Application Note

ZigBee 3.0 Light Bulbs

This Application Note provides example applications for light bulbs in a ZigBee 3.0 network that employs the NXP KW41Z wireless microcontrollers. An example application can be employed as:

- **A demonstration using the supplied pre-built binaries that can be run on FRDM-KW41Z boards**
- **A starting point for custom application development using the supplied C source files and associated project files**

The light bulbs described in this Application Note are based on ZigBee device types from the ZigBee Lighting & Occupancy (ZLO) Device Specification.

The ZigBee 3.0 nodes of this Application Note can be used in conjunction with nodes of other ZigBee 3.0 Application Notes, available from the NXP web site.

1 Introduction

A ZigBee 3.0 wireless network comprises a number of ZigBee software devices that are implemented on hardware platforms to form nodes. This Application Note is concerned with implementing the device types for light bulbs on the Kinetis KW41Z platforms.

This Application Note provides example implementations of light bulbs that use one of the following device types from the ZigBee Lighting & Occupancy (ZLO) Device Specification:

- Dimmable Light
- Extended Colour Light
- Colour Temperature Light

The above device types are detailed in the *ZigBee 3.0 Devices User Guide* and the clusters used by the devices are detailed in the *ZigBee Cluster Library User Guide*.

Note: If you are not familiar with ZigBee 3.0, you are advised to refer the *ZigBee 3.0 Stack User Guide* for a general introduction.

2 Development Environment

2.1 Software

To use this Application Note, you need to install the following software:

- IAR Embedded Workbench, version 7.80 or above.
- MCUXpresso, version 10.1.1
- KW41Z_ZigBee_3.0_Software_v.6.0.6

2.2 Hardware

Hardware boards are available from NXP to support the development of ZigBee 3.0 applications. The following board is recommended for running these applications:

- NXP FRDM-KW41Z Evaluation Board

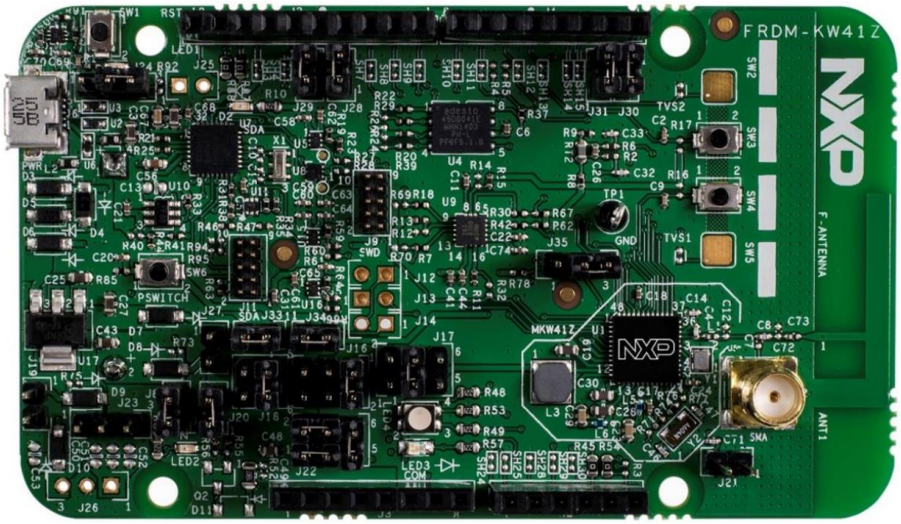


Figure 1. NXP FRDM-KW41Z Evaluation Board

3 Application Note Overview

The example applications provided in this Application Note are listed in the table below with the lighting device types that they support (for device type descriptions, refer to Section 4).

Application Name	Device Type
App_DimmableLight	Dimmable Light
App_ExtendedColorLight	Extended Colour Light
App_ColorTemperatureLight	Colour Temperature Light

Table 1: Example Applications

For each application, source files and pre-built binary files are provided in the software package. The pre-built binaries can be downloaded and run on FRDM-KW41Z board.

- To load the pre-built binaries into the evaluation board and run the demonstration application, refer to Section 4.
- To start developing you own applications based on the supplied source files, refer to Section 7.

3.1 Compatibility

The software provided with this Application Note has been tested with the following evaluation kits and SDK versions.

Product Type	Revision/Version	Supported Chips
FRDM-KW41Z Evaluation Board	Rev. A/A2/A3	KW41Z
IAR Embedded Workbench	v.7.80.4	KW41Z
MCUXpresso	v.10.1.1	KW41Z
KW41Z_Zigbee_3.0_Software_v6.0.6	v.6.0.6	KW41Z

Table 2: Compatibility Information

4 Supported Device Types

As indicated in Section 3, the supported ZLO device types in this Application Note are:

- Dimmable Light
- Extended Colour Light
- Colour Temperature Light

The above devices types are introduced in Sections 4.1, 4.2 and 4.3 respectively.

These lighting devices types must be paired for operation with switch/controller device types. Example applications for the paired device types are provided in the Application Note *ZigBee 3.0 Controller and Switch*. Two device types can be paired if they support the same cluster (Colour Control, Level Control or On/Off cluster) such that the cluster client on the switch/controller device type can access/control attributes of the cluster server on the lighting device type.

The table below lists the lighting device types (as well as the ZigBee Base Device) and, for each device type, indicates which types of cluster attributes can potentially be written/read.

Device Type	Attribute Types					
	OnOff	**Level**	**X & Y Colour**	**Hue & Saturation**	**Colour Temperature**	**Colour Loop**
Dimmable Light	Yes	Yes	No	No	No	No
Extended Colour Light	Yes	Yes	Yes	Yes	Yes	Yes
Colour Temperature Light	Yes	Yes	No	No	Yes	No
Base Device Router	Yes	No	No	No	No	No

Table 3: Lighting Device Types and Accessible Attributes

The table below lists the switch/controller device types (as well as the ZigBee Base Device) and, for each device type, indicates which types of cluster attributes can potentially be controlled (on the lighting device).

Device Type	Attribute Types					
	OnOff	**Level**	**X & Y Colour**	**Hue & Saturation**	**Colour Temperature**	**Colour Loop**
Dimmer Switch	Yes	Yes	No	No	No	No
Colour Scene Controller	Yes	Yes	Yes	Yes	Yes	Yes
Control Bridge	Yes	Yes	Yes	Yes	Yes	Yes
Base Device Coordinator	Yes	No	No	No	No	No
Base Device End Device	Yes	No	No	No	No	No

Table 4: Switch/Controller Device Types and Controllable Attributes

The ZLO device types used in this Application Note are outlined below.

