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

TABLE OF CONTENTS

LIST OF FIGURES

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

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:

- OTA Console 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 update to. The server responds to the query next image request with a *Query Next Image Response* message.

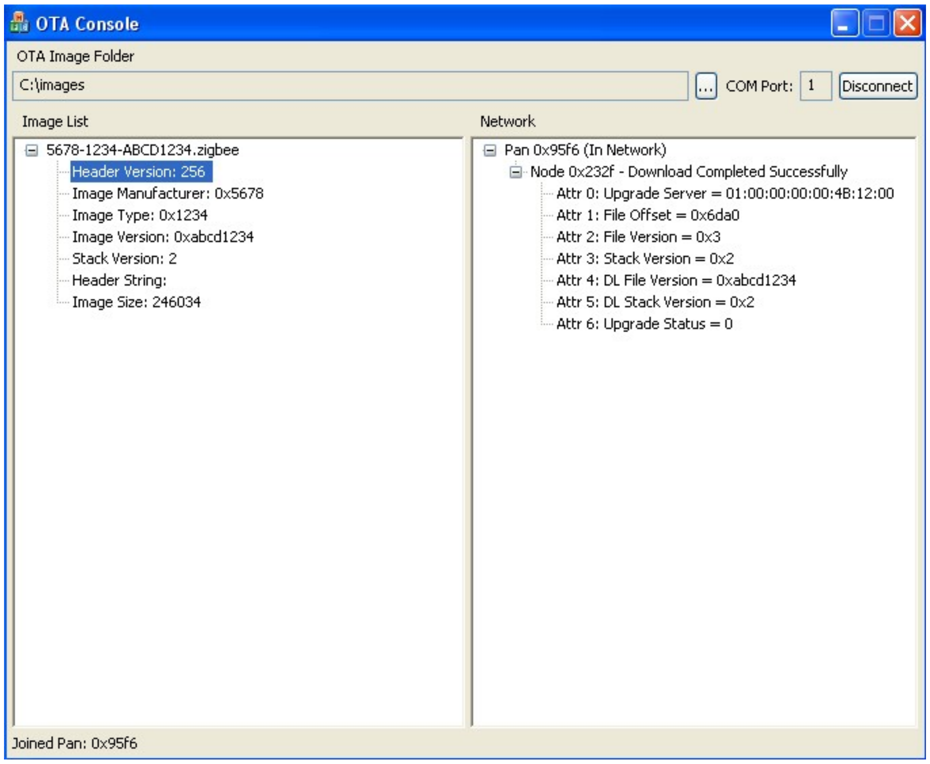
The *Query Next Image Response* message may instruct the client to download new firmware, or it may inform the client that no firmware is available. Should a download take place, the client controls the download. During the download, the client sends *Image Block Request* messages to the server and receives *Image Block Response* messages from the server with chunks of the upgrade image. The client writes the 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 bootloader on the client sees that the new image is available. Then the bootloader copies the new image from secondary storage to the operational memory space. Then the new firmware is started.

2.2 OTA Console Application

The OTA Console application is the front end of Texas Instruments' Z-Stack OTA Server sample application. The OTA Console Application requires Microsoft Windows XP and an RS-232 serial port. The OTA Console connects to a SmartRF05 or to an MSP430 EXP board running the OTA Dongle sample application via an RS-232 serial port.



```
C:\images    COM Port: 1

5678-1234-ABCD1234.zigbee
    Header Version: 256
    Image Manufacturer: 0x5678
    Image Type: 0x1234
    Image Version: 0xabcd1234
    Stack Version: 2
    Header String:
    Image Size: 246034

Pan 0x95f6 (In Network)
  Node 0x232f - Download Completed Successfully
    Attr 0: Upgrade Server = 01:00:00:00:00:4B:12:00
    Attr 1: File Offset = 0x6da0
    Attr 2: File Version = 0x3
    Attr 3: Stack Version = 0x2
    Attr 4: DL File Version = 0xabcd1234
    Attr 5: DL Stack Version = 0x2
    Attr 6: Upgrade Status = 0

Joined Pan: 0x95f6
```

Figure 1: OTA Console Application

2.2.1 COM Port

The *COM Port* box on the OTA Console is used to select the RS-232 serial port that the OTA Dongle is connected to.

To open the COM Port, press the C*onnect* button. After the port is opened, the text of the *Connect* button will change to *Disconnect*. To change to a different COM Port, first click *Disconnect*. Then change the value of the COM port. Then click *Connect*.

The default baud rate for OTA Console and the OTA server application is 38400 8-N-1.

2.2.2 Image List

The *Image List* contains a list of the images hosted by the OTA Server. The Images are loaded from the *OTA Image Folder*. To change the OTA Image Folder, click the button with three dots (…) to the right of the *OTA Image Folder*. Then, browse to the folder containing the images and click *OK*.

The OTA Console loads the images from the *OTA Image Folder* into memory. To reload the Images, right click in the Image List and click *Reload*.

The Image List can be used to view the header information in the OTA Image file. To see the header, click the expand item (plus sign) next to the name of the image file.

2.2.3 Network Manager

The *Network Manager* is used to manage the OTA Server's network. The following can be performed from the Network Manager:

- Scan for PANs
- Join a PAN
- Leave a PAN
- Scan for Devices
- Send an Image Notify
- Read OTA Attributes

2.2.3.1 Scanning for a PAN

The OTA Console can be used to scan for nearby PANs if the OTA Dongle is a router or an End Device. To scan for a PAN, right-click in the Network Manager and click *Scan for PANs*. The scan may take several seconds. If PANs are found, they will be added to the list of PANs on the network manager.

Scanning for a PAN is not necessary if the OTA Dongle is a coordinator. Coordinators form their own PANs.

2.2.3.2 Joining a PAN

The OTA Console can be used to join a PAN if the OTA Dongle is a router or an End Device. To join a PAN, first scan for PANs. Then right click on the PAN to join and click *Join PAN*.

Joining a PAN is not necessary if the OTA Dongle is a coordinator. Coordinators form their own PANs.

2.2.3.3 Leaving a PAN

To leave your PAN, right click on the PAN you are a member of and click *Leave PAN*.

2.2.3.4 Scan for Devices

To scan for devices, right click on the PAN you are a member of and click *Scan for Devices*. The OTA dongle will perform a match descriptor request for clients that implement the OTA cluster. All devices that support the support the OTA cluster as a client will respond to the match descriptor request and the dongle application will record the destination endpoint information. And end device announce received from the joining device will also trigger the dongle application to discover the device's "OTA capabilities" using the same match descriptor request mechanism. However, instead of sending it as a broadcast, the match descriptor request will be sent unicast to the joined device.

2.2.3.5 Send an Image Notify

The OTA Console can send the OTA Cluster Image Notify. To broadcast an Image Notify to all devices, right click on the PAN you are a member of and click *Broadcast Image Notify*. To send a unicast image notify. Right click on the node to send the Image Notify to and click *Image Notify*.

2.2.3.6 OTA Console Command Line Arguments

The OTA Console accepts command line arguments. These command line arguments permit use of the OTA Console without a GUI and can be useful for controlling the OTA server from a script. The following is a list of the OTA Console command line arguments:

- -w – Working Directory
- -c – COM Port
- -p – Join the specified PAN
- -l – Channel to scan (required when specifying a pan
- -n – Hide the GUI
- -b – Broadcast an Image Notify on startup

An example of running the OTA Console from the command line follows:

```
OtaServer.exe  -w c:\images -c2 -n -b -p0x1234 –l14
```

In the above example, the OTA Console will start up automatically without a GUI, load the images located in c:\images, connect to COM Port 2, join PAN ID 0x1234 on channel 14, and broadcast an image notify.

2.2.4 OTA Dongle

The OTA Console communicates via UART to a device running the OTA Dongle sample application firmware.

