BLE Developer's Guide for Over-the-Air Download for CC254x

Version 1.2

1. Purpose

The purpose of this document is to enable a developer working with the TI BLE stack to successfully implement the proprietary TI OAD Profile functionality in any sample or proprietary application using the CC254x SOC.

2. Functional Overview

OAD is an extended stack feature provided as a value-enhancing solution for updating code in deployed devices without the cost of physical access via a programming header. OAD is a client-server mechanism in which one device acts as the OAD image server (OAD manager) and the other device is the OAD image client (OAD target).

3. Assumptions

1. The BIM will never need to be updated.

4. Definitions, Abbreviations, Acronyms

Term	Definition
API	Application Programming Interface
BIM	Boot Image Manager – boot code that receives the reset interrupt vector and manages which valid image (Image-A or Image-B) shall run.
BEM	Boot Encrypted Manager – encrypted version of BIM
EBL	Encrypted Bootload – used to encrypt target images.
DL	Down-Loaded – an RC-image candidate that has been downloaded via the OAD procedure and stored in the non-volatile RC-image area.
INSTALL_DIR	The installation directory path of the protocol stack code specific to the version installed, but something like this: `C:\Texas Instruments\BLE-CC254x-1.2.1`
ISR	Interrupt Service Routine
IVEC	Interrupt Vectors – the table of ISR jump instructions to which the CPU physically vectors for ISR's.
LPRF	Low Power RF – a business unit within TI.
NV	Non-volatile storage
OSAL	Operating System Abstraction Layer
OA	Over-the-Air – an RF transmission that can be seen with a packet sniffer.
OBL	OAD Boot Loader – boot code that receives the reset interrupt vector and runs an existing valid RC-image or instantiates a DL-image.
OAD	Over-the-Air Download – a proprietary profile provided by TI for transmitting a candidate RC image over-the-air to an OAD-capable device.
PM	Power-Mode – a low power or sleep mode to reduce power consumption.
SOC	System on a Chip
SNV	Simple Non-volatile memory manager.
TI	Texas Instruments Incorporated

5. References

1. BLE Stack Functional Specification v1_2, Requirements TBD, 2-1, 2-3, 2-4
2. ANSI C Coding Standard (F8W-2005-0002).

6. Revision History

Date	Document Revision	Sections Effected	Summary of Changes
06/12/2012	0.1	All	New document.
10/20/2012	0.2	All	Removed OBL portions.
11/27/2012	0.3	All	Edited GATT architecture.
1/31/2013	1.0	All	General Revisions
3/18/2013	1.1	13	Added Encryption Info
8/4/2013	1.2	All	General Revisions

7. Design Constraints

7.1 External Constraints / Features

If the application image cannot fit into half of the available internal memory, external NV memory must be provided for storage of the RC-image; the means of which this is implemented is beyond the scope of this document.

7.2 Internal Constraints / Requirements

The OAD image must be a complete and integral BLE Stack Application (i.e. only sending the stub Application layer is not supported).

8. OAD Design

8.1 OAD System Overview

The OAD system, depicted in Figure 1, is comprised of two devices: the OAD Target and the OAD manager. The OAD target contains three image areas: the boot code (boot image manager) and two different application image areas. The boot code must occupy page 0 in order to intercept the non-relocatable 8051 reset interrupt vector. Neither of the application images can occupy the last page, the lock-bits page, because it is not programmatically writable or erasable. Image A and Image B, the two application images, both must occupy some of the non-banked code in bank 0. This is described below. The OAD manager contains the image that will be sent over-the-air and an application that performs the OAD process.

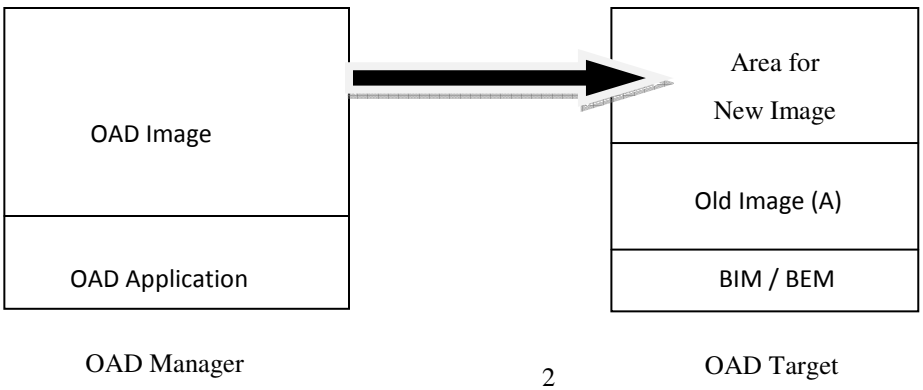


Figure 1: Brief Overview of OAD System

8.2 OAD Target Overview

The figure below shows the system context when implementing OAD with the boot image manager (BIM). The BIM design offers the flexibility of having two valid images ready to run which can ping-pong as to which image should run. It is possible for either image to execute an OAD of the other. The BIM also allows the possibility of having a very small, permanently resident "OAD-only Image" in order to have a much larger final application image. When either image can execute an OAD of the other, the user must carefully build and manage the download of Image-A while running Image-B and vice versa, where Image-A and Image-B are obviously built and linked to run differently. The other advantage of having a permanently resident Image-A which only implements the BLE-stack and OAD Target Profile is that the final customer's Image-B does not have to include the OAD Target Profile at all, saving flash size, and where the Image-A can occupy both the INTVECs page and lock bits page, saving even more flash size. Finally, Image-A may be able to be built with a minimal subset of the BLE stack functionality which is required only to execute the OAD of new Image-B updates, thus further maximizing the space for Image B.

