SimpleLink™ *Bluetooth*® low energy CC2640 wireless MCU

Over-the-Air Download User's Guide

For BLE-Stack™ Version: 2.1.0 / 2.1.1

Table of Contents

Table of Contents

1 Revision History

Date	Author's Name	Doc Revision	Section(s)	Summary
12/17/14	SP	1.0	5, 6, 7, 8, 9, 13, 14	Edited for correctness
2/16/2015	SP	2.0		Finalization
7/14/2015	CK	2.1	All	Reorganized chapters / Added Off-chip OAD / Removed OAD Manager / Added downloading methods / Added compiler dependencies / Added sequence diagrams
1/14/2016	ZH	2.1.1	All	Re-organized for user-friendliness / Separate Off-chip and On-Chip OAD chapters / Added Appendix chapter

2 References

1. Over-the-Air Download for CC254x.doc
2. Texas Instruments CC2640 Bluetooth® low energy Software Developer's Guide http://www.ti.com/lit/pdf/swru393

3 Definitions, Abbreviations, Acronyms

Term	**Definition**
BIM	Boot Image Manager, the software bootloader
BLE	Bluetooth low energy wireless protocol
CCC	Client Characteristic Configuration
CCFG	Customer Configuration Area, contains lock-bits on flash page 31
CCS	Code Composer Studio
MCU	Microcontroller Unit
OAD	Over the Air Download
RCFG	RTOS in ROM Configuration Table
ROM	Read Only Memory
RTOS	Real Time Operating System
SNV	Simple Non-Volatile storage
TI	Texas Instruments
TI-RTOS	Texas Instruments Real Time Operating System

4 Purpose

This document describes the process by which a developer can enable a SimpleLink™ Bluetooth® low energy CC2640 wireless MCU project and application to successfully implement the TI OAD Profile to remotely upgrade the image on a CC2640 BLE device, a process heretofore referred to as an Over-the-Air Download or OAD. This process provides tremendous value to a BLE product solution, as a target device does not need to be physically accessed to provide a software upgrade. Our purpose here is to make OAD simple by providing detailed instructions from enabling OAD in an application project to receiving the new image over the air, verifying its correctness, and running it from the bootloader.

5 Scope

This document provides instruction on how to setup a BLE-Stack™project to be OAD enabled, such as our example SimpleBLEPeripheral project for OAD. An overview of the OAD architecture and how to build, download, and debug the components shall be discussed here as well. Details about the BLE runtime system on CC26xx devices and the interrupt vector tables will be discussed as is necessary to elucidate how an OAD build differs from a project which does not provide OAD capability.

There are two different types of OAD. One is On-chip OAD which doesn't require any additional hardware and the other is Off-chip OAD which supports a system where an external flash memory is equipped. In the sample application projects, it is assumed that the hardware is SensorTag where 512-kB external flash memory is connected to the CC2650 via SPI.

Topics omitted from this document are reducing the size of an application which does not meet the size limitation (see 7.1), or an On-chip OAD on the BLE stack image or the bootloader. It is important to note that an On-chip OAD of the application image does not include an upgrade of the BLE Stack image as well. Also OAD for the SensorTag project is not covered in this guide, see the SensorTag User's Guide wiki for more information: http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User's_Guide.

Note: On-chip OAD is currently only supported by IAR with CC2640 devices, while Off-chip OAD works with both IAR and CCS.

Also, it is assumed that the developer is familiar with the CC2640 Software Developer's Guide [2], including the dual-image architecture used by the SDK.

For additional support, please visit the following online resources:

- TI Bluetooth LE Wiki-page: www.ti.com/ble-wiki
- TI E2E support forum: www.ti.com/ble-forum

6 Off- Chip OAD

6.1 Constraints and Requirements for Off-chip OAD

Using the internal flash of CC2640F128, the first page and the last page, or 8kB in total, of flash are reserved for the flash interrupt vectors and the BIM. The flash page 31 or the last page starting at address 0x1F000 where BIM is located is shared with the CCFG. Neither the first page nor the BIM is designed to be upgraded by Off-chip OAD.

An off-chip flash component of at least 120kB plus space for a 16 byte image metadata block is required for a full flash update. A SPI connection is used to communicate with the off-chip flash component.

The OAD image to be downloaded to the off-chip flash memory can be an application image, a stack image, a hex merge of application plus stack, an image intended for the upgrade of a network processor, or any type of image as far as it is supposed to eventually replace any part of the on-chip 120kB area between the first and the last pages. More than one image can be downloaded before the system reset followed by BIM's copying the downloaded images from the off-chip flash memory to the on-chip flash memory.

Since page 0 cannot be updated in Off-chip OAD, an application must include its own TI-RTOS instance in flash without dependency on the TI-RTOS ROM implementation. Also, it must include OAD profile so that further OAD upgrades are available when it runs on the on-chip flash memory since Off-chip OAD doesn't require any OAD-dedicated application like Image A for On-chip OAD.

The first and last flash pages must never be attempted to update because a power cycle during an update of either page could render the device unresponsive until physically reprogrammed.

6.2 System Overview

The OAD system, depicted in Figure 1, is comprised of OAD Image, Downloader and OAD Target. The OAD Image is the image file to be downloaded to OAD Target and should be in intel hex(.hex) format. The Downloader can be BLE Device Monitor running on PC with CC2540 dongle where HostTest application is programmed or any proprietary application or smartphone app where the OAD Profile client described in 6.3 is implemented. The OAD Target is the device to be upgraded with new image(s). To use Off-chip OAD, the OAD Target must have an off-chip flash memory component in it.

While On-chip OAD Target receives only one application OAD image, Off-chip OAD Target can receive up to 3 OAD images. Downloader generates metadata of OAD Images for the Off-chip OAD Images since they don't contain header information in the beginning unlike On-chip OAD image. The metadata is inserted into the beginning of the Off-chip OAD Image when transferred and is stored separately in the off-chip flash.

TI-proprietary OAD Profile, defined for communications between downloader and OAD Target, supports image identification, image block request/response and image count setup, with corresponding characteristics.

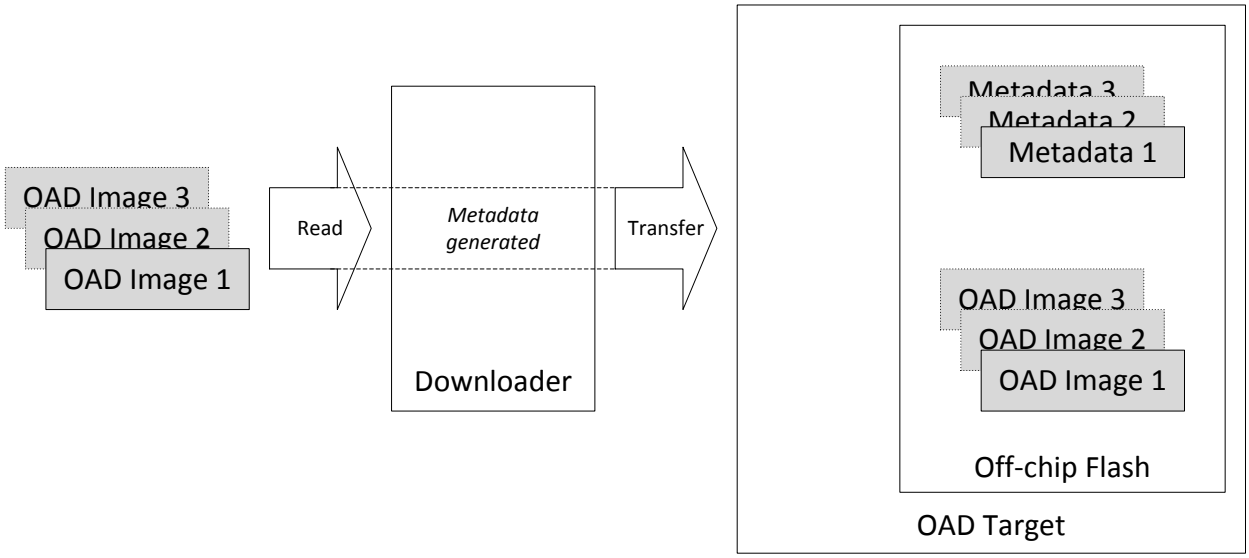

Figure 1. Off-chip OAD System Overview

6.3 OAD Profile

This Profile has been designed to provide a simple and customizable OAD Profile for the customer. In its most rudimentary form, for On-chip OAD, this profile is responsible for accepting an OAD interaction based on image header criteria, storing the image onto the on-chip flash in the OAD Target device and causing a device reset if the download is successful so that the downloaded application image is run by the BIM. For Off-chip OAD, this profile is responsible for accepting an OAD interaction based on image metadata criteria, storing the image(s) and the metadata(s) onto the off-chip flash component in the OAD Target device and causing a device reset if all the downloads are successful so that the downloaded images are copied onto the on-chip flash and then run by the BIM. Downloader and OAD Target perform Client role and Server role respectively.

6.3.1 Dependencies

The OAD Profile depends on the final customer application to determine the connection parameters and application specific criteria for initiating the OAD process.

6.3.2 Messages

The "write with no response" GATT message has been chosen as the default message type from Downloader to the OAD Target in order to reduce code size and increase data throughput as much as possible. This decision was made because adding the ability to send GATT Notifications requires adding the GATT_ClientInit() call which would increase code size noticeably. In a noisy or otherwise poor RF-environment, the "write with no response" GATT message may not be sufficient to successfully transmit an entire image and software driven timeouts and re-tries may need to be added. Since the Downloader will have already initialized the GATT_Client, notifications were chosen as the default message type from the OAD Target to the Downloader.

6.3.3 Characteristics

The OAD Profile has only three characteristics. The burden is on the Downloader to discover the handles of these three characteristics on the OAD Target. Structures of these characteristics are as shown in Figure 2.

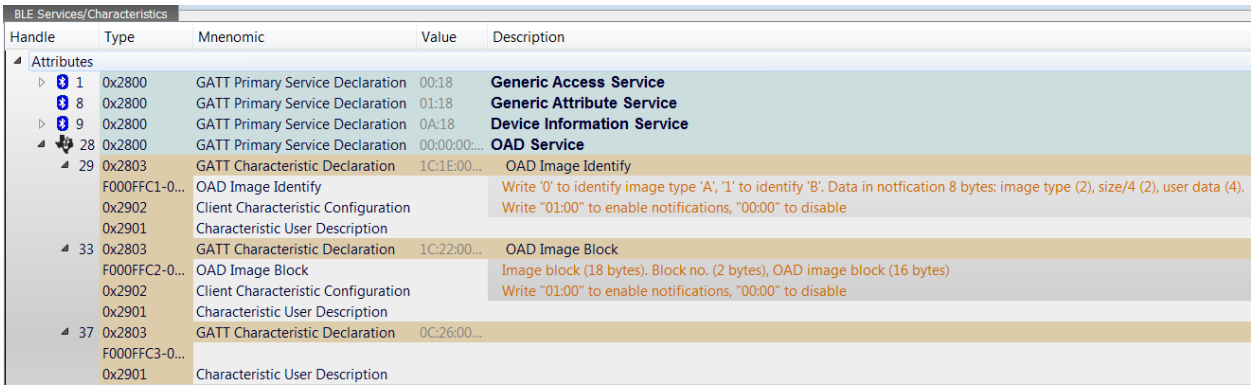

Figure 2. OAD Characteristics from BLE Device Monitor

6.3.3.1 OAD Image Identify Characteristic

The Image Identify characteristic is used to exchange either the image header information embedded in the OAD image for On-chip OAD or the metadata generated by Downloader for the OAD Image for Off-chip OAD in order for the OAD Target to decide if an OAD should occur. "01:00" shall be written to the CCC of this characteristic so that notification for reject is enabled.