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. 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 contains 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

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:
  - CC2530DB
  - CC2520DB
  - EXP5438
- -o<path> Output folder to put the image in
- -s<path> Location of the certificate used to sign the image

An example command line for the Image Converter Tool follows:

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

2.3.3.1 Image Manufacturer Identifier (-m)

The Manufacturer Identifier is a 16-bit hexadecimal 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 Option (-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:

- cc2530 – SmartRF05 with an attached CC2530EM
- msp26xx – SmartRF05 with an attached EM430F2618 and CC2520EM
- 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 OTA Console Tool. An example of the Output Directory option follows:

```
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. An example of the Signature Option follows:

```
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 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_signatures.c
- ota_signatures.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 OTA Console, and the OTA Image Converter.

2.4.3 OTA_SIGNATURES.C and OTA_SIGNATURES.H

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

2.4.4 Compile Time Configuration

The OTA firmware can be configured with compile time definitions to include OTA Client functionality by defining OTA_CLIENT=TRUE. The OTA firmware can be configured to include OTA Server functionality by defining OTA_SERVER=TRUE. The OTA firmware can be configured to require signatures by defining the OTA_MMO_SIGN=TRUE.

2.4.5 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 query servers for the next upgrade cluster.

2.4.5.1 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 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.

2.4.5.2 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.

2.4.5.3 zclOTA_SendImageNotify

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

2.4.5.4 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.

2.4.6 OTA Client Memory Partition

When an OTA Client downloads a new Upgrade Image, it must store the image in secondary storage. Later a bootloader copies the image from secondary storage into the operational space.

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

- SmartRF05 with attached CC2530EM
- EXP430F5438 with attached CC2520EM
- SmartRF05 with attached EM430F2618 with attached CC2520EM

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

2.4.6.1 SmartRF05 with CC2530EM

The Bootloader 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.

2.4.6.2 EXP430F5438 with CC2520EM

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

2.4.6.2.1 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.

2.4.6.2.2 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.

2.4.6.3 SmartRF05 with EM430F2618 and CC2520EM

The bootloader is loaded into address 0xFA00 – 0xFF3D on the EXP430F2618. The MSP430F2618 uses an off-chip serial flash as the secondary storage.

2.4.7 OTA Bootloader

The OTA bootloader used by Z-Stack is responsible for:

- Copying memory from the secondary storage space to the primary storage space.
- Performing a CRC check of the memory in the secondary and primary space to verify the integrity of the image.
- Forwarding interrupts to the primary memory space.
- Booting the application in the primary memory space.

The OTA Boot application cannot 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 v2.4.0 or greater
- IAR Embedded Workbench
- Two Evaluation Boards with EM modules

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

- *CC2530DB* - SmartRF05 with attached CC2530EM
- *EXP5438* - EXP430F5438 with attached CC2520EM
- *CC2520DB* - SmartRF05 with attached CCMSP-EM430F2618 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 SampleSwitchOta 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 *CC2530DB* platform, verify the two SmartRF05 boards have CC2530EM modules attached.
   b. For the *EXP5438* platform, verify the two EXP430F5438 boards have CC2520EM modules attached.
   c. For the CC2520DB platform, verify the two SmartRF05 boards have *CCMSP-EM430F2618* modules attached, and verify the CCMSP-EM430F2618 modules have CC2520EM modules attached.

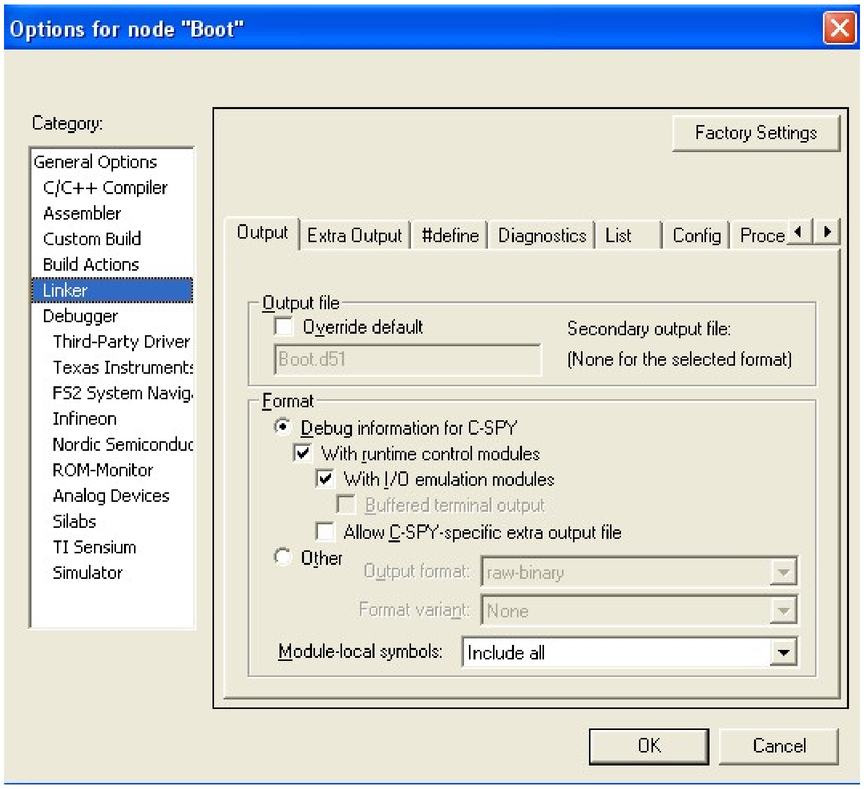
3.2.1 OTA Boot

The OTA Boot application is only required for the CC2530DB platform. The OTA bootloader is included in the IPD – OTA project for the EXP5438 and CC2520DB projects. The OTA Boot application is located in

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

To build and download the OTA Boot application:

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:

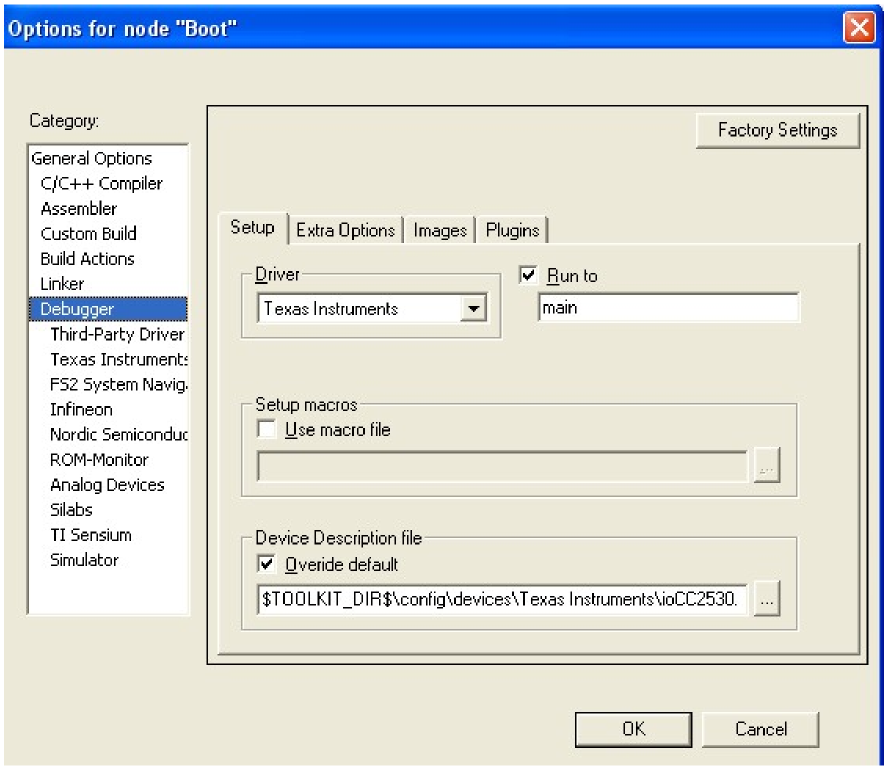


Output file
- [ ] Override default
Boot.d51
Secondary output file:
(None for the selected format)

Format
- (●) Debug information for C-SPY
  - [✓] With runtime control modules
  - [✓] With I/O emulation modules
  - [ ] Buffered terminal output
  - [ ] Allow C-SPY-specific extra output file
- ( ) Other
  - Output format: raw-binary
  - Format variant: None