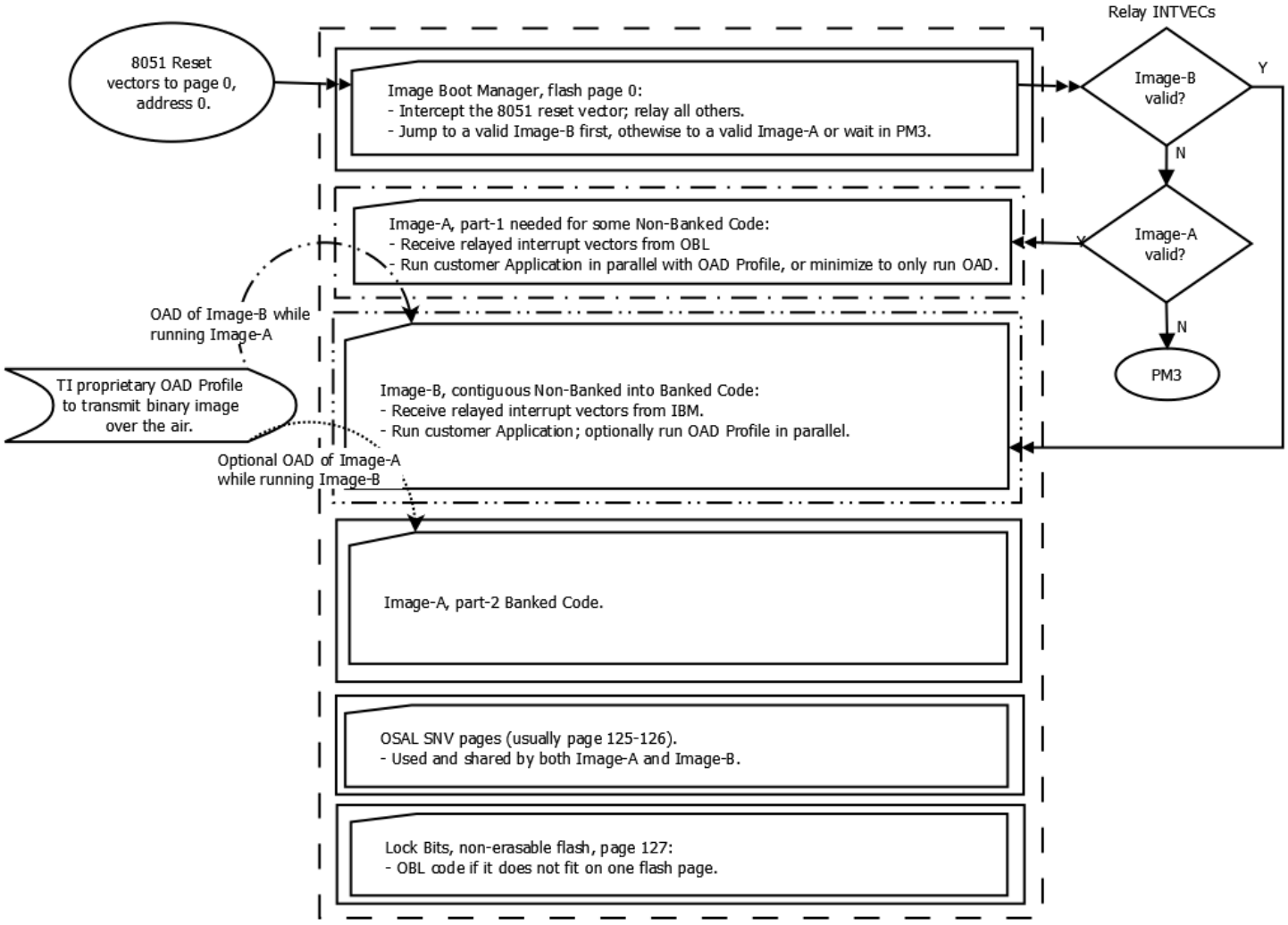


Figure 2: OAD Target Context Diagram using the BIM.

8.3 Functional Description

The OAD solution requires that permanently resident boot code exists on page 0 in order to provide a fail-safe mechanism for determining (in preferential order) the Image that is ready to run. The Image-A and/or Image-B must implement the proprietary TI OAD Target Profile.

In order to verify that an image is valid, a special 4-byte area known as the CRC and CRC-shadow will be interrogated. If the 2-byte CRC matches the 2-byte CRC shadow, 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 instantiated and/or whether the image should be commissioned as ready to run on the next reset.

8.3.1 Boot Image Manager (BIM)

As the permanently resident target of the 8051 interrupt vectors, the BIM provides a fail-safe mechanism for intercepting the reset vector, putting the hardware into a safe state, and taking the most appropriate action based on the status of the Image-B and/or Image-A and/or the entries on a designated Ledger Page.

Normally, Image-B will first be checked for readiness to run and then Image-A, but this order can be changed according to an entry on a Ledger Page or the setting of a GPIO. If the preferred image is not ready to run, then check the other image. If neither image is ready to run, there is no mechanism of image instantiation in the BIM design, so put the hardware into a safe and low power state and enter PM3.

8.3.1.1 Boot Image Manager ISR Latency

Although the IVEC-relay logic of the BIM has been implemented using optimized assembly code and fast access IDATA area memory, the BIM must choose on every single occurrence of every single ISR whether to forward the interrupt vector to the Image-A INTVEC table or to the Image-B INTVEC table. This extra cost per interrupt is 20 cycles at 32-MHz or 625 nano-seconds.

8.3.2 Download-Code Image

This is an image that will be serially loaded to the flash of the OAD Manager and sent over-the-air to replace an image in the OAD Target. In order to be a valid image, the CRC-shadow must be equal to the erased value of 0xFFFF and the CRC must not equal zero or 0xFFFF (and, of course, it must be the CRC calculated over the entire image, not including the 4 bytes of CRC and shadow). This image must be linked to run in the specific image A/B area, and therefore cannot be run in-place.

8.3.3 Images A and B

In order for one image (i.e. A) to OAD the other (i.e. B), the first image (A) must contain the OAD Target profile. These images will rely on a special linker file that properly reserves the code areas for each image and which links the downloaded code to run in the respective image area. Such linking includes placing the INTVEC table and CRC & CRC-shadow at the locations expected by the BIM. Note that these locations can never be changed via an OAD image update, since the BIM cannot be modified in a fail-safe manner after it is directly programmed with a tool.

