Application Note

ZigBee 3.0 Sensors

This Application Note provides example applications for sensors 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 sensors 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 sensors on the Kinetis KW41Z platform.

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

- Light Sensor
- Occupancy Sensor
- Light, Temperature & Occupancy Sensor (combination device type)

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*. The Light, Temperature & Occupancy Sensor is a combination device type based on the Light Sensor device type with the addition of the Temperature Measurement cluster and Occupancy Sensing cluster.

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.

The software and documentation resources referenced in this Application Note are available free-of-charge via the ZigBee 3.0 page of the NXP web site.

2 Development Environment

2.1 Software

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

- IAR Embedded Workbench, version 8.32.1 or above, or MCUXpresso, version 10.2.1
- KW41Z_ZigBee_3.0_Software_v.6.0.8

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

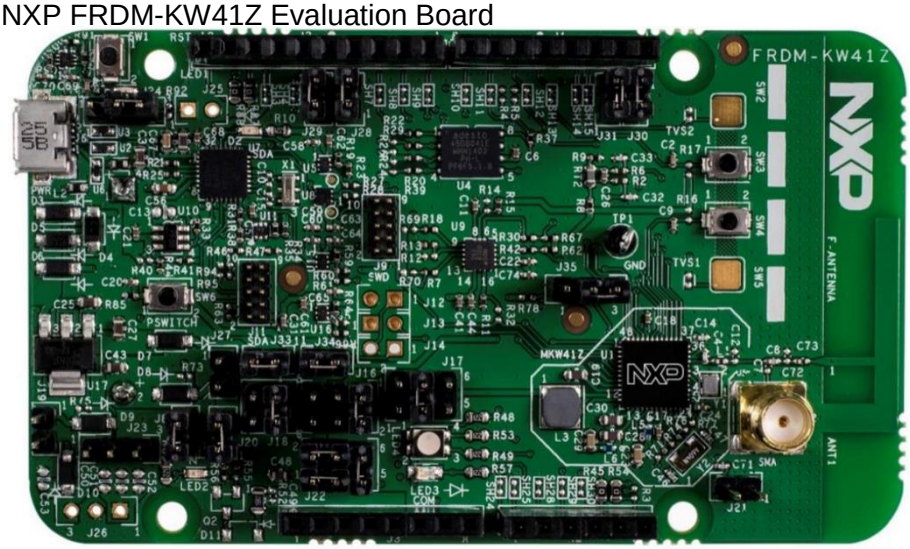


Figure 1. NXP FRDM – KW41Z Evaluation Board

3 Application Note Overview

The example applications provided in this Application Note are listed in the following table. For each application, the table indicates the required device type as well as the device type (on another node) with which it can be paired for operation.

Application	Device Type	Paired Device Type
App_LightSensor	Light Sensor	Control Bridge
App_OccupancySensor	Occupancy Sensor	Control Bridge
App_LightTemperatureOccupancySensor	Light, Temperature & Occupancy Sensor *	Control Bridge

Table 1: Example Applications and Device Types

* Light Sensor device type with addition of Temperature Measurement and Occupancy Sensing clusters

The Control Bridge is described in in the Application Note *ZigBee Control Bridge.*

For each application, source files and pre-built binary files are provided in the Application Note ZIP package. The pre-built binaries can be run on components of the FRDM - KW41Z board

- To load the pre-built binaries into the evaluation kit components and run the demonstration application, refer to Section 4.
- To start developing you own applications based on the supplied source files, refer to Section 5.

Note 1: By default, all sensors have the low power functionality enabled: the *cPWR_UsePowerDownMode* define is set to 1 in *config.h* file.

Note 2: On the FRDM-KW41Z board only the buttons SW3 and SW4 will wake the device from low power for 3 seconds. If the device is in low power, the events associated with the respective buttons will not be processed on the first button press.

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	8.32.1	KW41Z
MCUXpresso	10.2.1	KW41Z
KW41Z_ZigBee_3.0_Software_v6.0.8	v6.0.8	KW41Z

Table 2: Compatiblity Information

4 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.

4.1 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.

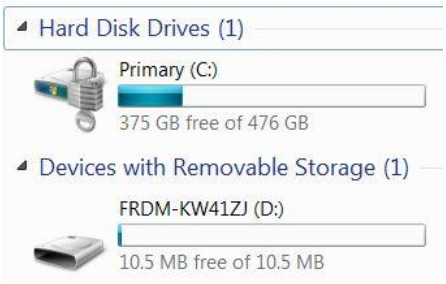
Application	Binary File	Expansion Board
App_LightSensor	**light_sensor_frdmkw41z.bin**	FRDM_KW41Z
App_LightTemperature OccupancySensor	**lto_sensor_frdmkw41z.bin**	FRDM_KW41Z
App_OccupancySensor	**occupancy_sensor_frdmkw41z.bin**	FRDM_KW41Z

Table 3: 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 on to a 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 the installation completes, a new drive letter will be assigned to the KW41Z bootloader. This can be checked using windows explorer



3. Using drag and 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.

4.2 Using the LightSensor Application

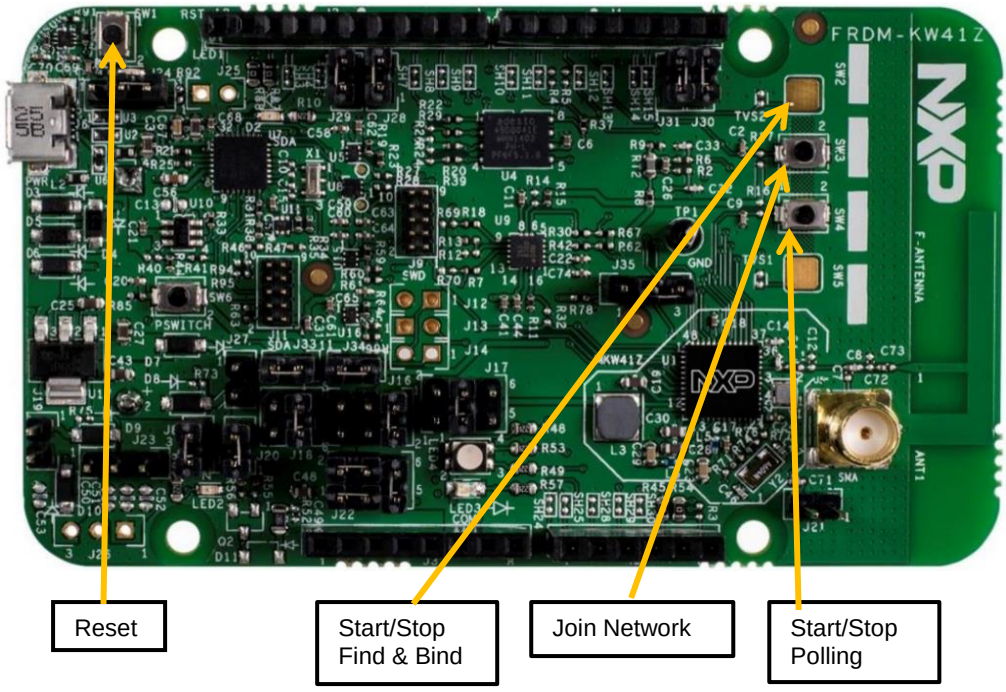
This section describes how to commission and operate the Coordinator application in a ZigBee3.0 network. To use this application, you must have programmed the application binary into the FRDM-KW41Z board.

4.2.1 Light Sensor Device Functionality

The Light Sensor can be used to provide regular illuminance measurements (from the Illuminance Measurement cluster) to a control application that adjusts the level of light emitted by light nodes – these lights can be any ZLO Lighting devices that support the Level Control cluster.

> **Note:** The FRDM-KW41Z board does not have an illuminance sensor. Hence, the reports sent by the Light Sensor have sample values for the corresponding attributes, the simulated measured illuminance ranging from 10 to 50.

The functionality of the LightSensor application is described and illustrated below.



4.2.2 Clearing Context Data on the Device

When loading the application for the first time, any persistent context data must be cleared in the device.

The context data can be cleared by holding down any of the SW2, SW3, SW4 or SW5 buttons for more than 8 seconds.