Module-local symbols: Include all

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



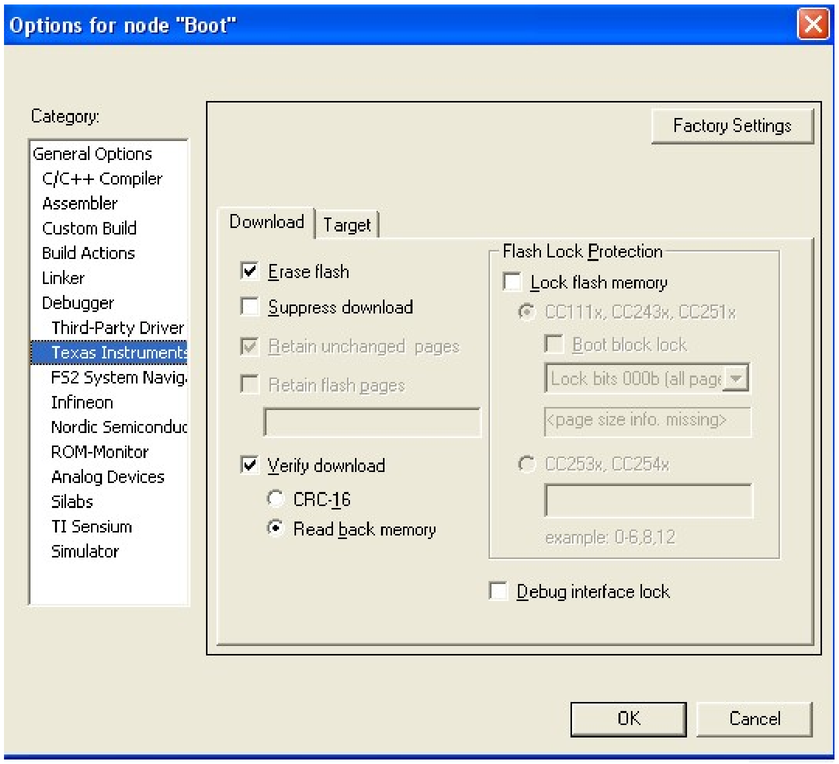
Driver: Texas Instruments

- [✓] Run to: main

Setup macros
- [ ] Use macro file

Device Description file
- [✓] Overide default
$TOOLKIT_DIR$\config\devices\Texas Instruments\ioCC2530.

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



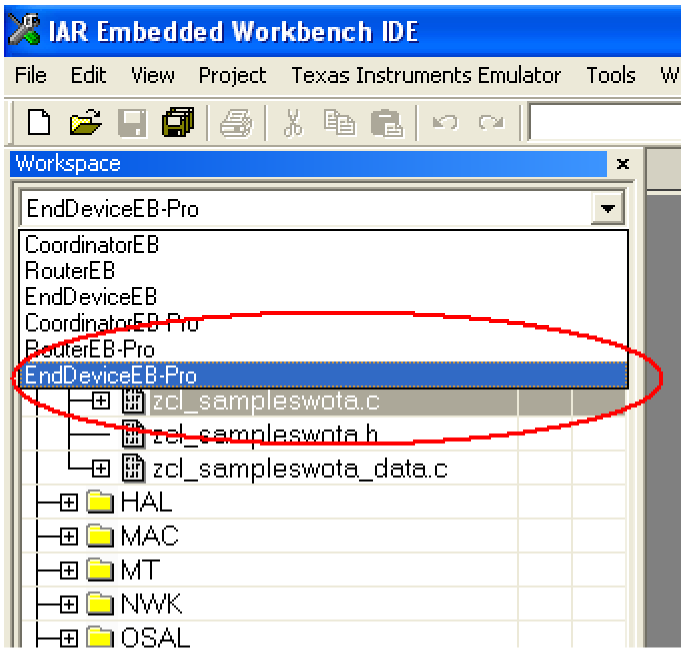
Download:
- [✓] Erase flash
- [ ] Suppress download
- [✓] Retain unchanged pages
- [ ] Retain flash pages
- [✓] Verify download
  - ○ CRC-16
  - ● Read back memory

Flash Lock Protection:
- [ ] Lock flash memory
  - ● CC111x, CC243x, CC251x
    - [ ] Boot block lock
    - Lock bits 000b (all page
    - <page size info. missing>
  - ○ CC253x, CC254x
    - example: 0-6,8,12
- [ ] Debug interface lock

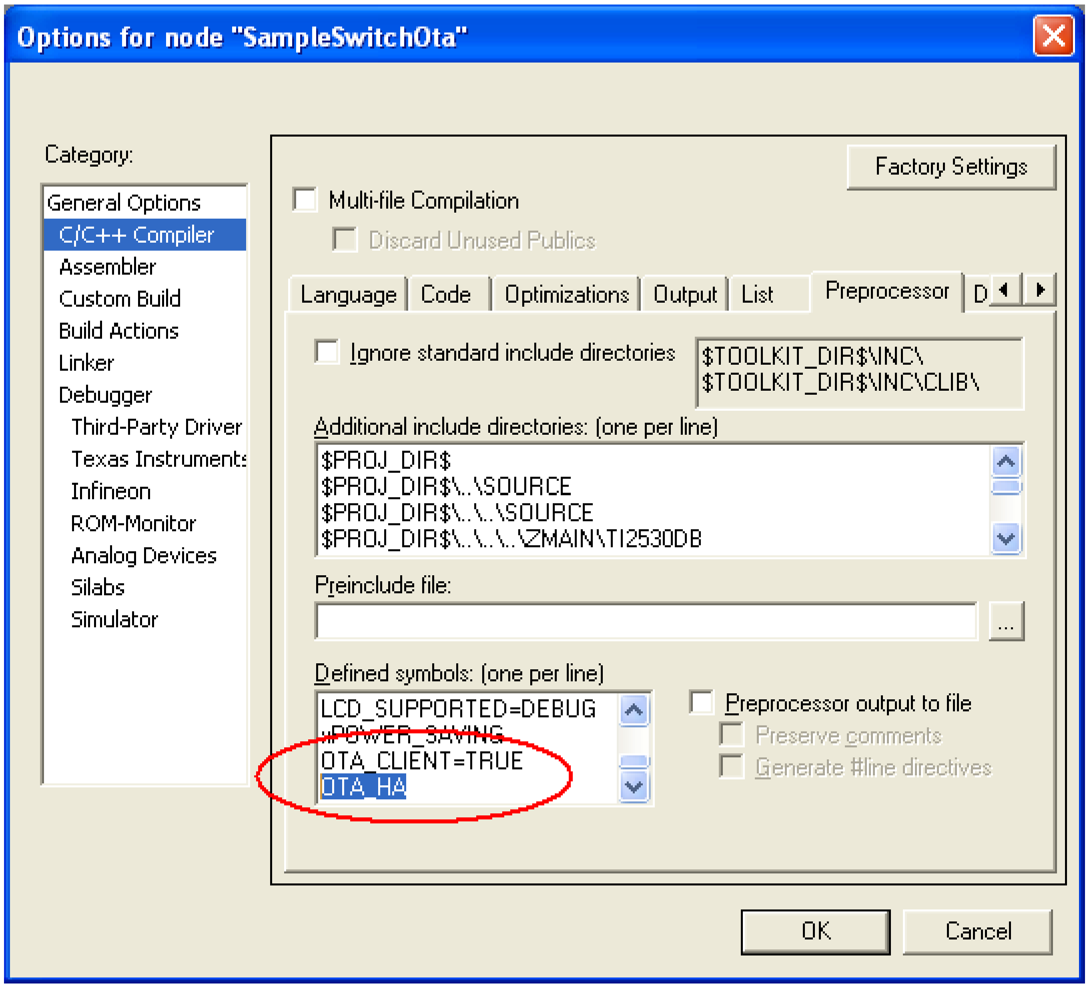
3) Build the Project.
   a. Verify the active project in the *Workspace* toolbar is *Debug*
   b. Click *Project>Rebuild All* from the menu.
   c. Wait for the build to complete.
4) Connect the SmartRF05 board labeled the C*lient* to the PC with a USB cable.
5) Download the image to the SmartRF05 board: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

