Z-Stack
OTA Upgrade
User's Guide

Document Number: SWRA353

Texas Instruments, Inc.
San Diego, California USA

Version	Description	Date
1.0	Initial release.	11/17/2010
1.1	Update for CC2538 run-in-place OTA images. Fix output location of generated binary file.	04/01/2013
1.2	Added additional clarifications for CC2538 OTA.	06/13/2013
1.3	Updated OTAConverter.exe location Revised OTA PC tool section(s) to reflect the new OTAServer 2.0.0 tool. Documented need to disable flow control on CC2538 for the OtaServer PC tool.	11/25/2013
1.4	Revised to align with consolidation of OTA IAR projects with existing Sample Switch application	06/09/2014
1.5	Added sections to describe “stitching” for CC2530 and CC2538	06/30/2014

TABLE OF CONTENTS

1. Introduction

This document is a Developer's Guide for the Over the Air (OTA), Upgrade Cluster in Texas Instruments' Z-Stack ZigBee platform.

1.1 How to Read this Document

This document is presented in 3 parts. The first part, ***OTA Overview***, gives a functional description of the OTA subsystem. The second part, ***Using the OTA Sample Application***, provides step by step instructions for building, installing, and running the Sample OTA application that comes with Texas Instruments' Z-Stack. The third part, ***Adding Client Functionality to an Application***, provides step by step instructions for configuring a Z-Stack application to operate as an OTA Client.

1.2 Definitions, Abbreviations, Acronyms

Term	Definition
IEEE 802.15.4	The IEEE 802.15.4 protocol specifies the physical and medium access layers for wireless Personal Area Networks targeted at low power, low data rate applications.
OTA	Over the Air
OTA Client	A device capable of down loading an image using the OTA cluster from an OTA Server.
OTA Server	A device capable of hosting an image for download via the OTA cluster by an OTA Client.
PAN	Personal Area Network
SoC	System on Chip – processor core and radio are integrated on same device

1.3 References

[1] ZigBee Alliance Document 095264, *ZigBee OTA Upgrade Cluster Specification*.

2. OTA Overview

The OTA, Over the Air, Upgrade Cluster provides a standard mechanism for wirelessly upgrading a ZigBee device's firmware. Over the air upgrade sessions take place between a client and server. The OTA Client downloads an OTA Upgrade Image. The OTA Server hosts OTA Upgrade Images.

Texas Instruments' Z-Stack ZCL, ZigBee Cluster Library, provides support for client and server operation of the OTA Upgrade Cluster. The Texas Instruments Z-Stack implementation of the OTA Upgrade Cluster consists of the following components:

- OtaServer PC Application
- OTA Image Converter Application
- ZCL Implementation of the OTA Protocol
- Boot loaders for the supported platforms
- Sample OTA Application

2.1 OTA Theory of Operation

This section provides a simple description of how the OTA Upgrade Cluster is used to update a device's firmware. For more detail, see the ZigBee Alliance Document 095264, *ZigBee OTA Upgrade Cluster Specification*.

Communication via the OTA Upgrade Cluster takes place using the following command messages:

- Image Notify
- Query Next Image Request
- Query Next Image Response
- Image Block Request
- Image Block Response
- Update End Request
- Update End Response

The *Image Notify* message is sent unicast or is broadcast by an OTA Server to notify OTA Clients that new images are available. The Image Notify does not contain information about the new images. The Image Notify only indicates new images are available. OTA Clients determine if these new images apply to them using the Query Next Image request and response messages.

Periodically, or after receipt of an Image Notify message, an OTA Client sends a *Query Next Image Request* message to an OTA Server. The Query Next Image Request message contains the version of the firmware currently running on the client. On receipt of the Query Next Image Request, the server decides if an image update should take place and which image the client should update to. The server responds to the query next image request with a *Query Next Image Response* message.

The *Query Next Image Response* message may instruct the client to download new firmware, or it may inform the client that no firmware is available. Should a download take place, the client controls the download. During the download, the client sends *Image Block Request* messages to the server and receives *Image Block Response* messages from the server with chunks of the upgrade image. The client writes the received image blocks to a secondary storage location. In the Z-Stack sample applications, this secondary storage can be on-chip or off-chip flash memory, depending on the hardware platform.

After the client has downloaded the entire upgrade image, the client sends the *Upgrade End Request* message to the server. The server then responds with an *Upgrade End Response* message. The Upgrade End Response message contains information about when the client should switch to the new firmware. The client may switch to the new firmware immediately or it may be instructed to wait a specified period of time.

In the Z-Stack sample applications, when it is time for a client to switch to the new image, the client writes a non-volatile, NV, memory location indicating new firmware is available. Then the client reboots. A boot loader on the client sees that the new image is available. Then the boot loader copies the new image from secondary storage to the operational memory space. Then the new firmware is started. On some platforms, such as the Cortex-M3 based CC2538, only a boot manager is required since the OTA images can be run in-place.

2.2 OtaServer PC Tool

The OtaServer PC tool is the front end of Texas Instruments' Z-Stack OTA Server sample application. The OtaServer PC tool requires Microsoft Windows XP or newer and an RS-232 serial port. The OtaServer connects to a SmartRF05, SmartRF06 or to an MSP430 EXP board running the OTA Dongle sample application via an RS-232 serial port (or in the case of the SmartRF06 it will use UART over USB).

2.3 OTA Image Converter

The OTA Image Converter tool converts IAR simple binary files into OTA Upgrade files. The image converter tool requires Microsoft Windows XP or newer. The OTA Image Converter is a command line utility.

2.3.1 OTA Upgrade Image File Format

This section contains a simple description of the OTA Upgrade Image File Format. For more detail, see the ZigBee Alliance Document 095264, *ZigBee OTA Upgrade Cluster Specification*.

OTA Update Image files are broken into the following parts:

- Image Header
- Image Data
- Certificate
- Signature

The *Image Header* contains the following information about the image:

- Header Version
- Manufacturer ID
- Image Type
- File Version
- Stack Version
- Header String
- Image Size
- Security Credentials
- Hardware Version

The *Image Data* contains the machine code for the new firmware from the IAR simple binary file. The *Signature* and *Certificate* are optional and contain an AES encrypted copy of an MMO hash of the Image and a certificate to verify the integrity of the OTA Image.

2.3.2 Generating an IAR Simple Binary

The IAR Simple Binary file used as an input to the OTA Image Converter is generated by IAR Embedded Workbench. To generate a simple binary, perform the following:

1. Select *Project>Options* from the IAR Embedded Workbench menu.
2. Select *Linker* from the Category list in the options dialog.
3. Select the *Extra Output* tab in the options dialog.
4. Check the *Generate extra output file* checkbox.
5. From the *Output format* pull-down list, select *simple-code*
6. Compile the project.

The simple binary will be located in <*project name*>/Exe/<*project name*>.bin

Note that the sample applications for the Cortex-M3 based targets already generate a suitable binary for the image converter utility.

2.3.3 OTA Image Converter Command Line Arguments

The first argument to the OTA Image Converter is the IAR simple file to convert.

The following arguments are mandatory:

- -m<id> - Image Manufacturer Identifier
- -t<id> - Image Type Identifier
- -v<id> - Image Version

The following arguments are optional:

- -p<platform> Hardware platform. Valid platforms are:
  - CC2538
  - CC2530DB
  - EXP5438
- -o<path> Output folder to put the image in
- -s<path> Location of the certificate used to sign the image

Note that for the CC2538 target, the '-p' argument is mandatory, or the image will not be processed correctly.

This is an example command line for the Image Converter Tool:

```
OtaConverter.exe SampleApp.bin –m0x1001 –t0x1234–pCC2530DB
```

2.3.3.1 Image Manufacturer Identifier (-m)

The Manufacturer Identifier is a 16-bit hexadecimal number that specifies the manufacturer of the device the OTA Upgrade Image is intended for. Manufacturer IDs are assigned to ZigBee device manufacturers by the ZigBee Alliance. The manufacturer identifier is specified in the –m command line argument to the OTA Image Converter.

2.3.3.2 Image Type Identifier (-t)

The Type Identifier is a 16-bit hexadecimal that specifies the type of device the OTA Upgrade Image is intended for. The type ID is manufacturer specific and typically corresponds to the model of the device being upgraded. The type identifier is specified in the –t command line argument to the OTA Image Converter.

2.3.3.3 Image Version (-v)

The Image Version is a 32-bit hexadecimal that specifies the version of the OTA Upgrade Image. The image version is specified in the –v command line argument to the OTA Image Converter.

2.3.3.4 Platform Target (-p)

The Platform the Image is intended for is specified with the –p option. The default platform is the SmartRF05 with an attached CC2530EM module. The following platforms can be specified to the OTA Image Converter:

- cc2538 – SmartRF06 with an attached CC2538EM
- cc2530 – SmartRF05 with an attached CC2530EM
- msp54xx – EXP430F5438 with attached CC2520EM

2.3.3.5 Output Directory Option (-o)

The Output Directory is optional. It specifies the folder the image will be put into after it is created. This can be the Image Folder used by the OtaServer PC Tool. This is an example of the Output Directory option:

```
OtaConverter.exe  app.bin –m0x1001 –t0x1234 –v0xABCD9876 –o"c:\Image Folder"
```

2.3.3.6 Signature Option (-s)

The Signature is optional, but required for ZigBee Smart Energy applications that utilize the OTA cluster as dictated by the Smart Energy 1.1 specification. The converter tool defaults to not using a signature. The location of the certificate must be provided with the signature option. This is an example of the Signature Option:

```
OtaConverter.exe  app.bin –m0x1001 –t0x1234 –v0xABCD9876 –s"c:\cert.txt"
```

Certificates are provided by Certicom Inc. A sample certificate is provided as part of the Z-Stack distribution in the following folder: Projects\zstack\SE\SampleApp\Source\OTA Cert.

2.4 Z-Stack Implementation of the OTA Upgrade Cluster

The embedded firmware for the OTA Upgrade cluster in Z-Stack is implemented in the following files:

- zcl_ota.c
- zcl_ota.h
- ota_common.c
- ota_common.h
- ota_signature.c
- ota_signature.h

2.4.1 ZCL_OTA.C and ZCL_OTA.H

The zcl_ota.c and zcl_ota.h files contain the following:

- Format OTA Upgrade Cluster messages
- Parse OTA Upgrade Cluster messages
- Logic to serve upgrade images
- Logic to download upgrade images

2.4.2 OTA_COMMON.C and OTA_COMMON.H

The ota_common.c and ota_common.h files contain OTA functionality like parsing OTA headers that are used by OTA Clients and OTA Server applications, the OtaServer PC Tool, and the OTA Image Converter.

2.4.3 OTA_SIGNATURE.C and OTA_SIGNATURE.H

The ota_signature.c and ota_signature.h files contain code to sign OTA images and validate signatures using AES encryption of an MMO Hash of an OTA image.

2.4.4 hal_ota.c and hal_ota.h

The hal_ota.c and hal_ota.h files contain platform dependent functions for managing and manipulating OTA image files including support for external versus internal flash etc.

2.4.5 Compile Time Configuration

The OTA firmware can be configured with the following compile time definitions:

- OTA_CLIENT=TRUE : Includes OTA Client functionality.
- OTA_SERVER=TRUE : Includes OTA Server functionality.
- OTA_MMO_SIGN=TRUE : Adds image signature handling.
- OTA_HA : Add HA-based OTA functionality.
- OTA_INITIAL_IMAGE: Used for CC2538. For specifics associated with a combined image that includes both the OTA bootloader and the ZigBee application with OTA client capabilities. See section 3.2.2 for more information.
- HAL_IMG_AREA=0 : Used for CC2538. Specifies IMAGE A properties.
- HAL_IMG_AREA=1 : Used for CC2538. Specifies IMAGE B properties.

2.4.6 OTA Upgrade API

Little interaction between the application and the ZCL OTA is necessary with Z-Stack's implementation of the OTA Upgrade Cluster. Z-Stack provides an OTA API to notify the application about the beginning and end of an OTA; to give application the ability to permit/disallow OTA operation; to give the application the ability to send an image notify; and to give the application the ability to query servers for the next upgrade cluster.

2.4.6.1 OSAL Callback Events

One application task can be registered with the OTA to receive callback events by calling the *zclOTA_Register* function. The following events are sent to the registered application task:

- ZCL_OTA_START_CALLBACK
- ZCL_OTA_DL_COMPLETE_CALLBACK

Events are sent as OSAL messages with the following body:

```
typedef struct
{
```



```
  osal_event_hdr_t hdr;
  uint8 ota_event;
} zclOTA_CallbackMsg_t;
```

The ZCL_OTA_START_CALLBACK is sent to indicate a successful or failed attempt to start a download. The ZCL_OTA_DL_COMPLETE_CALLBACK is sent when a download completes indicating the download completed successfully or failed to complete.

2.4.6.2 zclOTA_RequestNextUpdate

The zclOTA_RequestNextUpdate function is called by applications to send an OTA Query Next Image message to an OTA Server.

The method of discovering servers and determining when to query the server is left up to the application. In the Z-Stack sample applications, the application performs a match descriptor request on the OTA Upgrade Cluster to discover a Server. The application then calls zclOTA_RequestNextUpdate on all discovered servers until the application receives a successful ZCL_OTA_START_CALLBACK event.

2.4.6.3 zclOTA_SendImageNotify

The zclOTA_SendImageNotify function can be called on an OTA Server to send an Image Notify message.

2.4.6.4 zclOTA_PermitOta

The zclOTA_PermitOta function can be called to enable or disable OTA Upgrades. When OTA is disabled, the OTA Client ignores Image Notify messages and the OTA Server sends no image available responses to Query Next Image Request messages.

2.4.7 OTA Client Memory Partition

When an OTA Client downloads a new upgrade image, it must store the image in secondary storage. Later a boot loader copies the image from secondary storage into the operational space. Or in the case of a Cortex-M3 platform, a boot manager simply passes control to the newly downloaded image.