After the reset, the outgoing network frame counter value is kept.

4.2.3 Commissioning

The commissioning operations for this device, such as forming/joining a network and binding, are the same as for the other device types described in this Application Note and are detailed in Section 4.5. To incorporate the device into a ZigBee 3.0 network, work through Section 4.5 and then return to this point for device operation.

4.2.4 Operation

There are two modes of operation of the Light Sensor, as follows:

- **Normal Mode:** The Light Sensor sleeps and wakes up every 61 seconds (`ZLO_MAX_REPORT_INTERVAL`), when it obtains a new light simulated reading, updates the `u16MeasureValue` attribute with the new reading, if necessary. The sensor will send out a periodic report after 1 second if there are any changes in the attribute or every 61 seconds if there is no update. This is done to reduce the current consumption, which increases the battery life.
- **Keep-alive Mode:** The Light Sensor is permanently active and polling its parent every second.

  The light sensor will be reading the value of the driver at the rate of once every `LIGHT_SENSOR_SAMPLING_TIME_IN_SECONDS` (5 seconds) and updating its `u16MeasuredValue` attribute accordingly. Thus, the changes in light sensor reading should be done only at intervals of 5 seconds.

  If there is any change in the `u16MeasuredValue` attribute, the sensor will send out a report after 1 second (`ZLO_MIN_REPORT_INTERVAL`), otherwise every 61 seconds (`ZLO_MAX_REPORT_INTERVAL`).

The Light Sensor can but put into ‘keep-alive’ mode by pressing the **SW4** button on the board (an LED will start to blink), after it has been woken-up, if in low power, by first pressing **SW3** or **SW4** button.

To return to normal mode from ‘keep-alive’ mode, press the **SW4** button again or reset the device.

Note 1: On the FRDM-KW41Z board only the buttons SW3 and SW4 will wake the device from low power for 3 seconds. If the device is in low power, the events associated with the respective buttons will not be processed on the first button press and there will be needed two button presses - one for waking up the device and one for triggering the desired event.

Note 2: As a sleepy End Device, the Light Sensor goes through a sleep/wake cycle. If an attribute change occurs just before the device

enters sleep mode, the attribute report will be sent out on the next wake-up and there will therefore be a delay.

Note 3: If the Light Sensor is to report frequently, it is recommended that the bound transmission management feature is enabled in the ZCL by defining CLD_BIND_SERVER in the **zcl_options.h** file. This feature and the required resources are described in the *ZigBee Cluster Library User Guide*.

4.2.4.1 Light Sensor

The Light sensor is simulated using an array of values, and the application is only cycling trough these values.

4.2.5 Sensing Clusters

The Light Sensor uses the Illuminance Measurement cluster to hold its results. The Illuminance Measurement cluster contains the “Measured Value” attribute which is used to store the light level measured by the sensor. Binding to this cluster (Id **0x0400**) will enable a remote device to receive reports when the value of this attribute on the sensor changes.

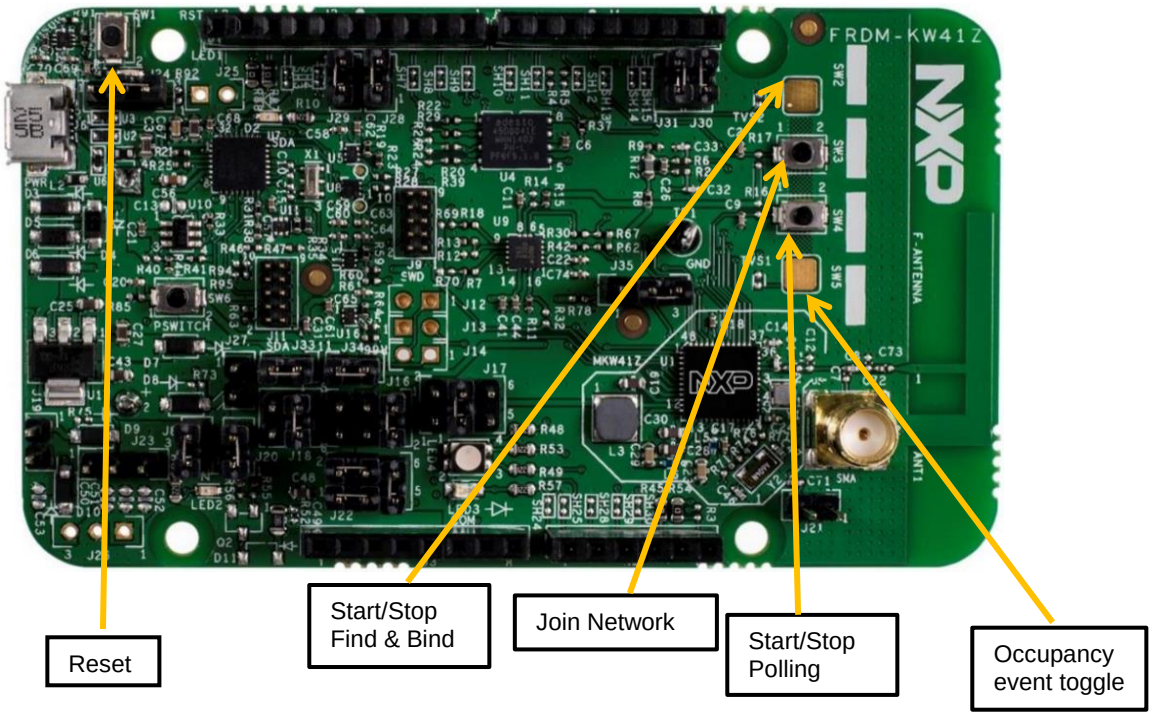
4.3 Using the OccupancySensor Application

This section describes how to commission and operate the OccupancySensor application in a ZigBee 3.0 network. To use this application, you must have programmed the application binary on to the KW41Z board, as described in Section 4.1:

- **occupancy_sensor_frdmkw41z.bin**

4.3.1 Occupancy Sensor Device Functionality

The functionality of the OccupancySensor application is described and illustrated below.



4.3.2 Clearing Context Data on the Device

When loading the application for the first time, any persistent context data must be cleared in the device.

The context data can be cleared by holding down the SW2, SW3, SW4 or SW5 buttons for more than 8 seconds.

The outgoing network frames counter value is kept even through a reset.

4.3.3 Commissioning

The commissioning operations for this device, such as forming/joining a network and binding, are the same as for the other device types described in this Application Note and are detailed in Section 4.5. To incorporate the device into a ZigBee 3.0 network, work through Section 4.5 and then return to this point for device operation.

4.3.4 Operation

Occupancy events are simulated by pressing a button on the board - button **SW5** is assigned as the output of a virtual PIR detector. There are two types of virtual occupancy sensor: Open Collector and PWM. Only the Open Collector sensor type is implemented in this demo.

Note that for both types of sensor, LED3 on the Board is used as an indicator of the state of the `u8Occupancy` attribute of the Occupancy Sensing cluster:

- If the `u8Occupancy` attribute is 0 (i.e. Unoccupied), the light is OFF
- If the `u8Occupancy` attribute is 1 (i.e. Occupied), the light is ON

Open Collector Sensor

An Open Collector sensor outputs a constant digital high/low signal for occupancy. In this demonstration, 'occupied' and 'unoccupied' states are toggled by pressing a button.

Simulating Unoccupied to Occupied Event:

To move the sensor from the unoccupied to occupied state, press the **SW3** or **SW4** button to wake the device from low power. Then within 3 seconds press the button **SW5** on the Board.

Simulating Occupied to Unoccupied Event:

Once in the occupied state, to simulate an 'unoccupied' event, press the **SW3** or **SW4** button to wake the device from low power, then press again the button **SW5**. If no further occupancy event is simulated by pressing the **SW5** button within a certain timeout period (180 seconds, by default), the sensor will automatically move to the unoccupied state.

Note 1: On the FRDM-KW41Z board only the buttons SW3 and SW4 will wake the device from low power for 3 seconds. If the device is in low power, the events associated with the respective buttons will not be processed on the first button press and there will be needed two button presses – one for waking up the device and one for triggering the desired event.