4.1 ZLO Dimmable Light

The Dimmable Light is a lighting device that can be switched on or off and the brightness of the light output varied. The device can be controlled by a bound controller device such as a dimmer switch. The complete lists of supported clusters, attributes and commands can be found in the ZigBee Lighting & Occupancy Devices Specification.

4.2 ZLO Extended Colour Light

The Extended Colour Light is a lighting device that can be switched on or off and the brightness of the light output varied. The colour of the light can also be adjusted via the colour commands. The full range of colour control is supported by the Extended Colour Light, XY, Hue and Saturation, Extended Hue and Saturation, Colour Temperature and Colour Loop commands. The device can be controlled by a bound controller device such as a colour controller. The complete lists of supported clusters, attributes and commands can be found in the ZigBee Lighting & Occupancy Devices Specification.

4.3 ZLO Colour Temperature Light

The Colour Temperature Light is a lighting device that can be switched on or off and the brightness of the light output varied. The colour of the light can also be adjusted via the colour temperature commands. The device can be controlled by a bound controller device such as a Colour Controller. The complete lists of supported clusters, attributes and commands can be found in the ZigBee Lighting & Occupancy Devices Specification.

5 Running the Demonstration Application

This section describes how to use the supplied pre-built binaries to run the example applications on components of the FRDM-KW41Z board.

5.1 Loading the Applications

The table below lists the application binary files supplied with this Application Note and indicates the hardware with which the binaries can be used. These files are located in the **../tools/wireless/binaries** directories for the relevant applications.

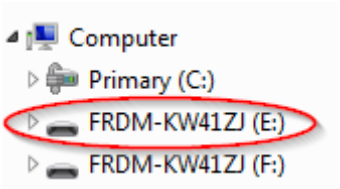
Application	Binary File	Expansion Board
App_DimmableLight	**dimmable_light_frdmkw41z.bin**	FRDM-KW41Z
App_ExtendedColorLight	**extended_color_light_frdmkw41z.bin**	FRDM-KW41Z
App_ColorTemperatureLight	**color_temperature_light_frdmkw41z.bin**	FRDM-KW41Z

Table 5: Application Binaries and Hardware Components

A binary file can be loaded into the Flash memory of a KW41Z device using the FRDM-KW41Z on-board bootloader (hosted by the OpenSDA chip) or using Test Tool 12 software, available via the NXP web site.

To load an application binary file into the KW41Z Flash memory using the existing on-board bootloader, follow the instructions below:

1. Connect a USB port of your PC to the micro-USB port on the FRDM-KW41Z board. At this point, you may be prompted to install the necessary drivers.
2. After driver installation completes, a new drive letter will be assigned to the FRDM-KW41Z bootloader. This can be checked using Windows Explorer.



3. Using drag-n-drop, copy the binary file to this new drive.
4. Once the download has successfully completed, disconnect and reconnect the USB cable.

Operating instructions for the different applications are provided in the sections below.

5.2 Commissioning the Network

Before one of the application devices can be used, it must first be commissioned into a network. This is a two-stage process:

1. First the device forms or joins a network, exchanging network parameters and security keys etc. This is called ‘Network Steering for a device not on a network’.
2. Then the process of service discovery is performed, in which controller type devices are bound to the light type devices they are to control. This is called ‘Finding and Binding’.

There is a further part to Network Steering for a device that is part of the network. This process ‘opens’ the network for new joiners for 180 seconds. This is initiated by one of the nodes in the network prior to the new device attempting to join the network.

The devices in this Application Note can be commissioned into either a Centralised Trust Centre network or a Distributed network (a network without a Trust Centre).

- In a Centralised network, the Trust Centre is responsible for deciding which devices are allowed onto the network and for provisioning the permitted devices with the appropriate network keys and application-level link keys. This decision is made by the Trust Centre according to its local security policies.
- In a Distributed network, there is no Trust Centre to manage which devices are allowed onto the network. The network key is passed from the parent node to the child node encrypted with the distribute link key.

Commissioning into a network is identical for all device types in this Application Note and is described in the following sections.

For FRDM-KW41Z board the commissioning into a network is possible by the means of the existing buttons, as depicted in Figure 2.

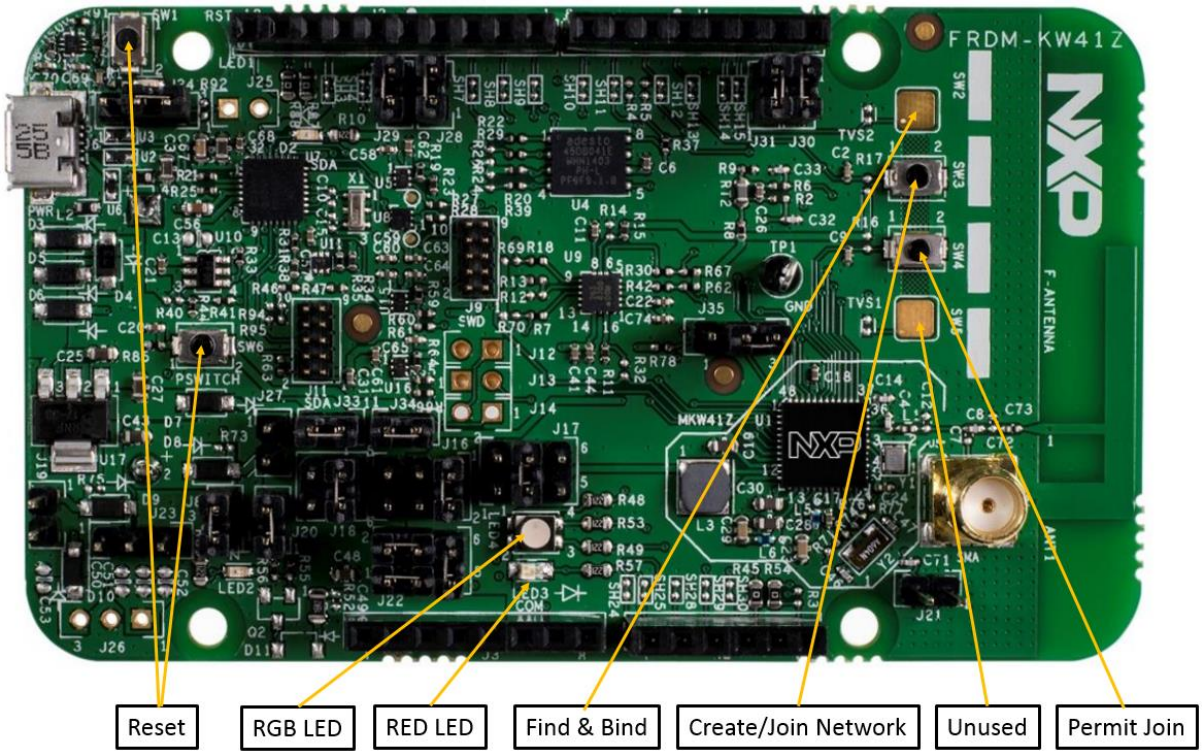


Figure 2: Using the FRDM-KW41Z buttons for commissioning the network

Note: The Reset switch from above will reset the board but will not erase the persistent data. To reset the board to factory-new state, refer to *Performing a Factory Reset* chapter.