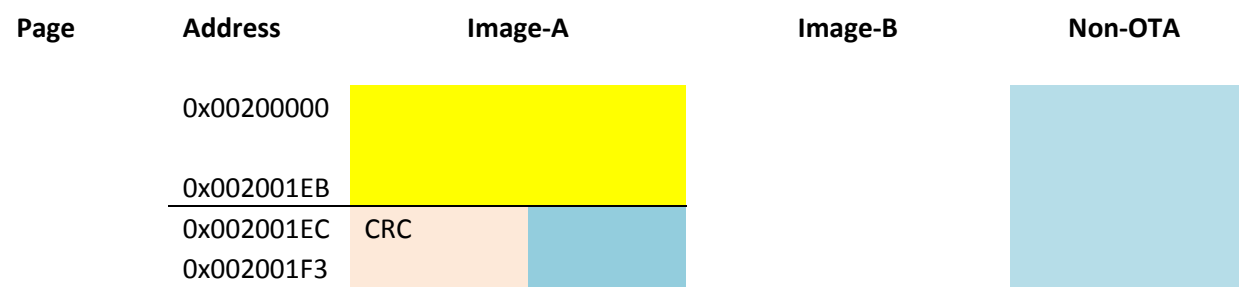
Z-Stack provides a sample OTA Client and boot loader for the following platforms:

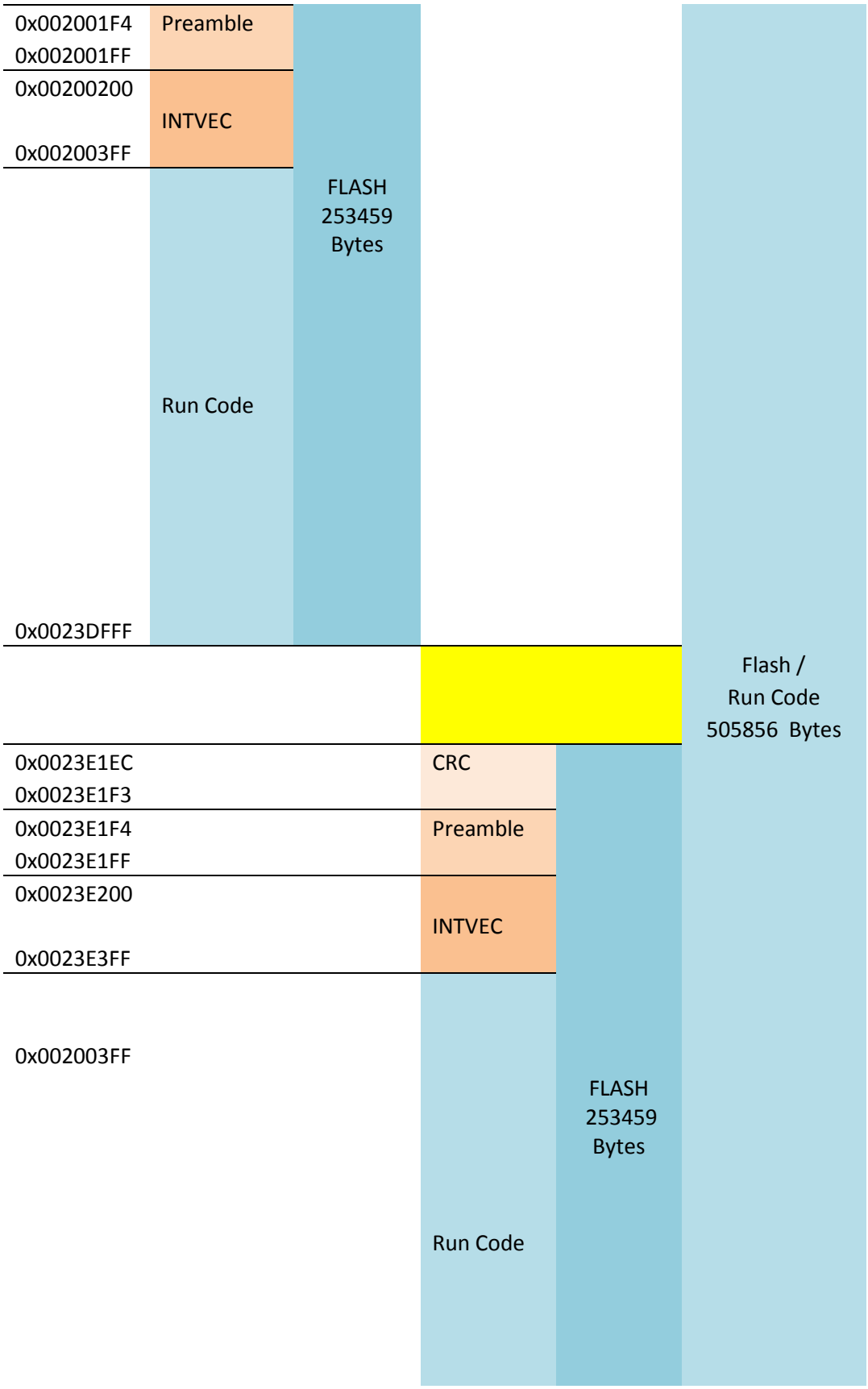
- CC2538 Cortex-M3 SoC
- SmartRF05 with attached CC2530EM
- EXP430F5438 with attached CC2520EM

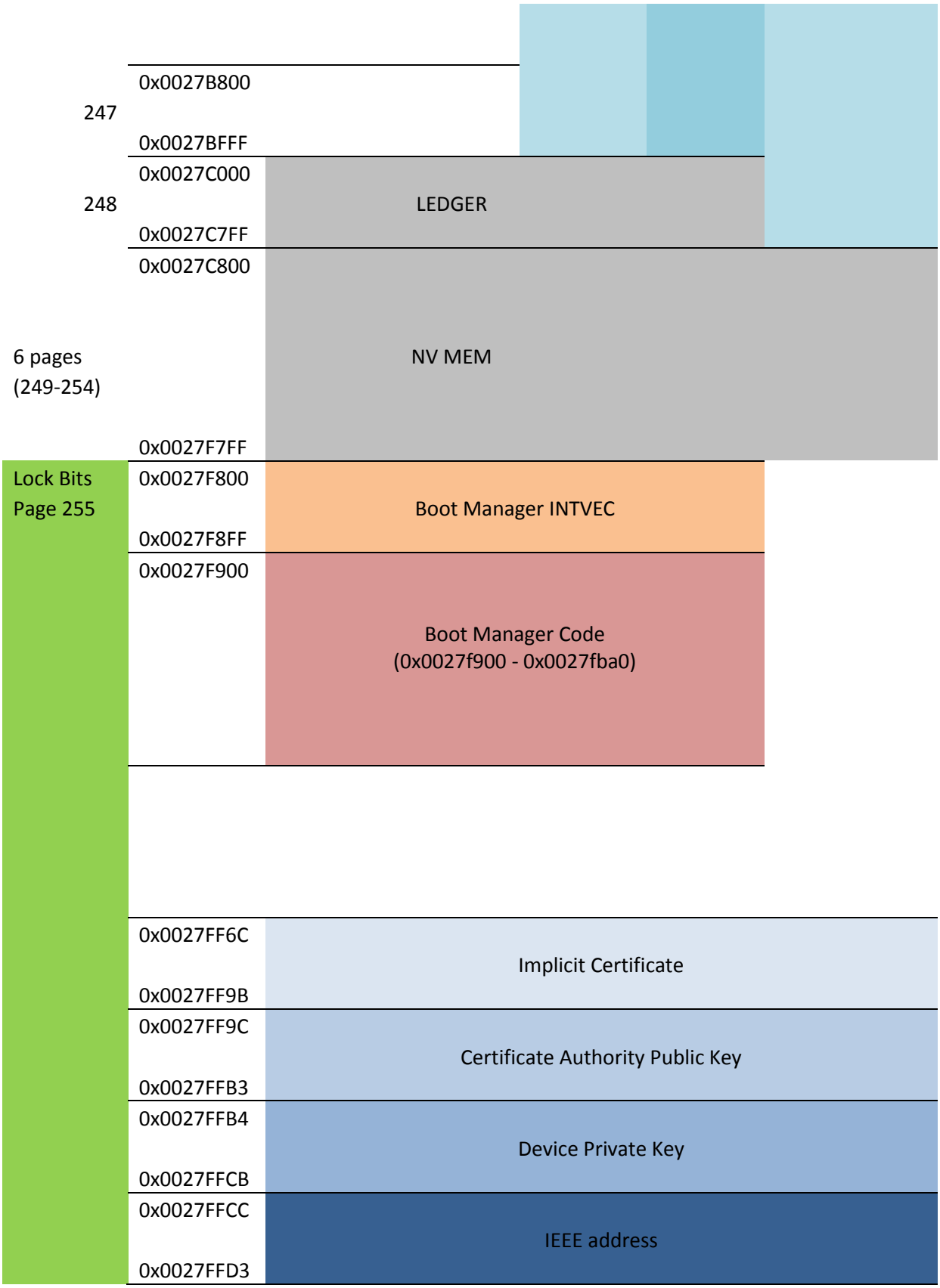
The following sections describe the memory partition on the above platforms.

2.4.7.1 CC2538 Cortex-M3 SoC

On the CC2538, downloaded images are not relocated and are instead “run-in-place”, a feature of the Cortex-M3. The OTA example software for the CC2538 demonstrates how to maintain two images in the on-chip flash. These are referred to as “Image A” and “Image B”. To accomplish this, the flash storage is allocated as depicted in the following diagram. Note that a non-OTA memory map is included for comparison:







FLASH_CCA	0x0027FFD4 0x0027FFD6	Reserved
	0x0027FFD7	Bootloader Backdoor
	0x0027FFD8 0x0027FFDB	Image Valid
	0x0027FFDC 0x0027FFDF	Application Entry Point
	0x0027FFE0 0x0027FFFF	Lock Bits

CC2538 Memory Map

Note that the Image Boot Manager (OTA boot loader), for CC2538, resides in the Lock Bits page. Also note the special "Ledger" page. This is used by the Boot Manager to maintain a record of the latest downloaded image and to determine which image in flash to execute at boot time. A ledger entry consists of two CRCs values for a particular image (shadow and actual) and a pointer to the top of the image's Interrupt Vector Table (INTVEC) which contains, among other things, an initial stack pointer and the list of interrupt handlers including the reset handler for that image.

Although the default settings are to evenly divide, in half, the flash not used by the NV and the Ledger, this is not mandator. The Image-A and Image-B code areas can be re-sized as desired by changing the definitions in the board-specific hal_board_cfg.h file as well as making corresponding changes to the two image linker control files):

Components\hal\target\CC2538\hal_board_cfg.h

Projects\zstack\Tools\CC2538DB\ CC2538-OTA-Image-A.icf

Projects\zstack\Tools\CC2538DB\ CC2538-OTA-Image-B.icf

To program the OTA Image Boot Manager, a combined binary image must be built that integrates the OTA Image Boot Manager, the OTA Client Application (eg. Z-Stack Home SampleSwitch or Z-Stack Energy SampleApp) and the optional certificates in the case of an SE OTA Client Application. This process is described in more detail in section 3.2.2 .

2.4.7.2 SmartRF05 with CC2530EM

The Boot loader is loaded into the first page of memory on the CC2530. This page is 2K in length and is located at addresses 0x0000 through 0x0800. The remaining 254K of the flash on the CC2530 is used as the operation memory space. The secondary storage resides on an Off-Chip serial flash.

The OTA Boot code is built with the OTA Boot IAR project for the CC2530 sample applications. The sample OTA Boot code and sample application must be loaded onto the CC2530 in a two-step process.

2.4.7.3 EXP430F5438 with CC2520EM

The boot loader is loaded into address 0xFA00 – 0xFE7B on the EXP430F5438. The 5438 can be configured to use internal or external flash as secondary storage.

2.4.7.3.1 Internal Flash

To configure the EXP430F5438 to use internal flash, the application must be smaller than 128KB. To enable use of the internal flash, set the HAL_OTA_XNV_IS_INT compile time flag.

2.4.7.3.2 External flash

When configuring the EXP430F5438 to use external flash, the application can use the entire 256KB of flash on the MSP430F5438. To enable use of the external flash, set the HAL_OTA_XNV_IS_SPI compile time flag. Note that the EXP430F5438 board does not have an external 256K flash chip so the user will have to provide the external flash chip connection.

2.4.8 OTA Boot loader

The OTA boot loader used by Z-Stack is responsible for:

- Copying memory from the secondary storage space to the primary storage space, if required for the given platform.
- Performing a CRC check of the runnable image to verify the integrity of the image.
- Ensuring interrupts handling is properly setup.
- Booting the application.

In the case of the Cortex-M3 targets, the boot manager simply looks in the ledger page for the run-in-place image to run.

Neither boot application can be upgraded wirelessly.

3. Using the Sample OTA Applications

The sample application supplied for the OTA update feature are:

1. OTA Dongle application: This example application contains the OTA Server.
2. OTA Smart Energy Client: This example application contains the OTA Client for the SE profile.
3. OTA Home Automation Client: This example application contains the OTA Client for the HA profile.

3.1 Required Materials

The following equipment and software are required for use of the Sample OTA Applications:

- Texas Instruments' Z-Stack Home 1.2.1 (or later) or Z-Stack Energy 1.1.0 (or later).
- IAR Embedded Workbench
- Two Evaluation Boards with EM modules

The OTA sample application is available for the following Texas Instrument's ZigBee Platforms.

- CC2538 Cortex-M3 SoC
- *CC2530DB* - SmartRF05 with attached CC2530EM
- *EXP5438* - EXP430F5438 with attached CC2520EM

Unless stated otherwise, the following instructions are the same for all three platforms listed above.

3.2 Building and Downloading Target Applications

This section describes the process of building and downloading the Sample OTA Application. Two boards need to be programmed. One board will run the OTA Dongle application and is called the *Dongle*. One board will run the HA SampleSwitch or In Premise Display – OTA SE SampleApp and is called the *Client*.

To prepare for download:

1) Place two Evaluation Boards on your work bench.
    a. Label one board the *Dongle*.
    b. Label the second board the *Client*.
2) Verify the boards contain Evaluation Modules and the modules have antenna attached.
    a. For the *CC2538* platform, verify the SmartRF06 board has a CC2538EM module attached.
    b. For the *CC2530DB* platform, verify the SmartRF05 board has a CC2530EM module attached.
    c. For the *EXP5438* platform, verify the EXP430F5438 board has a CC2520EM modules attached.

3.2.1 OTA Boot

The OTA Boot application is only required for the OTA Clients.

For the EXP5438 projects, the OTA boot loader is integrated into the IPD – OTA IAR project.

For CC2530 and CC2538 platforms, the OTA Boot application IAR project is located in one of the following relative directories:

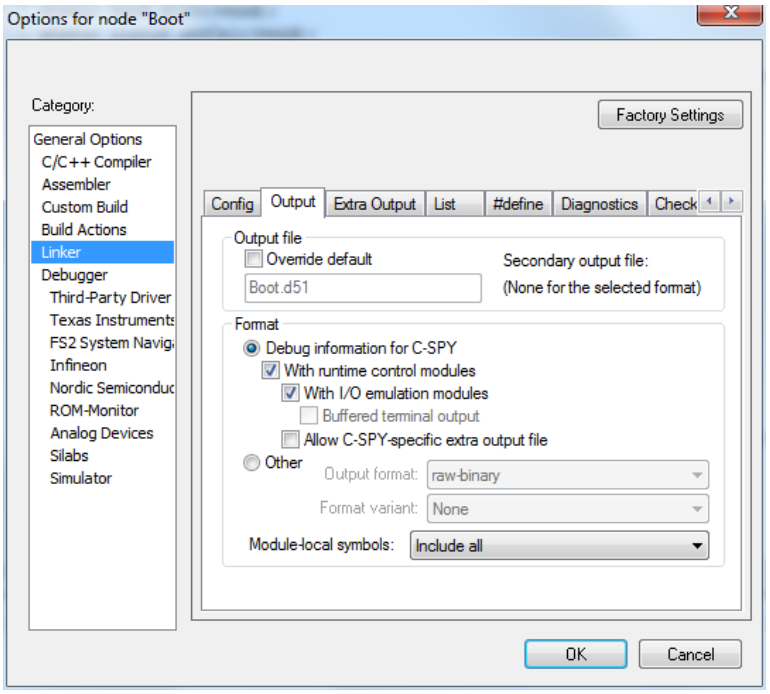
..\Projects\Z-Stack\OTA\Boot\CC2530DB

..\Projects\Z-Stack\OTA\Boot\CC2538

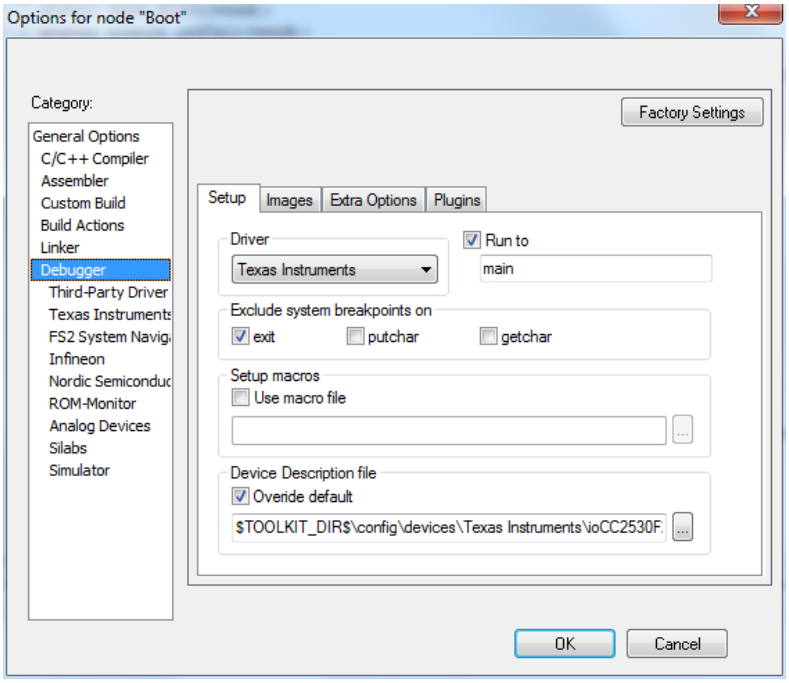
3.2.1.1 OTA Boot for CC2530

Follow these steps to build and download the OTA Boot application for CC2530:

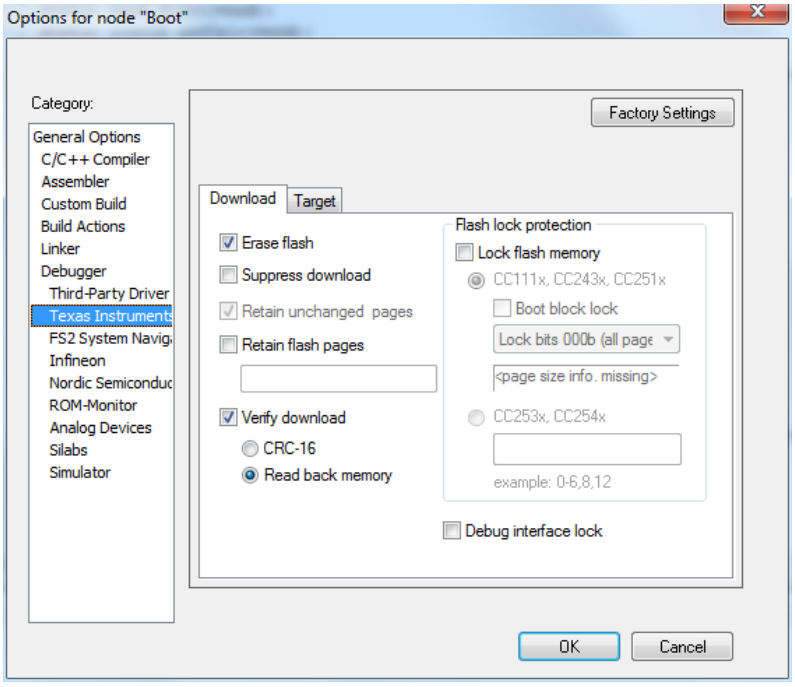
1) Open Boot.eww with IAR Embedded Workbench
2) Verify the project's options are configured properly.
   a. Click *Project>Options* from the menu.
   b. Click the Linker category. Verify the Output contains debug information for C-Spy:



c. Click the Debugger category. Verify the Driver is set to Texas Instruments:



d. Click the Texas Instruments category. Verify the flash WILL be erased, and the download will be verified:



3) Build the Project.
   a. Verify the active project in the *Workspace* toolbar is *OTA Boot Loader*
   b. Click *Project>Rebuild All* from the menu.
   c. Wait for the build to complete.
4) Connect the SmartRF05 board labeled the C*lient* to the PC with a USB cable.