Note 2: The timeout period can be customised using the macro `APP_OCCUPANCY_SENSOR_OCCUPIED_TO_UNOCCUPIED_DELAY`.

Once in the occupied state, the sensor can be kept in the occupied state with a single button-press - a single transition of **SW5** will reset the timer that keeps track of the timeout defined by `APP_OCCUPANCY_SENSOR_OCCUPIED_TO_UNOCCUPIED_DELAY`. This feature is provided to simulate maintaining the occupied state.

Additional Sensor Functionality

The Occupancy Sensor will report its attributes if it is bound to at least one device, the occupancy state is 'occupied' and one of the following actions occurs:

- It has joined a new network
- It has rejoined the network
- It has exited 'Finding and Binding'

The Occupancy Sensor will start a timer for the number of seconds defined by the macro `APP_OCCUPANCY_SENSOR_OCCUPIED_TO_UNOCCUPIED_DELAY` if the occupancy state is 'unoccupied' and one of the following actions occurs:

- It has joined a new network
- It has rejoined the network
- It has exited 'Finding and Binding'

Attribute Reporting

Attribute Reporting can be configured optionally by defining the macro `ZLO_SYSTEM_MIN_REPORT_INTERVAL` to be greater than zero. If defined to be greater than zero, the Occupancy Sensor reports the Occupancy attribute value to the light (to which it is bound) after 1 second (or `ZLO_SYSTEM_MIN_REPORT_INTERVAL`) if there is any change in the attribute or, otherwise, every 60 seconds (or `ZLO_SYSTEM_MAX_REPORT_INTERVAL`). On receiving the attribute report:

- If the `u8Occupancy` attribute is 0 (i.e. Unoccupied), the light will switch OFF
- If the `u8Occupancy` attribute is 1 (i.e. Occupied), the light will switch ON

4.3.5 Sensing Clusters

The occupancy sensing cluster contains the `u8Occupancy` attribute which is used to indicate the occupied/unoccupied status of the sensor. Binding to this cluster (Id **0x0406**) will enable a remote device to receive reports when the value of the `u8Occupancy` attribute on the sensor changes.

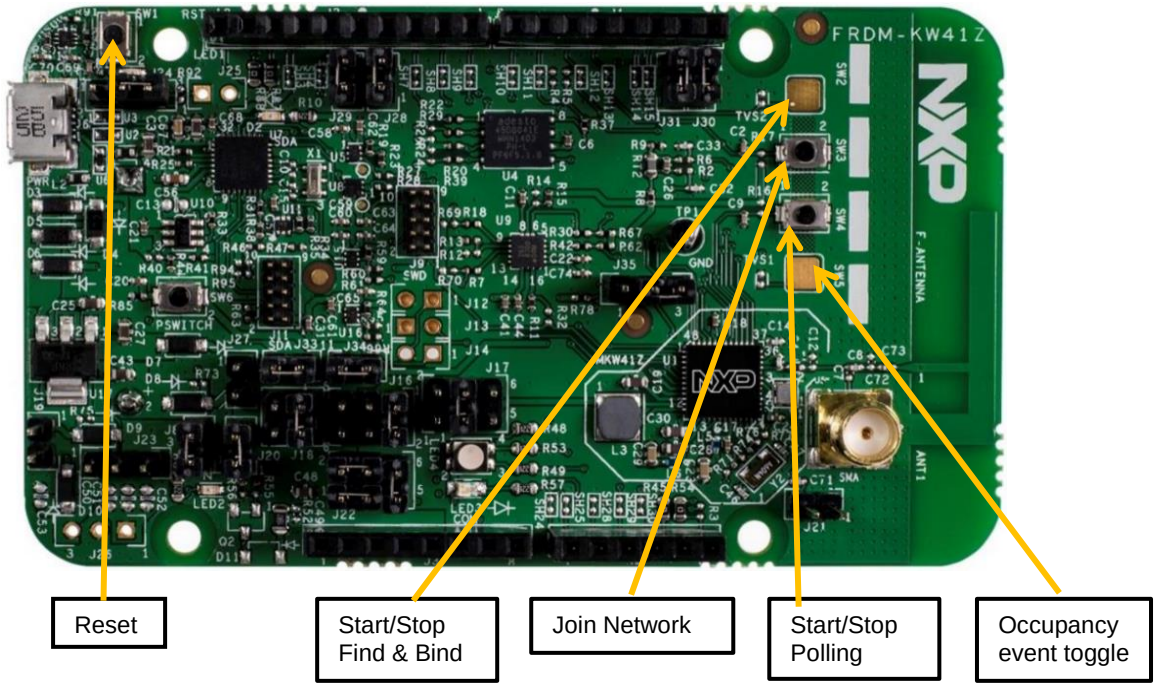
4.4 Using the LightTemperatureOccupancySensor Application

This section describes how to commission and operate the LightTemperatureOccupancySensor application in a ZigBee 3.0 network. To use this application, you must have programmed the relevant application binary into the FRDM-KW41Z board, as described in Section 4.1:

- **lto_sensor_frdmkw41z.bin**

4.4.1 Light, Temperature & Occupancy Sensor Device Functionality

The functionality of the LightTemperatureOccupancySensor application is described and illustrated below.



4.4.2 Clearing Context Data on the Device

When loading the application for the first time, any persistent context data must be cleared in the device.

The context data can be cleared by holding down any of the SW2, SW3, SW4 or SW5 buttons for more than 8 seconds.

After resetting the board, the outgoing network frames counter maintains its previous value.

4.4.3 Commissioning

The commissioning operations for this device, such as forming/joining a network and binding, are the same as for the other device types described in this Application Note and are detailed in Section 4.5. To incorporate the device into a ZigBee 3.0 network, work through Section 4.5 and then return to this point for device operation.

4.4.4 Operation

There are two modes of operation of the Light, Temperature & Occupancy (LTO) Sensor, as follows:

- **Normal Mode:** The Light, Temperature & Occupancy Sensor sleeps and wakes up every 61 seconds (`ZLO_MAX_REPORT_INTERVAL`). If the light level / temperature level changes since the last simulated reading then the `u16MeasuredValue` attribute is updated and the Light, Temperature & Occupancy Sensor sends out an attribute report after 1 second. If there is no change, it will send a report every 61 seconds (`ZLO_MAX_REPORT_INTERVAL`).
- **Keep-alive Mode:** The Light, Temperature & Occupancy Sensor is permanently active and polls its parent every second.

The Light, Temperature & Occupancy Sensor will be reading the value of driver at the rate of `LIGHT_SENSOR_SAMPLING_TIME_IN_SECONDS` / `TEMPERATURE_SENSOR_SAMPLING_TIME_IN_SECONDS` (5 seconds) and updating its `u16MeasuredValue` attribute accordingly. Thus, the changes in light sensor reading should be done only at intervals of 5 second.

If there is any change in the `u16MeasuredValue` attribute, sensor will send out report after 1 second (`ZLO_MIN_REPORT_INTERVAL`), otherwise every 61 seconds (`ZLO_MAX_REPORT_INTERVAL`).

The LTO Sensor can but put into 'keep-alive' mode by pressing the **SW4** button on the board (an LED will start to blink), after it has been woken-up, if in low power, by first pressing **SW3** or **SW4** button.

To return to normal mode from 'keep-alive' mode, press the **SW4** button again or reset the device.

Note: On the FRDM-KW41Z board only the buttons SW3 and SW4 will wake the device from low power for 3 seconds. If the device is in low power, the events associated with the respective buttons will not be processed on the first button press and there will be needed two button presses – one for waking up the device and one for triggering the desired event.

4.4.4.1 Occupancy Sensor

Occupancy events are simulated by pressing a button on the Carrier Board - button **SW5** is assigned as the output of a virtual PIR detector. There are two types of virtual occupancy sensor: Open Collector and PWM. By default, the Light, Temperature & Occupancy Sensor operates as an Open Collector sensor.

Open Collector Sensor

An Open Collector sensor outputs a constant digital high/low signal for occupancy. In this demonstration, 'occupied' and 'unoccupied' states are toggled by pressing a button.