3.2.2 HA SampleSwOta Sample Application

To build and download the HA OTA App:
1) Open SampleSwitchOta.eww with IAR Embedded Workbench
2) Select the In EndDeviceEB-Pro configuration
   a. From the Workspace dialog, click the configuration dropdown button.
   b. Verify the In EndDeviceEB-Pro 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. Varify that the following symbols are defined:
      i. OTA_CLIENT=TRUE
      ii. OTA_HA



- [ ] Ignore standard include directories

```
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
```

Additional include directories: (one per line)

```
$PROJ_DIR$
$PROJ_DIR$\..\SOURCE
$PROJ_DIR$\..\..\SOURCE
$PROJ_DIR$\..\..\..\ZMAIN\TI2530DB
```

Preinclude file:

Defined symbols: (one per line)

```
LCD_SUPPORTED=DEBUG
xPOWER_SAVING
OTA_CLIENT=TRUE
OTA_HA
```

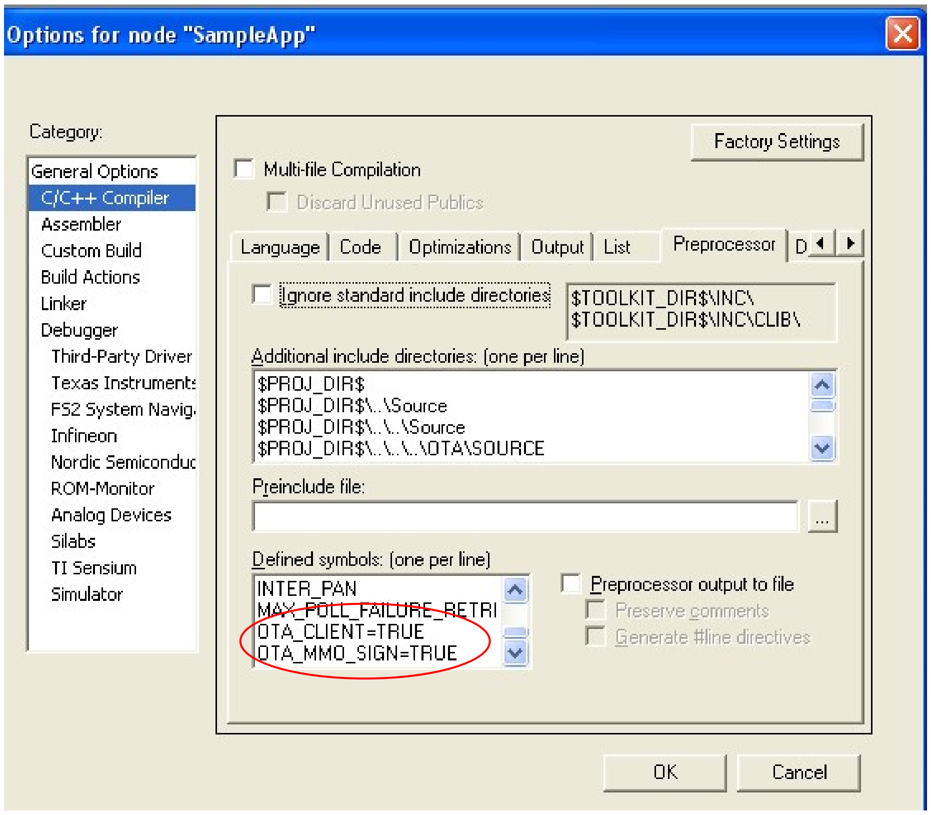
- [ ] Preprocessor output to file
- [ ] Preserve comments
- [ ] Generate #line directives

4) From here follow from on from 3) of the In Premise Display – OTA SE SampleApp instructions below.

3.2.3 In Premise Display – OTA SE SampleApp

To build and download the IPD OTA App:

1) Open SampleApp.eww with IAR Embedded Workbench
2) 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:
        iii. OTA_CLIENT=TRUE
        iv. OTA_MMO_SIGN=TRUE
    d. Verify that the following symbols are NOT defined
        v. OTA_HA

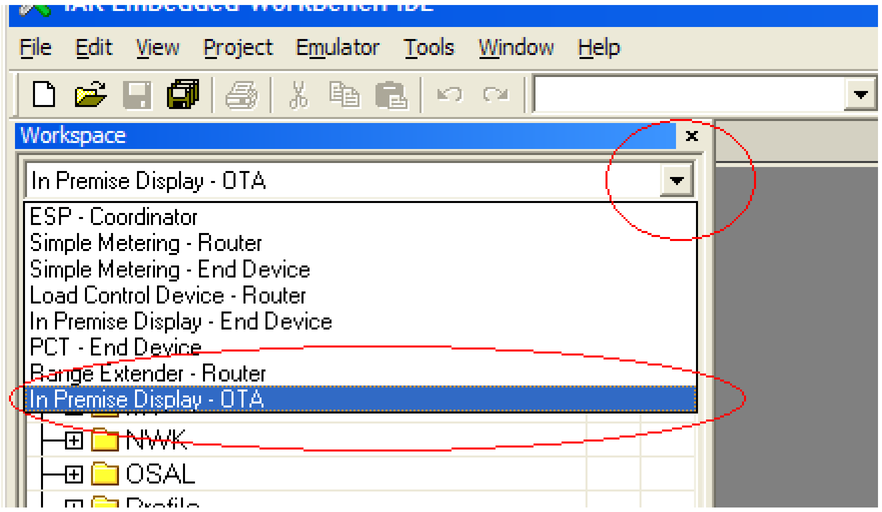


```
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
```

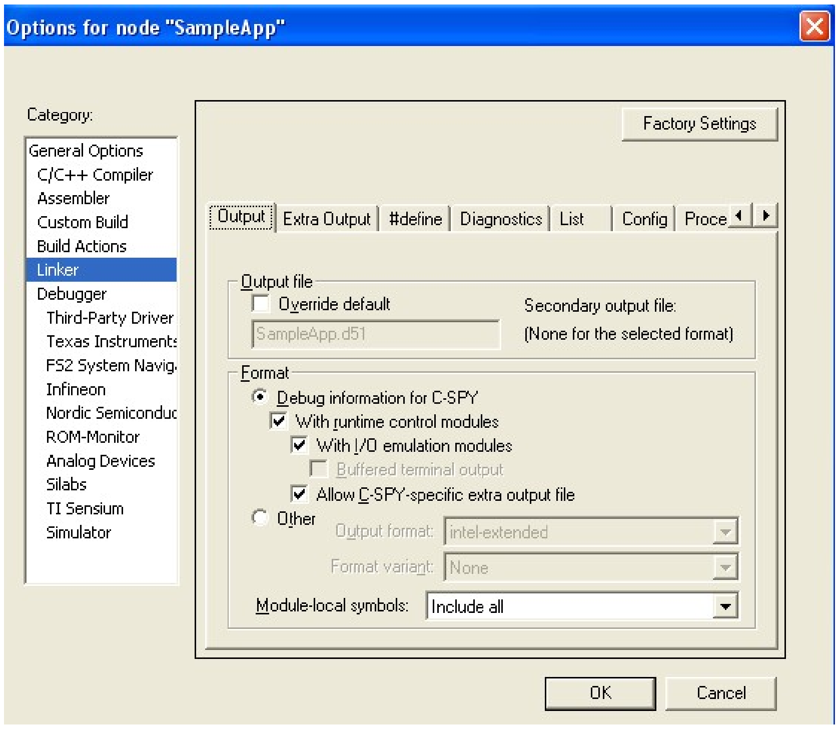
```
$PROJ_DIR$
$PROJ_DIR$\..\Source
$PROJ_DIR$\..\..\Source
$PROJ_DIR$\..\..\..\OTA\SOURCE
```

```
INTER_PAN
MAX_POLL_FAILURE_RETRI
OTA_CLIENT=TRUE
OTA_MMO_SIGN=TRUE
```

3) Select the In Premise Display – OTA configuration
   a. From the Workspace dialog, click the configuration dropdown button.
   b. Verify the In Premise Display – OTA configuration is selected



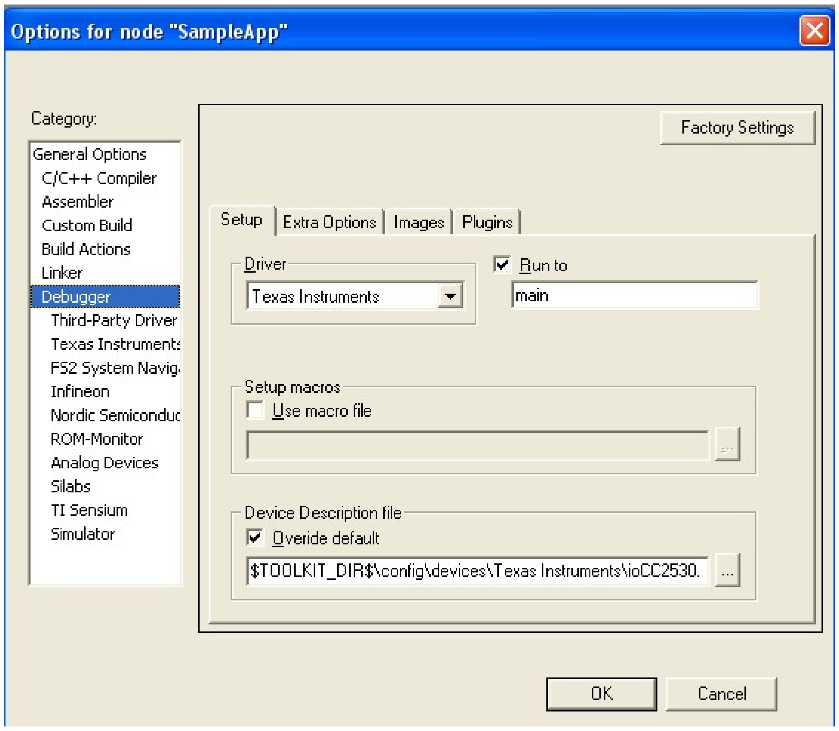
4) Verify the project’s options are configured properly.
   a. Click *Project>Options* from the menu.
   b. Click the Linker category. Verify the Output contains debug information for C-Spy, and that the “Allow C-SPY-specific extra output file” checkbox is selected:



Output file
☐ Override default
SampleApp.d51
Secondary output file:
(None for the selected format)

Format
● Debug information for C-SPY
☑ With runtime control modules
☑ With I/O emulation modules
☐ Buffered terminal output
☑ Allow C-SPY-specific extra output file
○ Other
Output format: intel-extended
Format variant: None
Module-local symbols: Include all

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

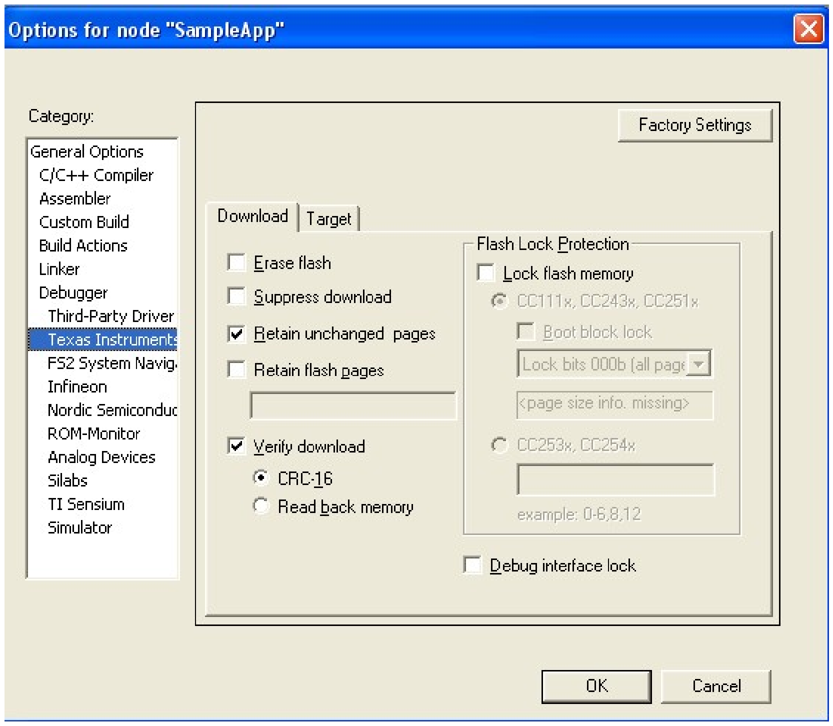


Driver: Texas Instruments
☑ Run to: main

Setup macros
☐ Use macro file

Device Description file
☑ Overide default
$TOOLKIT_DIR$\config\devices\Texas Instruments\ioCC2530.

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



Options for node "SampleApp"

Category:

General Options
C/C++ Compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Third-Party Driver
Texas Instruments
FS2 System Navig
Infineon
Nordic Semiconduc
ROM-Monitor
Analog Devices
Silabs
TI Sensium
Simulator

Factory Settings

Download | Target

- [ ] Erase flash
- [ ] Suppress download
- [x] Retain unchanged pages
- [ ] Retain flash pages
- [x] Verify download
  - (●) CRC-16
  - ( ) Read back memory

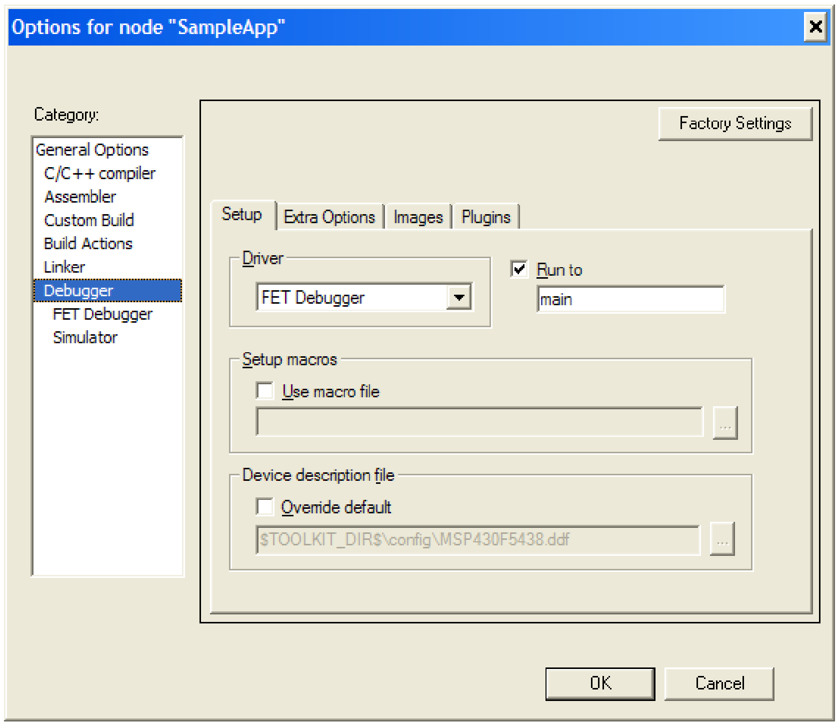
Flash Lock Protection
- [ ] Lock flash memory
  - (●) CC111x, CC243x, CC251x
    - [ ] Boot block lock
    - Lock bits 000b (all page
    - <page size info. missing>
  - ( ) CC253x, CC254x
    - example: 0-6,8,12

- [ ] Debug interface lock

OK Cancel

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.

e. For the EXP5438 or CC2520DB platforms, verify the Driver is set to FET Debugger:



Driver: FET Debugger
Run to: ✓ main
Use macro file: ☐
Override default: ☐ $TOOLKIT_DIR$\config\MSP430F5438.ddf

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 CC2520DB and 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.