5) Download the image to the SmartRF05 board:
    a. Click *Project>Debug.*
    b. If more than one SmartRF05 board is connected to the PC, IAR will ask which board to program. Choose the board designated to be the C*lient*.
    c. Wait for the download to complete.
6) Terminate the debug session.
    a. Click *Debug>Stop Debugging* from the menu.

3.2.1.2 Image Boot Manager for CC2538

For the CC2538 platform, we build the Image Boot Manager binary file but do NOT download the image to the target. This will be needed when we create a combined image that incorporates this binary file. This is explained further in the specific sections for building the sample applications for CC2538, below.

Follow these steps to build the OTA Boot application for CC2538:

1) Open Boot.eww with IAR Embedded Workbench
2) Build the Project.
    a. Click *Project>Rebuild All* from the menu.
    b. Wait for the build to complete.

3.2.2 Sample OTA Client Applications for CC2538

For the CC2538, the flash space is divided in half so that one image can be run while another is downloaded. These two halves are referred to as Image A and Image B. To support this design, we need to build two different types of images. An "initial" image that combines the Image Boot Manager with an Image A – based ZigBee application and optionally any certificate data, applicable for security enabled SE sample OTA client applications. This combined "initial" image is programmed via the IAR debugger.

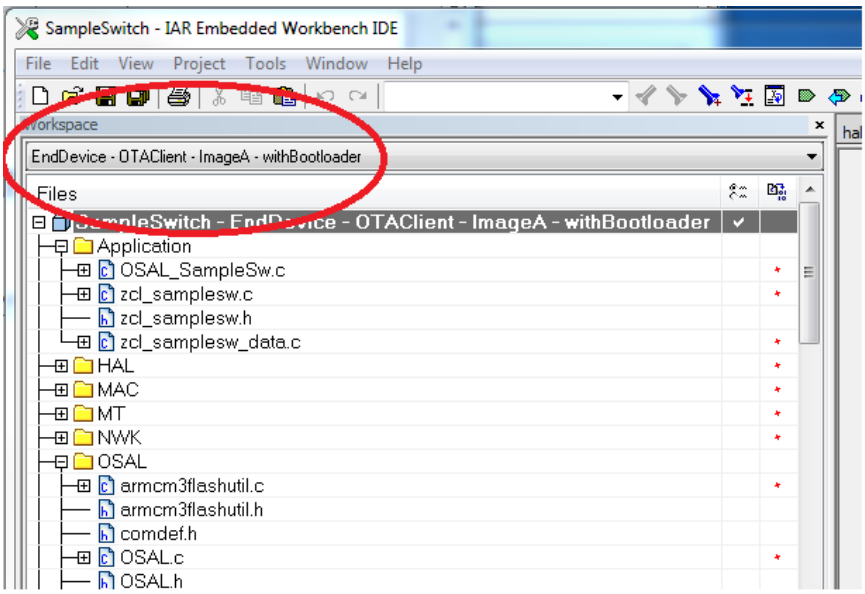
The second type of image we need to build is either an "A" or "B" version of our OTA client-enabled sample application. This image A or B will be built with a specific linker control file designed to place the image in either the upper (A) or lower (B) half of the flash memory as depicted in the memory diagram above. These images are specifically intended to be transferred via OTA

3.2.2.1 Building the Initial Sample Applications for CC2538

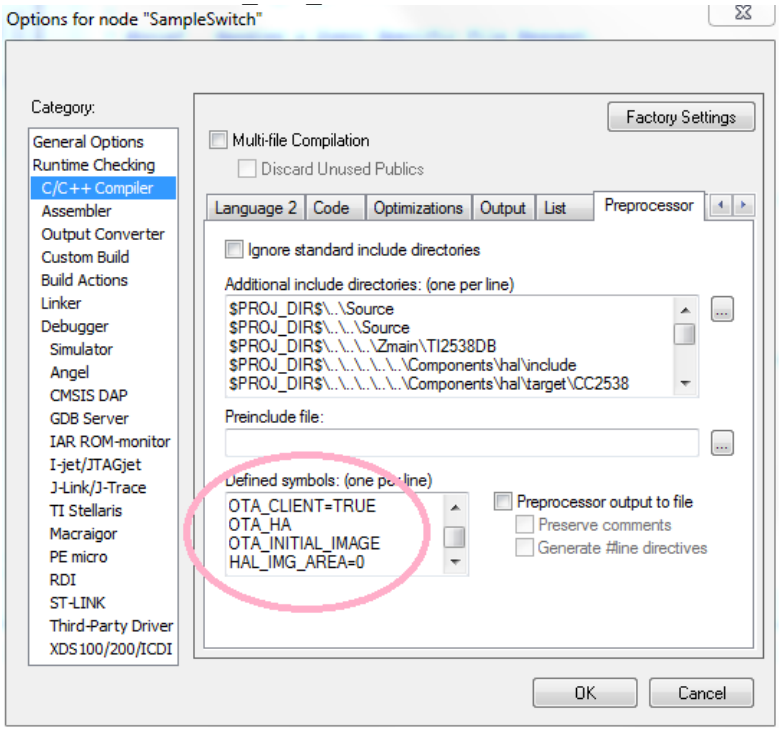
Step 1A: To prepare to build the initial OTA capable HA SampleSwitch App:

1) Open SampleSwitch.eww with IAR Embedded Workbench
2) Select the "EndDevice – OTAClient – ImageA - withBootloader" configuration
    a. From the Workspace dialog, click the configuration dropdown button.
    b. Verify the "EndDevice – OTAClient – ImageA - withBootloader" configuration is selected.

    Note: You can also select the "Router – OTAClient – ImageA – withBootloader" build configuration if your desire is to build a ZigBee Router.



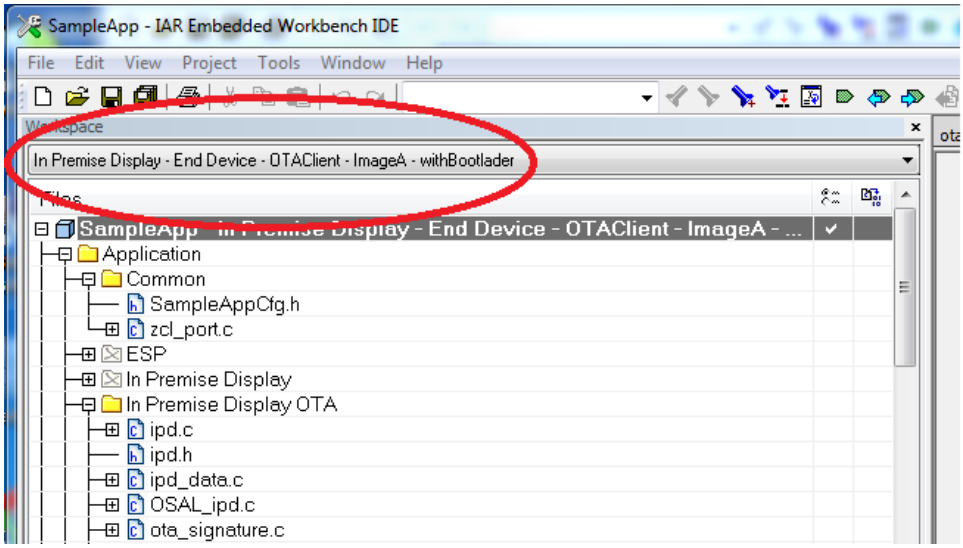
3) Verify the project's definitions are configured properly for HA.
   a) Click *Project>Options* from the menu.
   b) Click on C/C++ Compiler category
   c) Verify that the following symbols are defined in *Preprocessor* tab:
      - `OTA_CLIENT=TRUE`
      - `OTA_HA`
      - `OTA_INITIAL_IMAGE`
      - `HAL_IMG_AREA=0`



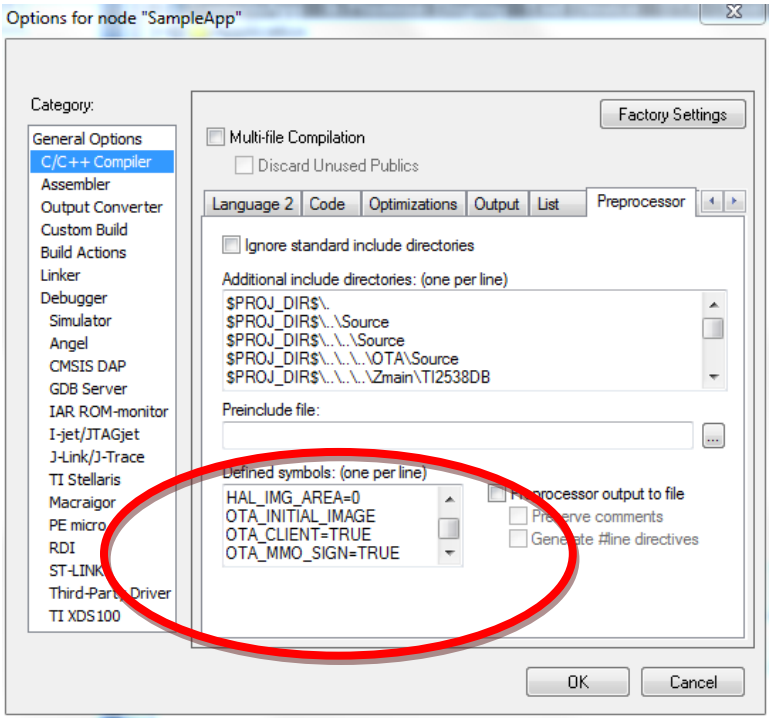
4) Go to Step 2

Step 1B: To prepare to build the initial SE SampleApp:

1) Open SampleApp.eww with IAR Embedded Workbench
2) Select the "In Premise Display – End Device – OTAClient - ImageA - withBootloader" configuration
    a. From the Workspace dialog, click the configuration dropdown button.
    b. Verify the "In Premise Display – End Device – OTAClient - ImageA - withBootloader" configuration is selected



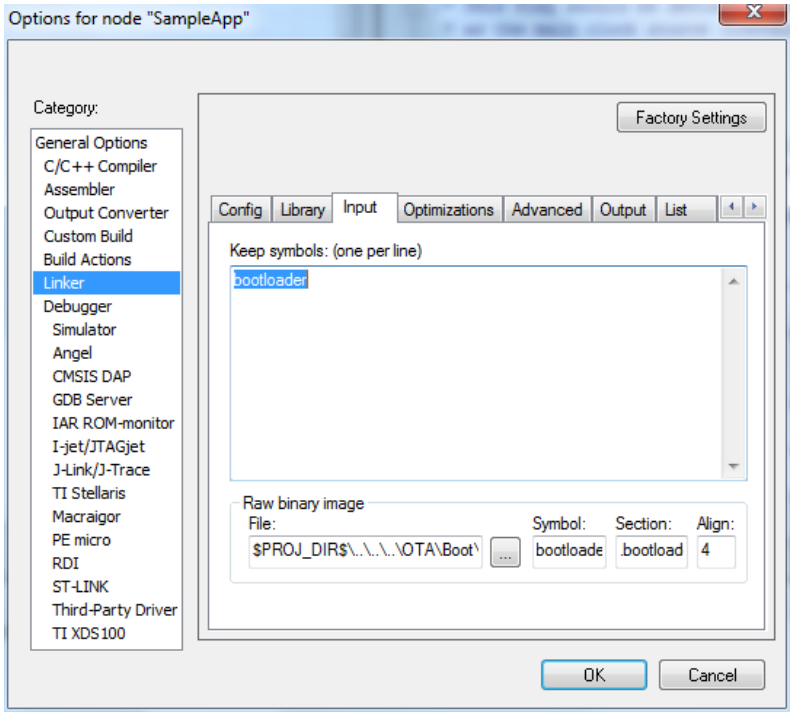
3) Verify the project's definitions are configured properly for SE.
    a. Click *Project>Options* from the menu.
    b. Click on C/C++ Compiler category
    c. Verify that the following symbols are defined in *Preprocessor* tab:
        - `OTA_CLIENT=TRUE`
        - `OTA_MMO_SIGN=TRUE`
        - `HAL_IMG_AREA=0`
        - `OTA_INITIAL_IMAGE`



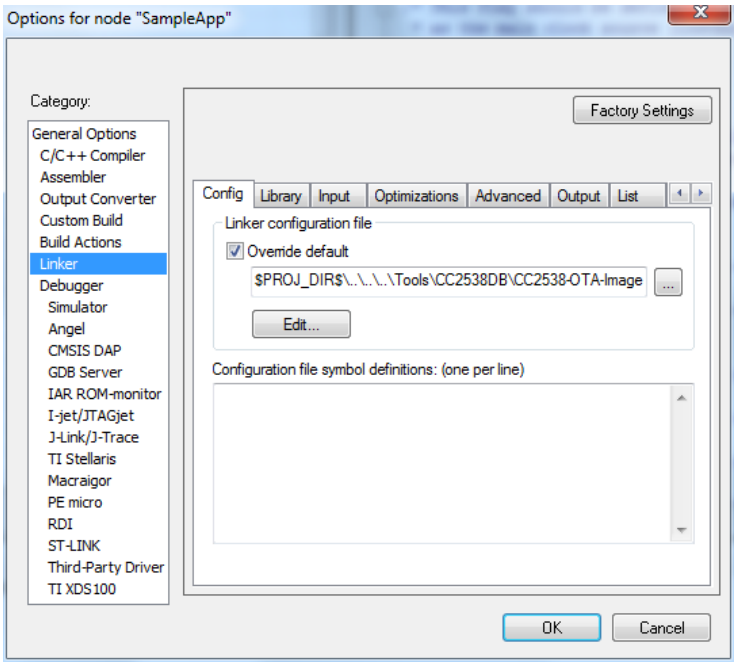
4) Go to Step 2

Step 2: Configure the linker options to incorporate the Boot Image Manger built above.