8.3.4 Image-A Concept

Image-A has a dedicated linker file which is necessarily different from the one for Image-B since this code will be downloaded (or physically programmed) and run in-place. Image-A will require some amount of non-banked code space, no matter how reduced of a stack or code base it is finally running. So Image-A has been arbitrarily selected to be the code image that has a non-contiguous address space which accommodates a contiguous Image-B in between its non-banked and banked code spaces (see Image 1). For this reason, Image-A should be the image made to only run the OAD Target Profile, if the design choice is made to have dissimilar code in Image-A and Image-B in order to maximize the space available for the Image-B which runs the code that implements the final BLE application. In such a scenario, the super efficient, and therefore very valuable, non-banked code space allocated for Image-A should be reduced to the

fewest pages possible, since Image-A only runs on the rare occasion of an OAD and Image-B effectively runs for the life of the product.

8.3.5 Image-B Concept

Image-B has been arbitrarily picked to be optimized to implement the final BLE application and may or may not have to implement the OAD Target Profile. As mentioned above, this image is linked to run in a contiguous address space that lies in between the two parts of the Image-A. If the probability of Image-B running is about as likely as that of Image-A running, then the non-banked code should be evenly split between the two (less, of course, the page 0 used by the BIM) – and this is the default setting for the dedicated linker files for the two images. As an ending exercise in this document, the default linker files will be changed to give more code space to Image-B by taking away an equal amount of code space from Image-A.

8.4 **External Interfaces**

The proprietary TI OAD Target Profile for the BLE stack is defined later in this document. The two functional units which will co-exist and cooperate on the CC254x SOC itself are the boot code and the run-code. These two images are independently compiled and linked by their respective IAR projects. The external interface between the two is just the CRC & CRC-shadow and the re-located INTVEC table.

The relative placement of the following structures has been arbitrarily chosen and is convenient within the scope of the sample applications. . If any location is to be changed, the #defines and all related linker files must be changed accordingly as well. Pay special attention to the in-line assembly instructions in the BIM as well as the placement and loading of the constants mapped as memory.

8.4.1 CRC

For each respective image, the image CRC is calculated by the IAR compiler and placed at a memory location before the beginning of the flash area reserved for code as defined by the image's respective linker file. This can be seen below for the case of Image A:

```
-D_CODE_BEG=0x0830
…
-Z(CODE)CHECKSUM=0x0800-0x0801
```

8.4.2 Image Header

The Image Header structure shall be this:

```
typedef struct {
   uint16 crc1;      // CRC-shadow must be 0xFFFF.
   uint16 ver;       // User-defined Image Version Number
   uint16 len;       // Image length in 4-byte blocks (i.e. HAL_FLASH_WORD_SIZE blocks).
   uint8  uid[4];    // User-defined Image Identification bytes.
   uint8  res[4];    // Reserved space for future use.
} img_hdr_t;
```

This image header has been placed immediately adjacent to the image CRC structure, thus allowing the CRC and the CRC shadow to be adjacent to each other in memory. If the location is to be changed, the #defines and all related linker files must be changed accordingly as well. The location must be on a HAL_FLASH_WORD_SIZE boundary. This image header is sent from the OAD Manager upon establishing a connection with an OAD Target device. The connecting device (OAD Target) is thereby notified of the details of the image which is available on the OAD manager. This connecting device will then either begin an OAD process or choose not to. This process of using the image header is elaborated upon in the OAD Target Profile section

.

8.4.3 INTVECS Table

The IAR generates the INTVECS Table and places it according to this directive in the linker command file:

```
-Z(CODE)INTVEC=_CODE_BEG-_CODE_END
```

The example code and linker files have arbitrarily placed the offset of the INTVEC Table at offset zero in the first image page.

8.4.4 Image Header Version

The uint16 'ver' field of the image header can be used by a device to determine if it should begin an OAD process with an OAD server. When implementing OAD with the BIM, it is recommended to use a numbering scheme such as all odd numbered versions are built to download and run in-place in the Image-B area while all even numbered versions are built to download and run in-place in the Image-A area.

8.4.5 Image Header Length

The uint16 'len' field of the image header represents the length of the image in HAL_FLASH_WORD_SIZE bytes.

8.4.6 Image Header Array

The byte-arrays of the image header, uid[] and res[], are reserved for proprietary use and could be used with the 'ver' field by a device to determine whether or not to begin a download process.

9. Producing Boot Code

The boot code is separately built and debugged or programmed via the IAR IDE. The project is located at:

$INSTALL_DIR\Projects\ble\util\BIM

The default configuration is with the download option to erase flash in order to start a CC254x SOC with clean flash (and thus clean NV). Before debugging or physically programming the OAD application code produced in the next section, this OAD boot code must first be programmed into the flash (but only this once, since, as the following section mentions, the default option for application code is to preserve this OAD Boot code on successive debugging or programming).

The .hex file produced by this project build can be programmed directly using a tool like the SmartRF Programmer. This .hex file can be merged with the .hex file from the build of the application image in order to produce a "super" .hex file that contains both images for factory programming of a device that is OAD-enabled with its first valid application image ready to run and the boot loader present to complete the final step of the OAD process.

10. Producing an Image-A & 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 areas and item placement (e.g. CRC, INTVECS, and Image Header) that are built into the BIM since the BIM is programmed once in the device's lifetime. Image A and Image B example target builds have already been added to the CC2541 project for the SimpleBLEPeripheral sample application. However, the following steps can be adapted to work with any application.

1. Select Project→Edit Configurations→New… and add a new build target: CC2541-OAD-ImgA. Make sure that this is based on the original CC2541 configuration.

2. Immediately compile the new build target. It should build and link without errors or warnings since it is an exact copy of an existing good target build. If this step does not succeed, first fix the problems with the original build target and application before re-starting from step 1.