Simulating Unoccupied to Occupied Event:

To move the sensor from the unoccupied to occupied state, press **SW3** or **SW4** button to wake the device from low power. Then within 3 seconds press the button **SW5** on the Board.

Simulating Occupied to Unoccupied Event:

Once in the occupied state, to simulate an 'unoccupied' event, press **SW3** or **SW4** button to wake the device from low power, then press again the button **SW5**. If no further occupancy event is simulated by pressing the **SW5** button within a certain timeout period (180 seconds, by default), the sensor will automatically move to the unoccupied state.

4.4.4.2 Temperature and Light Sensors

The Temperature and Light sensors are simulated using an array of values, and the application is only cycling trough these values.

4.4.5 Sensing Clusters

The Light, Temperature & Occupancy Sensor uses the Illuminance Measurement, Temperature Measurement and Occupancy Sensing clusters to hold its results.

- The Illuminance Measurement cluster contains the `u16MeasuredValue` attribute which is used to indicate the light level measured by the sensor. Binding to this cluster (Id **0x0400**) will enable a remote device to receive reports when the value of this attribute on the sensor changes.
- The Temperature Measurement cluster contains the `u16MeasuredValue` attribute which is used to indicate the temperature measured by the sensor. Binding to this cluster (Id **0x0402**) will enable a remote device to receive reports when the value of this attribute on the sensor changes.
- The Occupancy Sensing cluster contains the `u8Occupancy` attribute which is used to indicate the occupied/unoccupied status of the sensor. Binding to this cluster (Id **0x0406**) will enable a remote device to receive reports when the value of the occupancy attribute on the sensor changes.

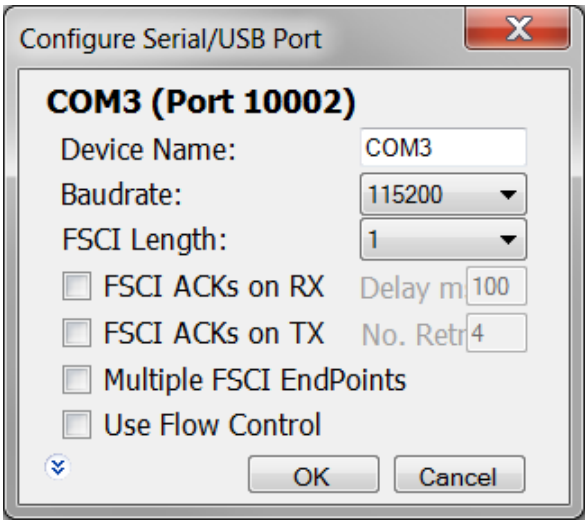
4.5 Network Commissioning Operations

This section describes the network commissioning operations for all the device types described in this Application Note. You should work through this section to incorporate a device in a network.

4.5.1 Forming and Joining a Network

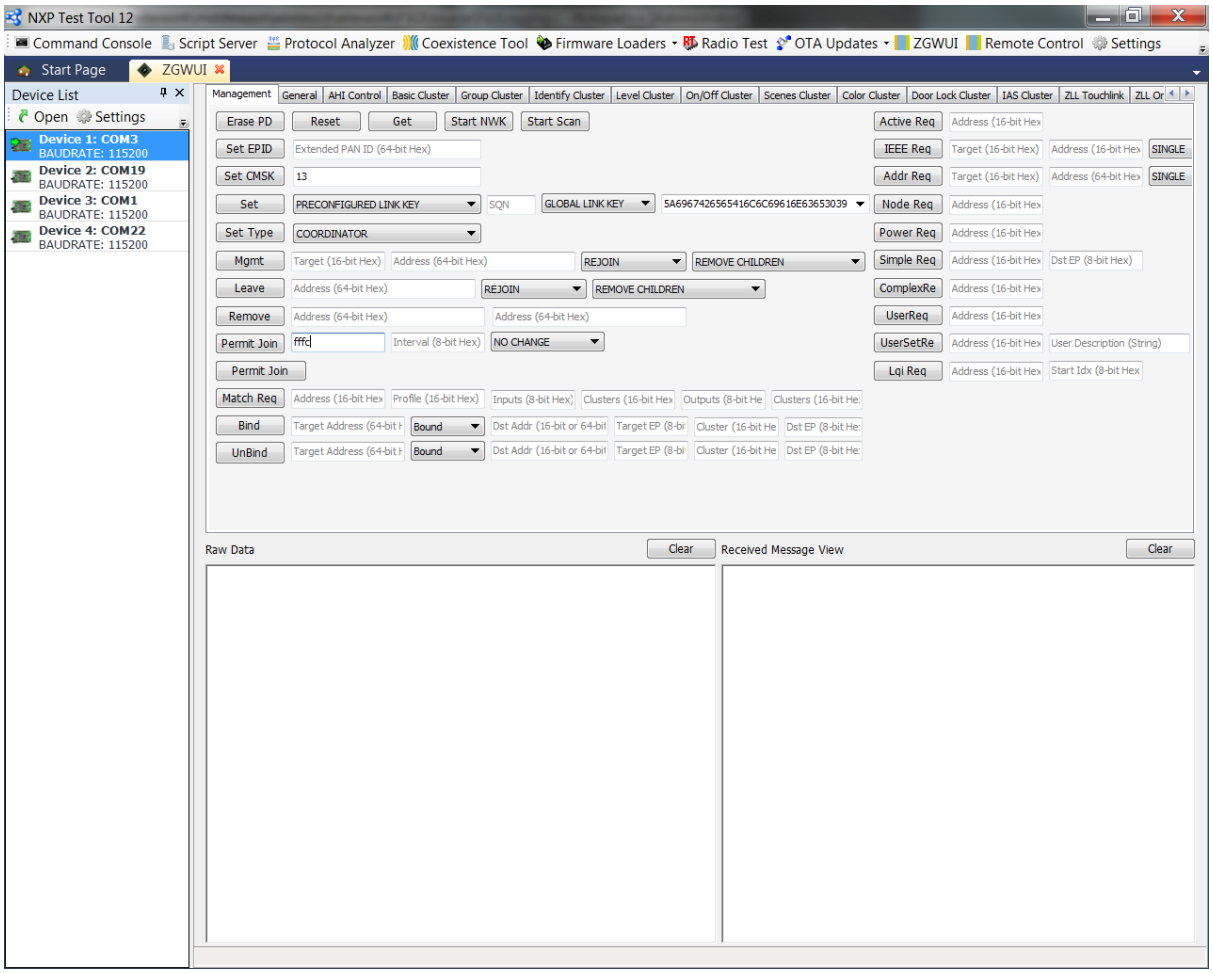
To create a network with the Control Bridge as the Coordinator and add one sensor node to it, follow the procedure below.

1. 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**
2. On the PC, start the NXP Test Tool 12 application, and select **ZGWUI** (top-right).
3. On the top-left select the COM device and click settings. In the resulting dialogue box, select the **Baud Rate** field to **115200**.



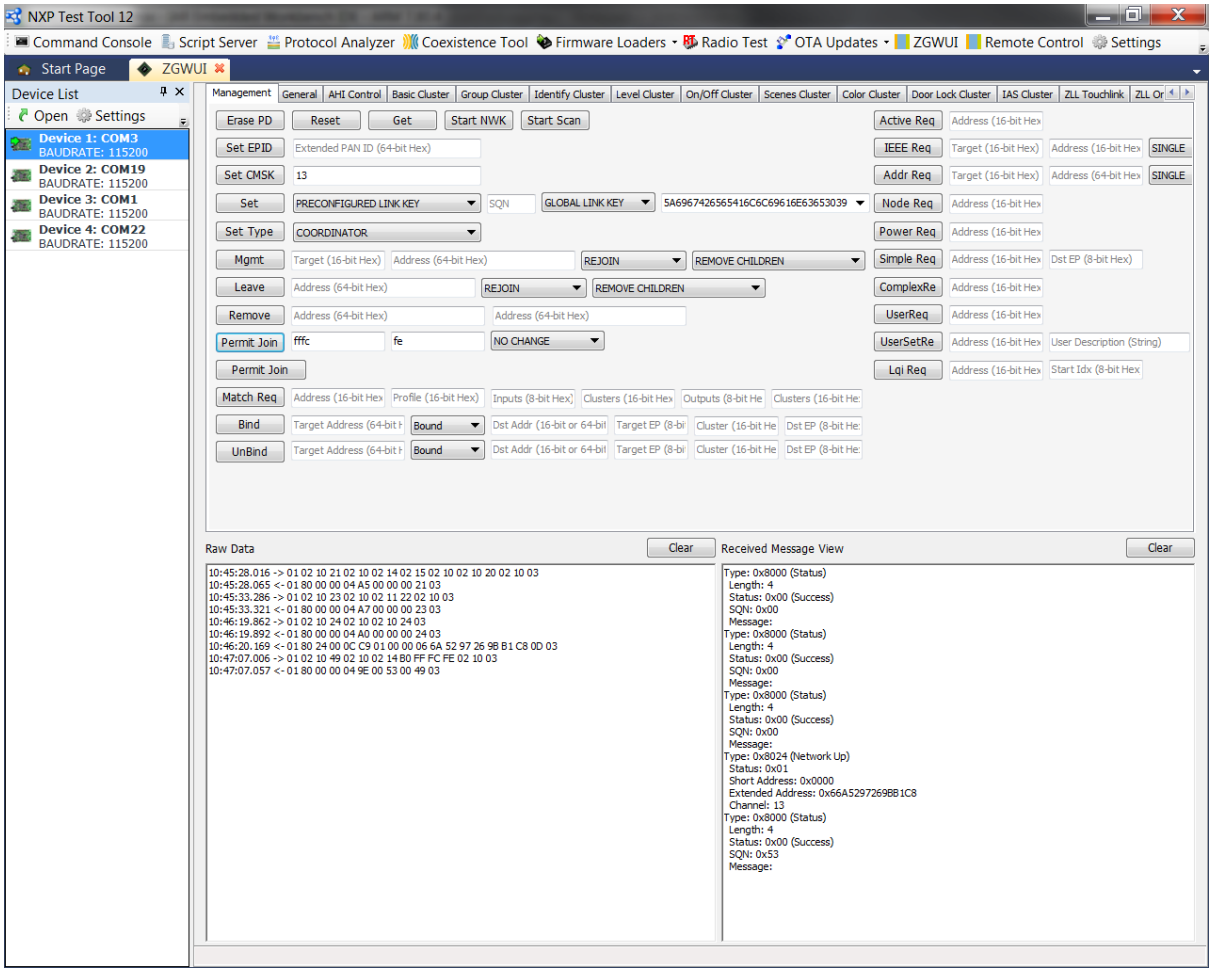
4. Click **OK** to apply the serial port configuration.
5. On top-left click **Open** or double-click the COM device to connect to the Control Bridge application.

6. In the interface, fill in the **Set CMSK** (channel mask) and **Permit Join** fields, and ensure the other fields are as shown below.

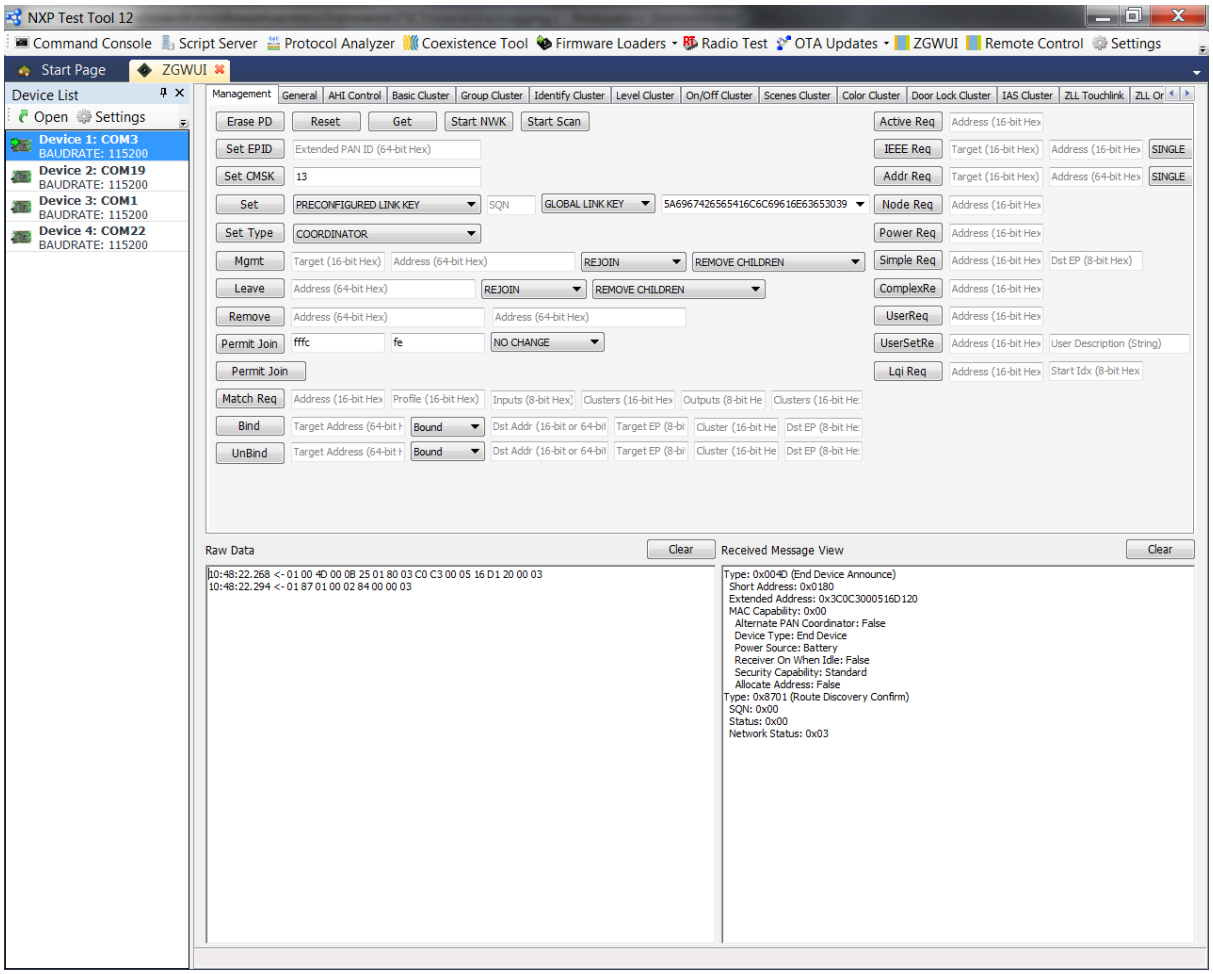


7. Reset the Control Bridge device by clicking the **Reset** button on the **Management** tab.
8. Set the Control Bridge device type to COORDINATOR in the **Set Type** field on the **Management** tab.

9. Start the network by clicking the **Start NWK** button on the **Management** tab. This should start the network and information should be output via the **Raw Data** and **Received Message View** panes, as shown below.



10. Open the network for devices to join by clicking the **Permit Join** button and start a sensor device. When the device has joined the network, an End Device Announce message will be displayed in the **Received Message View** pane, as shown below.



Note 1: If a sensor node does not find a suitable network to join, it goes into a low power state after scanning all the primary and secondary channels.

Note 2: If a sensor node loses its parents, it goes into a low power state after trying to re-join on all the channels.

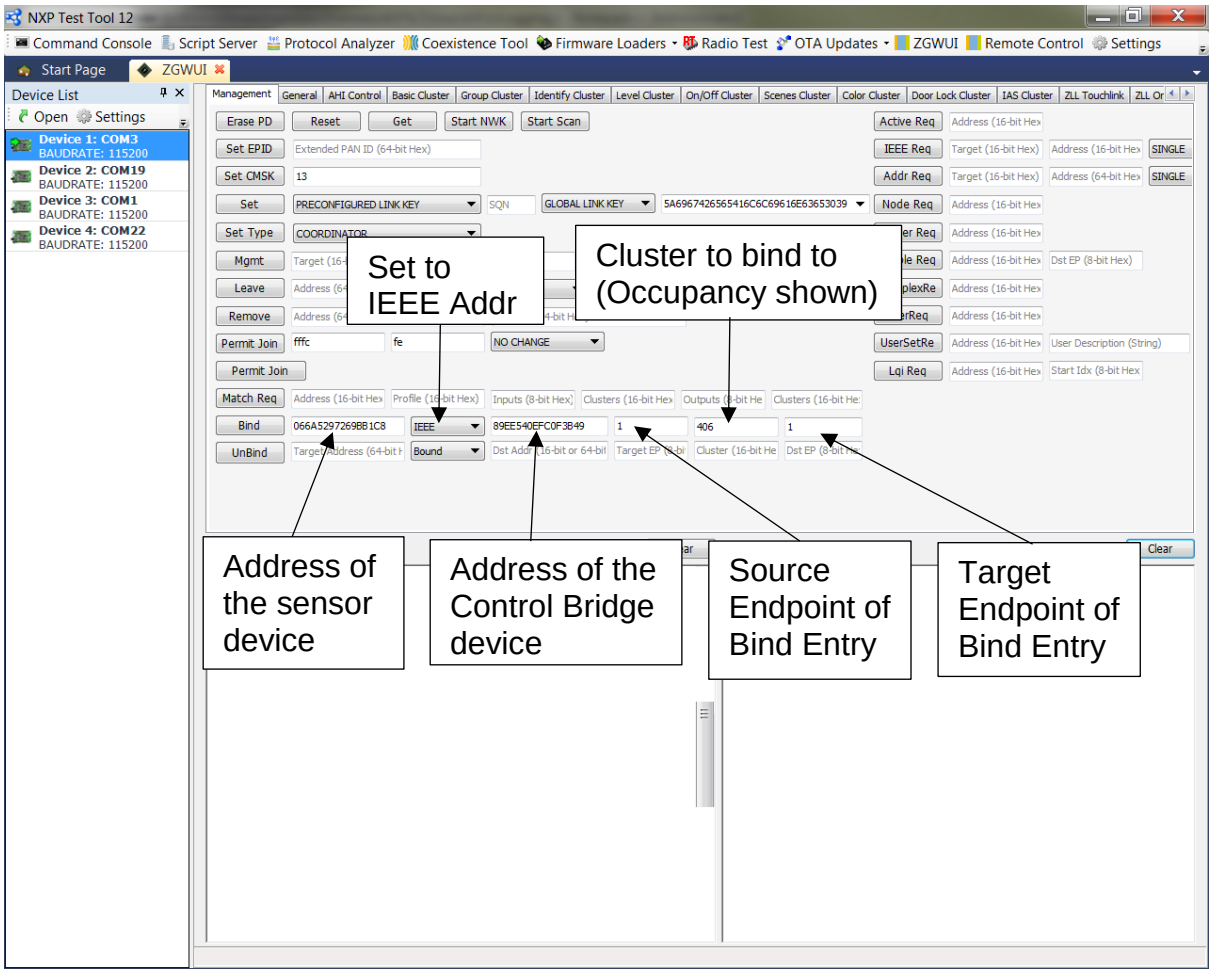
4.5.2 Allowing Other Nodes to Join

If you wish to add other nodes to the network, open the network for devices to join by clicking the **Permit Join** button in the interface again (see Step 10 above).

4.5.3 Binding Nodes (Control Bridge)

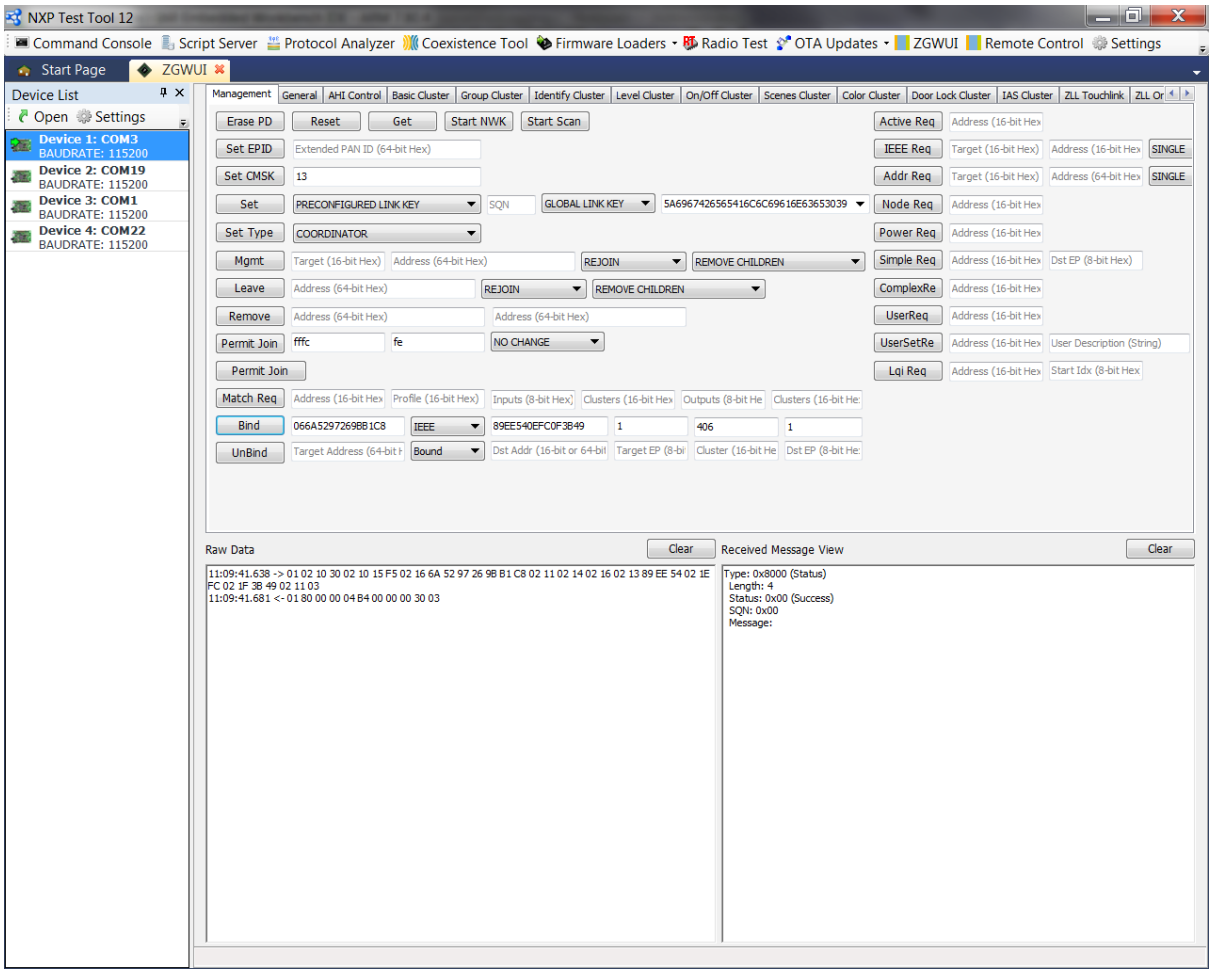
To create a binding between a cluster on a sensor node and the Control Bridge, follow the procedure below.

1. In the interface, fill in the **Bind** fields on the **Management** tab as shown below.

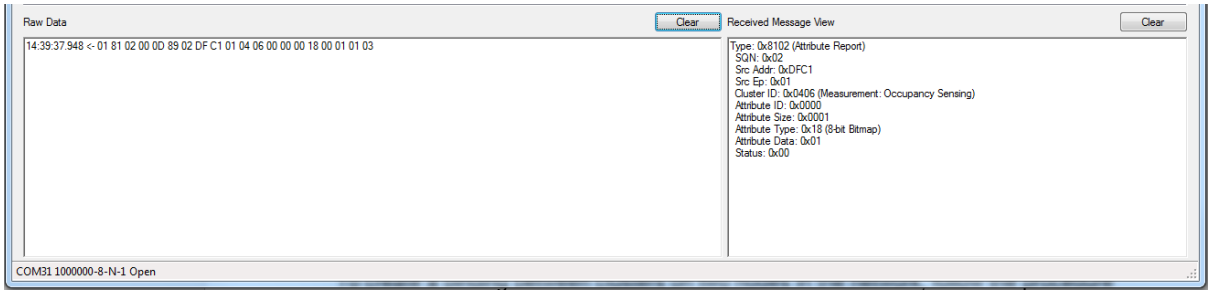


2. Place the sensor node in the persistent poll/keep-alive mode (by first pressing **SW3** or **SW4** button, if the device is in low power state) to ensure that it is able to receive a Bind Request from the Control Bridge, followed by a **SW4** button press after the device was waked-up from low power state.

3. Initiate binding by clicking the **Bind** button. If successful, the result will be shown in the **Received Message View** pane, as shown below.


NXP Test Tool 12
Command Console
Script Server
Protocol Analyzer
Coexistence Tool
Firmware Loaders
Radio Test
OTA Updates
ZGWUI
Remote Control
Settings
Start Page
ZGWUI
Device List
Open
Settings
Device 1: COM3
BAUDRATE: 115200
Device 2: COM19
BAUDRATE: 115200
Device 3: COM1
BAUDRATE: 115200
Device 4: COM22
BAUDRATE: 115200
Management
General
AHI Control
Basic Cluster
Group Cluster
Identify Cluster
Level Cluster
On/Off Cluster
Scenes Cluster
Color Cluster
Door Lock Cluster
IAS Cluster
ZLL Touchlink
Erase PD
Reset
Get
Start NWK
Start Scan
Set EPID
Extended PAN ID (64-bit Hex)
Set CMSK
13
Set
PRECONFIGURED LINK KEY
SQN
GLOBAL LINK KEY
5A6967426565416C6C69616E63653039
Set Type
COORDINATOR
Mgmt
Target (16-bit Hex)
Address (64-bit Hex)
REJOIN
REMOVE CHILDREN
Leave
Remove
Permit Join
fffc
fe
NO CHANGE
Match Req
Address (16-bit Hex)
Profile (16-bit Hex)
Inputs (8-bit Hex)
Clusters (16-bit Hex)
Outputs (8-bit He
Clusters (16-bit He
Bind
066A5297269BB1C8
IEEE
89EE540EFC0F3B49
1
406
1
UnBind
Target Address (64-bit
Bound
Dst Addr (16-bit or 64-bit
Target EP (8-bi
Cluster (16-bit He
Dst EP (8-bit He
Active Req
IEEE Req
Addr Req
Node Req
Power Req
Simple Req
ComplexRe
UserReq
UserSetRe
Lqi Req
Dst EP (8-bit Hex)
User Description (String)
Start Idx (8-bit Hex
SINGLE
Raw Data
Clear
Received Message View
Clear
11:09:41.638 -> 01 02 10 30 02 10 15 F5 02 16 6A 52 97 26 9B B1 C8 02 11 02 14 02 16 02 13 89 EE 54 02 1E FC 02 1F 3B 49 02 11 03
11:09:41.681 <- 01 80 00 00 04 B4 00 00 00 30 03
Type: 0x8000 (Status)
Length: 4
Status: 0x00 (Success)
SQN: 0x00
Message:

4. Once binding to the desired cluster is complete, the sensor node can be taken out of the persistent poll/keep-alive mode, by pressing **SW4** button again. At this point, any reports generated by a sensor node for bound clusters should be displayed in the **Received Message View** pane. An example of this is shown below for an Occupancy Sensor reporting a change in the state of its `u8Occupancy` attribute.



4.5.4 Binding Nodes (Finding and Binding)

To create a binding between clusters on two nodes in the network, follow the procedure below.

1. Put the target node (to which you wish to bind the sensor node) into identify mode.
2. Wakeup the device from low-power state by pressing **SW3** or **SW4.**
3. Put the sensor node into ‘Finding and Binding’ mode by pressing the button **SW2** on the Board (the sensor node will indicate that it is in ‘Finding and Binding’ mode by flashing an LED).

   To return to normal mode from ‘Finding and Binding’ mode, simply reset the device.

4.5.5 Network Steering

Discovery / Network steering on the sensor applications occurs when the **reset** button is pressed on the board, or by pressing **SW3** after the device was waked-up from low-power state (by first pressing **SW3** or **SW4** button).

4.5.6 Performing a Factory Reset

To reset the sensor node to its factory settings, hold down any of the following buttons for more than 8 seconds: SW2, SW3, SW4 or SW5.

5 Developing with the Application Note

The example applications provided in this Application Note were developed using the KW41Z-ZigBee_3.0_Software_v.6.0.8 and IAR Embedded Workbench 8.32.1 or MCUXpresso 10.2.1 as build Toolchain.

5.1 Code Common to All Sensors

app_zb_utils.c provides console output support.

5.2 Light Sensor Application Code

This section describes the application code for the LightSensor, 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 5.5.

5.2.1 Operational State Machine

The operational state machine (**sDeviceDesc.eNodeState**) is located within **app_light_sensor_state_machine.c**. If further operational modes are required, additional states must be added to this switch statement.

5.2.2 Button Press and Release Handling

Buttons are processed by the keyboard framework module, in **Keyboard.c** file. Any button events are then passed to the **KBD_Callback()** for processing in **app_main.c**, where the button event is eventually handled in **vAppHandleAppEvent()** contained in the file **app_event_handler.c**

5.2.3 Configuration Macros

The following macros, which can be found in **App_LightSensor.h**, are available for configuring the Light Sensor:

```
#define LIGHT_SENSOR_MINIMUM_MEASURED_VALUE                0x0001
#define LIGHT_SENSOR_MAXIMUM_MEASURED_VALUE                0xFAF
#define LIGHT_SENSOR_MINIMUM_REPORTABLE_CHANGE             0x01
#define LIGHT_SENSOR_SAMPLING_TIME_IN_SECONDS              5
```

The sampling time is the rate at which the `u16MeasureValue` attribute will be changed.

5.3 Occupancy Sensor Application Code

This section describes the application code for the OccupancySensor, 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 5.5.

5.3.1 Operational State Machine

The operational state machine (**sDeviceDesc.eNodeState**) is located within **app_occupancy_sensor_state_machine.c**. If further operational modes are required, additional states must be added to this switch statement.

5.3.2 Button Press and Release Handling

Buttons are processed by the keyboard framework module, in **Keyboard.c** file. Any button events are then passed to the **KBD_Callback()** for processing in **app_main.c**, where the button event is eventually handled in **vAppHandleAppEvent()** contained in the file **app_event_handler.c**.

5.3.3 Configuration Macros

The following macros, which can be found in **App_OccupancySensor.h**, are available for configuring the sensor:

```
#define APP_OCCUPANCY_SENSOR_UNOCCUPIED_TO_OCCUPIED_DELAY          10
#define APP_OCCUPANCY_SENSOR_TRIGGER_THRESHOLD                      5
#define APP_OCCUPANCY_SENSOR_OCCUPIED_TO_UNOCCUPIED_DELAY          180
```

5.4 Light, Temperature and Occupancy Sensor Application Code

This section describes the application code for LightTemperatureOccupancySensor, 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 5.5.

5.4.1 Operational State Machine

The operational state machine (**sDeviceDesc.eNodeState**) is located within **app_sensor_state_machine.c**. If further operational modes are required, additional states must be added to this switch statement.

5.4.2 Button Press and Release Handling

Buttons are processed by the keyboard framework module, in **Keyboard.c** file. Any button events are then passed to the **KBD_Callback()** for processing in **app_main.c**, where the button event is eventually handled in **vAppHandleAppEvent()** contained in the file **app_event_handler.c**.