1) Click *Project>Options* from the menu.
2) Click on Linker Compiler category
3) Select the "Input" tab and verify the following:
   a) In the "Keep symbols" panel the following symbol should be defined:
      - `bootloader`
   b) In the "Raw binary image" section, verify the following:
      - File: `$PROJ_DIR$\..\..\..\OTA\Boot\CC2538\Image Boot Manager\Exe\Boot.bin`
      - Symbol: `bootloader`
      - Section: `.bootloader`
      - Align: `4`



4) Select the "Config" tab and verify the following in the Linker configuration file section:
   a) The Override default check box should be checked.
   b) The specified linker configuration file should be:
      - `$PROJ_DIR$\..\..\..\Tools\CC2538DB\CC2538-OTA-Image-A.icf`



Step 3: To build and download the *initial* image A:

1) Click *Project>Clean*.
2) Click *Project>Rebuild All*
3) Click *Project>Download>Erase Memory*
   This is step is important to ensure any previous programming information in flash is removed.

4) Click *Project>Download and Debug*
5) When the download completes, you can disconnect the debugger.

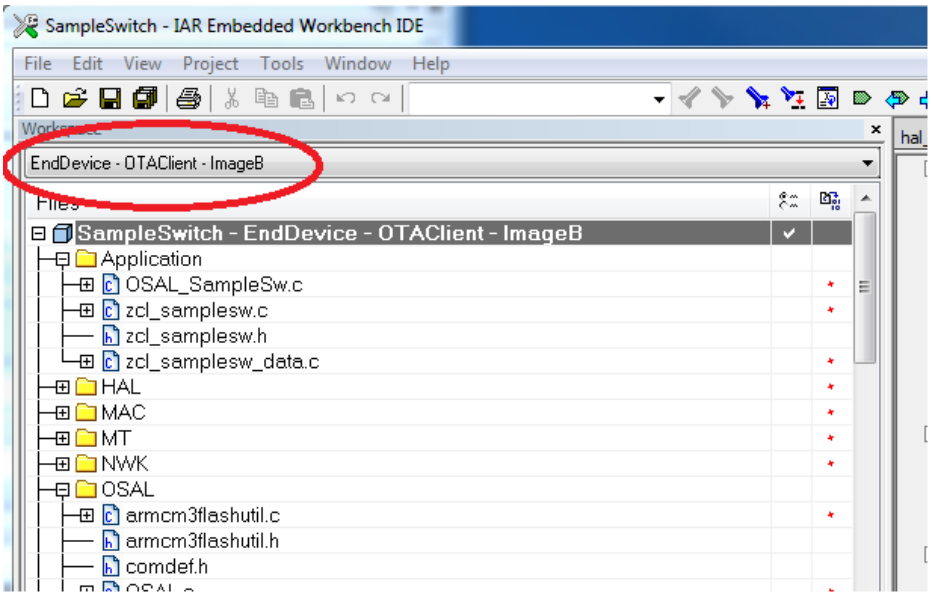
Your platform is now programmed with an initial Image A including the Boot Image Manager and optional certificate data.

3.2.2.2 Building Image A/B Sample Applications for CC2538

With the initial Image A OTA client with Bootloader now programmed into the target hardware, your first OTA transfer image must be built as an Image B. When the transfer is complete, the target will reset and boot the new Image B. The next OTA transfer must be an Image A. This ping-ponging between Image A and B is necessary to take advantage of the run-in-place feature of the CC2538.

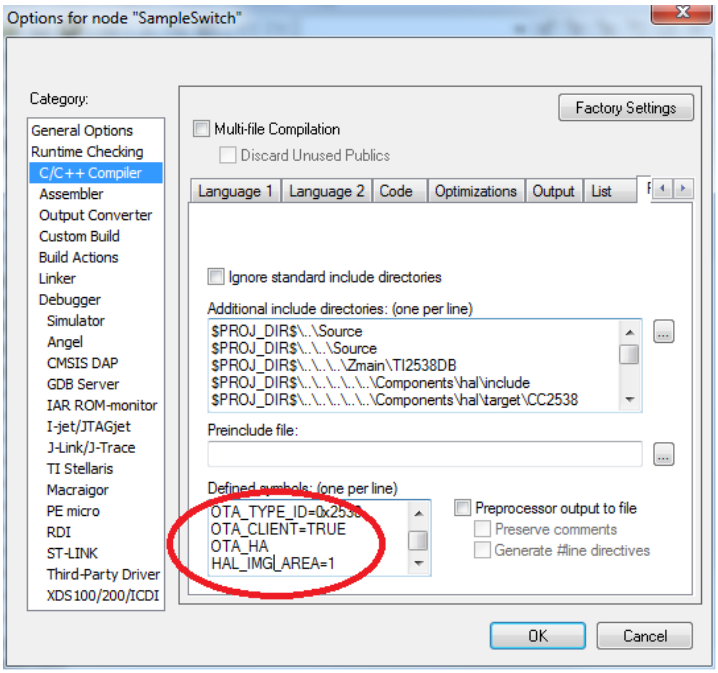
3.2.2.2.1 Building the HA SampleSwitch as Image B:

1) Open SampleSwitch.eww with IAR Embedded Workbench
2) Select the "EndDevice – OTAClient – ImageB" configuration
    a. From the Workspace dialog, click the configuration dropdown button.
    b. Verify the "EndDevice – OTAClient – ImageB" configuration is selected

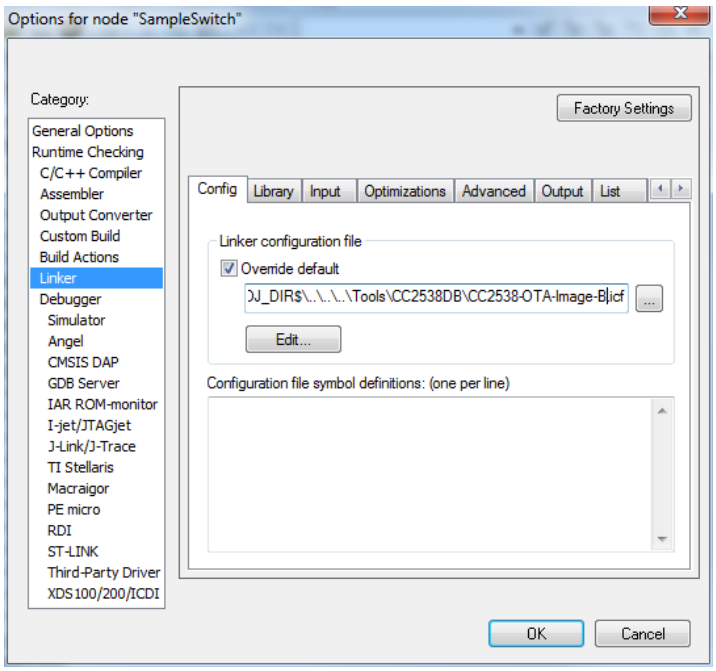


3) Verify the project's definitions are configured properly for Image B.
    1) Click *Project>Options* from the menu.
    2) Click on C/C++ Compiler category
    3) Verify that the following symbols are defined in *Preprocessor* tab:
        - `OTA_CLIENT=TRUE`
        - `OTA_HA`
        - `HAL_IMG_AREA=1`

    Important: HAL_IMG_AREA=1 refers to Image B.
    Very Important: Ensure that *OTA_INITIAL_IMAGE* is **NOT** defined.



4) Verify the project's linker configuration is set properly for Image B.
   a. Click *Project>Options* from the menu.
   b. Click on *Linker* category
   c. Verify that the following linker configuration file is specified in the *Config* tab:
      - Override default is checked.
      - `$PROJ_DIR$\..\..\..\Tools\CC2538DB\CC2538-OTA-Image-B.icf`



5) Verify the project's post-build action is set properly to post process the binary file to convert/prepare it for OTA transfer.
   a. Click *Project>Options* from the menu.

b. Click on *Build Actions* category
c. Verify that the following in the Post-build command line:

```
"$PROJ_DIR$\..\..\..\..\tools\OTA\OtaConverter\Release\OtaConverter.exe"
"$PROJ_DIR$\EndDevice - OTAClient - ImageB\Exe\EndDevice-OTAClient-
ImageB.bin" -o"$PROJ_DIR$\EndDevice - OTAClient - ImageB\Exe" -t0x2538 -
m0xBEBE -vBBBB0000  -pCC2538
```

Note: The values specified for the "-m" and "-v" options are just examples.

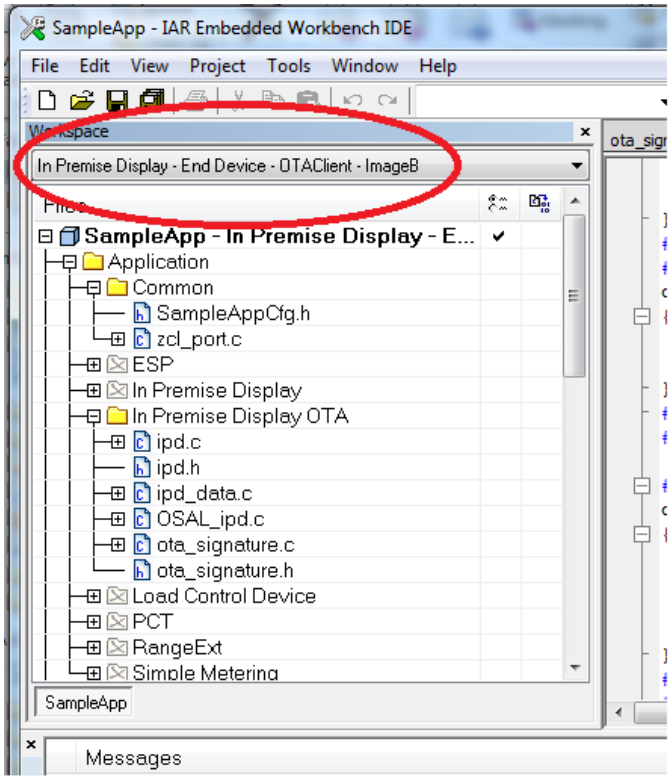
6) Build the Image B
   a. Click *Project>Clean*.
   b. Click *Project>Rebuild All*

Note: The converted image can be found here:

```
Projects\zstack\HomeAutomation\SampleSwitch\CC2538\EndDeviceSrc
- OTAClient - ImageB\Exe\BEBE-2538-BBBB0000.zigbee
```

3.2.2.2.2 Building the SE SampleApp as Image B:

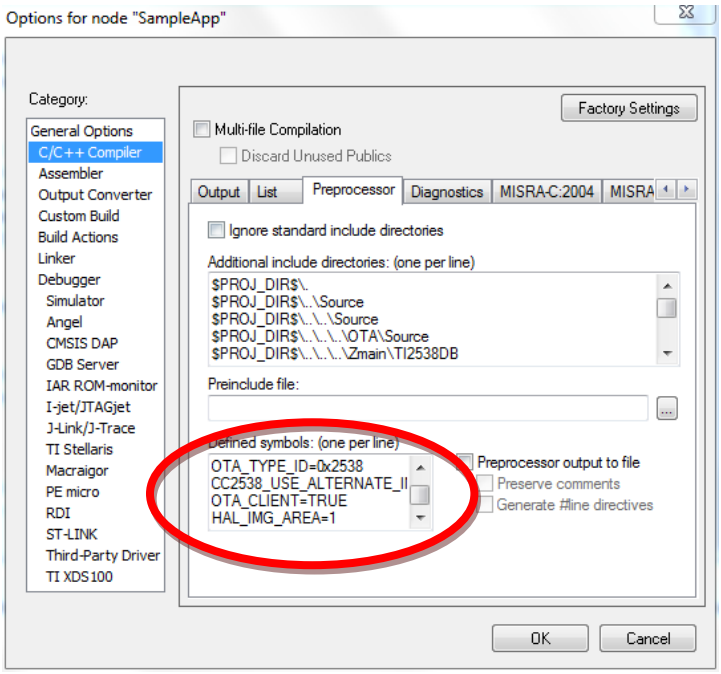
1) Open SampleApp.eww with IAR Embedded Workbench
2) Select the "In Premise Display – End Device – OTA Client - ImageB" configuration
   a. From the Workspace dialog, click the configuration dropdown button.
   b. Verify the "In Premise Display – End Device – OTA Client - ImageB" configuration is selected



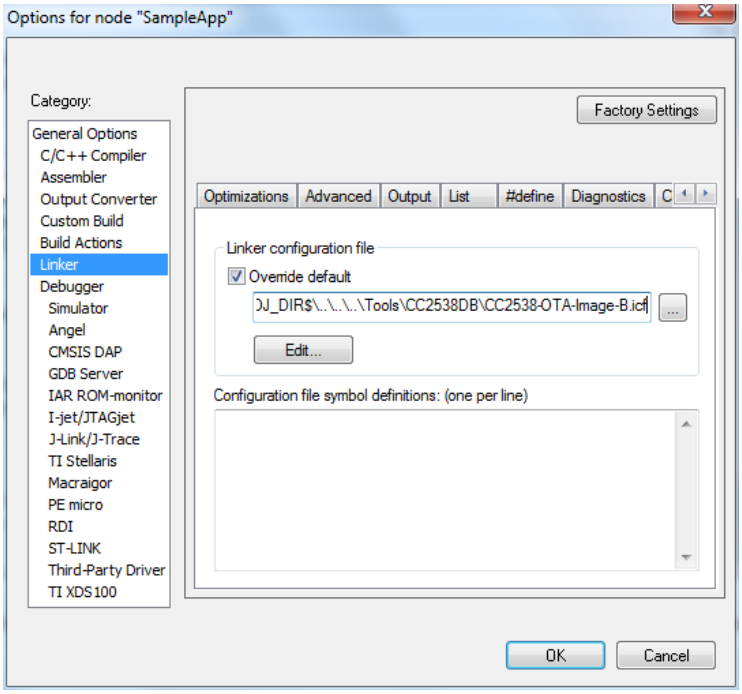
3) Verify the project's definitions are configured properly for Image B.
   1) Click *Project>Options* from the menu.
   2) Click on C/C++ Compiler category
   3) Verify that the following symbols are defined in *Preprocessor* tab:

- OTA_CLIENT=TRUE
- HAL_IMG_AREA=1

Important: HAL_IMG_AREA=1 refers to Image B.
Very Important: Ensure that *OTA_INITIAL_IMAGE* is **NOT** defined.



4) Verify the project's linker configuration is set properly for Image B.
   a. Click *Project>Options* from the menu.
   b. Click on *Linker* category
   c. Verify that the following linker configuration file is specified in the *Config* tab:
      - Override Default is checked.
      - `$PROJ_DIR$\..\..\..\Tools\CC2538DB\CC2538-OTA-Image-B.icf`



5) Verify the project's post-build action is set properly to post process the binary file to convert/prepare it for OTA transfer.
   a. Click *Project>Options* from the menu.
   b. Click on *Build Actions* category
   c. Verify that the following in the Post-build command line:

```
"$PROJ_DIR$\..\..\..\..\tools\OTA\OtaConverter\Release\OtaConverter.exe"
"$PROJ_DIR$\In Premise Display - End Device - OTAClient -
ImageB\Exe\SampleApp.bin" -o"$PROJ_DIR$\In Premise Display - End Device -
OTAClient - ImageB\Exe" -t0x2538 -m0xBEBE -pCC2538 -vBBBB0000 -
s"$PROJ_DIR$\..\Source\OTA Cert\cert.txt"
```

   Note: The values specified for the "-m" and "-v" options are just examples.

   Note: For the conversion of an SE OTA image, the converted file is signed. The example certificate data, used by the conversion tool to include a signature with the converted file, is specified by the "-s" option.