5.2.1 Forming a Network

For the devices in this Application Note to join a network, there must first be an existing network and this network must be open for new devices to join.

A suitable Coordinator/Trust Centre can be found in either the Application Note *ZigBee 3.0 IoT Control Bridge* or the Application Note *ZigBee 3.0 Base Device Template*.

A network can be formed from a factory-new Coordinator (Network Steering while not on a network) as follows:

- Press the **SW3** button on the FRDM-KW41Z board for the BDB Coordinator.
- On a Control Bridge, configure the device as a Coordinator, set up the channel mask and then start the network using the ZGWUI PC application

The Coordinator will then start a network. Using a packet sniffer, the periodic link status messages can then be observed on the operational channel.

5.2.2 Joining a Network

1. Once there is a Coordinator running, trigger 'Network Steering for a device on a network' to open the network for joiners. This is achieved by pressing the **SW4** button on the FRDM-KW41Z board for the BDB Coordinator application or the ZGWUI PC application for the Control Bridge application.

   This will cause a Management Permit Join Request to be broadcast to the network to open the Permit Join status for a period of 180 seconds.

2. Now start the device that is to be joined to the network, i.e. the light device. To join an existing network press SW3 on the FRDM-KW41Z board. The device will perform network discovery across the primary and secondary channels, and attempt to associate with any open networks that are discovered. Once associated, the joining device will receive a network key from the Trust Centre via the parent device. After this, the joining device will attempt to update its link key – the new key will be issued by the Trust Centre and sent to the new device via the parent device. If the association or exchange of security keys fail, then the new device will resume network discovery on any un-scanned channels to look for other networks to join. If the join was successful, it is indicated by the FRDM-KW41Z board with the LED3 in solid red state.

Once this process has successfully completed, the light will be part of the network but not fully commissioned, as it has not been bound to any controlling devices. This is described in Section 5.2.3.

5.2.3 Finding and Binding

'Finding and Binding' is the process in which 'controller' type devices (e.g. switches) are bound to 'controlled' type devices (e.g. lights). Bindings are created in the Binding table of the controller device for any matching operational clusters of the controlled device. An operational cluster is a cluster that directly controls the output of a device – for example, On/Off, Level Control and Colour Control are operational clusters. Support clusters, such as Groups, Identify and Commissioning, are not operational clusters.

As controlled devices, those in this Application Note implement the Finding and Binding process as targets.

The Finding and Binding process is as follows:

1. Trigger Finding and Binding as a target on a light of this Application Note by pressing the **SW2** switch on the FRDM-KW41Z board

   This will cause the device to identify itself for 180 seconds.

2. Once all the target devices are identifying, trigger Finding and Binding as an initiator on the controller device that you wish to bind to the target controlled devices.

3. The initiator will then send out an Identify Query Request in order to use the responses to create a list of devices that are currently identifying. Each device in this list will be sent a Simple Descriptor Request to determine the supported clusters on the target device.

4. For any matching operational clusters, bindings will be created in the initiator's Binding table. Depending on the type of binding being created, an Add Group Command may be sent to the target.
5. Once a binding has been created, the initiator may send an Identify command with time of zero to the target to take it out of identify mode and stop it responding to any further Identify Query Requests.

5.2.4 Start-up Behaviour

All light devices in this Application Note are ZigBee Router devices. On start-up, if a device is factory-new, it will wait for user input to form or join a network. The red LED (LED3) will be flashing as long as device is not part of any network. After network creation or joining is initiated, the RGB LED will also flash until the forming/joining of the network is completed. When completed, the RGB LED will become solid.

If the light device is already part of a network, it just restarts using the stored network parameters and continues on the operational channel. In this case, the red LED (LED3) will be ON.

5.2.5 Performing a Factory Reset

All light devices in this Application Note can be reset to their factory-new state as follows:

- On FRDM-KW41Z board, hold down any of the buttons SW2, SW3, SW4 or SW5 for more than 5 seconds.

The factory default state is indicated by the FRDM-KW41Z board blinking LED3.

The reset maintains the value of the outgoing network frame counter across the reset. This parameter is never cleared out by a reset.

5.3 Configurations Parameters

There are several attributes that control the start-up state of a light and whether the commands that alter the operational attributes of the light are obeyed if the light is in the Off state (these attributes are new for ZigBee 3.0 and the ZLO Devices specification).

Cluster	Attribute	Function	Defined Behaviour
OnOff	0x4003	Determines the OnOff attribute at power-up.	Restore to the status from before power was removed.
Level Control	0x000F	Determines whether level can be changed if light is Off. Determines whether level is coupled to colour temperature.	Level will not change when Off. Colour temperature and level are coupled.
Level Control	0x4000	Determines the light brightness at power-up.	Restore the previous brightness.
Colour Control	0x000F	Determines whether the colour can be changed if the light is Off.	Colour cannot be changed when Off.
Colour Control	0x4010	Determines the colour temperature at start-up.	Restore the previous colour temperature.

Note: These default behaviours are defined in the **zcl options.h** file and can be changed at compile-time, if required. For full details of the

available options, refer to the ZigBee Lighting & Occupancy Devices Specification.

The attributes are writeable and persisted in non-volatile memory, so they can be changed at run-time to suit the installed application.

The options that control whether the level or colour can be changed when the light is Off allow the lights to be configured to match the behaviour of either the previous ZigBee Light Link Specification (no change if Off) or ZigBee Home Automation Specification (change if Off). In addition, all cluster commands in the Level Control and Colour Control clusters that change the level or colour now carry optional parameters that define whether the local operational mode should be over-ridden by this command.

5.4 Attribute Reporting Configurations

The ZigBee Lighting & Occupancy Devices Specification mandates that some attributes are reportable - that is, once bound to a device to receive these reports, the attributes will periodically report their status and also report changes to their status. To this end, each reportable attribute has a default configuration to determine this schedule. This default configuration can be altered by the report receiving device at run-time, as required. The report configuration is stored in persistent memory.