5.4.3 Configuration Macros

The following macros, which can be found in **App_LightTemperatureOccupancySensor.h**, are available for configuring the sensor:

```
#define LIGHT_SENSOR_MINIMUM_MEASURED_VALUE                        0x0001
#define LIGHT_SENSOR_MAXIMUM_MEASURED_VALUE                        0xFAF
#define LIGHT_SENSOR_MINIMUM_REPORTABLE_CHANGE                     0x01
#define LIGHT_SENSOR_SAMPLING_TIME_IN_SECONDS                      5

#define APP_OCCUPANCY_SENSOR_UNOCCUPIED_TO_OCCUPIED_DELAY          10
#define APP_OCCUPANCY_SENSOR_TRIGGER_THRESHOLD                     5
#define APP_OCCUPANCY_SENSOR_OCCUPIED_TO_UNOCCUPIED_DELAY          180


#define TEMPERATURE_SENSOR_MINIMUM_MEASURED_VALUE                  0x0001
#define TEMPERATURE_SENSOR_MAXIMUM_MEASURED_VALUE                  0xFAF
#define TEMPERATURE_SENSOR_MINIMUM_REPORTABLE_CHANGE               0x01
#define TEMPERATURE_SENSOR_SAMPLING_TIME_IN_SECONDS                5
```

The sampling time is the rate at which the `u16MeasureValue` attribute will be changed.

5.5 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.

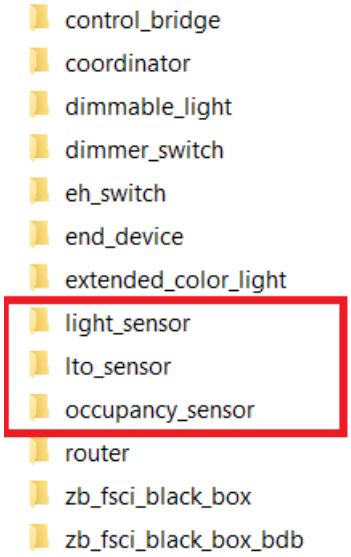
5.5.1 Pre-requisites

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_example\zigbee_3_0

The ZigBee sensor devices projects are highlighted below:



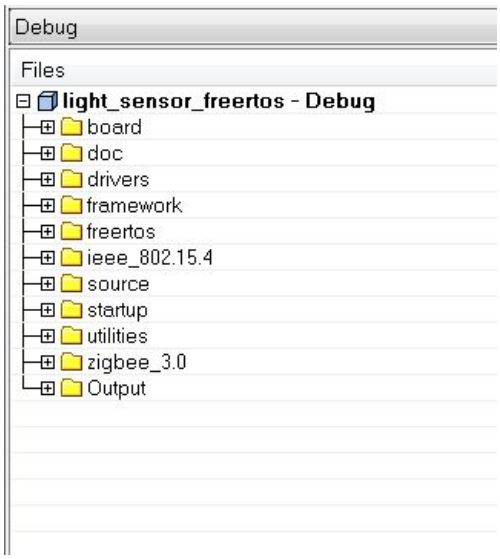
5.5.2 Build Instructions – IAR Embedded Workbench IDE

To build any of these light applications, open the application project file in IAR Embedded Workbench IDE. This application note uses the “dimmable light” project as an example. The project workspace file is located at the following relative path:

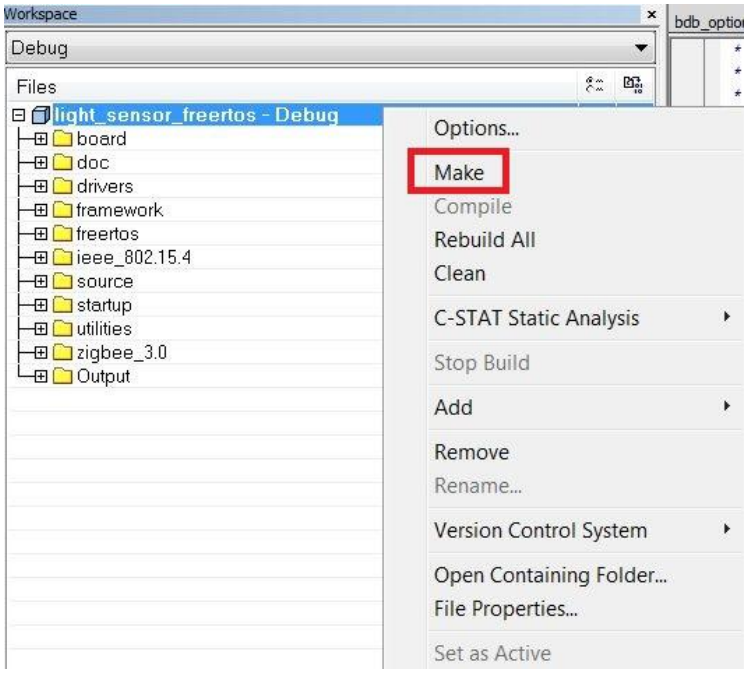
..\boards\frdmkw41z\wireless_examples\zigbee_3_0\light_sensor\freertos\iar

\light_sensor_freertos.eww

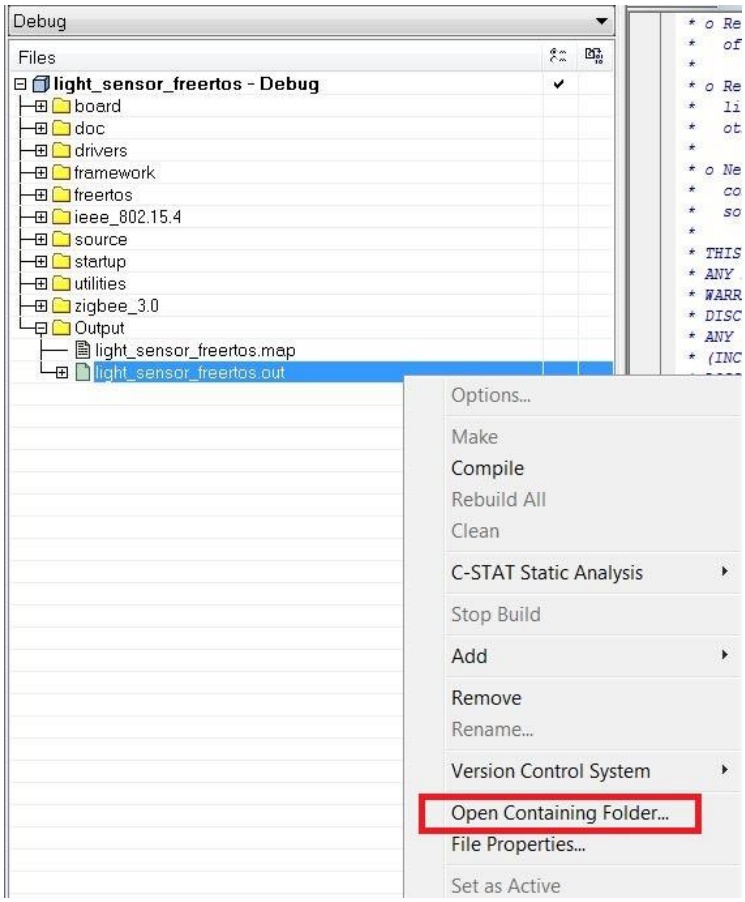
Open the project workspace file by double clicking or by drag and dropping it into IAR EW.



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 the *.out file and click “Open Containing Folder “.



The folder opened by Windows Explorer will also contain the binary file.

5.5.3 Build instructions – MCUXpresso

To build any of these light applications, please refer to the official documentation “Getting started with MCUXpresso.pdf” and follow the steps provided in chapter 7 “Running a demo using MCUXpresso IDE”

6 Application Code Sizes

The applications presented in this Application Note have the following memory footprints on the KW41Z device, when using the KW41Z_ZigBee_3.0_Software_v6.0.8 SDK

Application – Release Configuration, IAR 8.32.1.	Read-only code (Bytes)	Read-only data (Bytes)	Read-write data (Bytes)
App_LightSensor	207123	20507	29964
App_OccupancySensor	207345	20759	29923
App_LightTemperatureOccupancySensor	208819	21070	30629

7 Related Documents

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

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

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
4	Updates for KW41Z Zigbee 3.0 Maintenance Release1
5	Updates for KW41Z Maintenance Release2
6	Update code sizes for KW41Z Maintenance Release2

Important Notice

NXP Semiconductors

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

www.nxp.com