3. Select Project->Options→C/C++ Compiler→Preprocessor→Defined symbols and add the following 4 new definitions:

   ```
   FEATURE_OAD
   OAD_KEEP_NV_PAGES
   ```

```
FEATURE_OAD_BIM
HAL_IMAGE_A
```

Also include the path to the OAD Profile in the 'Additional include directories':

```
$PROJ_DIR$\..\..\Profiles\OAD
```

Select PROFILES→Add→Add Files and add oad_target.c, oad_target.h, and oad.h from the following directory:

```
INSTALL_DIR\Projects\ble\Profiles\OAD
```

4. Enable the OAD Profile in the Application by adding a call to the OAD_AddService() function wherever the other BLE Profiles are activated. As an example, this is done in the simpleBLEPeripheral.c module, SimpleBLEPeripheral_Init() function as shown below:

```
  SimpleProfile_AddService( GATT_ALL_SERVICES );  // Simple GATT Profile
#if defined FEATURE_OAD
  VOID OADTarget_AddService();                    // OAD Profile
#endif
```

5. Add an include of the OAD Profile header file in the module accessed in Step 4. As an example, this is done at the top of the simpleBLEPeripheral.c module as shown below:

```
#if defined FEATURE_OAD
  #include "oad.h"
  #include "oad_target.h"
#endif
```

6. Select Project->Options→Build Actions→Post-build command line and paste the following line (with **no carriage return** and **exactly one single space** between quote marks):

```
"$PROJ_DIR$\..\..\common\CC2540\cc254x_ubl_pp.bat" "$PROJ_DIR$" "ProdUBL"
"$PROJ_DIR$\CC2541-OAD-ImgA\Exe\SimpleBLEPeripheral"
```

Note that the relative path above will contain different names as you work with projects other than the CC2541 SimpleBLEPeripheral.

7. Select Project->Options→Linker→Config→Linker configuration file and paste the following line

```
$PROJ_DIR$\..\..\common\CC2541\cc254x_f256_imgA.xcl
```

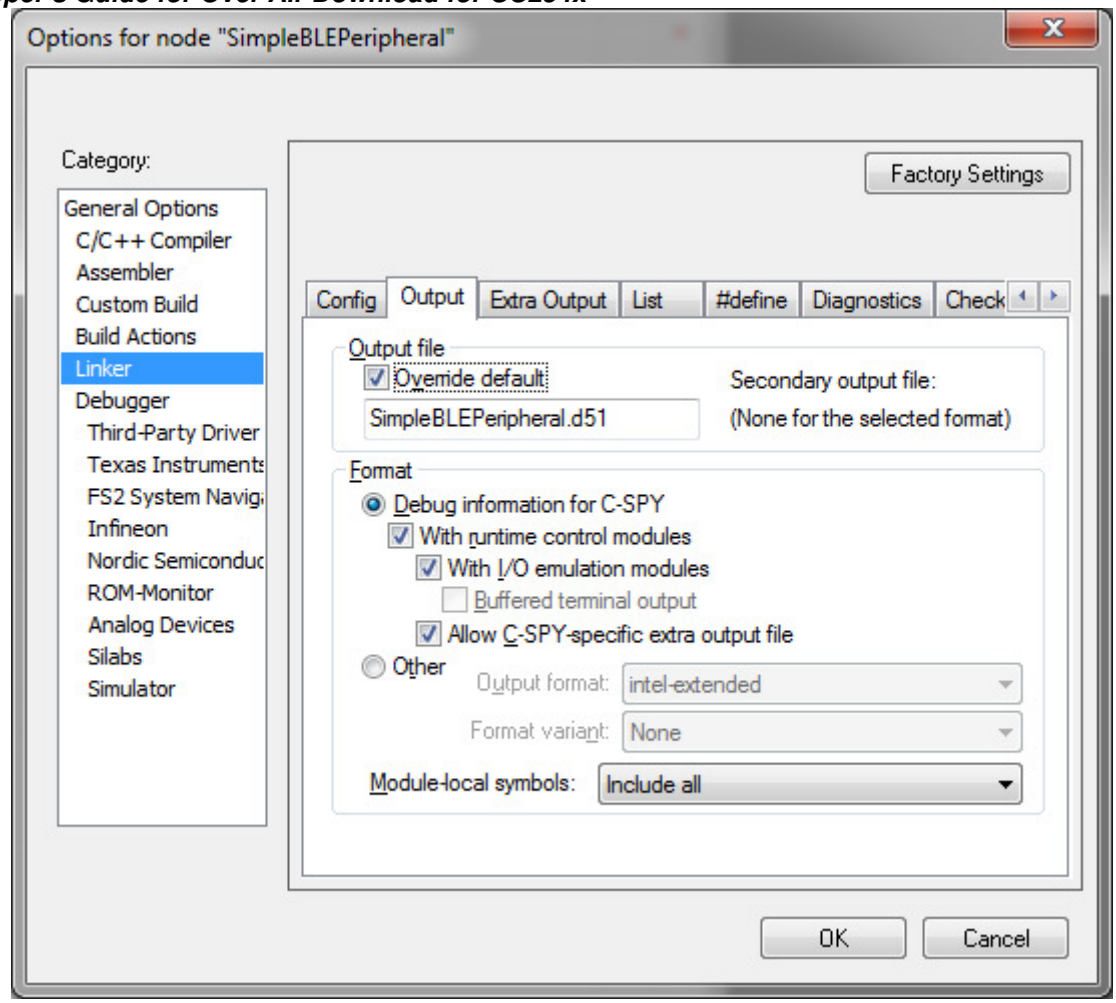


Figure 1: Enable extra linker output.

8. Select Project->Options→Linker→Output→Format→Allow C-SPY-specific extra output as shown above.

9. Select Project->Options→Linker→Extra Output→Format.

   To produce a .hex image that can be appended to the .hex image of the OBL or BIM .hex image and thereby produce a super .hex image for production programming:

   Select the 'Format→Output format' to be 'intel-extended' and check the box for 'Output file→Override default' and use the '.hex' suffix for the output file.

10. Select Project->Options→Debugger→Texas Instruments→Download.

Uncheck the box to 'Erase flash' and check the box to 'Retain unchanged pages' so that the BIM is not erased when this application image is downloaded from IAR. When running an OAD-enabled image for the first time after programming it directly (i.e. not after sending it via the OAD process), there will be a delay of about 15 seconds while the BIM verifies the image CRC.

11. Build and download the project on a SmartRF05 with a CC2541EM.

You may receive a post-build error. This is because you have not yet produced the .sim file. You can ignore this error at this point.