Cluster	Attribute	Attribute ID
OnOff	OnOff	0x0000
Level Control	Current Level	0x0000
Colour Control	Current Hue	0x0000
Colour Control	Current Saturation	0x0001
Colour Control	Current X	0x0003
Colour Control	Current Y	0x0004
Colour Control	Colour Temperature	0x0007

5.5 Using the IoT Control Bridge to control the Light devices

The following setup uses the IoT Control Bridge to exercise basic commands over the Light devices. Optionally, a ZigBee End Device (ZED) can be commissioned into the network to exercise the OnOff cluster. In this case, the ZED must be bound with the Light device using the ‘Finding and Binding’ procedure.

This setup requires at least two FRDM-KW41Z boards, one of the boards will be programmed with the Control Bridge application and one with any of Light Device applications.

First, the Control Bridge application needs to be downloaded to one of the FRDM-KW41Z boards. The Control Bridge binary file is located at the following relative path:

..\tools\wireless\binaries

On how to program the board, read the “Loading the Applications” chapter.

Then, the second FRDM-KW41Z board needs to be programmed. The recommended application is the Extended Color Light application because it exposes all the attribute types.

After both boards were programmed, connect the Control Bridge to a PC/laptop and open the ZGWUI (ZigBee Gateway User Interface) application.

The ZGWUI application is **available as a TestTool toolbar option:**

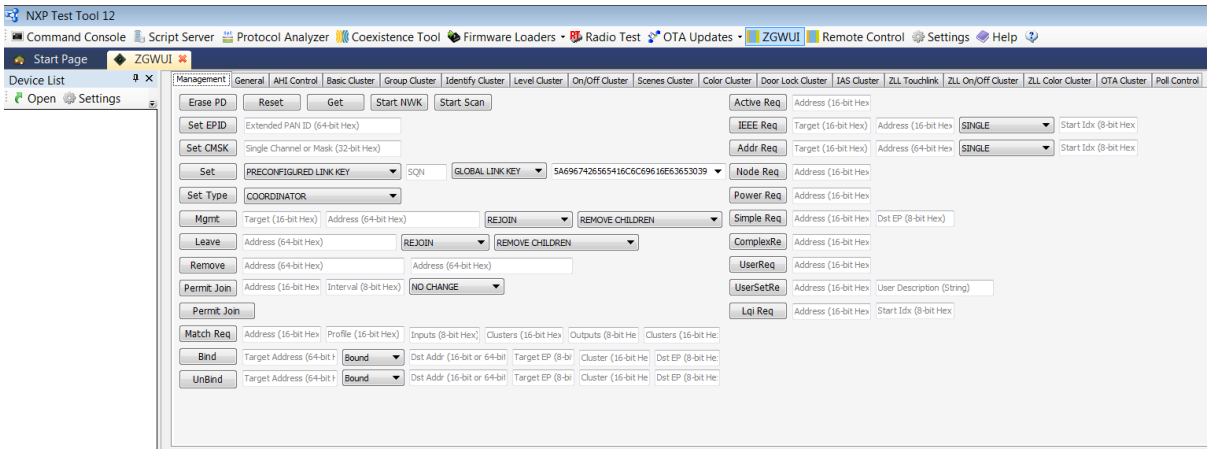
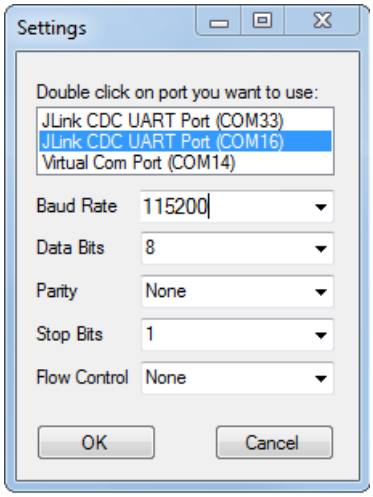


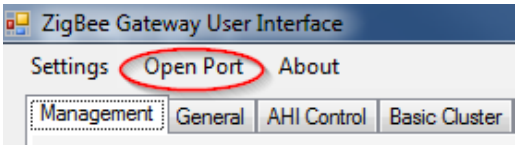
Figure 3 - ZigBee Gateway User Interface

To setup a new network, follow the below steps:

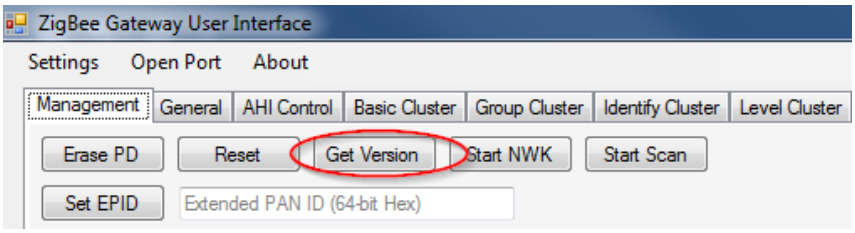
- Click on “Settings” menu item
- The “Settings” window will open
- Select the Virtual COM port assigned to FRDM-KW41Z Control Bridge



- Left the other parameters unchanged and click OK to close the window
- Click on “Open Port” menu item

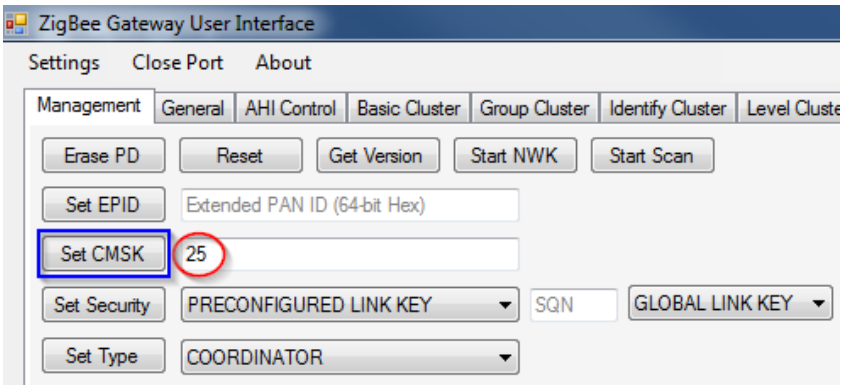


- Check the connectivity with the Control Bridge by pressing the “Get Version” button