6) Build the Image B
   a. Click *Project>Clean*.
   b. Click *Project>Rebuild All*

   Note: The converted image can be found here:

```
Projects\zstack\SE\SampleApp\CC2538\In Premise Display - End
Device - OTA Client - ImageB\Exe\BEBE-2538-BBBB0000.zigbee
```

3.2.2.2.3 Building the HA and SE applications as Image A

To build these sample applications as Image A rather than Image B, you'll need to make two changes:

1) Change the linker control file to: `$PROJ_DIR$\..\..\..\Tools\CC2538DB\CC2538-OTA-Image-A.icf`

2) Set the compiler processor HAL_IMG_AREA=0

3) Optionally, you can specify a different version option "-v" to the post-build command line with something that identifies the image file as an image A file. Eg. "–vAAAA0000"

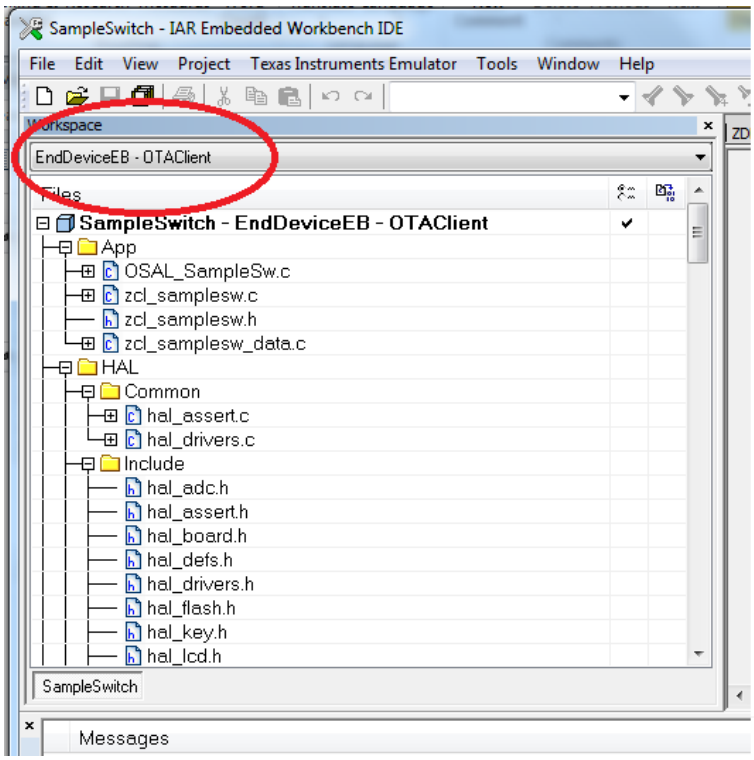
These settings are available as pre-sets in the corresponding build configuration ending in "… OTAClient – ImageA" for both the HA SampleSwitch workspace and the SE SampleApp workspace.

3.2.3 Sample OTA Client Applications for Other Platforms

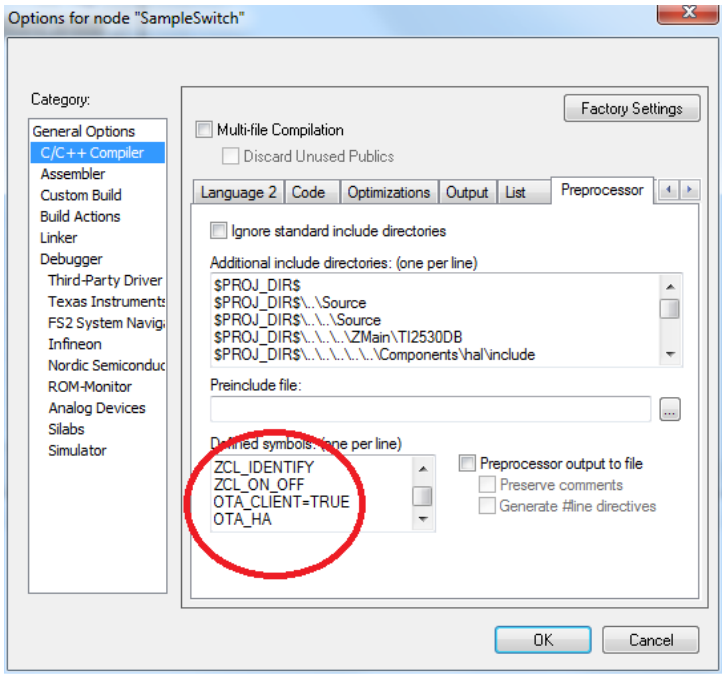
3.2.3.1 HA SampleSwitch OTA Client Sample Application for CC2530

To build and download the HA OTA App:

1) Open SampleSwitch.eww with IAR Embedded Workbench
2) Select the "EndDeviceEB – OTAClient" configuration
    a. From the Workspace dialog, click the configuration dropdown button.
    b. Verify the EndDeviceEB configuration is selected



3) Verify the project's definitions are configured properly for HA.
    a. Click *Project>Options* from the menu.
    b. Click on C/C++ Compiler category
    c. Verify that the following symbols are defined:
        - OTA_CLIENT=TRUE
        - OTA_HA

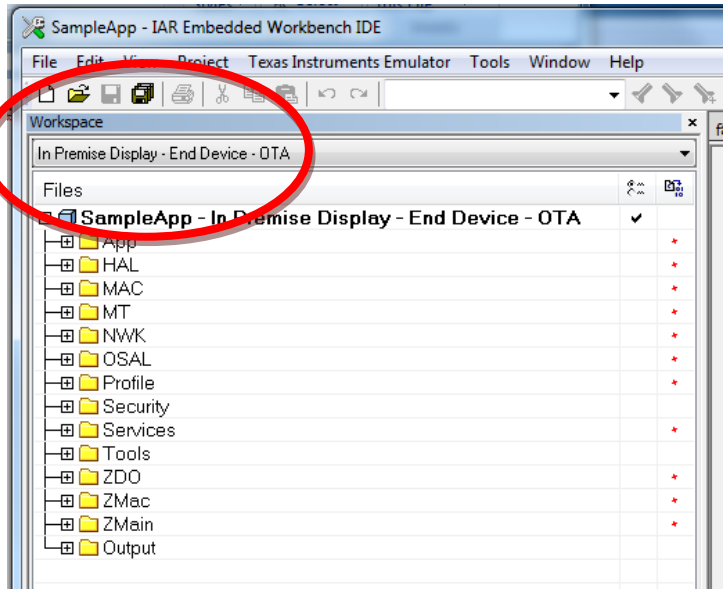


4) From here on, follow instructions starting from 4) of the In Premise Display – OTA SE SampleApp instructions below.

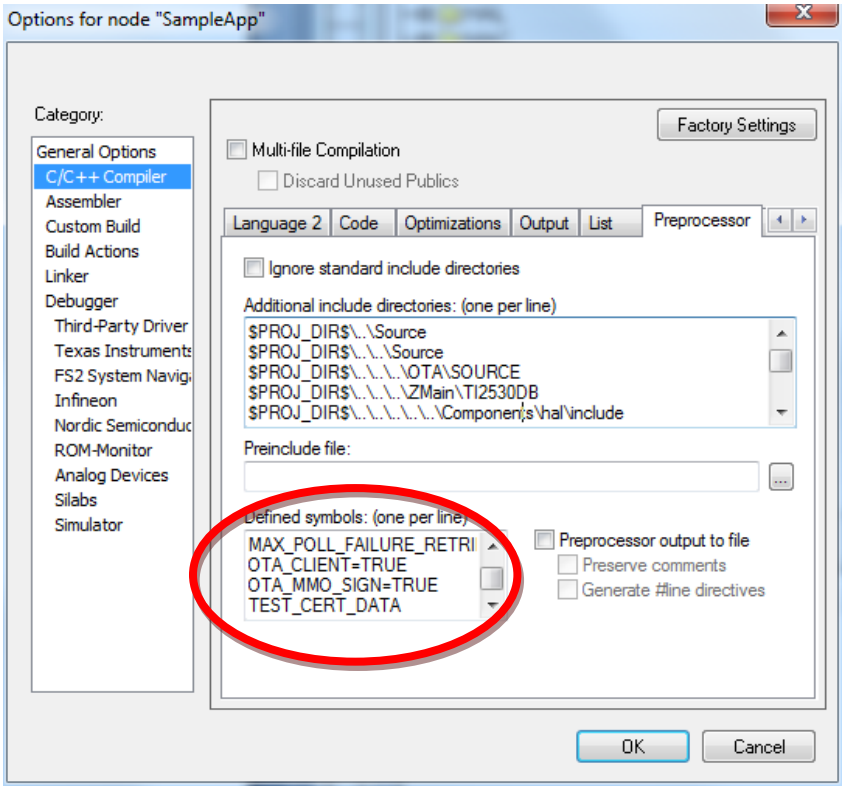
3.2.3.2 In Premise Display – OTA SE SampleApp

To build and download the IPD OTA App:

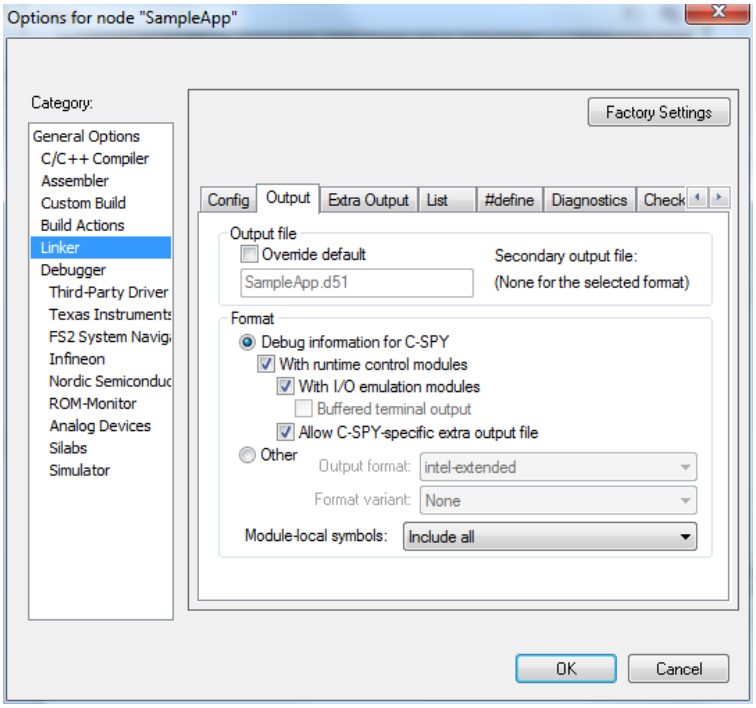
1) Open SampleApp.eww with IAR Embedded Workbench
2) Select the In Premise Display – End Device – OTA configuration
    a. From the Workspace dialog, click the configuration dropdown button.
    b. Verify the In Premise Display – End Device – OTA configuration is selected



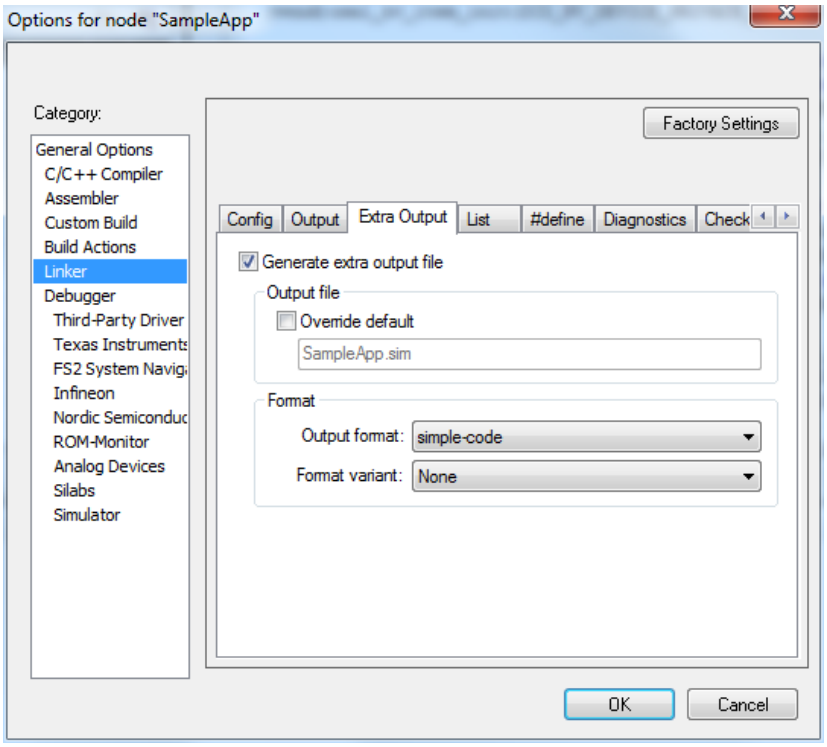
3) Verify the project's definitions are configured properly for SE.
   a. Click *Project>Options* from the menu.
   b. Click on C/C++ Compiler category
   c. Verify that the following symbols are defined:
      - OTA_CLIENT=TRUE
      - OTA_MMO_SIGN=TRUE
   d. Verify that the following symbols are **NOT** defined
      - OTA_HA



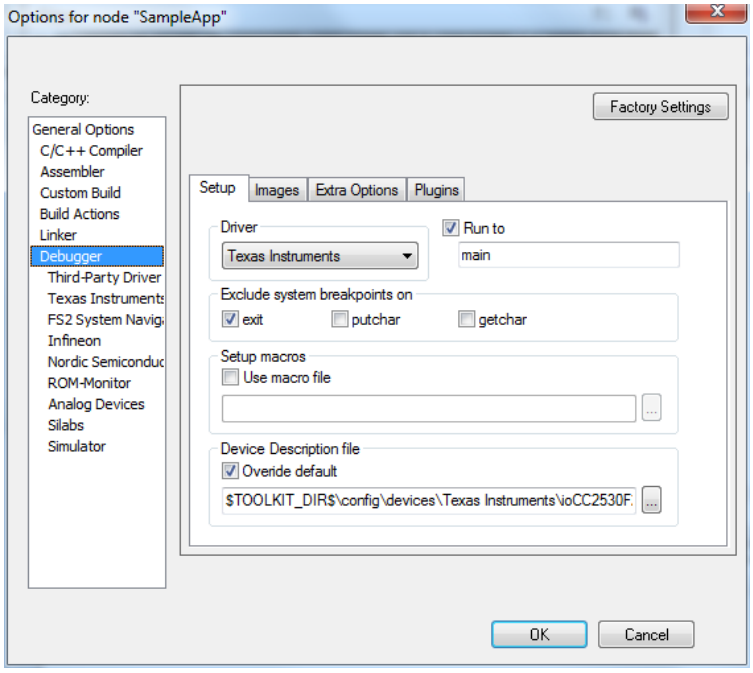
4) Verify the project's options are configured properly.
   a. Click *Project>Options* from the menu.
   b. Click the Linker category. Verify the Output contains debug information for C-Spy, and that the "Allow C-SPY-specific extra output file" checkbox is selected:



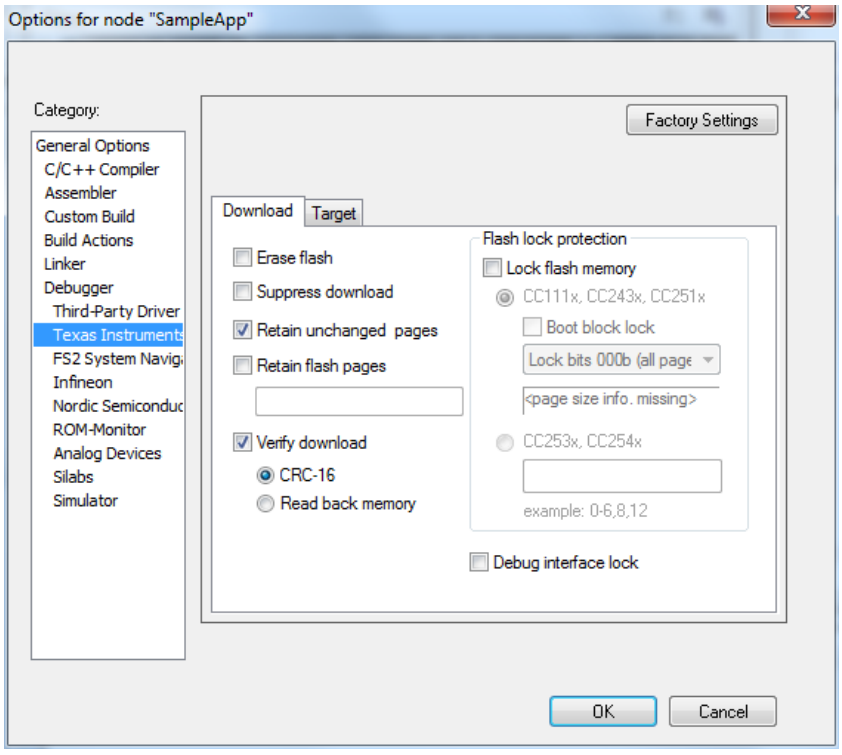
c. Click the Extra Output tab.
d. Verify the Output format is simple-code:



e. Click the Debugger category. For the CC2530DB platform, verify the Driver is set to Texas Instruments:

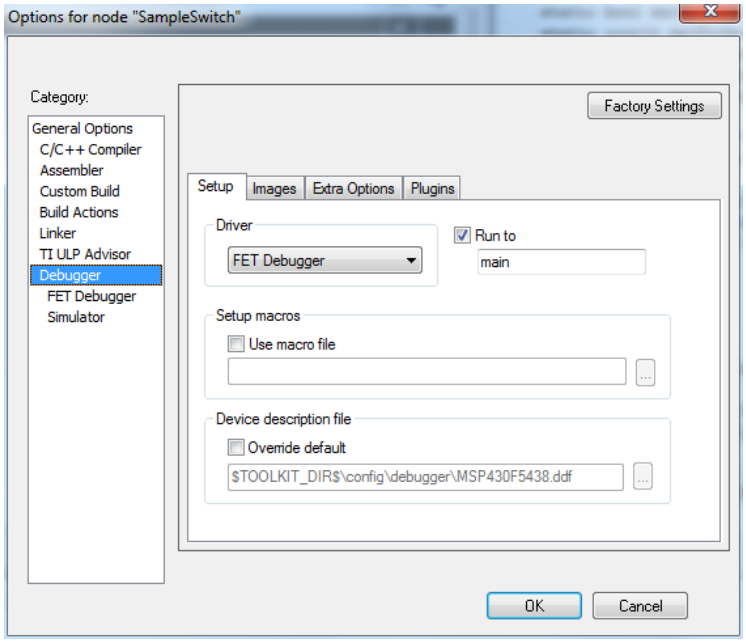


f. For the CC2530DB platform, click the Texas Instruments category. Make sure that the "Erase flash" box is unchecked and download will be verified:



Note: By selecting the "Retain Unchanged pages" setting, the OTA Boot application download only used the first page of memory on the CC2530. This page was not used by the IPD – OTA Client application. Subsequently both the IPD – OTA Client and OTA Boot are present on the C*lient* device.

g. For the EXP5438 platform, verify the Driver is set to FET Debugger:



h. Verify the project's post-build action is set properly to post process the binary file to convert/prepare it for OTA transfer.
   i. Click *Project>Options* from the menu.
   ii. Click on *Build Actions* category
   iii. Verify that the following in the Post-build command line:

For HA SampleSwitch on CC2530:

```
"$PROJ_DIR$\..\..\..\..\tools\OTA\OtaConverter\Release\OtaConverter.exe" "$PROJ_DIR$\EndDeviceEB - OTAClient\Exe\EndDeviceEB-OTAClient.sim" -o"$PROJ_DIR$\EndDeviceEB - OTAClient\Exe" -t0x1234 -m0x5678 -v0000ABCD -pCC2530DB
```

Note: The converted image can be found here:
`Projects\zstack\HomeAutomation\SampleSwitchOta\CC2530DB\EndDeviceEB - OTAClient\Exe`

For HA SampleSwitch on MSP430:

```
"$PROJ_DIR$\..\..\..\..\tools\OTA\OtaConverter\Release\OtaConverter.exe" "$PROJ_DIR$\EndDevice - OTAClient\Exe\EndDevice-OTAClient.sim" -o"$PROJ_DIR$\EndDevice - OTAClient\Exe" -t0x1234 -m0x5678 -pEXP5438
```

Note: The converted image can be found here:
`Projects\zstack\HomeAutomation\SampleSwitch\EXP5438\EndDevice - OTAClient\Exe`

For SE SampleApp on CC2530:

```
"$PROJ_DIR$\..\..\..\..\tools\OTA\OtaConverter\Release\OtaConverter.exe" "$PROJ_DIR$\In Premise Display - End Device - OTA\Exe\SampleApp.sim" -o"$PROJ_DIR$\In Premise Display - End Device
```

```
- OTA\Exe" -t0x1234 -m0x5678 -v0x6969 -pCC2530DB -
s"$PROJ_DIR$\..\Source\OTA Cert\cert.txt"
```

Note: The converted image can be found here:
`Projects\zstack\SE\SampleApp\CC2530DB\In Premise Display - End Device - OTA\Exe`

For SE SampleApp on MSP430:

```
"$PROJ_DIR$\..\..\..\..\tools\OTA\OtaConverter\Release\OtaConverter.e
xe" "$PROJ_DIR$\In Premise Display - End Device -
OTA\Exe\SampleApp.sim" -o"$PROJ_DIR$\In Premise Display - End Device
- OTA\Exe" -t0x1234 -m0x5678 -v0xabcd1234 -pEXP5438 -
s"$PROJ_DIR$\..\Source\OTA Cert\cert.txt"
```

Note: The converted image can be found here:
`Projects\zstack\SE\SampleApp\EXP5438\In Premise Display - End Device - OTA\Exe`

Note: The values specified for the "-m" and "-v" options are just examples.

Note: For the conversion of an SE OTA image, the converted file is signed. The example certificate data, used by the conversion tool to include a signature with the converted file, is specified by the "-s" option.

5) Build the Project.
   a. Click *Project>Rebuild All* from the menu.
   b. Wait for the build to complete.
6) Connect the Evaluation Board labeled the C*lient* to the PC
   a. For the CC2530DB, connect the SmartRF05 to the PS with a USB cable.
   b. For the EXP5438, connect the EXP430F5438 to an MSP-FET430UIF programmer, and connect the MSP-FET430UIF to the PC with a USB cable.
7) Download the image to the evaluation board:
   a. Click *Project>Debug*.
   b. If more than one board is connected to the PC, IAR will ask which board to program. Choose the board designated to be the C*lient*.
   c. Wait for the download to complete.
8) Terminate the debug session.
   a. Click *Debug>Stop Debugging* from the menu.

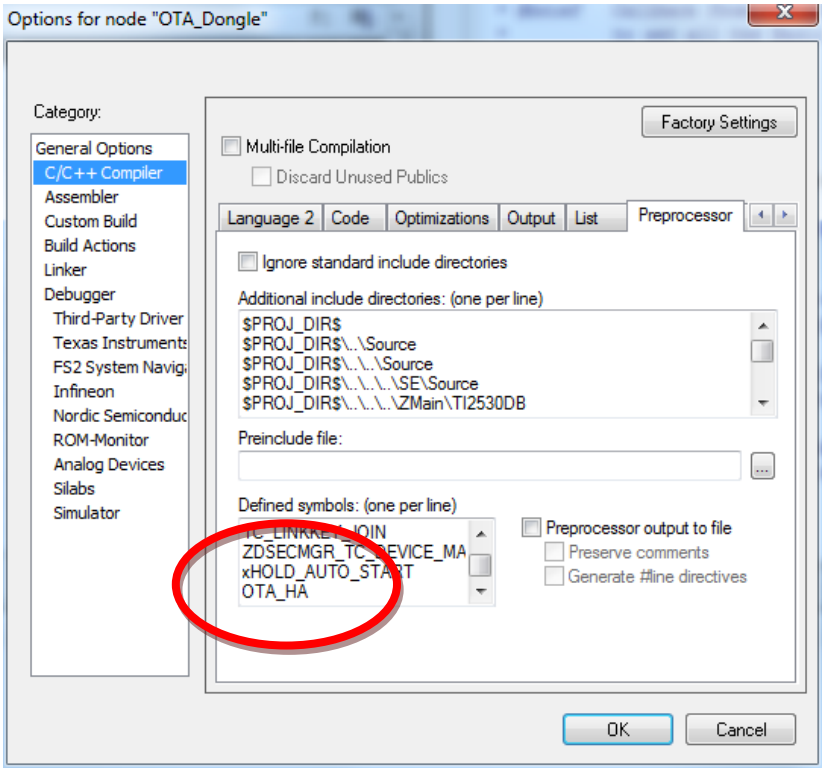
3.2.4 Dongle

The dongle runs OTA Dongle, application. The OTA Dongle application is located in:

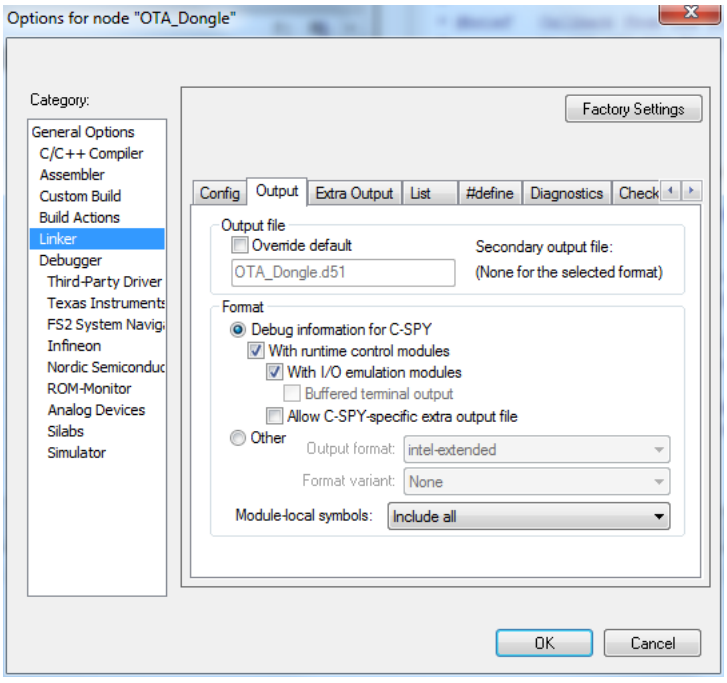
\ Projects\zstack\OTA\Dongle\<platform>\

To build and download the OTA Dongle application:

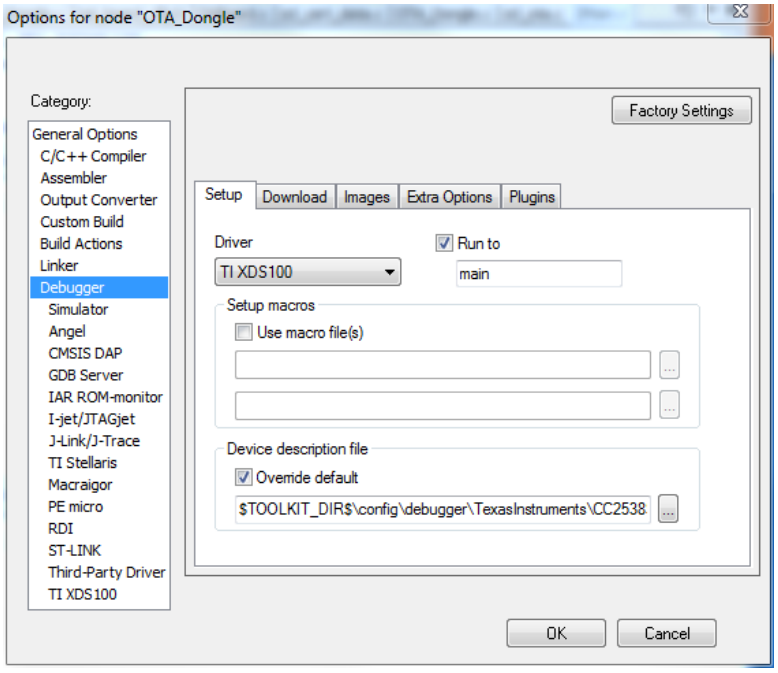
1) Open OTA_Dongle.eww with IAR Embedded Workbench
2) Verify the project's definitions are configured properly for the profile being used.
   c. Click *Project>Options* from the menu.
   d. Click on C/C++ Compiler category
   e. For HA profile verify that the following symbols are defined:
      - OTA_HA
   f. For SE profile verify that the following symbols are NOT defined
      - OTA_HA



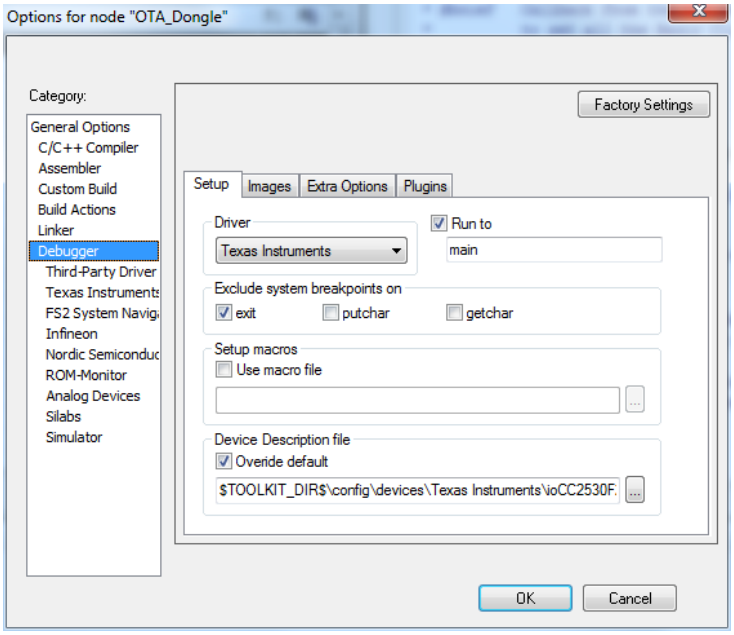
3) For CC2530, Verify the project's options are configured properly.
   a. Click *Project>Options* from the menu.
   b. Click the Linker category. Verify the Output contains debug information for C-Spy:



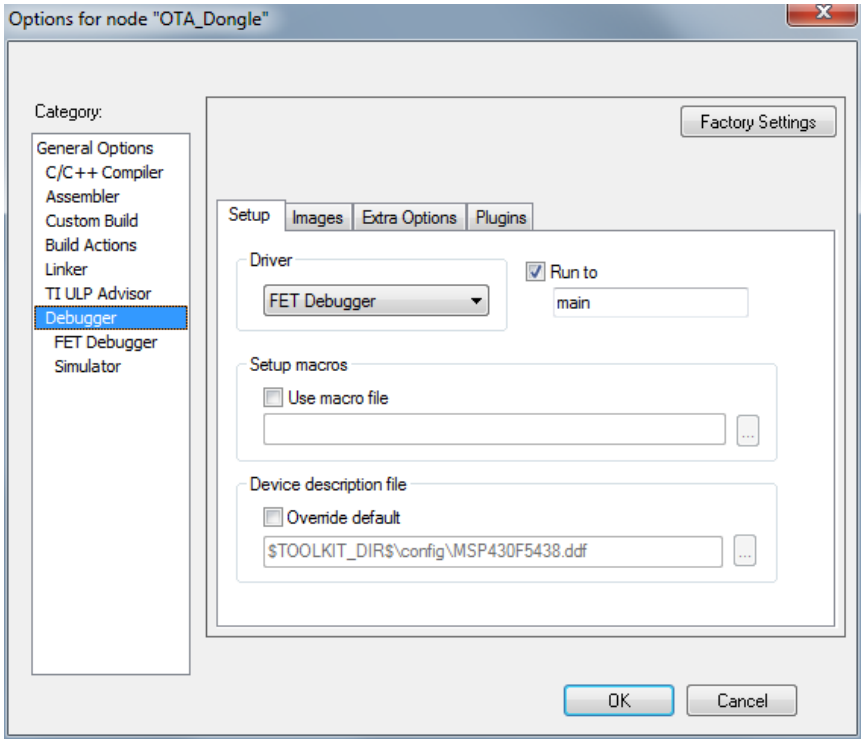
   b. Click the Debugger category.
      a. For the CC2538 platform, verify the Driver is set to TI XDS100:



b. For the CC2530DB platform, verify the Driver is set to Texas Instruments:



c. For the EXP5430 platform, verify the Driver is set to FET Debugger:



4) Build the Project.
   a. Click *Project>Rebuild All* from the menu.
   b. Wait for the build to complete.
5) Connect the evaluation board labeled the *Dongle* to the PC.
   a. For the CC2538, connect the SmartRF06 to the PS with a USB cable.
   b. For the CC2530DB, connect the SmartRF05 to the PS with a USB cable.
   c. For the EXP5438, connect the EXP430F5438 to an MSP-FET430UIF programmer, and connect the MSP-FET430UIF to the PC with a USB cable.
6) Download the image to the target board:
   a. Verify the *Coordinator* configuration is selected from the *workspace*.
   b. Click *Project>Debug*.
   c. If more than one SmartRF05 or SmartRF06 board is connected to the PC, IAR will ask which board to program. Choose the board designated to be the *Dongle*.
   d. Wait for the download to complete.
7) Terminate the debug session.
   a. Click *Debug>Stop Debugging* from the menu.
8) Connect the *Dongle's* UART to the PC
   a. For the CC2538 platform, connect the SmartRF06's mini USB connector to the PC.
   b. For the CC2530DB platforms, connect the SmartRF05 board's RS-232 connector to a COM port on the PC.
   c. For the EXP5438 platform, connect the EXP430F5438's micro USB connector to the PC.

3.3 Performing an Image Update

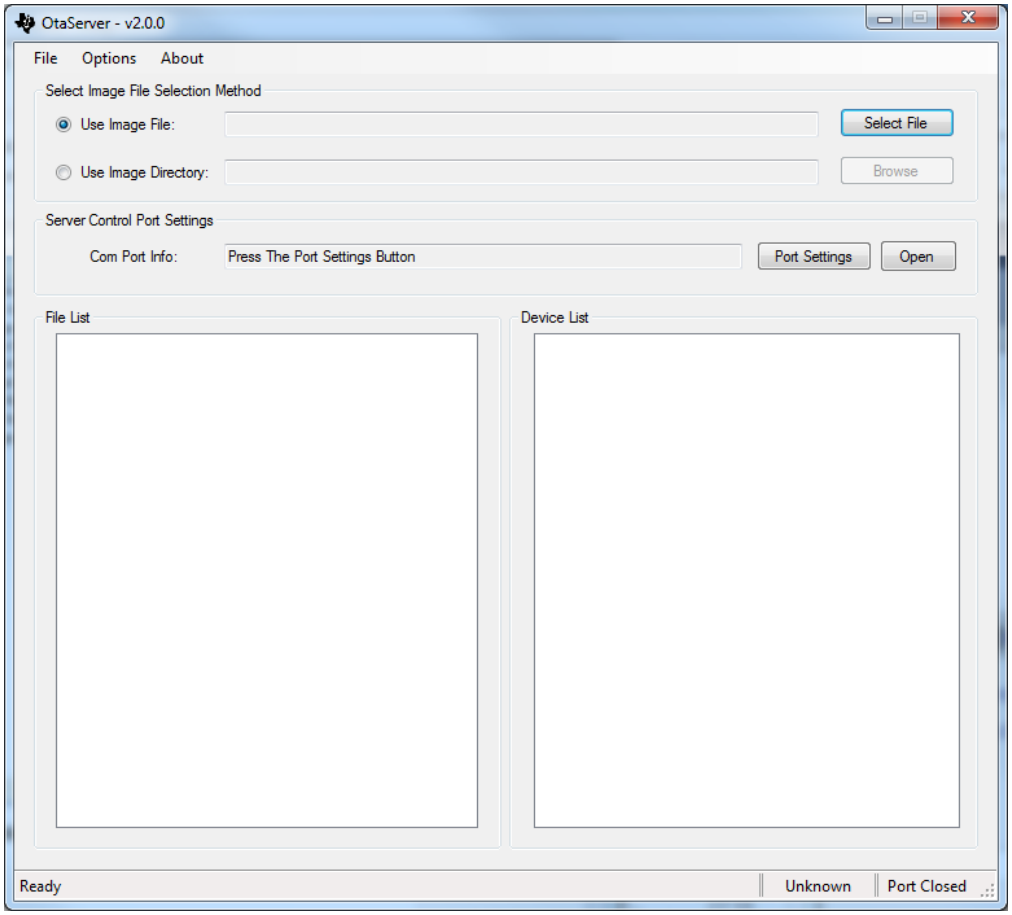
3.3.1 OtaServer Operation

1. Build an OtaDongle that is one of the following**:** a coordinator, router or an end point. An end point is supported but not recommended because the image update speeds will be much slower.

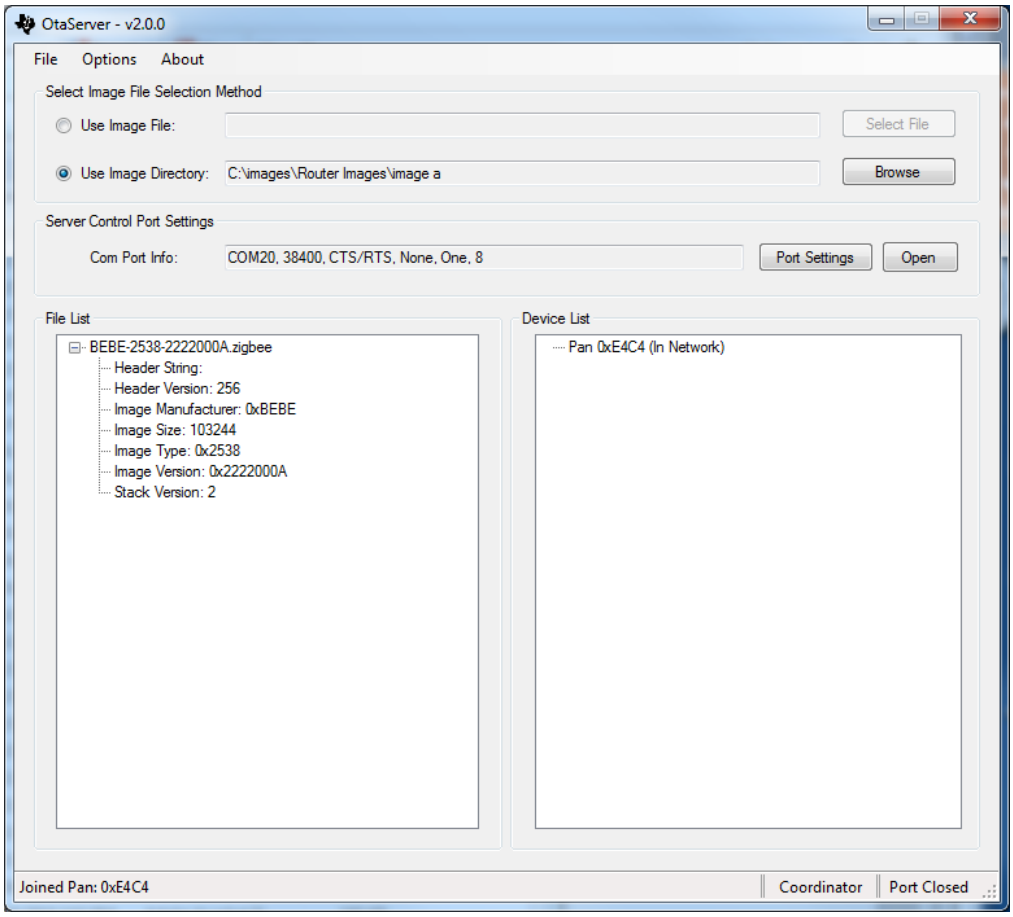
2. Select a file method:

   a. *Use Image File* – Select a single image file for the update. Using this option, it is possible downgrade an image. To downgrade uncheck the "Version Check Image File" in the Options menu.

   b. *Use Image Directory* – All images in the directory are automatically loaded and the newest compatible image is automatically selected during the Image Notify process. It is not possible to downgrade image files using this option.



3. The File List display area will now show the available image(s) for the updating of OTA devices. Expanding the plus sign on each image provides extra information about the image. It is not possible to select image files in the File List display area.



4. Check and modify the Port Settings as necessary. Pressing the Open button will open the port and OtaServer will begin monitoring the OtaDongle. If OtaServer does not see a proper regular message from OtaDongle, it will automatically close the port and report the issue to the user.

   The default baud rate for OtaServer and the OTA server application (Dongle) is 38400 8-N-1.

   NOTE: For a CC2538 hosted OTA Server (Dongle), you will need to disable flow control.

If your OtaDongle is a router (recommended) follow these steps:

1. The Device List will update within 5 seconds with the PAN IDs.
2. If any of PAN IDs say “In Network” your OtaDongle has joined that PAN ID.
3. If that is not the desired PAN ID or you wish to see all the PAN IDs available: Right click on the “In Network” PAN ID, and then select "Leave PAN". It may take up to a minute for the device to reset and scan for PANs. Right click on one of the displayed PAN IDs and select "Join PAN". When the PAN ID shows “In Network” you are ready to proceed to the next step.
4. Right click on the “In Network” PAN ID and select Device Scan.
5. As nodes respond to the request they will appear in the device list. Note that only OtaClient coordinators and routers will respond. OtaClients that are end devices will only appear as they join the network.

If your OtaDongle is a coordinator, follow these steps:

1. The Device List will update within 5 seconds with your coordinators PAN ID.
2. Any nodes that have already joined the pan will not be displayed. Any nodes that join after OtaServer displays the PAN ID will be displayed. The easiest thing to get all the nodes displayed is to right click on the PAN ID that is “In Network”. Selecting “Leave PAN” causes the OtaDongle to reset. It may take a number of minutes for all the nodes to rejoin..
3. As each node joins OtaServer will automatically read the attributes of each node. Press the plus next to the node name to see the information.

If your OtaDongle is an end device (not recommended), follow these steps:

1. The Device List will update within 5 seconds with the PAN IDs.
2. If any of PAN IDs say “In Network” your OtaDongle has joined that PAN ID.
3. If that is not the desired PAN ID or you wish to see all the PAN IDs available: Right click on the “In Network” PAN ID, and then select "Leave PAN". It may take up to a minute for the device to reset and scan for PANs. Right click on one of the displayed PAN IDs and select "Join PAN". When the PAN ID shows “In Network” you are ready to proceed to the next step.
4. Right click on the “In Network” PAN ID and select Device Scan.
5. As nodes respond to the request they will appear in the device list. Note that only OtaClient coordinators and routers will respond. OtaClients that are end devices will NOT appear as they join the network.

These remaining steps are for all configurations:

1. There are two ways to start the image download:

   Right click a Device List node and select Image Notify.

   <Or>

   Right click the “In Network” PAN ID and select Broadcast Image Notify.
2. The image download(s) will begin shortly and may take quite a while (minutes) to complete.
3. When the image download is successful, the node is deleted from the Device List and will reappear in a few minutes after the updated device completes its OTA image upgrade process. If your OtaDongle is an end device, you will need to do a Device Scan after the device completes the OTA process to see the node again.

3.3.2 OtaServer Command Line Operation

Using the command line option changes the OtaServer into command line mode - GUI will not appear and operation progress is sent to the console. OtaServer command line mode does not use or change any settings made while in GUI mode. It is suggested that the user perform some operations using the GUI before using command line mode.

Usage:

OtaServer <-p Port Name> <-s Port Settings> [<-ch Channel Number>] [<-pa PAN ID>] <-de Device Id> <-e Endpoint> <-f Ota Image File> <-d Ota Image Directory> [<-l Log File>] [<-NoVersionCheck>] [<-help>]

Example:

OtaServer -p "COM1" -s "38400,CTS/RTS,None,One,8" -ch 0x0B -pa 0xFFFF -de 0xFFFF -e 0xff -f "c:\my ota files\ota image.zigbee" -l "logfilename” -NoVersionCheck

Options:

<-p Port Name> Required. The Communications Port Name As Displayed In The OtaServer GUI

Example: -p "COM1"

<-s Port Settings> Required. The Communications Port Settings As Displayed In The OtaServer GUI Field Breakdown: "baudrate,flowofcontrol,parity,stopbits,databits"

Example: -s "38400,CTS/RTS,None,One,8"

[<-ch Channel Number>] Optional Only If OtaServer Is A Coordinator. The Channel Number

Example: -ch 0x0B

[<-pa PAN ID>] Optional Only If OtaServer Is A Coordinator. The PAN ID

Example: -pa 0x1234

<-de Device Id> Required. The Network Node Id

Example: -de 0x4321

<-e Endpoint> Required. The End Point Value

Example: -e 0x0E

<-f Ota Image File> Required If No Image Directory Is Provided. The Image File Name With Full Path Information

Example: -f "c:\my ota files\ota image.zigbee"

<-d Ota Image Directory> Required If No Image File Is Provided

The Image Directory Path Information

Example: -f "c:\my ota files"

[<-l Log File>] Optional. The Log File Name With Full Path Information And Without A File Extension

Example: -l "C:\my ota files\logfilename

[<-NoVersionCheck>] Optional And Cannot Be Used With An Image Directory. Supresses Image File Version Number Checking

Example: -NoVersionCheck

[<-help>] Optional. Displays This Help Screen

Returns:

0 = success & 1 = failure

After the download completes, the Client device will copy the new firmware to the operational program space. This may take several minutes. Then the new firmware will start running.

4. Adding Client Functionality to an Application

The following steps must to be taken to add OTA Client functionality to a Z-Stack application:

1) Add the OTA source code to the application.
2) Add the OTA linker configuration file.
3) Add OTA include directory to the list of include directories.
4) Add the configuration OTA compile flags
5) Add the OSAL zclOTA_event_loop and zclOTA_Init task functions for the OTA Task

4.1 Adding OTA Client Source Code

To add the OTA source code to an application's project, perform the following:

1) Open the project workspace in IAR Embedded Workbench
2) In the workspace toolbar:
    a. Right click on the *Profile* folder.
    b. Select *Add>Add Files…* from the pull down menu.
    c. Browse to Components\stack\zcl.
    d. Select the following files:
        i. zcl_ota.h
        ii. zcl_ota.c

4.2 Add the OTA Linker Configuration File

To add the OTA linker configuration file to an application, perform the following:

1) Open the project workspace in IAR Embedded Workbench
2) Click *Projects>Options…* from the menu.
    a. Select the *Linker* category.
    b. Select the *Config* tab.
    c. Depending on the target platform, set the Linker command file to:

        For CC2530 use:

        $PROJ_DIR$\..\..\..\Tools\CC2530DB\ota.xcl

        For MSP430 use:

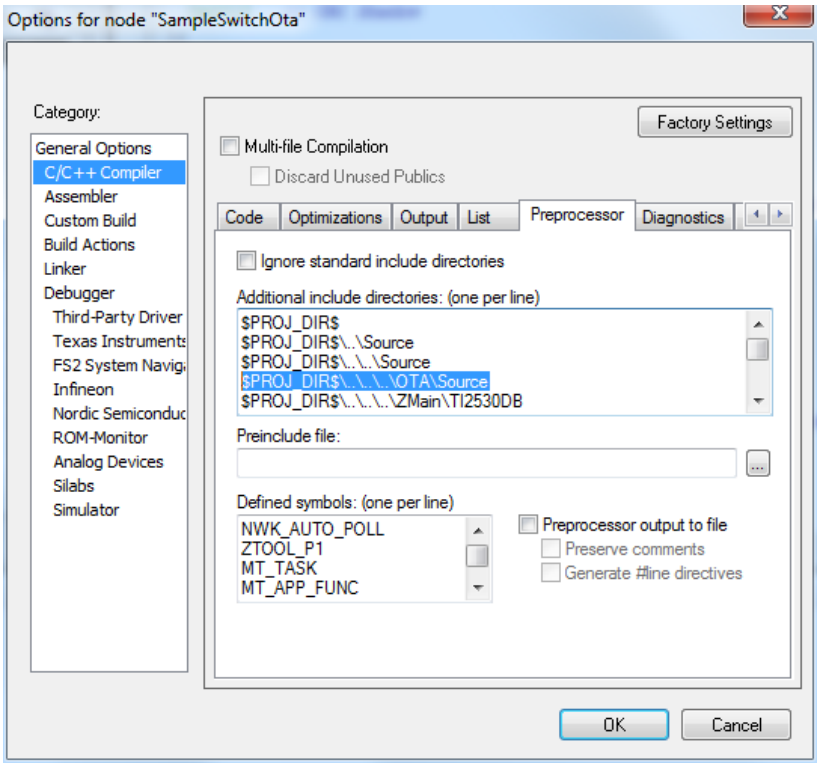
        $PROJ_DIR$\..\..\..\Tools\MSP5438\ota.xcl

        For CC2538 use:

        $PROJ_DIR$\..\..\..\Tools\CC2538DB\CC2538-OTA-Image-A.icf
        Or
        $PROJ_DIR$\..\..\..\Tools\CC2538DB\CC2538-OTA-Image-B.icf

4.3 Add OTA\Source to the Includes Directories

1) Open the project workspace in IAR Embedded Workbench
2) Click *Projects>Options…* from the menu.
    a. Select the *C/C++ Compiler* category.
    b. Select the *Preprocessor* tab.
    c. Add *$PROJ_DIR$\..\..\..\OTA\Source* to the list of *Additional include directories*.



4.4 Adding Conditional Compile Time Configuration

1) Open the project workspace in IAR Embedded Workbench
2) Click *Projects>Options…* from the menu.
    a. Select the *C/C++ Compiler* category.
    b. Select the *Preprocessor* tab.
    c. Add OTA_CLIENT=TRUE to the list of Defined Symbols.
    d. Add OTA_MMO_SIGN=TRUE to enable Smart Energy AES-MMO signing of OTA images.

4.5 Add OSAL Initialize and Task Functions for the OTA Task

1) Open the project workspace in IAR Embedded Workbench
2) Add zclOTA_event_loop to the *tasksArr* array:
    a. Press CTRL-SHIFT-F.
    b. Enter *tasksArr* as the search term.
    c. In the Find in Files dialog, double click on the line containing the definition for *tasksArr*.
    d. Add zclOTA_event_loop to the bottom of the list.
3) Add zclOTA_Init to the *osalInitTasks*
    a. *osalInitTasks* should be a few lines below the *taskArr*.
    b. Add: zclOTA_Init (taskID++); to the end of the *osalInitTask* function.

5. Creating a Combined Hex File

To facilitate more convenient programming, the OTA Boot and Client applications can be combined into a single hex image suitable for the TI Flash Programmer.

5.1 CC2530

The following instructions detail the steps to create and combine OTA Boot and Client application hex images for the CC2530. The basic process will involve creating hex images for each and then, with some very minor edits, the two images can be concatenated with a simple text editor.

5.1.1 Create Hex File Output for OTA Boot Application

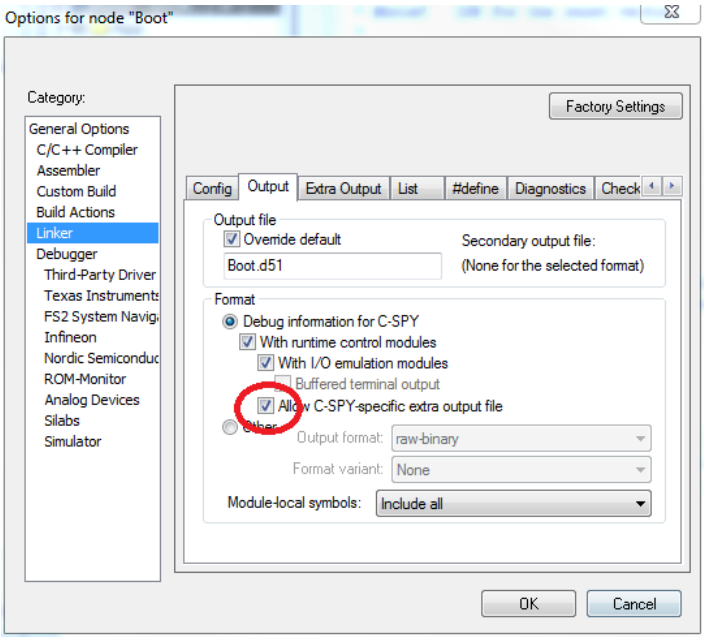
We'll basically repeat the steps outlined in section 3.2.1.1 with the following changes:

1) Generate extra output from the linker.

   With the boot.eww workspace open in IAR:
   a) Click *Project>Options* from the menu.

   b) Click the Linker category.

   c) Select the "Output" tab and check the "Allow C-SPY-specific extra output file" check box:



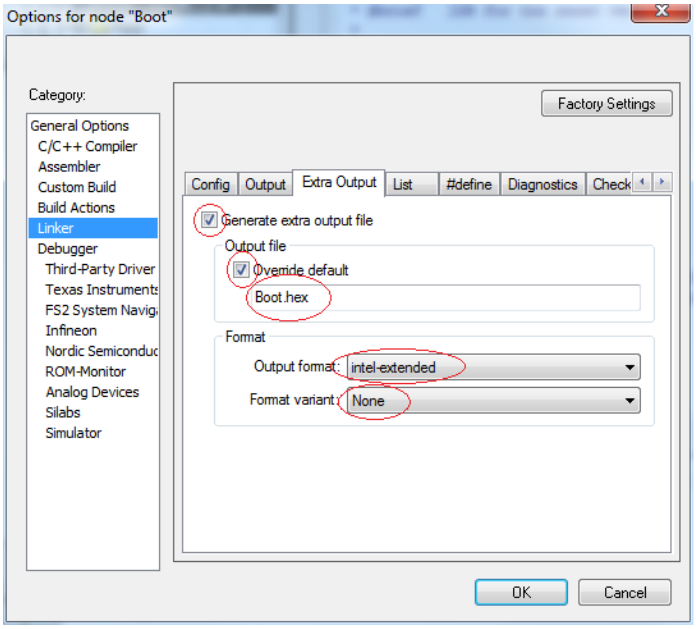
   d) Select the "Extra Output" tab and:

      i) Check the "Generate extra output file" check box

      ii) Check the "Override default" check box and enter "Boot.hex" in the box below.

      iii) Select "Intel-extended" for the "Output format".

      iv) Select "None" for the "Format variant".



2) Click *Project>Rebuild All* from the menu to rebuild the project and generate the new hex file for the OTA Boot application.

Note: the new hex file will be located here: Projects\zstack\OTA\Boot\CC2530DB\OTA-Boot\Exe\Boot.hex

5.1.2 Create Hex File Output for OTA Client Application

We'll basically repeat the steps outlined in section 3.2.3.1with the following changes:

1) Generate extra output from the linker.

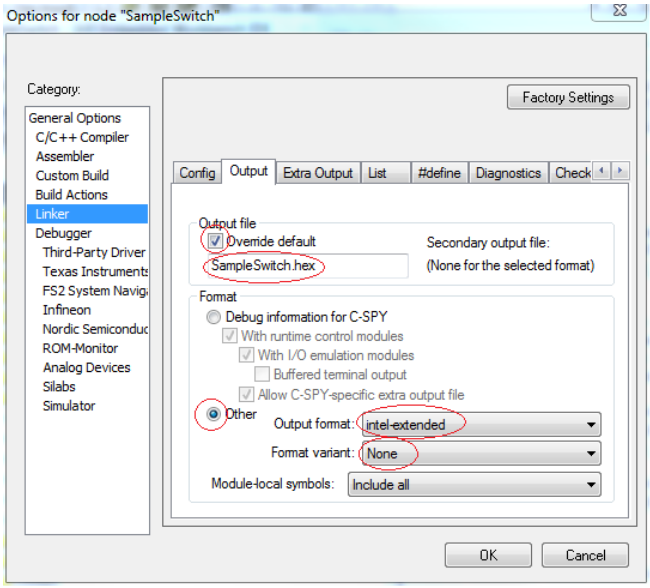
   With the SampleSwitch.eww workspace open in IAR and either of the OTA Client build configurations selected (RouterEB – OTA Client or EndDeviceEB – OTA Client), make the following changes:

   a) Click *Project>Options* from the menu.

   b) Click the "Linker" category.

   c) Check the "Override default" check box and enter "SampleSwitch.hex" in the box beneath the check box.

   d) Check the "Other" radio button in the Format section and select "Intel-extended" for the "Output format" and "None" for the "Format variant".



2) Modify the linker control file.

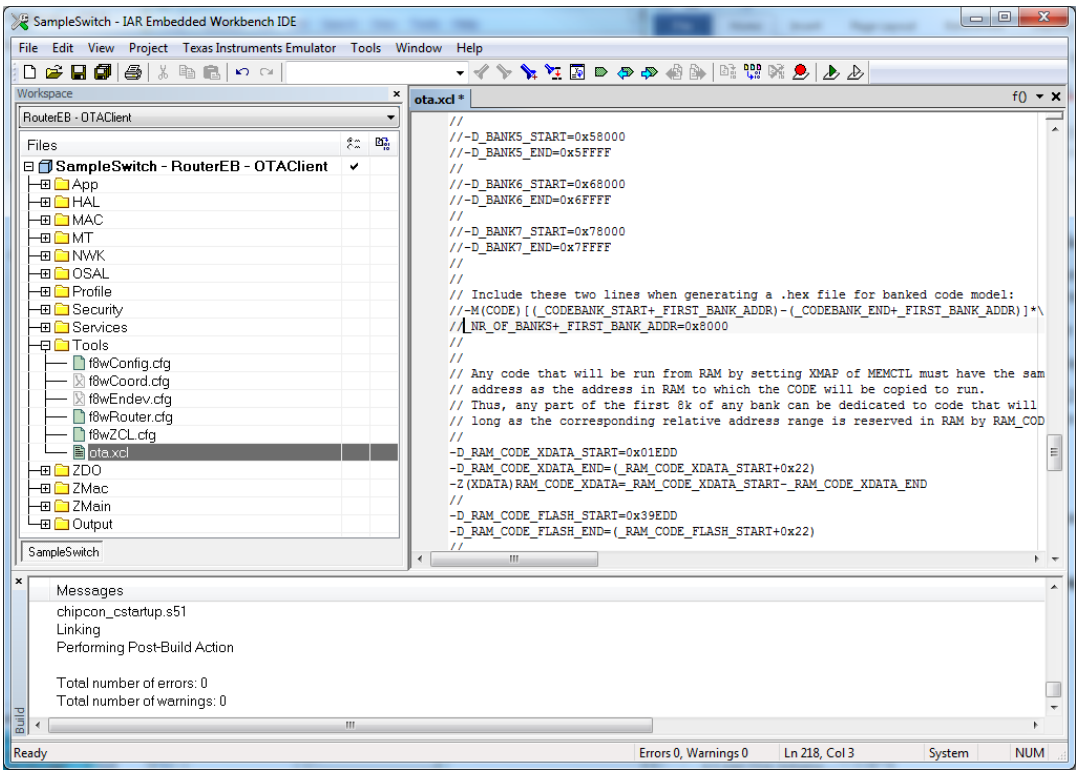
We need to uncomment two lines in the linker control file, for the Sample Switch: OTA Client (Router or End Device) build configuration, to properly create the hex file output.

a) Click "File > Open > File"

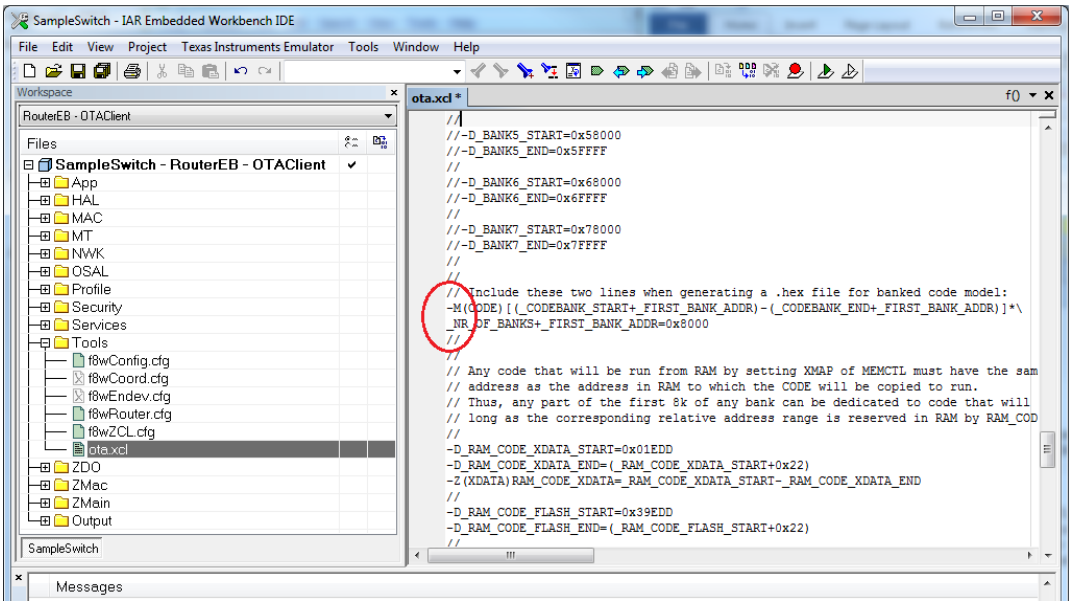
b) In the dialog box select: Projects\zstack\Tools\CC2530DB\OTA.xcl

c) Near line 216, remove the comment from the line that begins with "-M". Note that the line is actually wrapped to a second line so two comments need to be removed:

Before:



After:



3) Click *Project>Rebuild All* from the menu to rebuild the project and generate the new hex file.

Note: the new hex file will be located here: Projects\zstack\HomeAutomation\SampleSwitch\CC2530DB\<selected build configuration\Exe\SampleSwitch.hex

Note: The changes to the .xcl file should be removed before compiling and programming with the debugger again.

5.1.3 Combine Hex Files

Using a simple text editor, we can now combine the .hex files created in the previous two steps:

1) Use any text editor to open the application hex file produced above: Projects\zstack\HomeAutomation\SampleSwitch\CC2530DB\<selected build configuration\Exe\SampleSwitch.hex

2) Delete this first line from the file:

   :020000040000FA

3) In a separate text editor window, open the OTA Boot application hex file produced above: Projects\zstack\OTA\Boot\CC2530DB\OTA-Boot\Exe\Boot.hex

4) Delete the last two lines from Boot.hex file. They should resemble the following:

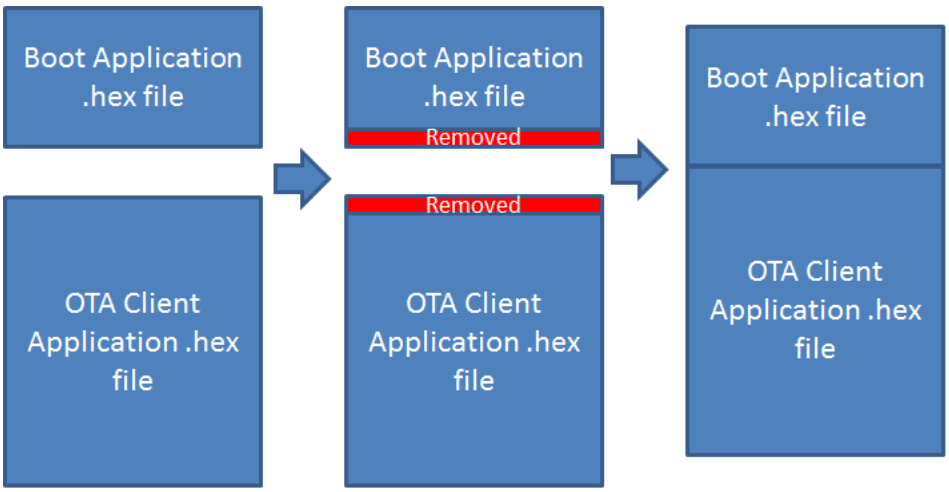
   :0400000500000738B8

   :00000001FF

5) Copy the edited contents of the OTA Boot Application hex file to the top of the Application Code file and save it.

6) Use the SmartRF Programmer to install the edited hex image into the CC2530 SoC.

The following diagram shows what we're doing:



5.2 CC2538

The CC2538 Home Automation Sample Switch IAR workspace contains two build configurations that produce a combined (OTA Boot and OTA Client applications) image. These are:

- Router – OTA Client – Image A – withBootloader
- EndDevice – OTA Client – Image A – withBootloader

To produce a combined .hex image, suitable for the programming with the SmartRF Flash Programmer 2 tool, we need to make a very minor change to the build configuration in IAR:

1) Enable extra output from the linker.

   With the SampleSwitch.eww workspace open in IAR and one of the above build configurations selected:
   a) Click *Project>Options* from the menu.

   b) Click the Output Converter category.

   c) On the Output tab, check the "Generate additional output" check box.

   d) Select "Intel extended" for the Output format.

   e) Check the "Override default" check box for the Output file and enter any name you prefer: e.g. "SampleSwitch_Combined.hex"

2) Click *Project>Rebuild All* from the menu to rebuild the project and generate the new hex file. NOTE: the new hex file will be located here: Projects\zstack\HomeAutomation\SampleSwitch\CC2538\<selected build configuration>\Exe