6.3.3.1.1 Image Metadata

Image Metadata is initially generated by Downloader, written to OAD Image Identify characteristic, and then stored in the off-chip flash with updated CRC Shadow by OAD profile on the OAD Target. The State field can be modified by the BIM later. Unlike OAD Image for On-chip OAD, OAD Image for Off-chip OAD doesn't have the metadata embedded. Instead, it has just 2-byte CRC and 14-byte 0xFF's in the beginning. Anyway, the first 16 bytes can be ignorable. Metadata's fields are described in Figure 3.

Field	Size (in byte)	Description
CRC	2	
CRC Shadow	2	
Version	2	
Length	2	The actual length of the image is 4 x Length in byte, where Length is multiple of 4.
UID	4	"EEEE"
Start Address	2	The start address in the on-chip flash memory the downloaded image is to be copied to. The actual start address will be 4 x Start Address, where Start Address is multiple of 4 for 16-byte alignment.
Image Type	1	1: Application or Application+Stack 2: Stack 3: Network Processor
State	1	0xFF: Image should be copied to on-chip flash 0x80: Image has already been copied.

Figure 3. Image Metadata for Off-chip OAD

6.3.3.2 OAD Image Block Characteristic

The OAD Image Block characteristic is used to request and transfer a block of the OAD image. If "01:00" is written to the CCC of this characteristic, notification for reject will be enabled. "01:00" shall be written to the CCC of this characteristic so that notification for block request is enabled.

6.3.3.3 OAD Image Count Characteristic

The OAD Image Count characteristic is used to set the number of OAD images to be downloaded. This is used for only Off-chip OAD and the default value of the characteristic is 1.

6.3.4 Initiation of the Off-chip OAD Process

After establishing a new connection and updating the connection interval for a faster OAD and enabling notifications of OAD Image Identify and OAD Image Block characteristics on the OAD Target, optionally the user can have Downloader write the number of OAD Images to OAD Image Count characteristic. Then, the Downloader shall write to the Image Identify characteristic of the OAD Target. The message data will be the metadata of the OAD Image available for OAD. The metadata is generated based on the address retrieved from the OAD Image of .hex format and the image type specified on the Downloader by the user.

Upon receiving the write request to the Image Identify characteristic, the OAD Target will check the version information in the metadata because only the OAD Image of the version 0 or the newer version than the current running image can be accepted.

If the OAD Target determines that the image available for OAD is acceptable, the OAD Target will initiate the OAD process by notifying the Image Block Transfer characteristic to the Downloader requesting the first block of the new image. Otherwise, if the OAD Target finds that the new image does not meet its criteria to begin the OAD process, it shall respond by notifying the Image Identify characteristic with its own Image metadata as sign of rejection. In that case, the OAD procedure will end at the moment where dotted 'X's are placed as depicted in **Figure 4.** The gray backgrounded procedures in Figure 4 will be repeated as many times as OAD Image Count characteristic value.

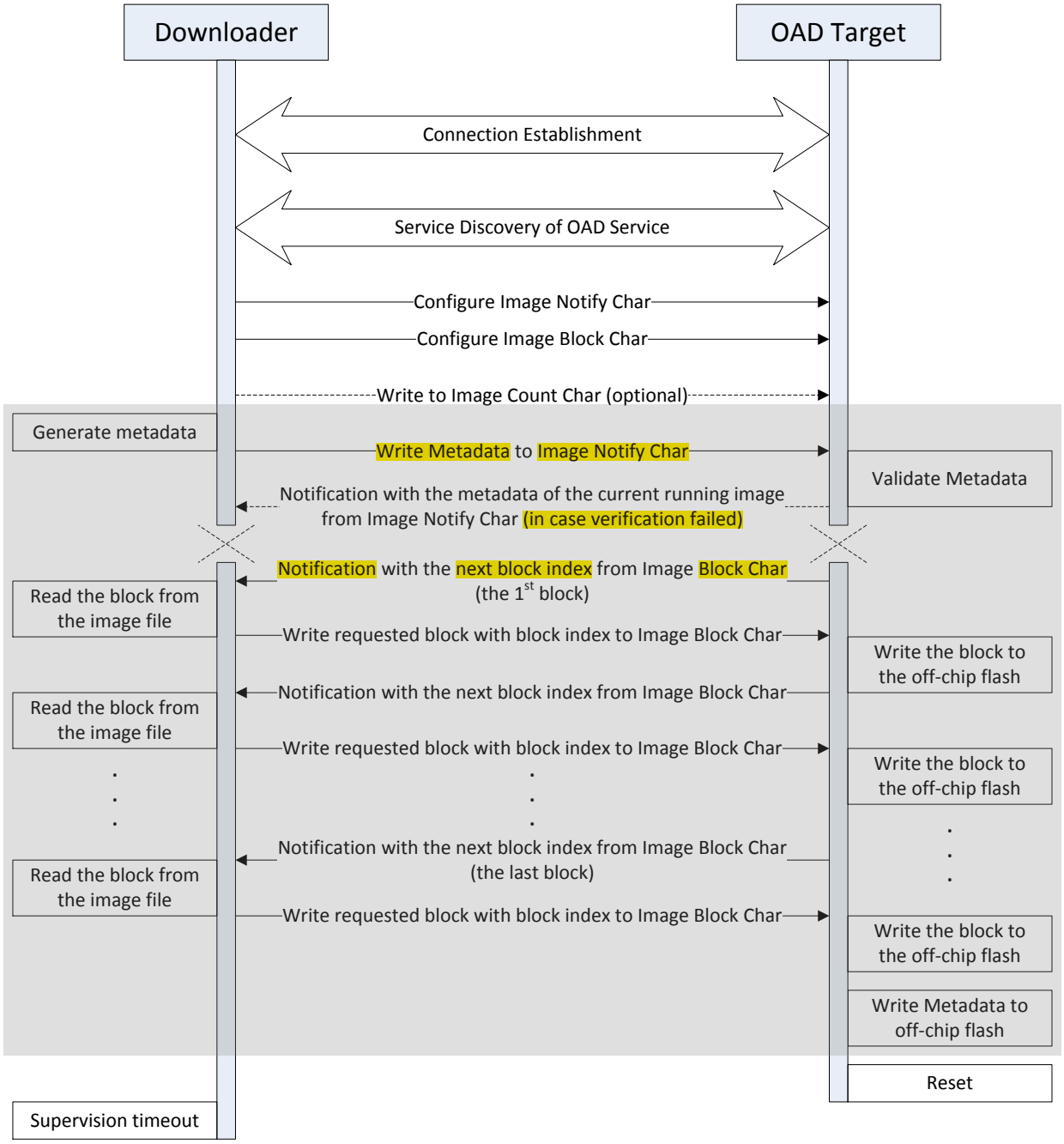

Figure 4. Off-chip OAD sequence diagram

6.3.5 Image Block Transfers

The Image Block Transfer characteristic allows the two devices to request and respond with the OAD image, one block at a time. The image block size is defined to be 16 bytes – see OAD_BLOCK_SIZE in oad.h. The OAD Target will request an image block from the Downloader by notifying the OAD Image Block characteristic with the correct block index. The Downloader shall then respond by writing to the OAD Image Block characteristic. The message's data will be the requested block's index followed by the corresponding 16-byte block of the image. Whenever the OAD Target is ready to digest another block of

the OAD image, it will notify the Image Block Transfer characteristic with the index of the desired image block. The Downloader will then respond.

6.3.6 Completion of the Off-chip OAD Process

After the OAD Target has received the final block of an OAD Image, it will verify that the image is correctly received and stored by calculating the CRC over the stored OAD image. The OAD Target will then store the corresponding metadata in the off-chip flash as well so that BIM can copy the stored image into the on-chip flash after system restart. If the number of downloaded images has reached the value of OAD Image Count characteristic, the OAD Target will reset itself. Otherwise, it will repeat the procedures that are gray backgrounded in Figure 4 until all the OAD Images available for OAD are downloaded.

6.4 OAD Target

6.4.1 Overview of OAD Target for Off-chip OAD

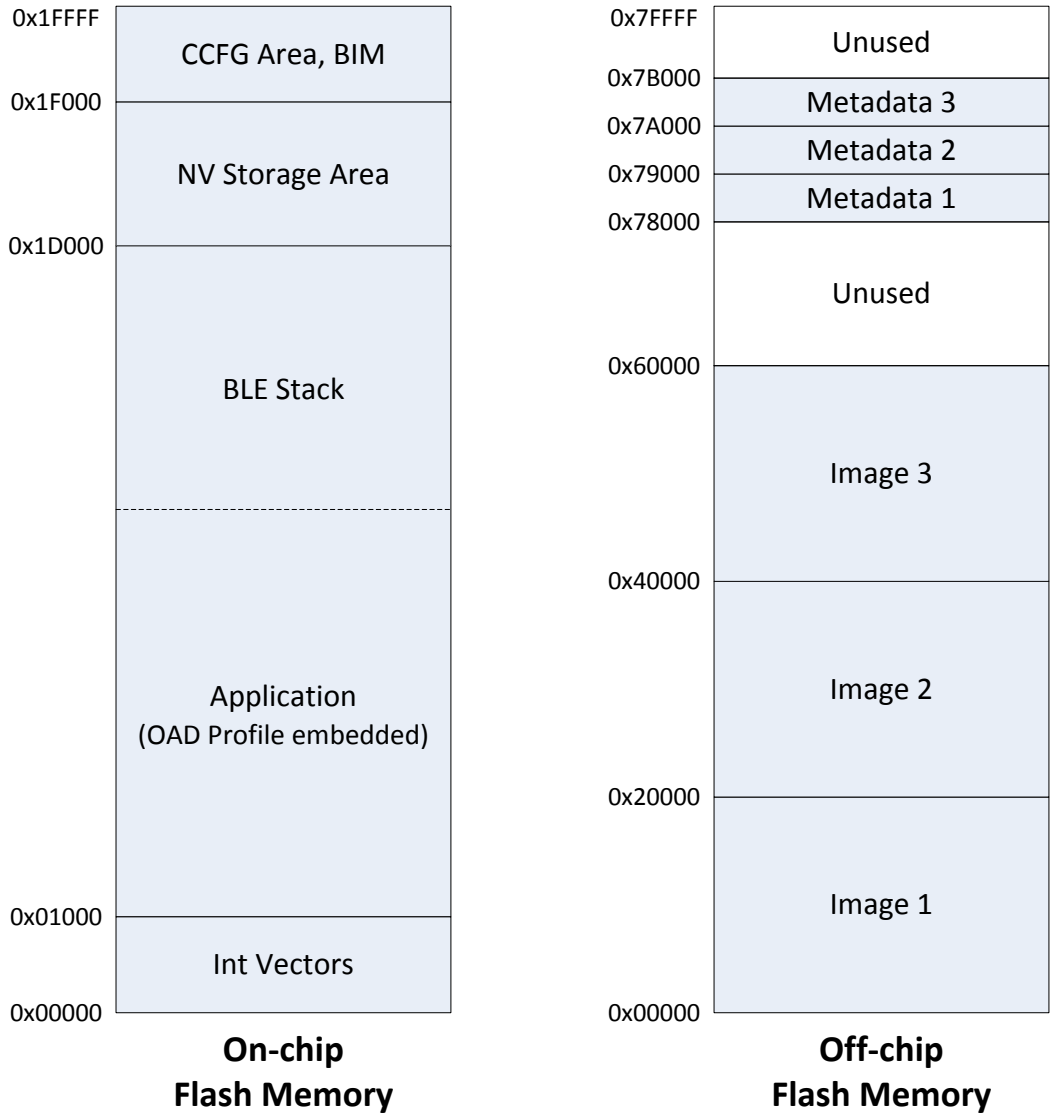

Figure 5. Off-chip OAD Target Memory Partition

The Off-chip OAD Target has both on-chip flash memory and off-chip flash memory device. The on-chip flash memory contains the Interrupt Vectors, the BLE Stack, the Application where OAD Profile is embedded, the BLE stack image, the NV Storage Area, the BIM and the CCFG.