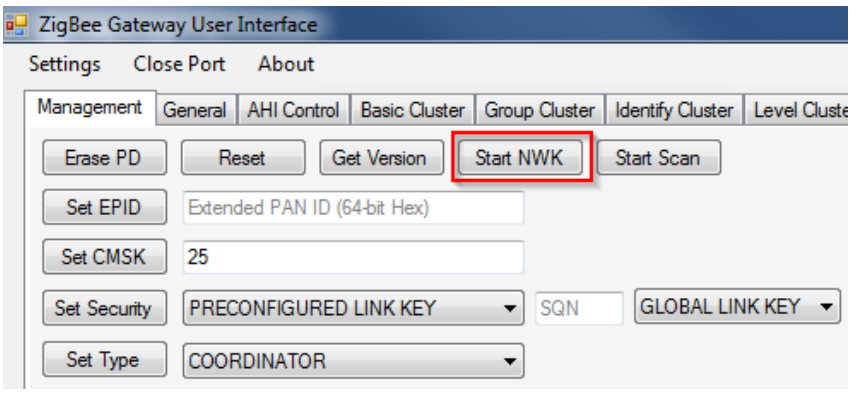


- In the Received Message View, if the communication is functional, the following message shall appear:

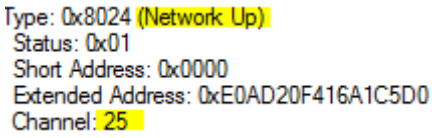
- Set the desired channel (or channel mask) for the new ZigBee 3.0 network to be created by editing the below box followed by the press of “Set CMSK”



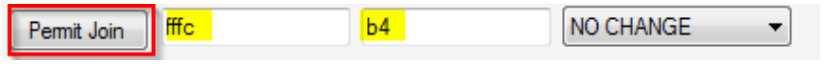
- Start the network on selected channel by pressing the “Start NWK” button



- In the Received Message View, we got the confirmation that the network is up and running on the selected channel

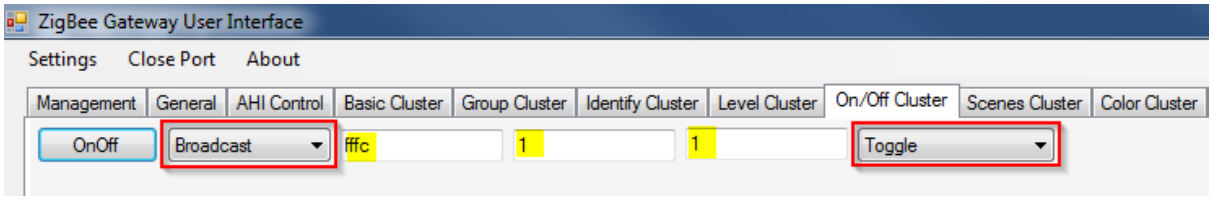


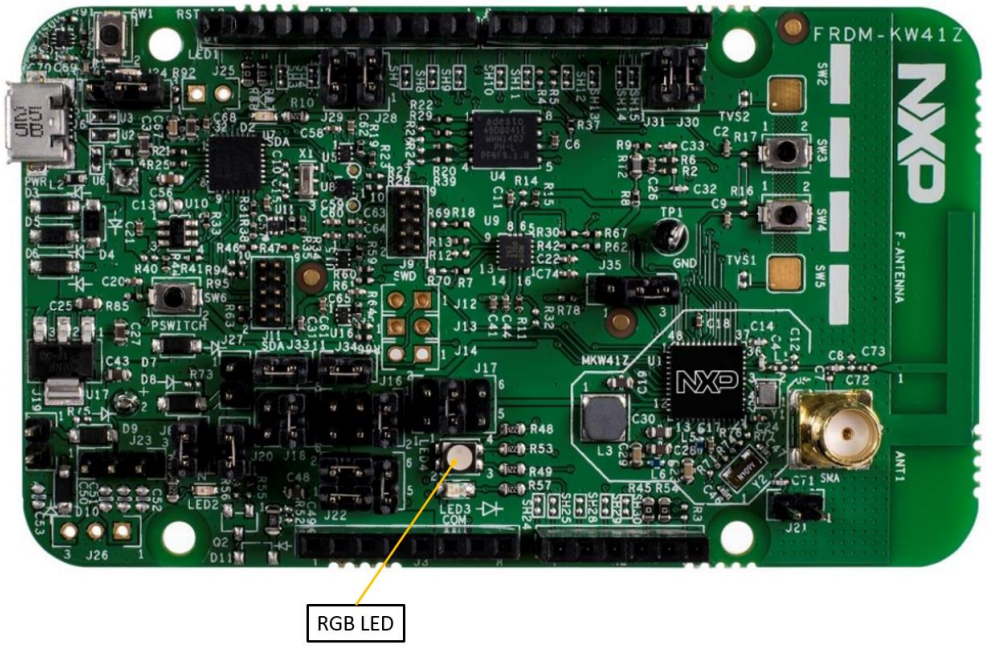
- The next step is to join the FRDM-KW41Z Extended Color Light device to the network just created. To do so, the network must be first “opened” to joiners. This can be achieved by editing the fields as below and then by pressing the “Permit Join” button.



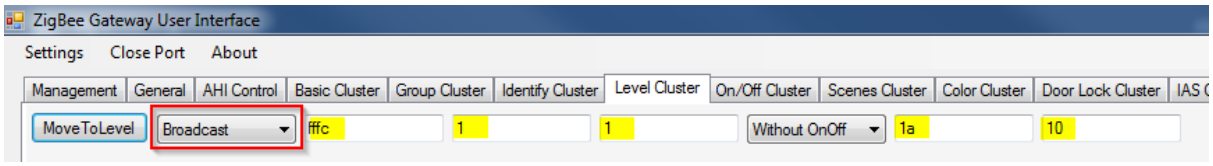
The value of 0xFFFC mask means “all joiners” and the value 0xB4 is the timeout (0xB4 = 180 decimal = 3 minutes). After this period the network will “close” to joiners until another “permit join” is issued.

- Within the specified timeout period (3 sec.), press **SW3** switch on the FRDM-KW41Z Extended Color Light device to initiate the join procedure to the existing network. When the joining procedure completes, the RGB LED (LED4) will become solid.
- At this point we have created a small ZigBee network with just two devices, the IoT Control Bridge (as Coordinator) and the Extended Color Light device (as Router). Further devices can be added to network.
- Send commands to the On/Off Cluster by selecting the On/Off Cluster tab and configure the fields as below. After the fields are configured, each press of the “OnOff” button will toggle the RGB LED on the FRDM-KW41Z Extended Color Light device.