10.1 Create the Application Image-B build target.

Similar steps as above will be followed to create the Image-B build target, using the Image-A build target as the source. Other than changing the name of the build target, the preprocessor definition needs to be changed from HAL_IMAGE_A to HAL_IMAGE_B, the post-processing build actions path to the EXE must be changed, and the linker control file must be the one for Image-B:

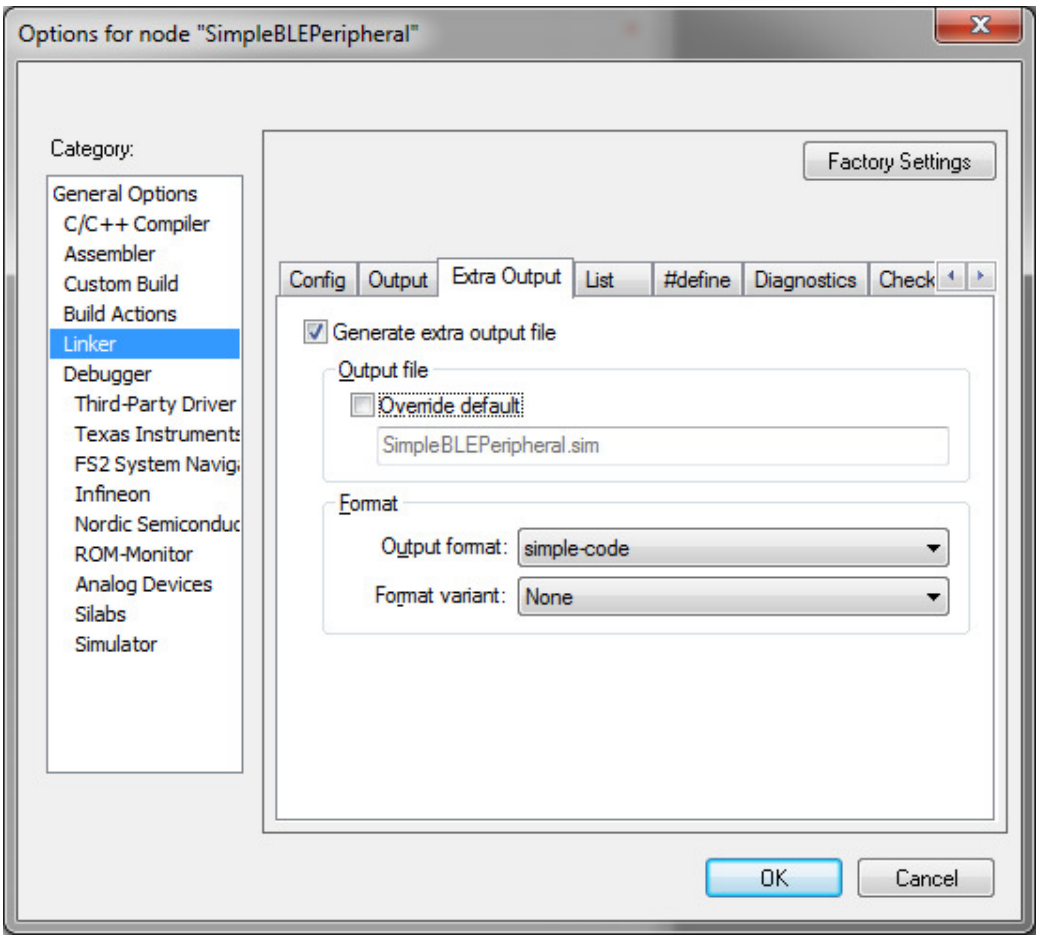
```
$PROJ_DIR$\..\..\common\CC2541\cc254x_f256_imgB.xcl
```

Once you have set up the Image-B project configuration, build and download this project also. At this point, the "super" hex image on the CC254x contains the BIM, Image A, and Image B.

11. Build the Application Image to send over the air

After programming the BIM (Section 9) and then programming images A and B (Section 10) with the options to preserve unused flash (and thereby to not over-write the BIM in flash), the image that will be sent over-the-air must be built and loaded into the OAD Manager. The following steps will allow you to produce a .bin image for sending over-the-air as an OAD update. For this example, we will be producing an over-the-air Image B.

1. Change the 'Format→Output format' to be 'simple-code' and do not check the box for 'Output file→Override default' as shown below.



2. Open the 'oad_target.c' module and change the version field in the Image Header from 0x0000 to 0x0002 as shown in the below figure. This will mimic an updated version of the image when the BIM checks the image header.

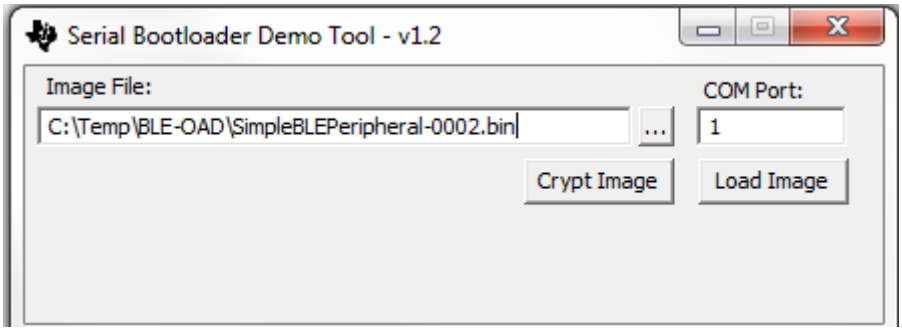
```
/*********************************************************************
 * CONSTANTS
 */

#define OAD_FLASH_PAGE_MULT  ((uint16)(HAL_FLASH_PAGE_SIZE / HAL_FLASH_WORD_SIZE))

#if defined (FEATURE_OAD_SECURE) && defined (HAL_IMAGE_A)
  // Enabled to ONLY build a BOOTSTRAP Encrypted Image-A (for programming over
  // BEM, not BIM). Comment line below to build a non-bootstrap Encrypted Image-A.
  #define BOOTP_E_IMAGE_A
#endif

#if !defined (OAD_IMAGE_VERSION)
  #define OAD_IMAGE_VERSION    0x0002
#endif

#if !defined (OAD_IMAGE_A_USER_ID)
  #define OAD_IMAGE_A_USER_ID  {'A', 'A', 'A', 'A'}
#endif

#if !defined (OAD_IMAGE_B_USER_ID)
  #define OAD_IMAGE_B_USER_ID  {'B', 'B', 'B', 'B'}
#endif

/*********************************************************************
```

3. Build but **do not download** the target build. This will produce the output binary file for OAD in the output directory here:

   `INSTALL_DIR\Projects\ble\SimpleBLEPeripheral\CC2541DB\CC2541-OAD\Exe\SimpleBLEPeripheral.bin`

   If this is your first time building with simple-code (.sim) output, you will have to build the project twice for the binary file to be produced.

4. Launch the SBL Demo Tool (download from http://processors.wiki.ti.com/index.php/Category:SBLTool) and load the image file as shown below. Make sure to choose the correct COM Port for your serial cable.



5. Load and run the OAD Manager application into a separate CC254xEM on a SmartRF05 board which is connected via a serial cable to the PC running the SBL Demo tool. The OAD Manager application can be found here:

   `INSTALL_DIR\Projects\ble\OADManager`

6. On the device which is loaded with the OAD manager, press the joystick down, away from the LCD, and not that the LCD message indicates that the OAD Dongle is now in SBL mode.

7. Press the 'Load Image' button on the SBL Demo tool and verify successful completion (the LCD should indicate this).

8. On the device which is loaded with the BIM and the Image A version 0x0000, begin advertising if you aren't already.

9. On the OAD Manager, begin scanning by pressing the joystick up, toward the LCD and follow the rest of the instructions for creating a BLE connection which are described for the SimpleBLECentral (e.g. press the joystick to the left to toggle through the devices found and press the joystick down into the board to connect).

10. Upon connection, the OAD Manager will read Image A's image header and realize that it has a newer version than Image A. The over-the-air update will then begin. Note the LCD message which shows the BLE OAD progress. The default behavior of the OAD Target Profile is to automatically start the OAD if the Dongle has an image with a higher version number. This behavior can (and probably should) be enhanced per the requirements of the specific project using it.

11. After the OAD download completes, the connection will be dropped because the running image version 0x0000 will invalidate itself and reset so that the BIM can get control and jump to the image version 0x0002 as the new running image. This is further explained in the OAD Target profile section below.

12. When the OAD target indicates that it is running again, establish another connection with it from the OAD Manager and note that this time an OAD process is not started, but the OAD-enabled device simply reports back its current Image Header information.

12. Maximize the Image-B area (while minimizing the Image-A area)

The following is an exercise to get insight into the inter-dependency between the image area constants and the linker control files as well as the in-line assembly code in the BIM. The inter-dependency between the C-code constants and the linker control file memory placement must be manually coordinated because the IAR IDE cannot be configured to automatically manage this.

1. Minimize the area available for Image-A in its corresponding linker file, cc254x_f256_imgA.xcl as shown below:

```
//
// CODE
//
-D_CODE_BEG=0x0830    // First page is for the BIM which intercepts the H/W INTVECS.
-D_CODE_END=0x2FFF    // Next 5 pages of Bank 0.
```

2. Maximize the area available for Image-B in its corresponding linker file, cc254x_f256_imgB.xcl as shown below:

```
//
// CODE
//
-D_CODE_BEG=0x3030    // Last 10 pages of Bank 0.
-D_CODE_END=0x7FFF
//
```

3. Change the location of the OAD data structures to the new Image-B 1st page as shown below:

```
-Z(CODE)CHECKSUM=0x3000-0x3001
-Z(CODE)IMAGE_HEADER=0x3002-0x300F
-Z(CODE) AES_HEADER=0x3010-0x302F
```

4. Change the range of the CRC calculation as shown below:

```
-J2,crc16,=3004-_BANK4_END
```

5. Change the BIM sizes for the Images in ibm_main.c as shown below:

```
#define BIM_IMG_A_PAGE        1
#define BIM_IMG_A_AREA        60

#define BIM_IMG_B_PAGE        6
#define BIM_IMG_B_AREA       (124 - BIM_IMG_A_AREA)
```

6. Make a corresponding change to the OAD Profile sizes for the Images in oad.h as shown below:

```
#if !defined OAD_IMG_A_PAGE
```

```
#define OAD_IMG_A_PAGE        1
#define OAD_IMG_A_AREA        60
#endif

#if !defined OAD_IMG_B_PAGE
// Image-A/B can be very differently sized areas when implementing BIM vice OAD boot
loader.
#if defined FEATURE_OAD_BIM
#define OAD_IMG_B_PAGE        6
```

7. Change the address to jump to Image B in the BIM in the main function of bim_main.c:

```
asm("LJMP 0x3030");
```

8. Change the Image-B INTVEC Table relocation address in bim_ivecs.s51

```
IMGB  EQU  0x3030
```

13. Adding Encryption to OAD

13.1 Summary

This section describes the steps that are needed to use the hardware AES engine to add encryption to the OAD mechanism. This is performed by signing a binary image, encrypting this binary image, uploading this image to an OAD Manager, sending the image over the air, decrypting the image on the OAD Target, and finally verifying the downloaded image by checking its signature. The SimpleBLEPeripheral project included with the BLE v1.3 stack has OADEncrypted project configurations for images A and B but you can create these configurations for any project with the following steps (assuming you followed the sections of this guide creating the general OAD configurations).

13.1 Producing Boot Code

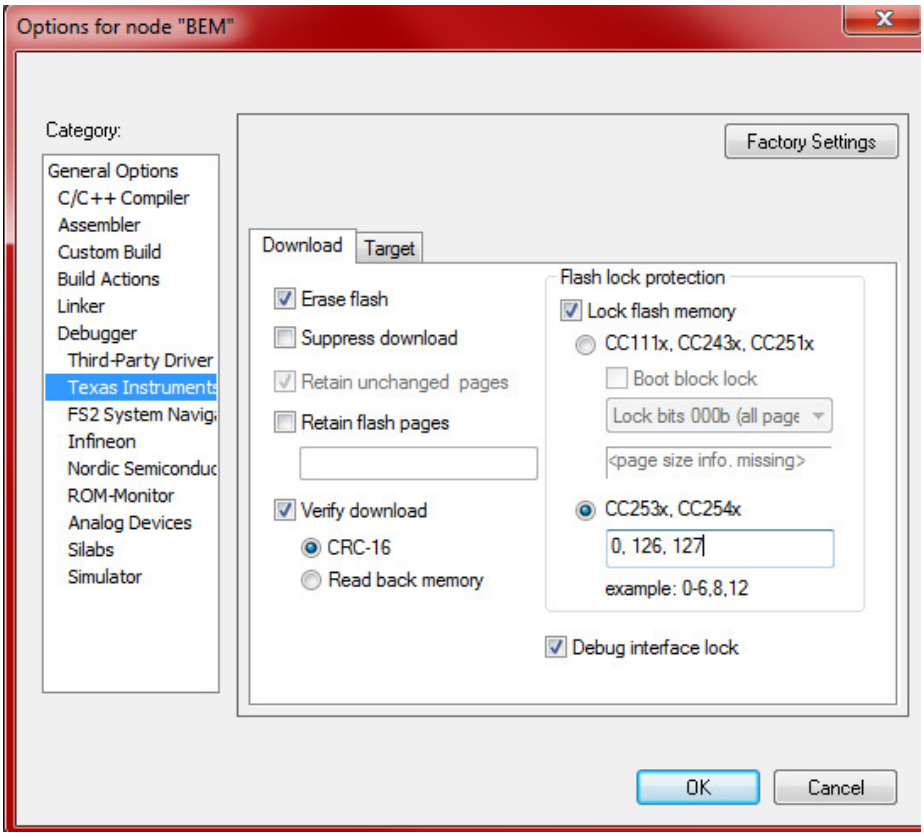
The Boot Encrypted Manager, or BEM, is separately built and debugged or programmed via the IAR IDE. The project is located at:

$INSTALL_DIR\Projects\ble\util\BEM

The default configuration is with the download option to erase flash in order to start a CC254x SOC with clean flash (and thus clean NV). Before debugging or physically programming the OAD application code produced in the next section, this OAD boot code must first be programmed into the flash (but only this once, since, as the following section mentions, the default option for application code is to preserve this OAD Boot code on successive debugging or programming).

The .hex file produced by this project build can be programmed directly using a tool like the SmartRF Programmer. This .hex file can be merged with the .hex file from the build of the application image in order to produce a "super" .hex file that contains both images for factory programming of a device that is OAD-enabled with its first valid application image ready to run and the boot loader present to complete the final step of the OAD process.

1. When compiling this project, be sure to lock the debug interface so that it is impossible to read the keys from flash. Also, you can choose to provide software write protection by setting the lock bits for the flash pages which contain the BEM (page 0) and the aesKey (pages 126, 127). Select "Lock Flash Memory" and "Debug interface lock" as shown below:



2. Two data structures, the "aesKey" and the "ivNonce," are used to sign and decrypt / encrypt the binary images. If desired, you can change these in the BEM.

   The aesKey is defined in bem_main.c as:

```
71 static const uint8 aesKey[KEY_BLENGTH] = {
72   // This dummy key must be replaced by a randomly generated key that is kept secret.
73   0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F
74 };
75
```

   The ivNonce is defined in bem_main.c as:

```
// A0: L-encoding of L-1 = 2-1 = 1; starting 2-byte CTR at 1.
uint8 ivNonce[KEY_BLENGTH] = { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 };
```

13.2 Producing Bootstrapped Image A

1. Select Project→Edit Configurations→New… and add a new build target: CC2541-OAD-Encrypted-ImgA. Make sure that this is based on the CC2541-OAD-ImgA configuration created in Section 10.

2. Immediately compile the new build target. It should build and link without errors or warnings since it is an exact copy of an existing good target build. If this step does not succeed, first fix the problems with the original build target and application before re-starting from step 1.

3. Select Project->Options→C/C++ Compiler→Preprocessor→Defined symbols and add the following new definition:

   `FEATURE_OAD_SECURE`

4. Select Project->Options→Build Actions→Post-build command line and edit path to the .sim file

```
"$PROJ_DIR$\..\..\common\CC2540\cc254x_ubl_pp.bat" "$PROJ_DIR$" "ProdUBL"
"$PROJ_DIR$\CC2541-OAD-Encrypted-ImgA\Exe\SimpleBLEPeripheral"
```

Note that the relative path above will contain different names as you work with projects other than the CC2541 SimpleBLEPeripheral.

5. Select Project->Options→Linker→Config→Linker configuration file and paste the following line:

```
$PROJ_DIR$\..\..\common\CC2541\cc254x_f256_imgAe.xcl
```

6. Build and download the project onto the SmartRF which will now be running the BEM and Image A.

7. Repeat steps 1-6 for Image B. You will need to change the path in step 4 to **CC2541-OAD-Encrypted-ImgB**. You will also need to change the linker file in step 5 to:

```
$PROJ_DIR$\..\..\common\CC2541\cc254x_f256_imgBe.xcl
```

13.3 Producing Image A and Image B images to send over the air

1. In the Image A project, comment out the following line in oad_target.c since this code will not be bootstrapped (it will be downloaded over the air).

```
67 #if defined (FEATURE_OAD_SECURE) && defined (HAL_IMAGE_A)
68   // Enabled to ONLY build a BOOTSTRAP Encrypted Image-A (for programming over
69   // BEM, not BIM). Comment line below to build a non-bootstrap Encrypted Image-A.
70   //#define BOOTP_E_IMAGE_A
71 #endif
```

2. Change the version number so that an OAD will occur.

```
73 #if !defined (OAD_IMAGE_VERSION)
74   #define OAD_IMAGE_VERSION    0x0003
75 #endif
```

3. Compile, but do not download, to produce the .bin file.

4. Repeat steps 2-3 for Image B. You will need to change the version number to something other than the previous two that were used:

```
73 #if !defined (OAD_IMAGE_VERSION)
74   #define OAD_IMAGE_VERSION    0x0002
75 #endif
```

At this point, you will have produced the unsigned .bin files for images A and B to be sent over the air.

13.4 Signing and Encrypting Images A and B.

In this step, we will use another project to sign and encrypt Images A and B: the Encrypted Bootloader (EBL) located at $INSTALL_DIR\Projects\ble\util\EBL. You will load this onto a separate Smart RF since we just programmed the BEM and Image A onto the first Smart RF. The following steps will describe how to do this as well as describe how to customize the signing and encryption to use your own keys.

1. The EBL is not set up as a signer by default. To enable this, select Project->Options→C/C++ Compiler→Preprocessor→Defined symbols and add the following new definitions:

```
SBL_SIGNER
SBL_SECURE=FALSE
```

2. If you altered the aesKey in the BEM, you should replicate it in sbl_exec.c:

```
static const uint8 aesKey[KEY_BLENGTH] = {
#if defined AES_TEST_VECS
  0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF
#else
  // This dummy key must be replaced by a randomly generated key that is kept secret.
  0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F
#endif
};
```

3. If you altered the ivNonce in the BEM, you should replicate it in sbl_exec.c:

```
// A0: L-encoding of L-1 = 2-1 = 1; starting 2-byte CTR at 1.
uint8 ivNonce[KEY_BLENGTH] = { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 };
```

   These two data structures define the signing and encryption and must be the same in both the EBL and BEM.

4. Launch the SBL Demo Tool (download from http://processors.wiki.ti.com/index.php/File:BL_Encrypter.zip) and load the image file by choosing the "Load Image" button. Make sure to choose the correct COM Port for your serial cable.

5. Once the image is loaded onto the EBL, choose the Crypt button to read back an encrypted binary image. You may want to rename the target file if you do not want to overwrite the original.

6. Do this for both the Image A and Image B .bin images created from section 13.3.

13.5 Performing the OAD

1. Load and run the OAD Manager application into a separate CC254xEM on a SmartRF05 board (or overwrite the EBL) which is connected via a serial cable to the PC running the SBL Demo tool. The OAD Manager application can be found here:

   ```
   INSTALL_DIR\Projects\ble\OADManager
   ```

2. On the device which is loaded with the OAD manager, press the joystick down, away from the LCD, and note that the LCD message indicates that the OAD Dongle is now in SBL mode.

3. In the SBL Demo Tool, browse to the signed Image B image from section 13.4. Press the 'Load Image' button on the SBL Demo tool and verify successful completion (the LCD should indicate this).

4. On the device which is loaded with the BEM and the Image A version 0x0000, begin advertising if you aren't already.

5. On the OAD Manager, begin scanning by pressing the joystick up, toward the LCD and follow the rest of the instructions for creating a BLE connection which are described for the SimpleBLECentral (e.g. press the joystick to the left to toggle through the devices found and press the joystick down into the board to connect).

6. Upon connection, the OAD Manager will read Image A's image header and realize that it has a newer version than Image A. The over-the-air update will then begin. Note the LCD message which shows the BLE OAD progress. The default behavior of the OAD Target Profile is to automatically start the OAD if the Dongle has an image with a higher version number. This behavior can (and probably should) be enhanced per the requirements of the specific project using it.

7. After the OAD download completes, the connection will be dropped because the running image version 0x0000 will invalidate itself and reset so that the BEM can get control and jump to the image version 0x0002 as the new running image. This is all dependent on successful decryption and signature calculation. If the signature that the BEM calculates doesn't match the signature supplied by the downloaded Image B, Image B will be invalidated and Image A will continue to run.

8. When the OAD target indicates that it is running again, establish another connection with it from the OAD Manager and note that this time an OAD process is not started, but the OAD-enabled device simply reports back its current Image Header information. If you were to repeat steps 1-8 in this section using the signed Image A from section 13.4, you would be able to perform another successful OAD because the image version is different.

14. Proprietary TI OAD Target BLE Profile

The design has been made as simple and as versatile as possible in order to minimize code size and complexity and to provide room for customer customizations and enhancements (such as encrypting the image that is given to an OAD server to send over the air) which are too costly to be included by default for a general audience.

14.2 Dependencies

The OAD Profile depends on the final customer application to determine and control connection parameters and criteria.

14.3 Messages

"Write with no response" has been chosen as the default message type from the OAD Manager 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 Notifications requires adding the GATT_ClientInit() which currently costs about 4.5 Kb. In a noisy or otherwise lossy environment, the "write with no response" may not be sufficient to successfully transmit an entire image and OSAL Timer driven timeouts and re-tries may need to be added. Since the OAD Target will have already initialized the GATT_Client, notifications were chose as the default message type from the OAD Target to the OAD Manager.

14.4 Characteristics

The OAD Target Profile has only two characteristics: OAD Image Identify and OAD Image Block Transfer. The burden is on the OAD Manager to discover the handles of these two characteristics on the OAD Target. The Image Identify characteristic is used to exchange the image header information in order for the OAD Target to decide if an OAD should occur. The OAD Image Block Transfer characteristic is used to request and transfer a block of the OAD image.

14.5 Initiation of OAD process

After establishing a new connection, updating the connection interval for a faster OAD, and enabling notifications on the OAD Target, the OAD Manager shall write to the Image Identify characteristic of the OAD Target. The message data will be the image header of the image that is available for OAD. 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. If the OAD Target determines that the image available for OAD is better than its own, the OAD Target will initiate the OAD process by notifying the Image Block Transfer characteristic to the OAD Manager. Otherwise, if the OAD Target does not start an OAD process, it shall respond by notifying the Image Identify characteristic with its own Image Header.

14.6 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 OAD manager by notifying the Image Block Transfer characteristic with the correct block index. The OAD Manager shall then respond by writing to the Image Block Transfer characteristic. The message data will be the image block index requested followed by the 16-bytes of image that correspond to that block.

14.7 Completion of OAD Process

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 OAD Manager will then respond as described in the previous section. 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 OAD Manager, which will suffer a lost BLE connection to the OAD Target during this verification and instantiation process, to re-start scanning and then to re-establish a connection and verify that the new image is indeed running.