to figure out its type. Also, refer to CySmart User Guide.

Upgrade Project Images with CySmart

More detailed information about the CySmart usage can be found in the CySmart User Guide (Section 2.7 “Updating Peripheral Device Firmware”). You can download the CySmart tool and its user guide from here http://cypress.com/cysmart.

To upgrade the device firmware OTA:

1. Make sure the OTA device is running the Stack project image and is ready to receive updates.
2. Connect the BLE Dongle.
3. Open the CySmart software, and select the BLE Dongle.
4. Press the **Start Scan** button to start scanning for the peripheral device.
5. When the device is listed in the Discovered Device list, stop scanning by clicking the **Stop Scan** button.
6. Select the device from the Discovered Device list.
7. Click the **Update Firmware** button.
8. Select the firmware update type on the OTA firmware update window.

Firmware Update Type	Description
Application and Stack (Combined) update	The Application and Stack firmware co-exist in the same memory location. This option updates complete firmware.
Application only update	The Application and Stack firmware exist in independent memory locations. This option updates only the application firmware.
Application and Stack update	The Application and Stack firmware exist in independent memory locations. Select this option to update first the tSack firmware and then the Application.

9. Click **Next**. In the next page, browse and select the new firmware image (*.cyacd) file. It is located in the project folder ([*project folder*]\CortexM0\[*compiler name*]).
10. Click the **Update** button.
11. Wait for the device firmware to be updated.

Note If the update fails at authentication, then check if the security settings in CySmart match the settings used in the BLE component of the stack project. To change the security settings in CySmart, click on the **Configure Master Settings** button in the tool and go to the **Security parameters** page.

Expected Results

After a firmware update, the device should work as it did before, except the LED indication – now the color used for advertisement indication is blue instead of green.

If there are more changes to the *BLE_OTA_External_Memory_Bootloadable* project, repeat all the steps described for an OTA firmware update in Section "Project Description".