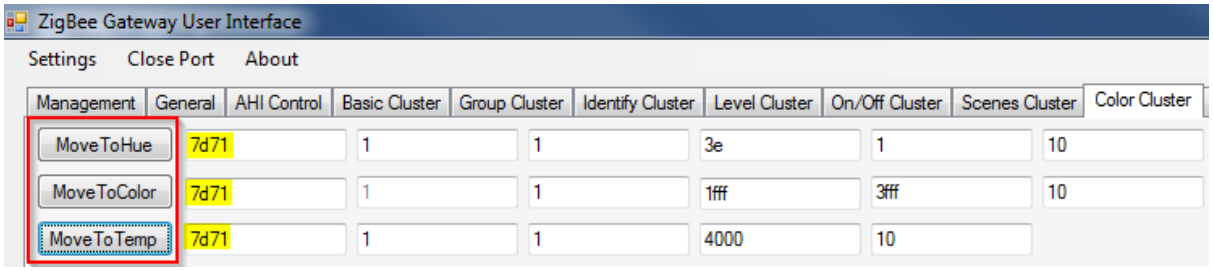




- Send commands to the Level Cluster by selecting the Level Cluster tab and configure the fields as below. After the fields are configured, press of the “MoveToLevel” button. The RGB LED on the FRDM-KW41Z Extended Color Light device will decrease or increase its level (intensity) to the specified value.



- Send commands to the Color Cluster by selecting the Color Cluster tab and configure the fields. Unlike the other Cluster commands, for Color Cluster the IEEE 802.15.4 short address should be used. This address can be retrieved from the Received Message View when the Light device is joining the network or using an IEEE 802.15.4 packet sniffer tool. The Extended Color Light supports all the commands from below: move to hue, move to color and move to temperature.



6 Over-The-Air (OTA) Upgrade

Over-The-Air (OTA) Upgrade is the method by which a new firmware image is transferred to a device that is already installed and running as part of a network. This functionality is provided by the OTA Upgrade cluster. In order to upgrade the devices in a network, two functional elements are required.

- **OTA Server:** First the network must host an OTA server, which will receive new OTA images from manufacturers, advertise the OTA image details to the network, and then deliver the new image to those devices that request it.
- **OTA Clients:** The second requirement is for OTA clients, which are located on the network devices that may need to be updated. These devices periodically interrogate the OTA server for details of the firmware images that it has available. If a client finds a suitable upgrade image on the server, it will start to request this image, storing each part as it is received. Once the full image has been received, it will be validated and the device will boot to run the new image.

New images are always pulled down by the clients, requesting each block in turn and filling in gaps. The server never pushes the images onto the network.

.

Note: Currently OTA Upgrades are supported only for Dimmable Light project. More details are available in the ZigBee 3.0 IoT Control Bridge Application Note, chapter 4.2.8.

7 ZigBee Green Power (GP) Support

This section describes the provision for the addition of ZigBee Green Power (GP) devices to the network. To support GP devices, the ZigBee devices in the network must also act as GP 'infrastructure devices' to facilitate the reception, routing and handling of GP frames from GP devices.

The GP Combo Basic support is available for Extender Colour Light, Colour Temperature Light, and Dimmable Light device types. Lights with GP Combo Basic support act as GP sink nodes that can be controlled by both GP switches and ZLO switches, after successful commissioning. These lights support the following GP commands: On, Off, Identify, Move Up with On/Off, Move Down with On/Off, Level Control/Stop. The commands are detailed the in ZigBee Green Power Specification.

ZigBee Green Power is described in the ZigBee Green Power User Guide (ZB3GPUG).

The Dimmable Light application is pre-built with GP Combo Basic support.

To enable the GP Combo Basic support for the Extended Color Light and Color Temperature Light applications described in this application note the following macros need to be defined (**config.h** or project options):

```
#define CLD_GREENPOWER
#define GP_COMBO_BASIC_DEVICE
```

7.1 Green Power Configuration Settings

The security keys to be used by the Combo Basic and Proxy Basic devices must be configured in the **zcl_options.h** file on the devices, as described below.

To set the security key type to be used, add the following line to the above file:

```
#define GP_KEYTPE <key_type>
```

where `<key_type>` is any one of the following enumerated values:

```
typedef enum
{
E_GP_NO_KEY = 0x00,
E_GP_ZIGBEE_NWK_KEY,
E_GP_ZGPD_GROUP_KEY,
E_GP_NWK_KEY_DERIVED_ZGPD_GROUP_KEY,
E_GP_OUT_OF_THE_BOX_ZGPD_KEY,
E_GP_DERIVED_INDIVIDUAL_ZGPD_KEY = 0x07
}teGP_GreenPowerSecKeyType;
```

The key will be derived from the ZigBee network key or from a key sent by the GP device, except for a group key. When the key type is set to E_GP_ZGPD_GROUP_KEY, a shared group key must be specified using the GP_SHARED_KEY macro, as follows:

```
#define GP_SHARED_KEY <key>
```

where `<key>` is the value of the key.

8 Developing with the Application Note

The example applications provided in this Application Note were developed using the KW41Z_Zigbee_3.0_Software_v.6.0.6, IAR Embedded Workbench 7.80 as Build toolchain and MCUXpresso v10.1.1.

Throughout your ZigBee 3.0 application development, you should refer to the documentation listed in Section 9.

8.1 App_DimmableLight Application Code

This section describes the application code for App_DimmableLight, which is provided in the **Source** directory for the application. You may wish to use this code as a basis for your own application development. You can rebuild your customised application as described in Section 8.5.

App_DimmableLight.c is specific to the Dimmable Light. It includes endpoint registration and constructor, reporting configuration, Basic cluster attribute initialisation, and Identify handler.

bdb_options.h defines the parameters used by the ZigBee Base Device, such as primary and secondary channel masks.

zcl_options.h defines the ZCL options, such as which clusters are supported, whether a client and or a server, and which optional commands and attributes are supported. Mandatory commands and attributes of the selected cluster will be automatically included.

8.2 App_ExtendedColorLight Application Code

This section describes the application code for App_ExtendedColorLight, which is provided in the **Source** directory for the application. You may wish to use this code as a basis for your own application development. You can rebuild your customised application as described in Section 8.5.

App_ExtendedColorLight.c is specific to the Extended Colour Light. It includes endpoint registration and constructor, reporting configuration, Basic cluster attribute initialisation, and Identify handler.

bdb_options.h defines the parameters used by the ZigBee Base Device, such as primary and secondary channel masks.

zcl_options.h defines the ZCL options, such as which clusters are supported, whether a client and or a server, and which optional commands and attributes are supported. Mandatory commands and attributes of the selected cluster will be automatically included.