The off-chip flash memory contains up to 3 OAD Images and up to 3 Metadatas corresponding to the OAD Images. The size of each OAD Image placeholder is 128kB. The memory partition of the OAD Target for Off-chip OAD is depicted in Figure 5. Each OAD image, if it's of either Application only or Application+Stack, must support OAD Profile so that further OAD is enabled after it is downloaded to the Off-chip memory, copied to the On-chip memory and executed.

6.4.2 BIM for Off-chip OAD

The OAD solution requires that permanently resident boot code, the BIM, exists in order to provide a fail-safe mechanism for determining whether to run the existing application image or to copy a new image or images from off-chip flash to on-chip flash. It is assumed that a valid image exists either in off-chip flash ready to be copied or already placed in on-chip flash at any given time. Given this assumption, the initial image placed in internal flash which does not exist in external flash will have invalid external image metadata, and so the bootloader will choose to jump to the existing image's entry point.

At startup, BIM checks if the application image metadata in off-chip flash has a status indicating that the image is to be copied to the on-chip flash. If it is, copies the image if a valid CRC and CRC Shadow are found. If not, assumes the application in the on-chip flash is valid to run. If a 2 byte value is found that is neither 0x0000 nor 0xFFFF, but a 0xFFFF shadow checksum is found, the BIM computes the CRC over the image. Image length is determined by the metadata that is also stored contiguous with the CRC in on-chip flash that was copied over during the original write of the image from the off-chip flash.

If off-chip flash contains a "bad" image to be downloaded, but this image is undesirable, BIM can be programmed with symbol NO_COPY to skip image checking and jump directly into the image already placed in on-chip flash; at which point the on-chip flash image could invalidate the bad image's metadata or OAD a new image in its place. BIM will not be able to load any new images while NO_COPY is defined in the build.

BIM is only responsible for making an application image failsafe upon entry. This could mean an app and stack image, or just the application. BIM has exactly one entrance to the application image.

The BIM occupies the last flash page with CCFG and uses the interrupt vectors at the start of flash where the Reset Interrupt Vector calls the BIM startup routine to ensure its control of the system upon a device reset.

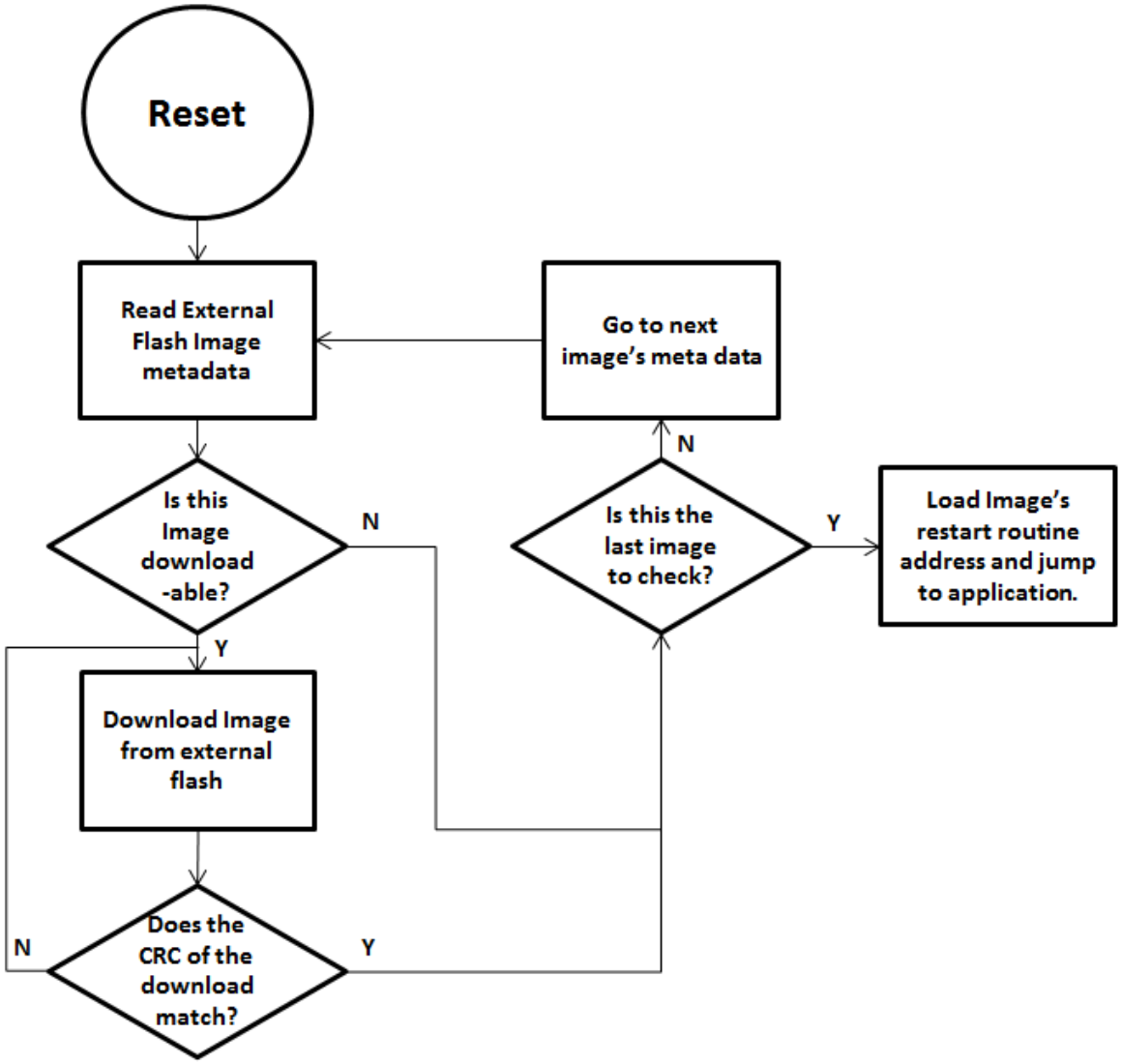

Figure 6. Functional Overview of Off-chip BIM

6.5 Building Off-chip OAD

6.5.1 Building BIM

The boot code is separately built, debugged and programmed via the IAR or CCS IDE. The projects are located at:

<INSTALL_DIR>\Projects\ble\util\BIM_extflast\CC26xx

Choose "FlashOnly" configuration and build. "FlashOnly_SensorTag" configuration is out of scope in this document.

By default, the symbol NO_COPY described in 6.4.2 is undefined as it is very unlikely there is a valid image, to be copied onto the on-chip flash, in the off-chip flash for the first-time use.

6.5.2 Building the BLE Stack

There is nothing specially to be done for an existing BLE Stack project to make it capable of running incorporated with OAD-enabled application.

For IAR, the example BLE Stack project is included in SimpleBLEPeripheral workspace. For CCS, SimpleBLEPeripheralStack project separately exists. The only change that should be made for the stack to be flashed on the SensorTag platform is Debugger setting. With IAR, select "TI XDS110 Emulator" in *Project→Options→Runtime Checking→Debugger→TI XDS→Setup→Emulator*. With CCS, select "Texas Instruments XDS110 USB Debug Probe" in *Properties→General→Main→Device→Connection*.

6.5.3 Building the Application Image

SimpleBLEPeripheral project contains a configuration of 'FlashOnly_OAD_ST_ExtFlash' designed for the application to run on SensorTag hardware platform and utilize the external flash component. The Stack built as described in 6.5.2 works with this application.

The SimpleBLEPeripheral with FlashOnly_OAD_ST_ExtFlash is made through following procedures. The procedures can be applied to convert any existing application to downloadable On-chip OAD Image. In addition to the procedures described in 6.5.3.1 and 6.5.3.2, registration and callbacks for OAD service should be added to the application. Changes to be made for those are found under FEATURE_OAD in simpleBLEPeripheral.c. Note that the last step VI should be done manually even for the pre-made example 'FlashOnly_OAD_ST_ExtFlash'.

The application image contains a python script that creates a combined image of Application and Stack. To create an image that includes Application and Stack and BIM to load the first time, see Appendix section 8.3 on Building Super Image.

6.5.3.1 Building the Application Image using IAR

Using IAR, the Application Image can be built through the following procedure.

I. Select *Project→Options→C/C++ Compiler→Preprocessor* and add the following new definitions to *Defined symbols*:

```
FEATURE_OAD
HAL_IMAGE_E
TI_DRIVERS_PIN_INCLUDED
TI_DRIVERS_I2C_INCLUDED
TI_DRIVERS_SPI_INCLUDED
SENSORTAG_HW
```

Add the following lines to *Additional include directories*:

```
$PROJ_DIR$/../../../../../../../Projects/ble/Profiles/OAD/CC26xx
$PROJ_DIR$/../../../../../SensorTag/CC26xx/Source/Application/Board_pat
ch/interfaces
$PROJ_DIR$/../../../../../SensorTag/CC26xx/Source/Application/Board_pat
ch/devices
$PROJ_DIR$/../../../../../SensorTag/CC26xx/Source/Application/Board_pat
ch/CC26XXST_0120
```

And remove the following line:

```
$TI_RTOS_DRIVERS_BASE$\ti\boards\SRF06EB\CC2650EM_7ID
```

II. Select *Project→Options→Build Actions→Post-build command line* and paste the line below:

```
python "C:\Python27\Scripts\hexmerge.py" -o
"$PROJ_DIR$\FlashOnly_OAD_ST_ExtFlash\Exe\OAD_FULL_IMAGE.hex" -r
"1000:1CFFF" --overlap=error
"$PROJ_DIR$\FlashOnly_OAD_ST_ExtFlash\Exe\SimpleBLEPeripheral_OADExtFla
sh.hex":1000:EFFF
"$PROJ_DIR$\..\..\Stack\CC2640\FlashROM\Exe\SimpleBLEPeripheralStackFla
shROM.hex":F000:1CFFF
```

The above process builds an Application-Stack combined image, named OAD_IMAGE_FULL.hex which has a flash range of 0x1000 to 0x1CFFF with the default configurations.

III. Select *Project→Options→Linker→Config→Linker configuration file* and paste the following line:

```
$PROJ_DIR$\..\..\..\..\..\common\cc26xx\IAR\cc26xx_ble_app_oad.icf
```

And add the following symbols to *Configurable file symbol definitions*:

```
APP_IMAGE_START=0x1000
```

IV. Add the OAD profile modules to the PROFILES folder of the workspace. These files are located here: `<INSTAL_DIR>/Project/ble/Profiles/OAD/CC26xx`.

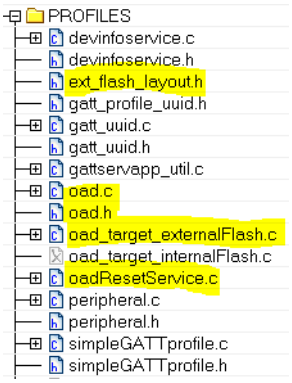

Figure 7. OAD modules in the IAR project

V. Add or replace hardware-related files as SensorTag is used as reference hardware platform:

- Add board files to Board folder of the workspace. Board files are located at `<INSTALL_DIR>\Projects\ble\SensorTag\CC26xx\Source\Application\Board_patch` and subdirectories.
- Add driver files to Drivers folder of the workspace. Driver files are found at `<TI_RTOS_DRIVERS_BASE>\ti\drivers` and subdirectories.
- Replace the existing Board.c with the one located at `<INSTALL_DIR>\Projects\ble\SensorTag\CC26xx\Source\Application\Board_patch`.

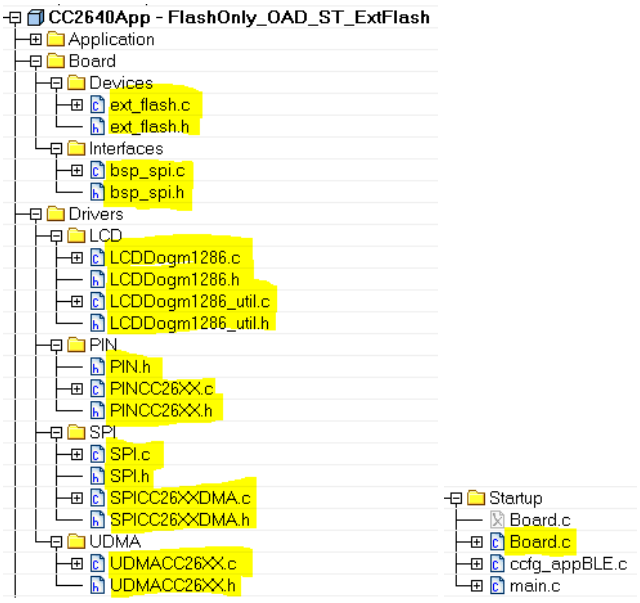

Figure 8. Board files in the IAR project

VI. See Appendix section 8.2 on how to switch the RTOS in Flash configuration.

6.5.3.2 Building the Application Image using CCS

Using CCS, the Application Image can be built through the following procedure.

I. Select *Project→Properties→Build→ARM Compiler→Include Options* and add the following lines to *Add dir to #include search path*:

```
${ORG_PROJ_DIR}/../../../../../../../Projects/ble/Profiles/OAD/CC26xx
${TI_RTOS_BOARD_BASE}/interfaces
${TI_RTOS_BOARD_BASE}/devices
${TI_RTOS_BOARD_BASE}/CC26XXST_0120
```

And remove the following line:

```
${TI_RTOS_DRIVERS_BASE}\ti\boards\SRF06EB\CC2650EM_7ID
```

II. Select *Project→Properties→Build* and paste the lines below to *Steps→Post-build steps*:

```
"${CG_TOOL_HEX}" -order MS --memwidth=8 --romwidth=8 --intel -o
"${ProjName}.hex" "${ProjName}.out"

python "C:/Python27/Scripts/hexmerge.py" -o
"${PROJECT_LOC}/FlashOnly_ST_OAD_ExtFlash/OAD_IMAGE_FULL.hex" -r
"1000:1CFFF" --overlap=error
"${PROJECT_LOC}/FlashOnly_ST_OAD_ExtFlash/${ProjName}.hex":1000:EFFF
"${PROJECT_LOC}/../SimpleBLEPeripheralStack/FlashROM/SimpleBLEPeriphera
lStack.hex":F000:1CFFF
```

The above process builds an Application-Stack combined image, named OAD_IMAGE_FULL.hex which has a flash range of 0x1000 to 0x1CFFF with the default configurations.

III. Use cc26xx_ble_app_oad.cmd as a linker command file instead of cc26xx_ble_app.cmd. APP_BASE is hardcoded to be 0x1000 in that file.

IV. Add the OAD profile modules to the PROFILES folder of the workspace. These files are located here: `<INSTAL_DIR>/Project/ble/Profiles/OAD/CC26xx`.

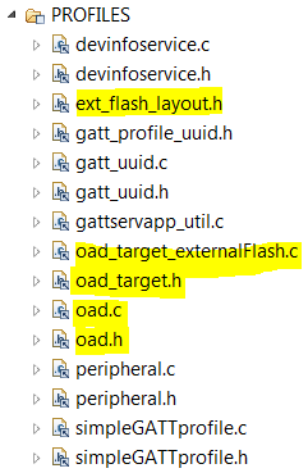

Figure 9. OAD modules in the CCS project

V. Add or replace hardware-related files as SensorTag is used as reference hardware platform:
- Add board files to Board folder of the workspace. Board files are located at `<INSTALL_DIR>\Projects\ble\SensorTag\CC26xx\Source\Application\Board_patch` and subdirectories.
- Add driver files to Drivers folder of the workspace. Driver files are found at `<TI_RTOS_DRIVERS_BASE>\ti\drivers` and subdirectories.
- Add ext_flash_layout.h to PROFILES folder of the workspace. The file is located at `<INSTALL_DIR>\Projects\ble\SensorTag\CC26xx\Source\Application`.
- Replace the existing Board.c with the one located at `<INSTALL_DIR>\Projects\ble\SensorTag\CC26xx\Source\Application\Board_patch`.

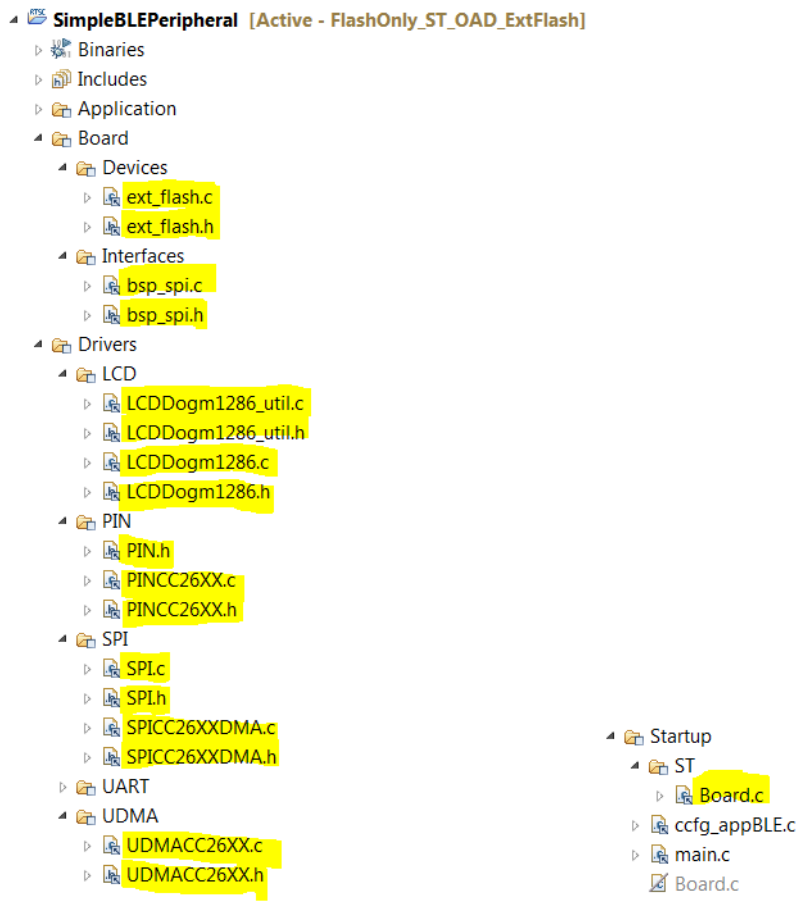

Figure 10. Board files in the CCS project

VI. Switch the RTOS in Flash configuration by following steps in section 8.2.

Modify resetVectorAddress in appBLE.cfg file as follows. This is necessary to link the interrupt vectors into the application's flash region as determined in cc26xx_ble_app_oad.cmd linker file.

```
/* Put reset vector at start of Flash */
M3Hwi.resetVectorAddress  = 0x00001010;
```

VII. Fix the linker error with the post build python script
In order to build the hex file, make sure Python 2.7.x is installed and added to your system path environment variables. See Appendix section 8.1 on installing python.

6.5.4 Adjusting Off-chip Flash Memory Layout

The Off-chip OAD described in this document is based on the assumption that it is running on SensorTag hardware where 512kB off-chip flash memory is equipped. If the size of the off-chip flash is different, there may be a need of changing the layout of the off-chip flash. The layout is defined in ext_flash_layout.h and referred by both BIM and Application.

As an example, assuming the external flash size is 128kB, the OAD image and the Metadata should fit in the 128kB space. Since the size of the OAD Image cannot exceed 120kB in Off-chip OAD design, there is at least 8kB available space for the Metadata. We can place the Metadata in the last page of the off-chip flash memory. This can be done by modifying EFL_ADD_META from 0x78000 to 0x1F000.

6.6 Sending an OAD Image over the air

By following the steps in section 6.5.3, the Off-chip OAD image is ready to be sent as an upgrade to the remote OAD Target device. Currently Windows-based BLE Device Monitor is provided as Downloader that works with the BIM for Off-chip OAD model. However, OAD client can be implemented on any OS platform since OAD Profile is platform-agnostic.

6.6.1 Using BLE Device Monitor

When the OAD Target device which supports OAD Service of UUID 0xFFC0 is connected to the BLE Device Monitor, select *File→Program (OAD)*. Choose a .hex Image desired to be downloaded in *Add Flash Image(s)*. For Off-chip OAD, you can choose up to 3 OAD Images of each different image type in case of Off-chip OAD.

Select *Options→GAP Settings* and press *Apply* to modify default values so that minimum interval and slave latency are applied. Additionally, set *Blk/conn* rate to 1 in *Program*.

Press *Start* in *Program* to start downloading.

For more information, see BLE Device Monitor wiki page.

7 On-Chip OAD

7.1 Constraints and Requirements for On-chip OAD

Using the internal flash of CC2640F128, the first 6 pages, or 24KB, of flash are, by default, reserved for the flash interrupt vectors, the BIM and the permanently resident OAD Target App using an instance of TI-RTOS partially implemented in ROM. BIM and the OAD Target App also use the remaining space on flash page 31, starting at address 0x1F000, shared with the CCFG. Neither BIM nor the OAD Target App is designed to be upgraded by On-chip OAD.

The OAD Image to be downloaded is, by default, allocated 9 flash pages, or 36KB, from address 0x6000 to 0xEFFF. Because page 0 cannot be updated, an application must include its own TI-RTOS instance in flash without dependence on the TI-RTOS ROM implementation. This image also shares the CCFG referenced in the above paragraph. It is not possible to update the CCFG parameters via an OAD.

The OAD Target App and the OAD image should share the same RAM range as only one is used per device reset. The OAD Image must be a complete application image, capable of running independently of the permanently resident OAD target App.

The BLE protocol stack defaults to a range of 15 flash pages, or 60kB, ranging from address 0xF000 to 0x1DFFF and 1-page, by default, SNV area ranges from address 0x1E000 to 0x1EFFF. If the OAD Image is too large to fit in its allocated space, it might be considered that the BLE stack reduce its size by removing some features. This will be discussed further in 7.5.5.

The OAD Target App, or the Image A, and the Image B shall share the same BLE stack. It is not possible to perform an On-chip OAD of the BLE Stack image.

The first and last flash pages must never be erased as doing so puts the device in an unsafe state and a reset at this time will "brick" the chip and prevent it from restarting without the help of a JTAG or serial boot loader.

7.2 System Overview

The OAD system, depicted in Figure 11, is comprised of OAD Image, Downloader and OAD Target. The OAD Image is the image file to be downloaded to OAD Target and should be in intel hex(.hex) format. The Downloader can be BLE Device Monitor running on PC with CC2540 dongle where HostTest application is programmed or any proprietary application or smartphone app where the OAD Profile client described in section 7.3 is implemented. The OAD Target is the device to be upgraded with new image(s).

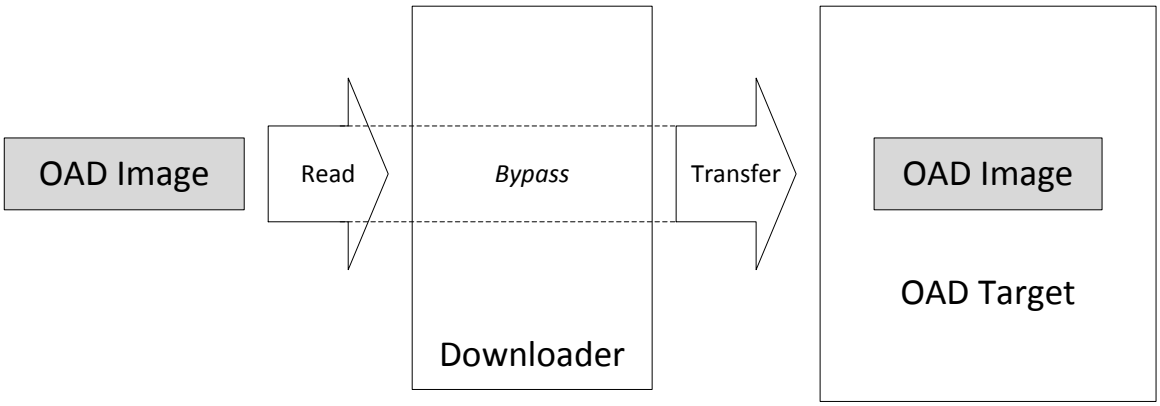

Figure 11. On-chip OAD System Overview

TI-proprietary OAD Profile, defined for communications between downloader and OAD Target, supports image identification, image block request/response and image count setup, with corresponding characteristics.

7.3 OAD Profile

This Profile has been designed to provide a simple and customizable OAD Profile for the customer. In its most rudimentary form, for On-chip OAD, this profile is responsible for accepting an OAD interaction based on image header criteria, storing the image onto the on-chip flash in the OAD Target device and causing a device reset if the download is successful so that the downloaded application image is run by the BIM. Downloader and OAD Target perform Client role and Server role respectively.

7.3.1 Dependencies

The OAD Profile depends on the final customer application to determine the connection parameters and application specific criteria for initiating the OAD process.

7.3.2 Messages

The "write with no response" GATT message has been chosen as the default message type from Downloader to the OAD Target in order to reduce code size and increase data throughput as much as possible. This decision was made because adding the ability to send GATT Notifications requires adding

the GATT_ClientInit() call which would increase code size noticeably. In a noisy or otherwise poor RF-environment, the "write with no response" GATT message may not be sufficient to successfully transmit an entire image and software driven timeouts and re-tries may need to be added. Since the Downloader will have already initialized the GATT_Client, notifications were chosen as the default message type from the OAD Target to the Downloader.

7.3.3 Characteristics

The OAD Profile has only three characteristics. The burden is on the Downloader to discover the handles of these three characteristics on the OAD Target. Structures of these characteristics are as shown in Figure 12.

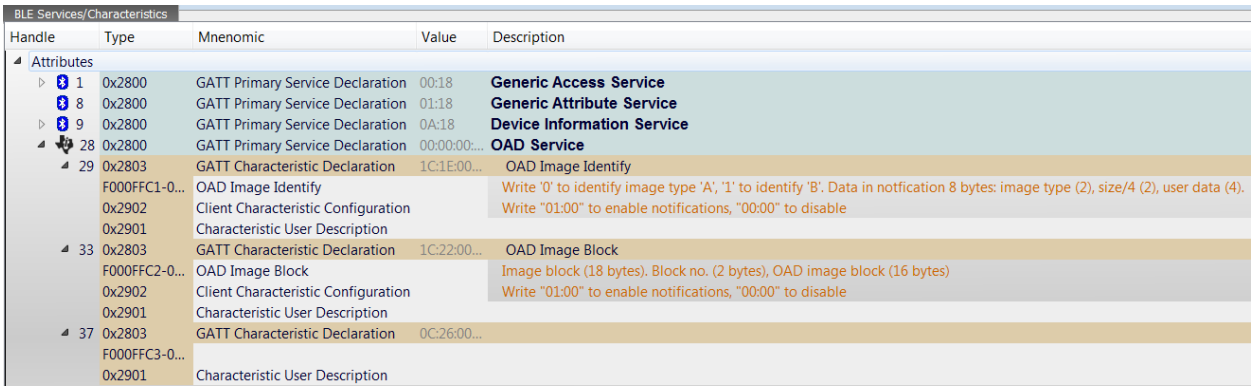

Figure 12. OAD Characteristics from BLE Device Monitor

7.3.3.1 OAD Image Identify Characteristic

The Image Identify characteristic is used to exchange either the image header information embedded in the OAD image for On-chip OAD or the metadata generated by Downloader for the OAD Image for Off-chip OAD in order for the OAD Target to decide if an OAD should occur. "01:00" shall be written to the CCC of this characteristic so that notification for reject is enabled.

7.3.3.1.1 Image Header

Image Header is originally located in the beginning part of the OAD Image for On-chip OAD. It starts exactly at 4th byte or right after the CRC and the placeholder for CRC shadow which are 4-bytes long in total and reside in the beginning of the OAD Image. This information is retrieved from the OAD Image file and put in the payload by Downloader when it writes to OAD Image Identify characteristic. Image Header's fields are described in Figure 13.

Field	Size (in byte)	Description
Version	2	
Length	2	The actual length of the image is 4 x Length in byte, where Length is multiple of 4.
UID	4	"AAAA" for Image A or "BBBB" for Image B
Reserved	4	

Figure 13. Image Header for On-chip OAD

7.3.3.2 OAD Image Block Characteristic

The OAD Image Block characteristic is used to request and transfer a block of the OAD image. If "01:00" is written to the CCC of this characteristic, notification for reject will be enabled. "01:00" shall be written to the CCC of this characteristic so that notification for block request is enabled.

7.3.3.3 OAD Image Count Characteristic

The OAD Image Count characteristic is used to set the number of OAD images to be downloaded. This is used for only Off-chip OAD and the default value of the characteristic is 1.

7.3.4 Initiation of the On-chip OAD Process

After establishing a new connection, updating the connection interval for a faster OAD and enabling notifications of OAD Image Identify and OAD Image Block characteristics on the OAD Target, the Downloader shall write to the Image Identify characteristic of the OAD Target. The message data will be the header retrieved from the OAD Image available for OAD.

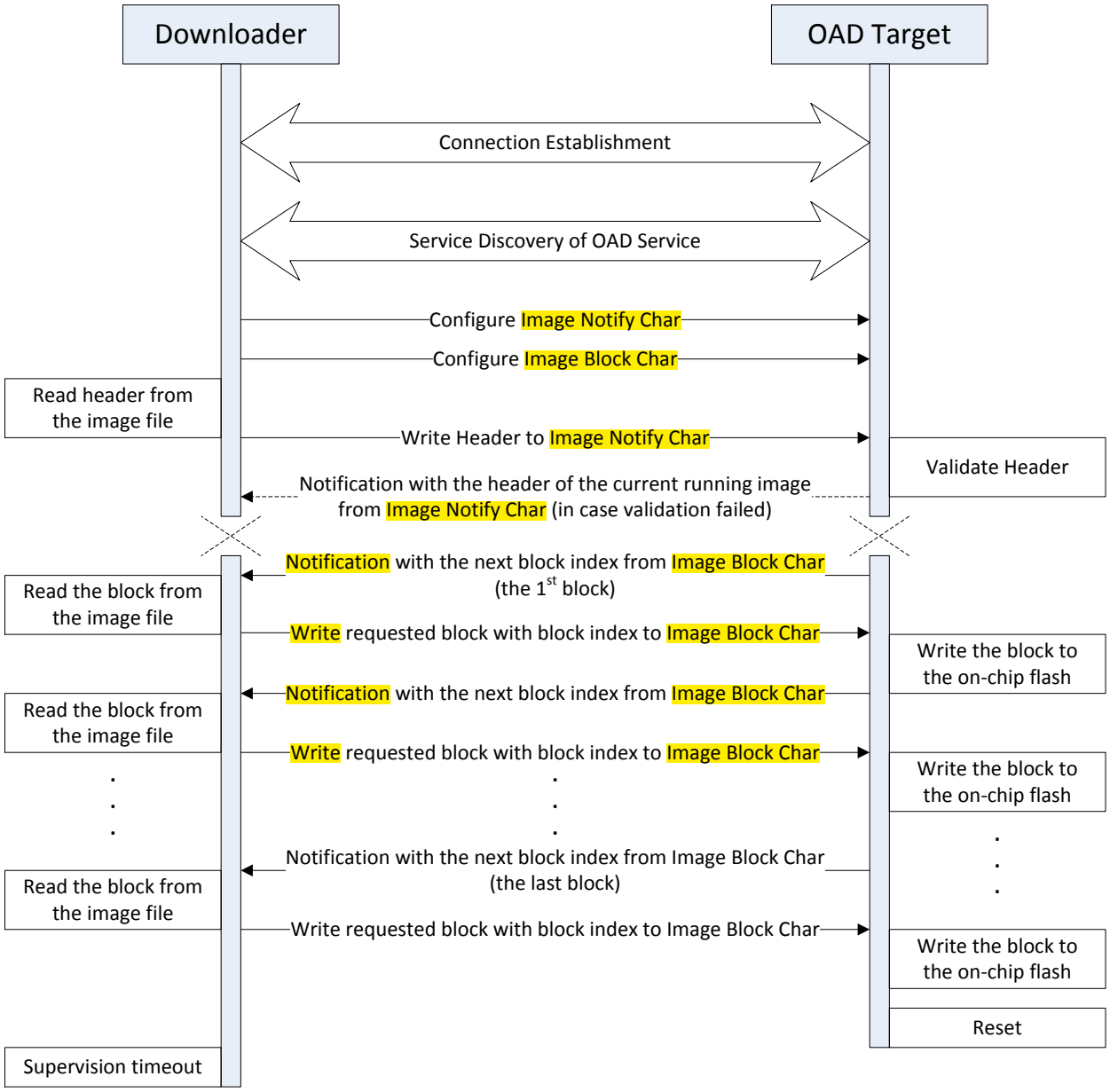

Figure 14. On-chip OAD sequence diagram

Upon receiving the write request to the Image Identify characteristic, the OAD Target will compare the image available for OAD to its own running image. By default, only the image size and version number, which implies whether the image is of type A or B, are checked to determine if the new image is acceptable to download.

If the OAD Target determines that the image available for OAD is acceptable, the OAD Target will initiate the OAD process by notifying the Image Block Transfer characteristic to the Downloader requesting the first block of the new image. Otherwise, if the OAD Target finds that the new image does not meet its criteria to begin the OAD process, it shall respond by notifying the Image Identify characteristic with its own Image Header data as sign of rejection. In that case, the OAD procedure will end at the moment where dotted 'X's are placed as depicted in Figure 14.

7.3.5 Image Block Transfers

The Image Block Transfer characteristic allows the two devices to request and respond with the OAD image, one block at a time. The image block size is defined to be 16 bytes – see OAD_BLOCK_SIZE in oad.h. The OAD Target will request an image block from the Downloader by notifying the OAD Image Block characteristic with the correct block index. The Downloader shall then respond by writing to the OAD Image Block characteristic. The message's data will be the requested block's index followed by the corresponding 16-byte block of the image. Whenever the OAD Target is ready to digest another block of the OAD image, it will notify the Image Block Transfer characteristic with the index of the desired image block. The Downloader will then respond.

7.3.6 Completion of the On-chip OAD Process

After the OAD Target has received the final image block, it will verify that the image is correctly received and stored by calculating the CRC over the stored OAD image. The OAD Target will then invalidate its own image and reset so that the BIM can run the new image in-place. The burden is then on the Downloader, which will suffer a lost BLE connection to the OAD Target during this verification and instantiation process, to restart scanning and the to reestablish a connection and verify that the new image is indeed running.

7.4 OAD Target

7.4.1 Overview of OAD Target for On-chip OAD

The flash memory of OAD Target for On-chip OAD contains the Interrupt Vectors, RCFG, the permanently resident OAD Target App which is also called Image A, the Image B which is initially empty and the place holder for the downloaded OAD Image, the BLE stack, the NV Storage Area, the BIM and the CCFG as shown in Figure 15.

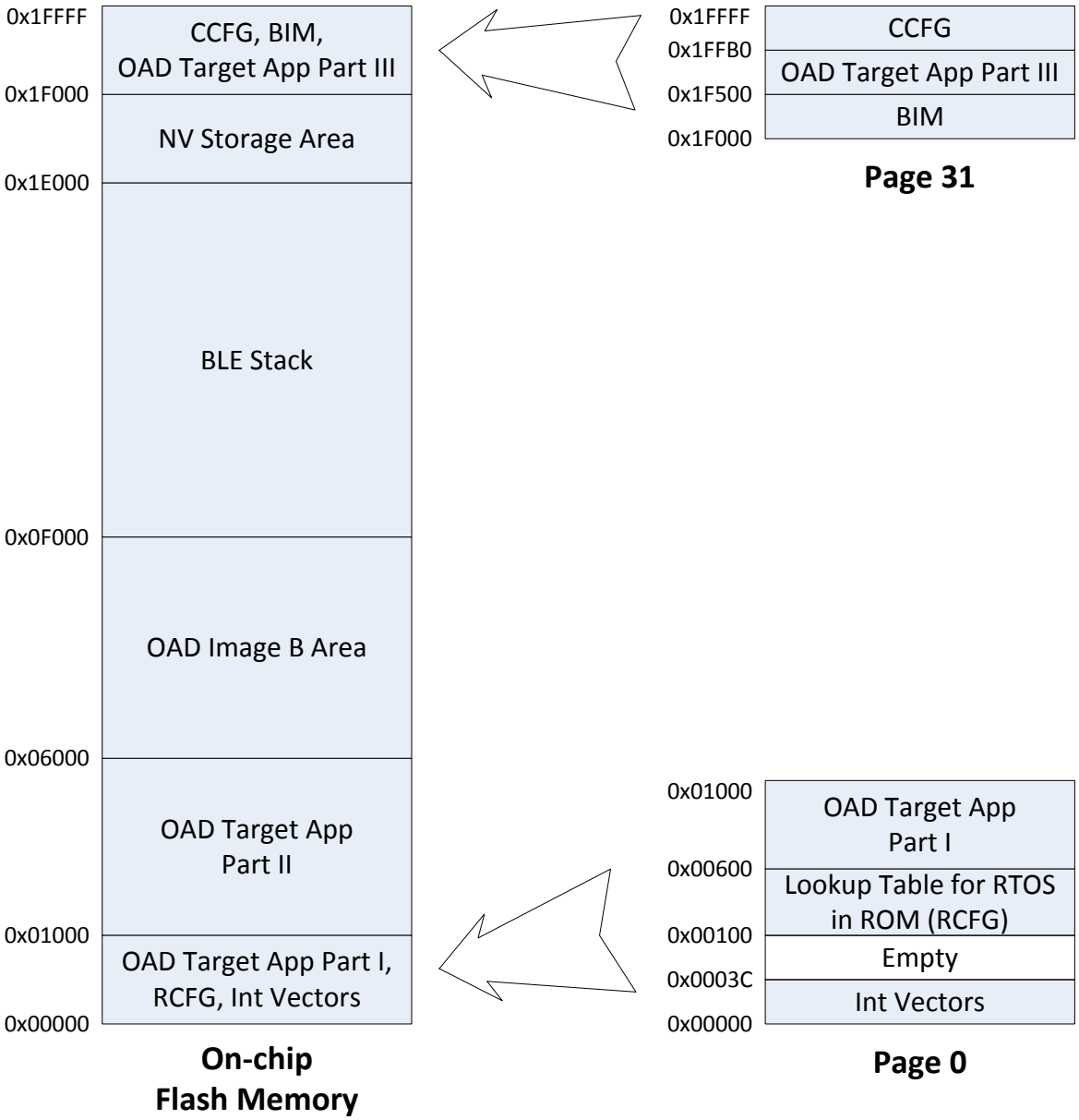

Figure 15. On-chip OAD Target Memory Partition

BIM's design offers the flexibility of having two valid images ready to run; the choice as to which image will run is decided in the BIM. Only the OAD Image B can be downloaded. The OAD Target application, Image A, is a permanent resident which relies on code in the first and last flash page – which if erased during a power down would break the device. The advantage of a permanently resident Image A whose sole purpose is to implement the BLE Stack and OAD Profile is that it increases the amount of available flash for Image B. The developer of a custom Image B does not have to include the OAD Profile implementation. The only reference to OAD feature that Image B needs is a valid image header described in Figure 13. The reference to the valid image header is necessary to use OAD Reset Service described in section 7.6.2. Both Image A and Image B must be developed using exactly the same BLE Stack build, linked at the same location in memory.

7.4.2 BIM for On-chip OAD

The OAD solution requires that permanently resident boot code, the BIM, exists in order to provide a fail-safe mechanism for determining (in preferential order) the image which is ready to run. When a valid image is found, the BIM jumps to that image at which point the image takes over execution. Either Image A or Image B must implement the proprietary TI OAD Profile. By default, this is Image A's role. When an image with the OAD Profile downloads a new image, a system reset can be executed to return to BIM to verify the correctness of the download and begin execution.

The BIM co-occupies the last flash page with CCFG and additional OAD Target application code. BIM uses the interrupt vectors at the start of flash where the Reset Interrupt Vector calls the BIM startup routine to ensure its control of the system upon a device reset.

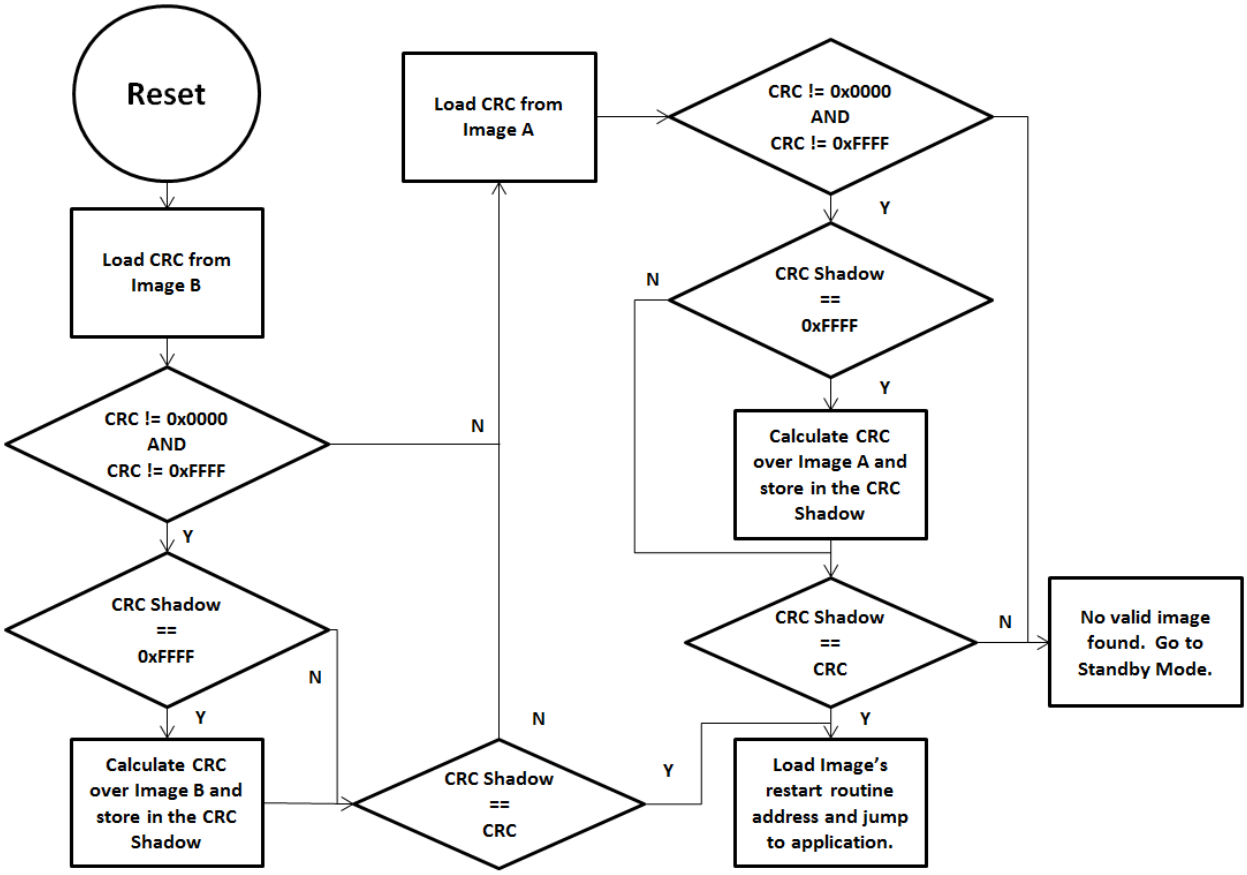

Figure 16. Functional Overview of On-chip BIM

As the permanent owner of the flash interrupt vectors, BIM provides a fail-safe mechanism for intercepting the reset vector, putting the hardware into a safe state, and taking the most appropriate action by reading the headers of Image A and Image B.

By default, BIM gives precedence to Image B, as Image A is only expected to be run when a newer instance of Image B is ready to OAD or no valid Image B exists. If the preferred image is not ready to run, then the other image is checked. If neither image is ready to run – an unlikely scenario because Image A, the OAD Target App, need not ever be erased – then BIM puts the device into a low power Standby mode. Also by default, a CRC check is not performed on Image A because it is expected that the OAD

Target App will be used as a fixed image. The check on Image A will only read the checksum placed by IAR to see if an image exists, it will not calculate the CRC shadow.

In order to verify that an image is valid, a fixed 4-byte area known as the CRC and CRC-shadow will be queried. If the 2-byte CRC16 output calculated at build time matches the 2-byte CRC16 shadow calculated by BIM, then the image is commissioned to run immediately. If the CRC is not zero and not the erased-flash value of 0xFFFF and the CRC-shadow is the erased-flash value of 0xFFFF, then the CRC can be calculated over the entire image (not including this 4-byte area) and the result can be compared to the valid CRC to determine whether the image should be commissioned as ready to run.

7.5 Building On-chip OAD

7.5.1 Building BIM

The boot code is separately built, debugged and programmed via the IAR IDE. There is a project connection within the OAD Target workspace; however, the standalone project is located at:

<INSTALL_DIR>\Projects\ble\util\BIM\CC26xx

On CC26xx, flash pages are erased before written when downloading code. This complicates the download of BIM as it shares the last flash page with code generated by the OAD Target Application. This is resolved by generating a hex file from the build and merging this hex file with the OAD Target App's hex file, as is described in 7.5.3."

By default, BIM defines symbol FEATURE_FIXED_IMAGE as a way to bypass the CRC check on Image A, as it is expected that a fixed image will be used. It is not suggested that this symbol be removed unless architectural changes to OAD made by a user require a CRC check to prove the validity of Image A. Image B will always be checked, for security and integrity reasons.

7.5.2 Building the BLE Stack

The OAD Target App and the OAD image share the BLE Stack image. This image can never be upgraded or modified via OAD. This emphasizes the importance of developing and testing an OAD image with the same stack that is downloaded onto the device. Like BIM, this project outputs a hex image to be merged with the OAD Target App so all three can be downloaded simultaneously. To reduce the size of the BLE Stack image, the NV memory storage space is only one flash page. This carries all the same functionality as the two page NV system, but with two differences. First, cache memory is reserved by the system and should not be disabled or modified by the user application. Second, during NV memory compaction, it is possible to lose all data in NV memory if power is lost before the process completes. A two page NV module is too large to fit without reducing the space reserved for the OAD image. Further restrictions on the BLE Stack require that it uses the CC2640_BLE_peri_HL_CL_FlashROM.a library with minimal features. By default, the OAD Target's CC2640Stack project is configured for these requirements and it is generally encouraged that it be used as the shared BLE Stack image.

7.5.3 Building the OAD Target Application

The OAD Target is the permanently resident application image designed to perform OAD of an image into the Image B area. The project is located at "<INSTALL_DIR>/Projects/ble/OADTarget/CC26xx/IAR". For simplicity, the OAD Target starts in the first flash page following its RCFG. In the BIM functional design (Figure 16), the OAD Target app is Image A so that by default the downloaded Image B always runs, if a valid instance exists. In the post build instruction of this project a python script is executed to

merge the OAD Target image, the BLE Stack Image and BIM into one .hex file. The BLE Stack Image and BIM must be built before building the OAD Target app. If any of these projects are modified they must be rebuilt along with the OAD Target app to update the "super" hex file. The output hex file is named "OAD_merge.hex". See Appendix on how to use python script to merge hex files.

Use the Flash Programmer 2 to program the hex file onto a CC2640 device. If this tool is not already installed, download it from here: http://www.ti.com/tool/flash-programmer. Under the main tab, click browse, navigate to the location of the merged hex file and select it. Click "Refresh" under "Connected devices" and if your device is connected, it should show up under "Connected devices". Select your device by clicking on CC2650 and it will become highlighted. Check the "Erase" box and select "All unprotected Pages". Check the "Program" box and select "Entire source file". Click the play button on the bottom right to program the device. See Figure 17 for how this should look.

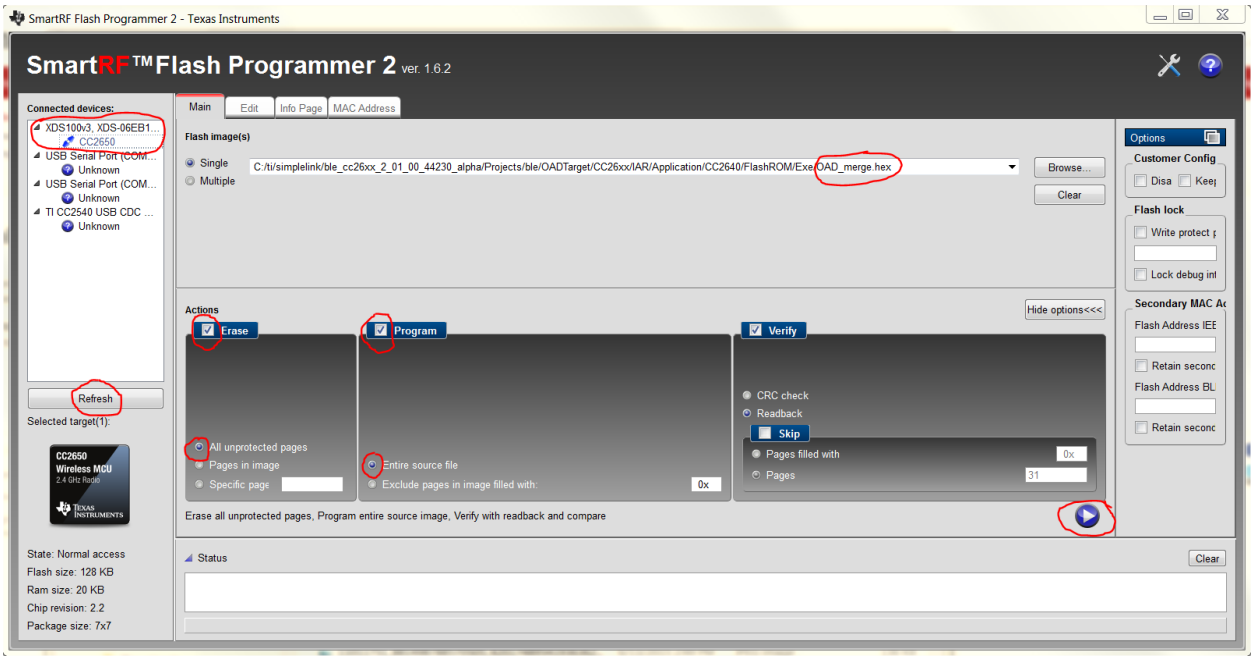

Figure 17. SmartRF Flash Programmer 2

7.5.4 Building OAD Image B

Although the OAD-enabled application image is built and linked separately from the supporting BIM, it must forever adhere to the constraints of the image boundaries and relative locations of external interfaces (e.g. CRC and Image Header) that are expected by BIM. The OAD Target App image is also dependent on BIM existing on the device when it is downloaded as only BIM places its interrupt vectors at the start of flash. Without interrupt vectors at this location, the device will break and become unusable. The example project for building an Image B included in SimpleBLEPeripheral workspace as 'FlashOnly_OAD_ImgB' configuration is made through following procedures. The procedures can be applied to convert any existing application to downloadable On-chip OAD Image. Note that the step VII and should be done manually even for the preset 'FlashOnly_OAD_ImgB' to switch the RTOS in Flash configuration.

I. Select *Project→Options→C/C++ Compiler→Preprocessor→Defined symbols* and add the following new definitions:

```
ICALL_STACK0_ADDR=0xF000
FEATURE_OAD_ONCHIP
IMAGE_INVALIDATE
HAL_IMAGE_B
```

Add the following line to the "Additional include directories":

```
$PROJ_DIR$/../../../../../../../Projects/ble/Profiles/OAD/CC26xx
```

II. This step is for OAD Manager which is out of scope of this document. Select *Project→Options→Build Actions→Post-build command line* and paste the line below to build the binary image, assuming the configuration name is 'FlashOnly_OAD_ImgB' and the output hex file name is 'SBP_OAD_ImgB.hex':

```
python "C:\Python27\Scripts\hex2bin.py" -r "6000:EFFF"
"$PROJ_DIR$\FlashOnly_OAD_ImgB\Exe\SBP_OAD_ImgB.hex"
"$PROJ_DIR$\FlashOnly_OAD_ImgB\Exe\OADbin.bin"
```

III. Select *Project→Options→Linker→Config*. Paste the following line to 'Linker configuration file':

```
$PROJ_DIR$\..\..\..\..\..\common\cc26xx\IAR\cc26xx_ble_app_oad.icf
```

And add the following symbol to 'Configuration file symbol definitions':

```
FLASH_ONLY_BUILD=1
```

IV. Setup the Linker for an image's flash and RAM usage. By default the linker guarantees 9 flash pages, or 36KB, to the OAD image starting at 0x6000. It is generally recommended that the values for Image B starting address are not changed from the default settings unless OAD Target App needs to be modified in its size.

V. Select *Project→Options→Linker→Checksum* to configure a CRC16 calculation over the application image. Make sure that the start address does not include the CRC and CRC Shadow locations and that the checksum ends at the last address of the specified image region of the OAD Target. By default, then, the CRC covers the range 0x6004 through 0xEFFF. Make sure that the Algorithm uses CRC16 with 0x1021. If this value is set to any other value than 0x1021, it can be modified by setting the algorithm first to CRC polynomial, and then setting the Algorithm back to CRC16.

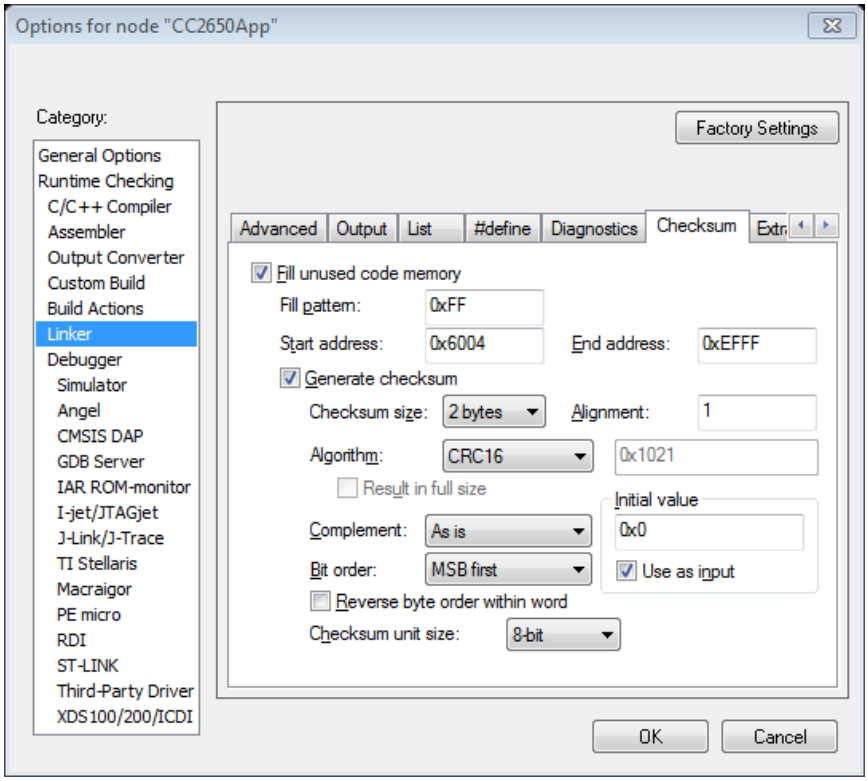

Figure 18. Default Checksum Settings for Image B

VI. Add the OAD profile modules to the PROFILES folder of the workspace. Include the oad.h and oad.c modules. These files are located here:

`<INSTAL_DIR>/Project/ble/Profiles/OAD/CC26xx.`

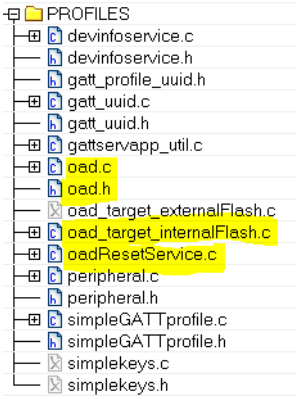

Figure 19. OAD Modules in the Image B Workspace

VIII. Switch the RTOS in Flash configuration by following steps in section 8.2.

7.5.5 Adjusting Stack and Application Sizes

By default, 36kB is available for Image B because the BIM, the flash interrupt vectors and the OAD Target App occupy 28kB, the BLE protocol stack takes 60kB and the SNV takes 4kB in the 128kB on-chip flash as shown in Figure 15. If Image B needs more flash, the only way to make it possible is to reduce the size of other components. Since modifying BIM or OAD Target App is not an option, we can consider downsizing the BLE Stack and/or reducing the number of SNV pages.

With keeping OAD Target App functional, the maximum space for Image B can be achieved by removing the GAP Bond Manager module from the BLE Stack and 1 page of the SNV. This enables the Stack to occupy only 52kB, ranging from address 0x11000 to 0x1EFFF and hence allows for an OAD Image of up to 11 flash pages, or 44kB, ranging from address 0x6000 to 0x10FFF. Note that without the GAP Bond Manager and NV storage area, bonding is no longer possible.

The following example procedures shows how to set up the Stack project, Image A project and Image B project to achieve the maximum 44kB for the Image B, using the size reduction method mentioned above.

Stack project

- buildConfig.opt
  - Comment out –DGAP_BOND_MGR.
- Linker option
  - In Config tab, set ICALL_STACK0_ADDR to 0x11000.

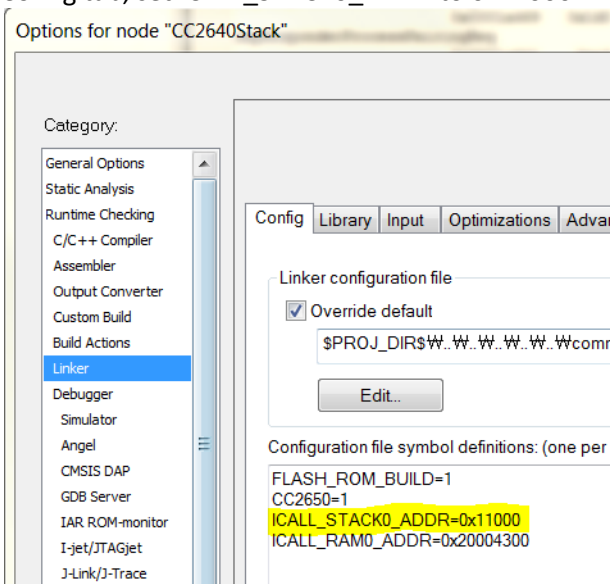

Figure 20. Linker option for On-chip OAD Stack

- Compiler option
  - In Preprocessor tab, undefined OSAL_SNV=1 and define NO_OSAL_SNV.

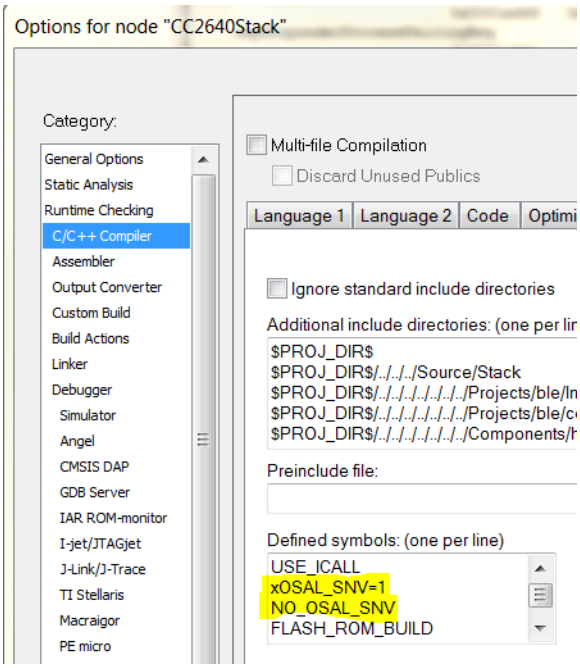

Figure 21. Compiler option for On-chip OAD Stack

OAD Target App(Image A) project

- Compiler option
  - In Preprocessor tab, set ICALL_STACK0_ADDR and OAD_IMG_B_AREA to 0x11000 and 11 respectively.

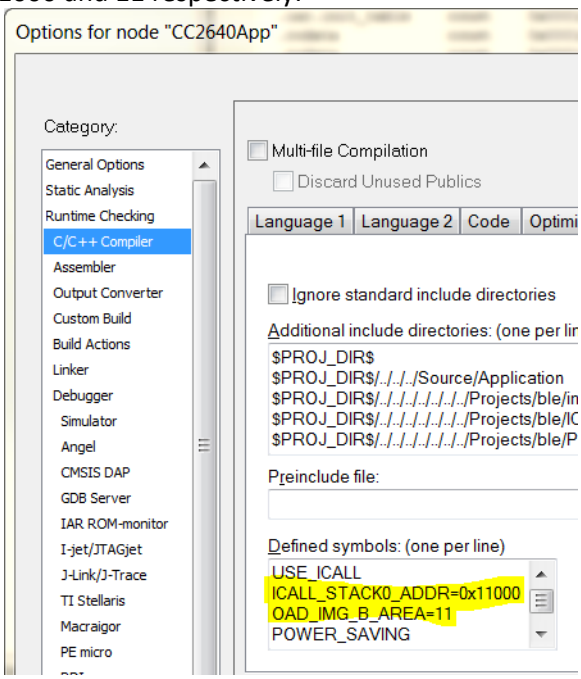

Figure 22. Compiler option for On-chip OAD Image A

- Build Actions option
  - At the end of Post-build command line, modify the range to 11000:1EFFF.

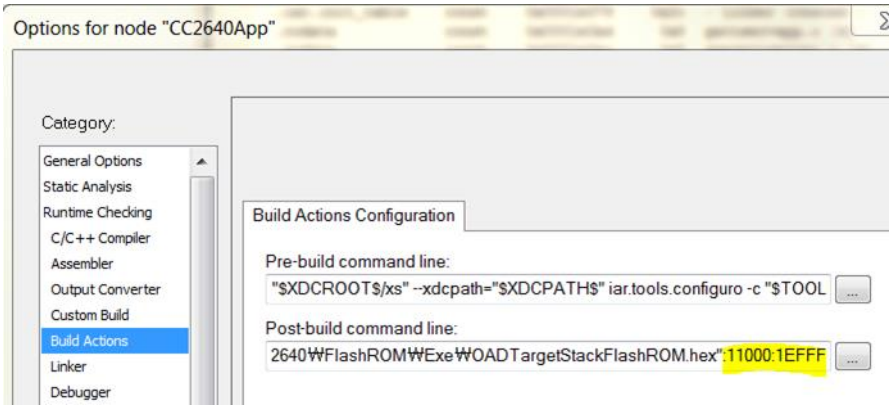

Figure 23. Post-build option for On-chip OAD Image A

Application(Image B) project

- Compiler option
  - In Preprocessor tab, set ICALL_STACK0_ADDR and OAD_IMG_B_AREA to 0x11000 and 11 respectively.

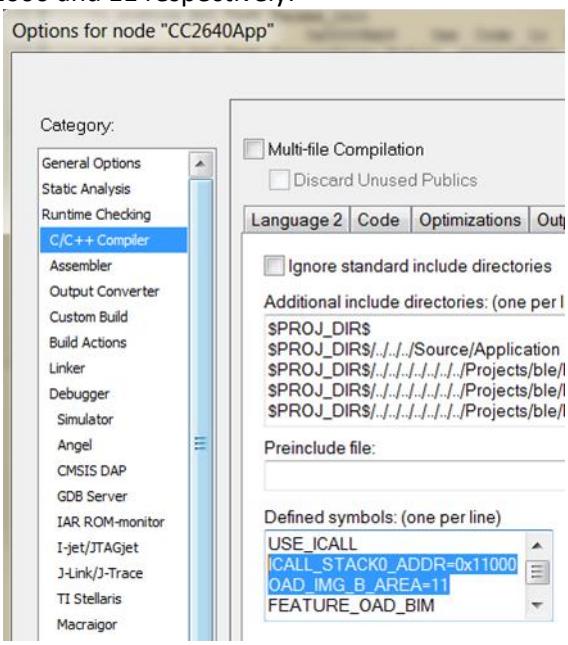

Figure 24. Compiler option for On-chip OAD Image B

- Linker option
  - In Checksum tab, set End address to 0x10FFF.

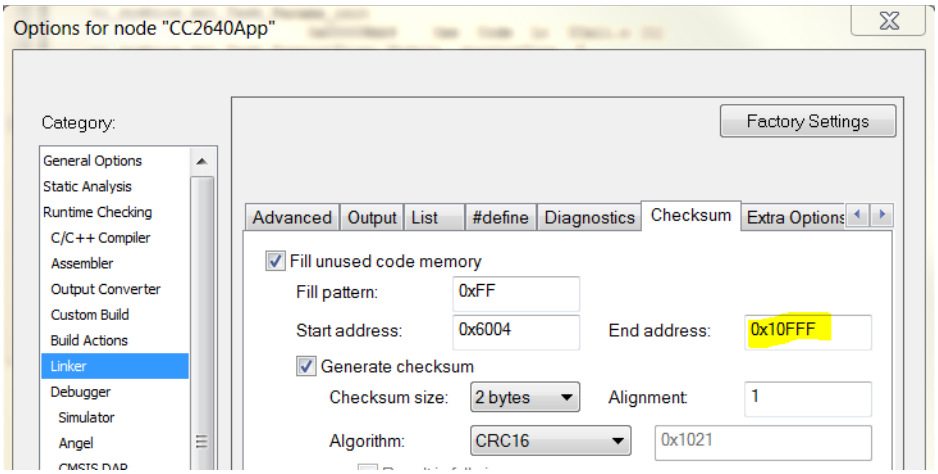

Figure 25. Linker option for On-chip OAD Image B

- Build Actions option
  - In Post-build command line, set the range to 6000:10FFF.

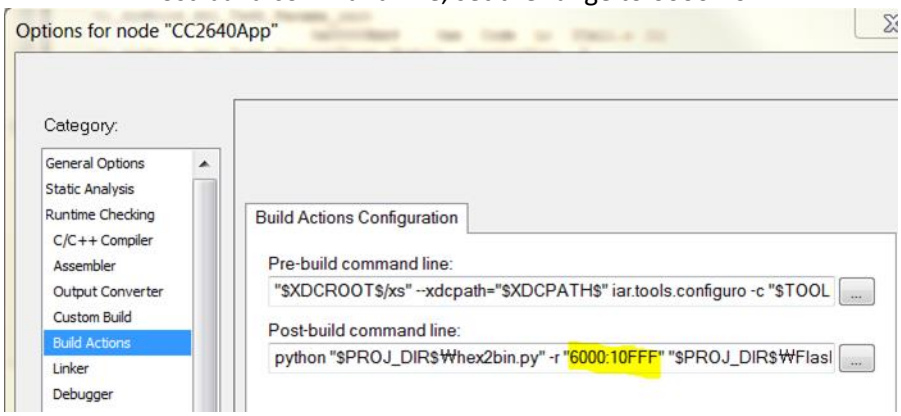

Figure 26. Post-build option for On-chip OAD Image B

7.6 Sending an OAD Image over the air

By following the steps in section 7.5.4, the On-chip OAD image is ready to be sent as an upgrade to the remote OAD Target device. Currently Windows-based BLE Device Monitor is provided as Downloader that works with the BIM for Off-chip OAD model. However, OAD client can be implemented on any OS platform since OAD Profile is platform-agnostic.

7.6.1 Using BLE Device Monitor

When the OAD Target device which supports OAD Service of UUID 0xFFC0 is connected to the BLE Device Monitor, select *File→Program (OAD)*. Choose a .hex Image desired to be downloaded in *Add Flash Image(s)*.

For On-chip OAD, select *Options→GAP Settings* and press *Apply* without modifying default values so that minimum interval and slave latency are applied. Additionally, set *Blk/conn* rate to 1 in *Program*. Also do not use fast mode (used for SensorTag project only).

Press *Start* in *Program* to start downloading.

For more information, see BLE Device Monitor wiki page.

7.6.2 Invalidating Image B for On-chip OAD

Once Image B is successfully downloaded and validated, the BIM will always jumps to the Image B which doesn't have capability to upgrade itself. That means no further OAD upgrade for Image B is possible since the Image A which has OAD Profile for receiving a new Image B will not get a chance to run. The only way to make BIM to jump to the Image A again is to invalidate the Image B. For this, the OAD Image B provides OAD Reset Profile whose service UUID is 0xFFD0. It has only one characteristic, the UUID of the value attribute of which is 0xFFD1. Writing any value to that attribute will invalidate the Image B by making the CRC that is supposed to be checked by the BIM wrong and enable Image A to be chosen to run by the BIM.

8 Appendix

8.1 Installing Python

Python scripts are used to create hex and bin files from project output. Make sure Python 2.7.x is installed and added to your system path environment variables. Also required is the Python IntelHex script hex_merge.py, freely available on the web at https://launchpad.net/intelhex/+download. The expected location of the script is "C:\Python27\Scripts\" as the post build procedure assumes the script is there.

For IAR, the python command can be found in Project Options -> Build Actions -> Post-build command line. For CCS, the python command can be found in Project Options -> CCS Build -> Steps -> Post-build steps. Change the above commands to include the full path of the python executable if a post-build issue occurs due to file path issues. See 8.3 for an example.

NOTE: To build the hex file, make sure Python 2.7.x is installed and added to your system path environment variables.

8.2 Switch the RTOS in Flash configuration

Switch the RTOS in Flash configuration. In the Application Workspace, under the "TOOLS" folder, open appBLE.cfg.

Comment out the first two lines as follows. This is because the Image B should not use TI-RTOS in ROM if the flash page 0 cannot be updated accordingly when the Image B is updated.

```
//var ROM = xdc.useModule('ti.sysbios.rom.ROM');
//ROM.romName = ROM.CC2650;
```

8.3 Building Super Image

Make sure python is installed in your system (see 8.1). To build a super image of multiple projects, extend the python script to add additional project hex files. For example, to add the BIM to the OAD_IMAGE_FULL hex file use the below script:

"C:\Python27\python" "C:/Python27/Scripts/hexmerge.py" -o "${PROJECT_LOC}/FlashOnly_ST_OAD_ExtFlash/SUPER_IMAGE_FULL.hex" -r "0000:1FFFF" --overlap=error "${PROJECT_LOC}/FlashOnly_ST_OAD_ExtFlash/${ProjName}.hex":1000:1EFFF "${PROJECT_LOC}/../SimpleBLEPeripheralStack/FlashROM/SimpleBLEPeripheralStack.hex":1000:1EFFF "${PROJECT_LOC}/../../../../util/BIM_extflash/CC26xx/CCS/FlashOnly/BIM_ExtFlash.hex":0000:1FFFF

Note the above command uses the full path ("C:\Python27\python") to the python executable. If just "python" command is called, the path should be added to the system path environment variables.

8.4 Additional updates

For more up to date information or errata to this document, please check the wiki page:
http://processors.wiki.ti.com/index.php/CC2640_OAD_User%27s_Guide