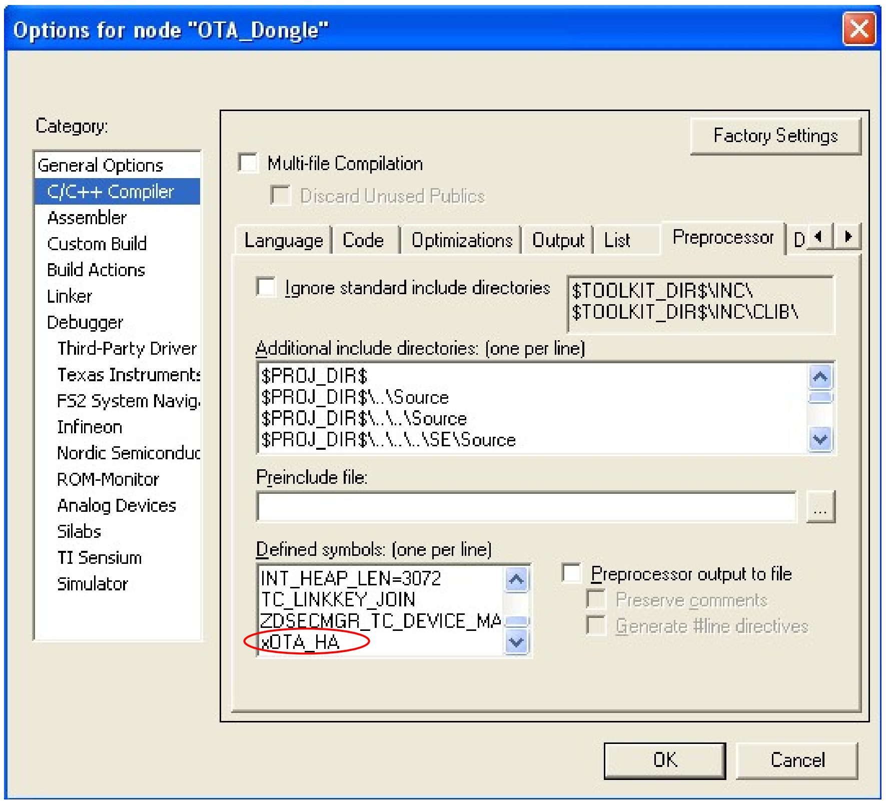
3.2.4 Dongle

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.
   b. Click *Project>Options* from the menu.
   c. Click on C/C++ Compiler category

d. For HA profile verify that the following symbols are defined:
    vi. OTA_HA
e. For SE profile verify that the following symbols are NOT defined
    vii. OTA_HA

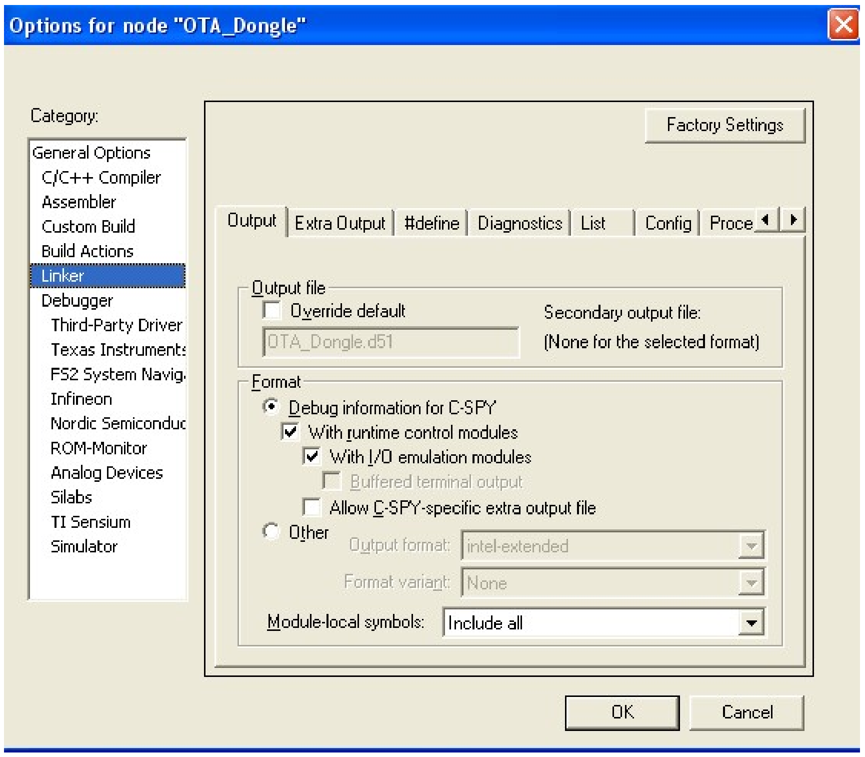


```
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
```

```
$PROJ_DIR$
$PROJ_DIR$\..\Source
$PROJ_DIR$\..\..\Source
$PROJ_DIR$\..\..\..\SE\Source
```

```
INT_HEAP_LEN=3072
TC_LINKKEY_JOIN
ZDSECMGR_TC_DEVICE_MA
xOTA_HA
```

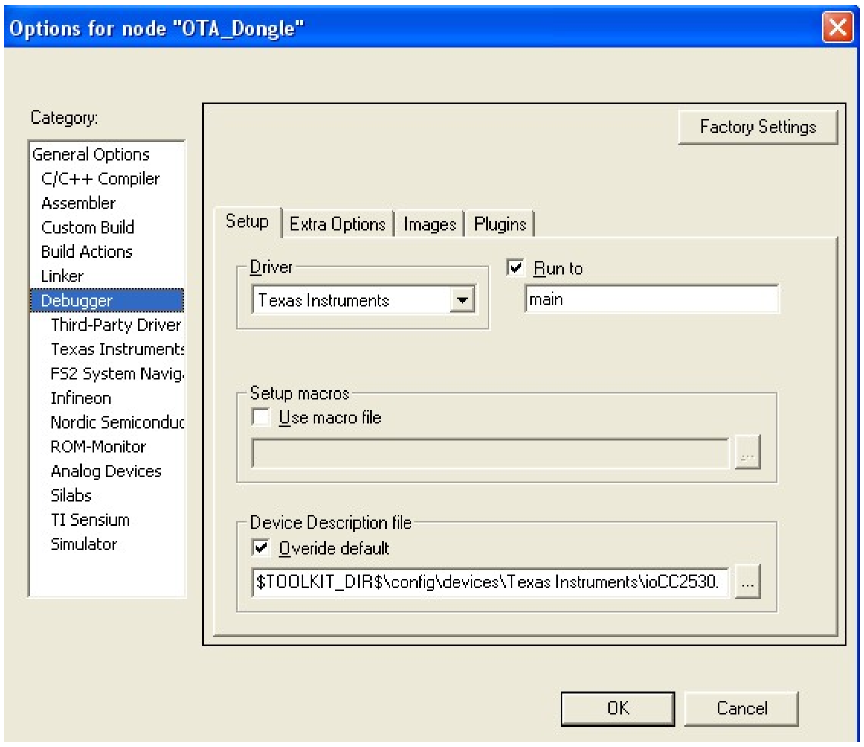
3) Verify the project's options are configured properly.
    a. Click *Project>Options* from the menu.
    b. Click the Linker category. Verify the Output contains debug information for C-Spy:



Output file
- [ ] Override default
- OTA_Dongle.d51
- Secondary output file: (None for the selected format)

Format
- (●) Debug information for C-SPY
  - [✓] With runtime control modules
  - [✓] With I/O emulation modules
  - [ ] Buffered terminal output
  - [ ] Allow C-SPY-specific extra output file
- ( ) Other
  - Output format: intel-extended
  - Format variant: None
- Module-local symbols: Include all

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

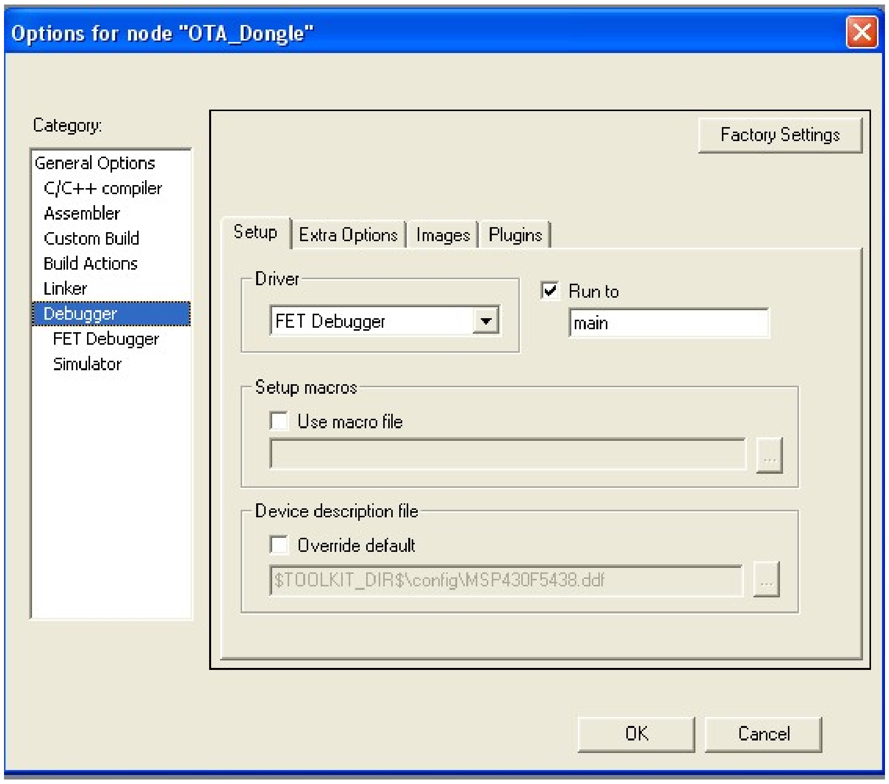


- Driver: Texas Instruments
- [✓] Run to: main

Setup macros
- [ ] Use macro file

Device Description file
- [✓] Overide default
- $TOOLKIT_DIR$\config\devices\Texas Instruments\ioCC2530.

c. For the EXP5430 or CC2520DB platforms, verify the Driver is set to FET Debugger:


Options for node "OTA_Dongle"

Category: General Options, C/C++ compiler, Assembler, Custom Build, Build Actions, Linker, Debugger (selected), FET Debugger, Simulator

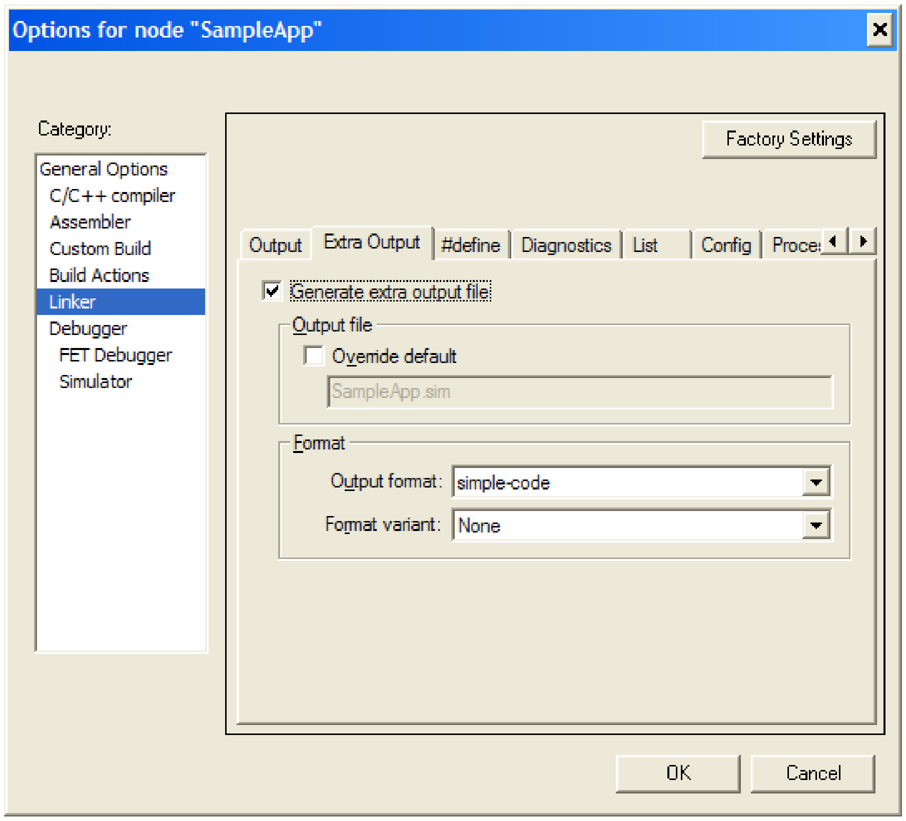
Setup tab — Driver: FET Debugger; Run to: ✓ main; Setup macros: Use macro file ☐; Device description file: Override default ☐, $TOOLKIT_DIR$\config\MSP430F5438.ddf

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 CC2520DB and 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.
6) Download the image to the SmartRF05 board:
   a. Verify the *Coordinator* configuration is selected from the *workspace*.
   b. Click *Project>Debug.*
   c. 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 *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 CC2520DB and CC2530DB platforms, connect the SmartRF05 board's RS-232 connector to a COM port on the PC.
   b. For the EXP5438 platform, connect the EXP430F5438's micro USB connector to the PC.

3.3 Generating a Binary File

The firmware to be transferred via OTA must be linked by IAR Embedded Workbench into a Simple Binary File. To generate a Simple Binary File, perform the following:

1) Open the IPD – OTA application (SampleApp.eww) project in IAR Embedded Workbench
2) Verify the project's options are configured properly.
   a. Click *Project>Options* from the menu.
   b. Click the Linker category.
   c. Click the Extra Output tab.
   d. Verify the Output format is simple-code:



☑ Generate extra output file

Output file

☐ Override default

SampleApp.sim

Format

Output format: simple-code

Format variant: None

3) Build the Project.
   a. Verify the active project in the *Workspace* toolbar is *In Premise Display - OTA*
   b. Click *Project>Rebuild All* from the menu.
   c. Wait for the build to complete.

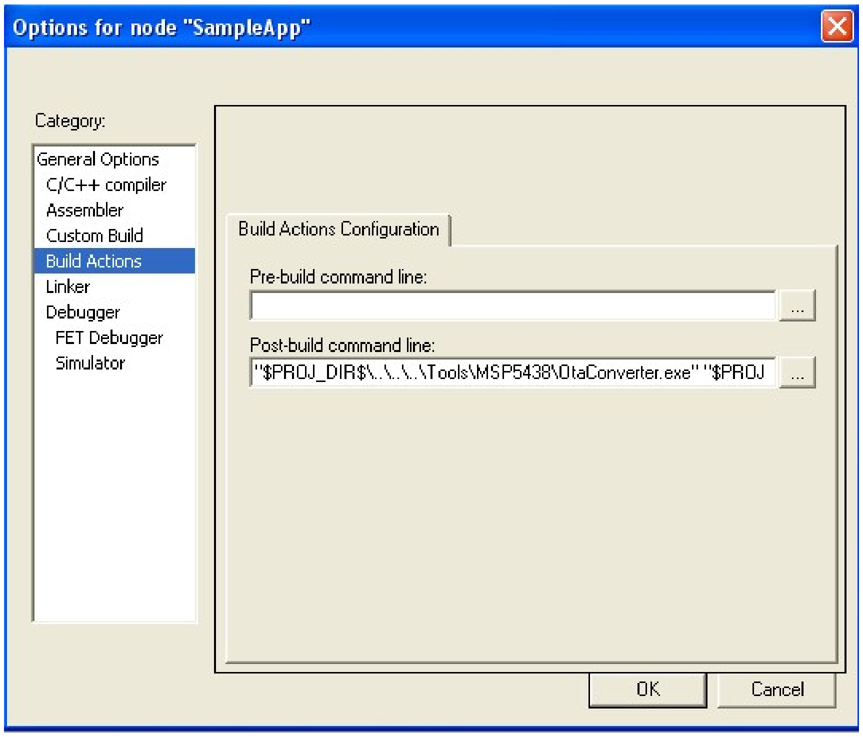
The output from the build will be located in:

Projects\zstack\SE\SampleApp\<platform>\In Premise Display - OTA\SampleApp.sim

3.4 Converting the image

Take the following steps to modify a project to automatically convert an image:
1) Open the IPD – OTA application (SampleApp.eww) project in IAR Embedded Workbench.
2) Modify the post build rule to convert the image
   a. Click Project>Options… from the menu
   b. Select Build Actions from the Category
   c. Put a Post-build command line to convert the image.
      i. For the IPD-OTA sample app, use the following:

ii. " PROJ_DIR$\..\..\..\Tools\MSP5438\OtaConverter.exe" "$PROJ_DIR$\In Premise Display - OTA\Exe\SampleApp.sim" -o"$PROJ_DIR$\In Premise Display - OTA\Exe" -t0x1234 -m0x5678 -v0xabcd1234 -pEXP5438



Pre-build command line:

Post-build command line:
"$PROJ_DIR$\..\..\..\Tools\MSP5438\OtaConverter.exe" "$PROJ

3.5 Performing an Image Update

Take the following steps to perform an image update:

1) Create an image folder
    a. Create a new folder in a location of your choosing, for example c:\images. This folder will be called the *image folder*.
    b. Copy the converted image created in Section 3.4 into the image folder.
2) Open the OtaServer.exe
    a. OtaServer.exe is located in \Tools\OTA\Console
3) Set the COM port the port attached to the *Dongle* and click *Connect*.
4) Press the browse button: ...
5) Browse to the image folder created in step 1. For example:
    a. c:\images
6) The Dongle is the coordinator and must form a network.
    a. Power cycle the device labeled *Dongle*.
    b. The yellow LED will stop blinking and the red LED will light.
    c. The PAN identifier for the network will appear in the Network Manager on the OTA Console.
7) The device labeled *Client* is an end device. Have the *Client* join the network.
    a. Power cycle the Client.
    b. The yellow LED will be blinking indicating the *Client* is in a hold state
    c. Press the Joystick to the up on *Client*.

    d. The yellow LED will stop blinking and the red LED will light.
    e. The short address for the *Client* will appear in the Network Manager on the OTA Console.

8) Right click on the newly added node in the OTA Console
9) Click *Image Notify*
10) Wait for the download to complete.

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
6) Add the preamble to the application

4.1 Adding OTA Client Source Code

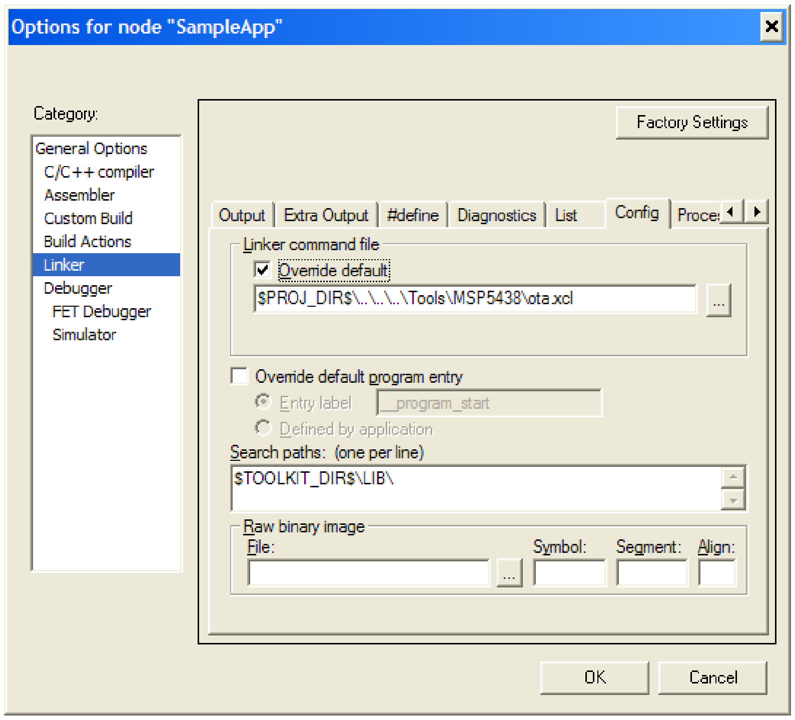
To add the OTA source code to an application's project, perform the following:

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. Set the Linker command file to: *$PROJ_DIR$\..\..\..\Tools\<platform>\ota.xcl*



Linker command file
[✓] Override default
$PROJ_DIR$\..\..\..\Tools\MSP5438\ota.xcl

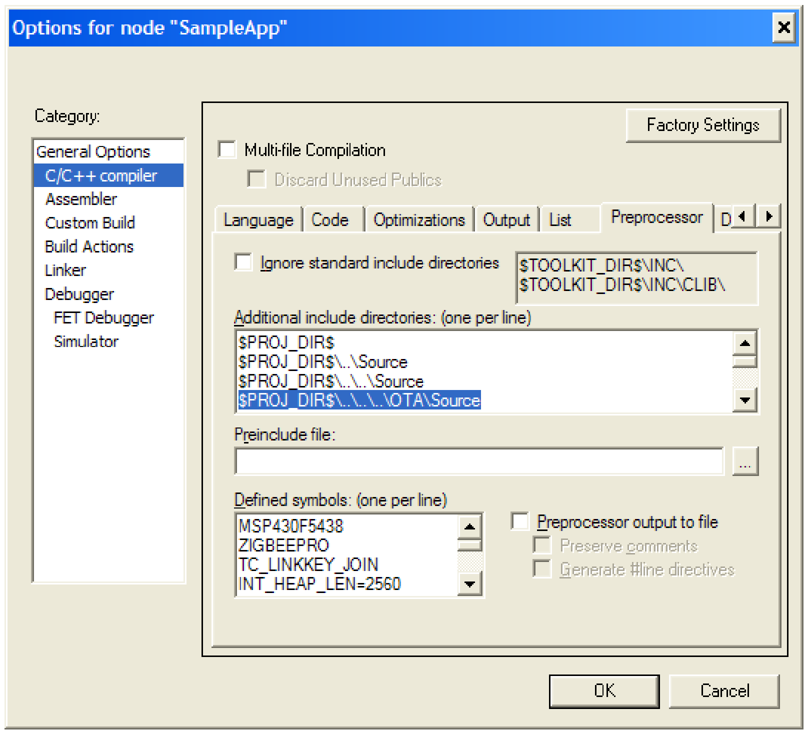
[ ] Override default program entry
Entry label: __program_start
Defined by application

Search paths: (one per line)
$TOOLKIT_DIR$\LIB\

Raw binary image
File: Symbol: Segment: Align:

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*.



```
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
```

Additional include directories: (one per line)

```
$PROJ_DIR$
$PROJ_DIR$\..\Source
$PROJ_DIR$\..\..\Source
$PROJ_DIR$\..\..\..\OTA\Source
```

Defined symbols: (one per line)

```
MSP430F5438
ZIGBEEPRO
TC_LINKKEY_JOIN
INT_HEAP_LEN=2560
```

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.

4.6 Adding the OTA Preamble to the Application

1) Open the project workspace in IAR Embedded Workbench
2) Open the application task C file (typically named <app>.c)
3) Copy and paste the following global variables into the header of the application C file:

```
#pragma location="CRC"
const CODE otaCrc_t OTA_CRC =
{
  0xFFFF,         // CRC
  0xFFFF,         // CRC Shadow
};
#pragma required=OTA_CRC

#pragma location="PREAMBLE"
const CODE preamble_t OTA_Preamble =
{
  0xFFFFFFFF,            // Program Length
  OTA_MANUFACTURER_ID,   // Manufacturer ID
  OTA_TYPE_ID,           // Image Type
  0x00000003             // Image Version
};
 #pragma required=OTA_Preamble
```

4) Modify the image version field of the OTA_Preamble as desired.