8.3 App_ColorTemperatureLight Application Code

This section describes the application code for App_ColorTemperatureLight, which is provided in the **Source** directory for the application. You may wish to use this code as a

basis for your own application development. You can rebuild your customised application as described in Section 8.5.

App_ColorTemperatureLight.c is specific to the Colour Temperature Light. It includes endpoint registration and constructor, reporting configuration, Basic cluster attribute initialisation, and Identify handler.

bdb_options.h defines the parameters used by the ZigBee Base Device, such as primary and secondary channel masks.

zcl_options.h defines the ZCL options, such as which clusters are supported, whether a client and or a server, and which optional commands and attributes are supported. Mandatory commands and attributes of the selected cluster will be automatically included.

8.4 Common Code

Code common to all light device types is held in the **Common_Light\Source** directory. This code contains the main application event handlers, as well as chip and stack initialisation.

app_start_light.c manages the chip start-up, calls the initialisation functions and launches the main program loop.

app_main.c hosts the main program loop, and defines and initialises system resources, queues, timers etc.

zlo_light_node.c hosts the event handlers for the application and the ZigBee Base Device callback. This callback receives ZigBee Base Device events and AF Stack events after the Base Device has completed any processing that it requires. These events can then be further processed by the application. These events include data indications that are passed to the ZCL for processing, and network management events, such as Joined or Failed to Join events, in order to keep the application informed of the network state. The application event queue is processed to receive button-press events. Sleep scheduling and polling for data are also handled here.

zlo_zcl_light_task.c hosts the ZCL initialisation and the ZCL callback functions. The callbacks notify the application of the results of any received ZCL commands or responses, so that the application can take the appropriate action. The ZCL tick timer is used to provide a ticks for the ZCL to manage timer-dependent events or state transitions.

App.zpscfg provides the configuration data to initialise the ZigBee stack. It sets the sizes of the various stack tables, such Neighbour, Routing and Key tables.

DriverBulb is the source for various light-bulb hardware interfaces.

app_buttons.c is the driver software to read the switches on the FRDM-KW41Z board and present events to the application.

irq.s defines which of the hardware interrupts are supported, serviced and at which priority. This is defined by two tables - an interrupt priority table and a table of handler functions.

app_light_interpolation.c smooths out the updates to the LED outputs to remove flicker. The ZCL updates the light at 10Hz. This code will further divide the update into 10 steps, updating the bulb at 100Hz.

app_manage_temperature.c manages the chip for changes in temperature to maintain accuracy.

app_power_on_counter.c counts how many times the device is switched on within a short period, generates application events to trigger Network Steering or a factory-reset depending on how many.

app_reporting.c manages the setting up of the default reporting configuration, updates and saves the configuration if it is changed remotely. The actual sending of reports is managed by the ZCL itself.

app_scenes.c provides the application interface between the ZCL and Persistent Data Manager (PDM). It contains a routine to save and restore scene cluster data to the PDM. The actual handling of scenes and sending scene data to the light-bulb hardware is managed by the ZCL.

PDM_IDs.c provides unique identifiers for all persistent data records used by the PDM

8.5 Rebuilding the Applications

This section describes how to rebuild the supplied applications, which you will need to do if you customise the applications for your own use.

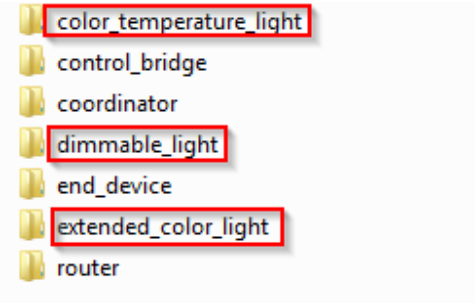
8.5.1 Pre-requisites

It is assumed that you have installed the relevant NXP development software on your PC, as detailed in Section 2.

The application projects are located at the following relative path:

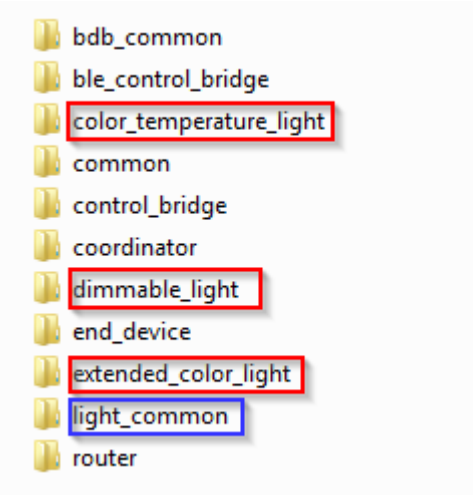
..\boards\frdmkw41z\wireless_examples\zigbee_3_0

The ZigBee light devices projects are highlighted below:



The applications code is located at the following relative path, where "x.y.z" is the software current version.

..\middleware\wireless\zigbee_3_0_x.y.z\examples



The **light_common** folder contains the code that is common to all lighting applications, including the driver bulb implementation.

Each of the applications has 2 subfolders, **Config** and **Source**.

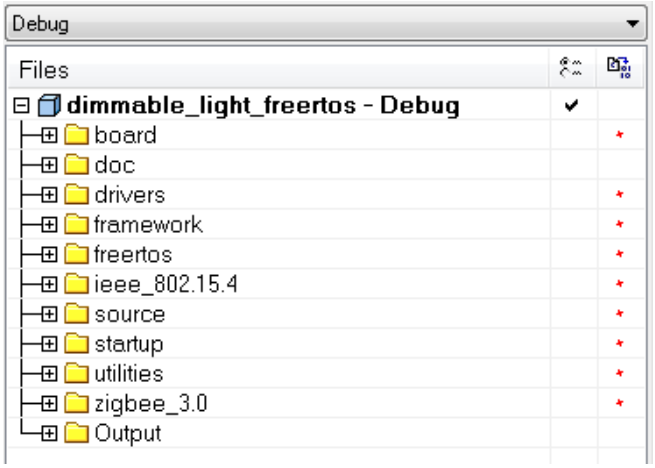
8.5.2 Build Instructions

To build any of these light applications, open the application project file in IAR Embedded Workbench IDE or MCUXpresso. In this application note, the Dimmable Light project is considered. The project workspace file is located at the following relative path:

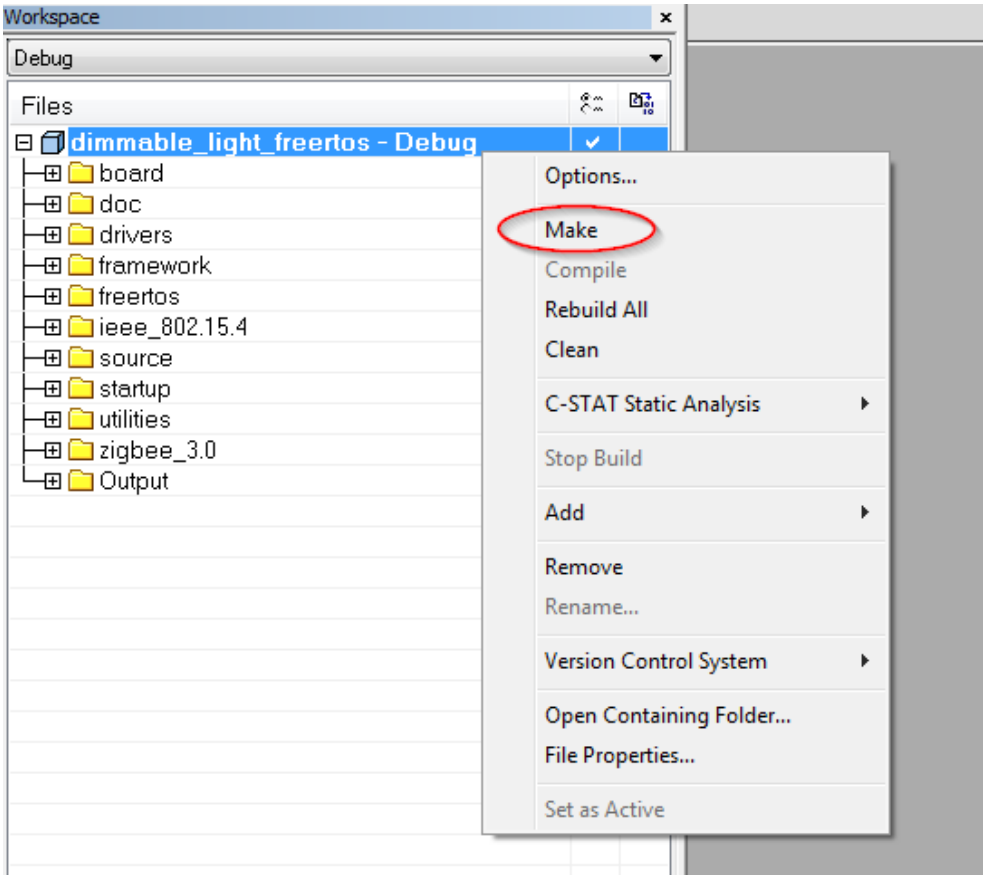
..\boards\frdmkw41z\wireless_examples\zigbee_3.0\dimmable_light\freertos\iar\dimmable_light_freertos.eww

Open the project workspace file by double click or by drag and drop into the IAR EW IDE.

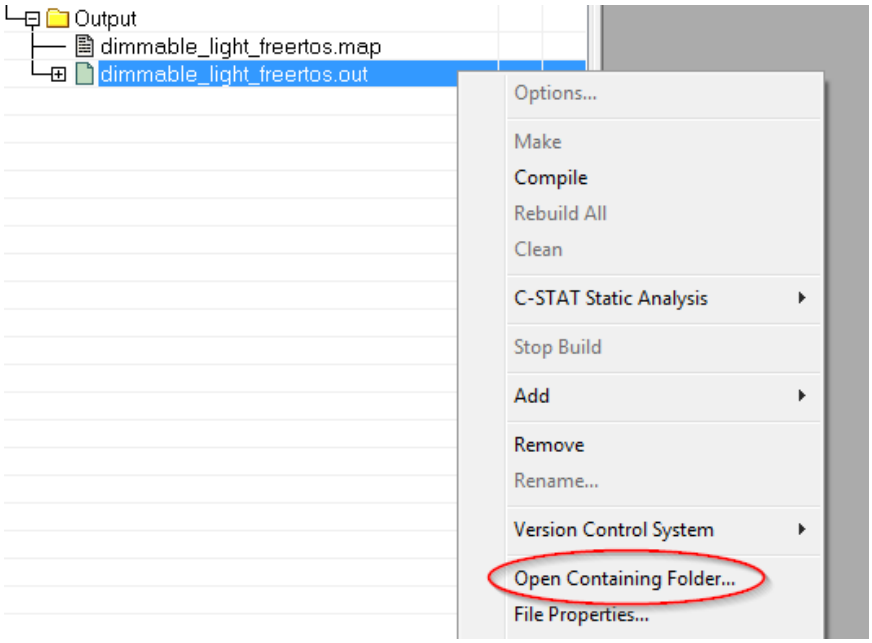
The project will be opened.



Right click on the workspace name, and then select “Make”.



After the build process completes, the output folder will contain the executable application (*.out) and the map file. The binary file is also generated, but is not visible in this view. To access it, right click on the *.out file and then click on “Open Containing Folder”



The containing folder will be opened in Windows Explorer and exactly in the same folder, the binary file is located.

In order to load a project using MCUXpresso, please refer to the documentation provided, “Getting started with MCUXpresso SDK.pdf “and follow the steps described in chapter 7 “Run a demo using MCUXpresso IDE”.

9 Application Code Sizes

The applications of this Application Note have the following memory footprints on the KW41Z device, when using the KW41Z_ZigBee_3.0_Software_v6.0.6 SDK.

Application – Release Configuration, IAR 7.80.4.	**Read-only code (Bytes)**	**Read-only data (Bytes)**	**RW data (Bytes)**
App_DimmableLight	291178	19077	35175
App_ExtendedColorLight	252742	16538	34097
App_ColorTemperatureLight	240740	16119	33292

10 Release Details

10.1 Compatibility

Product Type	Version	Supported Chips
Version 1001		
FRDM-KW41Z	Rev A/A2/A3	KW41Z
IAR Embedded Workbench	v7.80.4	KW41Z
MCUXpresso	V10.1.1	KW41Z
KW41Z_Zigbee_3.0_Software_v6.0.6	v6.0.6	KW41Z

11 Related Documents

The following manuals will be useful in developing custom applications based on this Application Note:

- ZigBee 3.0 Stack User Guide
- ZigBee Device User Guide
- ZigBee Cluster Library User Guide

All the above manuals are available as PDF documents from the ZigBee 3.0 page of the NXP web site.

Revision History

Version	Notes
0	First KW41Z release
1	Updates for KW41Z ZigBee 3.0 Alpha/EAR Release
2	Updates for KW41Z ZigBee 3.0 Beta/PRC Release
3	Updates for KW41Z ZigBee 3.0 GA/RFP Release

Important Notice

NXP Semiconductors

For the contact details of your local NXP office or distributor, refer to: