Application Note: JN-AN-1245

DK006: JN5189 / JN5188 / K32W061 / K32W041

Zigbee 3.0 Controller and Switch

This Application Note applies to the JN5189, JN5188, K32W061 and K32W041 Zigbee 3.0 wireless microcontrollers used with the DK006 (Development Kit) platform. These microcontrollers will be referred to as the DK006 microcontrollers throughout this document.

This Application Note provides example applications for a controller and a switch in a Zigbee 3.0 network that employs the NXP DK006 wireless microcontrollers. An example application can be employed as:

- **A demonstration using the supplied pre-built binaries that can be run on nodes of the DK006 Development Kits**
- **A starting point for custom application development using the supplied C source files and associated project files**

The controller and switch described in this Application Note are based on Zigbee device types from the Zigbee Lighting & Occupancy (ZLO) Device Specification.

The Application Note also includes an example of a typical Zigbee Green Power (GP) Energy Harvesting switch and an example of low power coin cell switch in a Zigbee 3.0 network

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 a controller and a switch on the NXP DK006 platform.

This Application Note provides example implementations of a controller and a switch that use the following device types from the Zigbee Lighting & Occupancy (ZLO) Device Specification:

- Color Scene Controller
- Dimmer Switch

The above device types are detailed in the *Zigbee 3.0 Devices User Guide [JN-UG-3131]* and the clusters used by the devices are detailed in the *Zigbee 3.0 Cluster Library User Guide [JN-UG-3132]*.

Note: If you are not familiar with Zigbee 3.0, you are advised to refer the *Zigbee 3.0 Stack User Guide [JN-UG-3130]* for a general introduction.

For information on the Zigbee Green Power (GP) switch provided in this Application Note, refer to Zigbee Green Power (GP) Switch.

2 Development Environment

2.1 Software

In order to use this Application Note, you need to install the Eclipse-based Integrated Development Environment (IDE) and Software Developer's Kit (SDK) that are appropriate for your DK006 wireless microcontroller:

- MCUXpresso IDE
- JN518x Zigbee 3.0 SDK
- K32W061/K32W041 Zigbee 3.0/Bluetooth SDK

The MCUXpresso software and installation instructions are described in the *Zigbee 3.0 Getting Started Application Note [JN-AN-1260]*. *JN-AN-1245 Release Notes* (included in this folder) indicate the versions of MCUXpresso and SDK that you will need to work with this Application Note.

The DK006 specific resources and documentation are available via the MCUXpresso website to authorised users.

Note: Prebuilt application binaries are supplied in this Application Note package see the *Zigbee 3.0 Getting Started Application Note [JN-AN-1260]* for instructions on how to compile the application binaries on your own system.

2.2 Hardware

Hardware kits are available from NXP to support the development of Zigbee 3.0 applications. The following kits provide a platform for running these applications:

- JN518x-DK006 Development Kit, which features JN5189 devices
- IoT_ZTB-DK006 Development Kit, which features K32W061 devices

This kit supports the NFC commissioning of network nodes (see NFC Hardware Support).

The Coin Cell Switch application can be run on DR1236 platform which is a JN5189 device in a cost effective 2-layer PCB running from a coin cell.

3 Application Note Overview

The example applications provided in this Application Note are listed in the table below with the switch/controller device types that they support.

Application	Device Type
ColorSceneController	Color Scene Controller
DimmerSwitch	Dimmer Switch
GpEhSwitch	Green Power Energy Harvesting Switch

Table 1: Example Applications

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 DK006 Development Kits.

- To load the pre-built binaries into the evaluation kit components and run the demonstration application, refer to Loading the Applications
- To start developing your own applications based on the supplied source files, refer to Developing with the Application Note

3.1 NFC Hardware Support

All the microcontrollers supplied with the DK006 Development Kits contain an internal NFC tag (NTAG) that connects to an NFC antenna on the OM15076-3 Carrier Boards. NTAG enabled applications use this internal NTAG by default.

The OM15076-3 Carrier Boards are also fitted with an external NTAG which can be used instead of the internal NTAG or with chip variants that do not have an internal NTAG (these are not supplied in the DK006 Development Kits). Minor hardware modifications are required to the OM15076-3 Carrier Boards to enable this external NTAG.

4 Supported Device Types

The supported ZLO device types in this Application Note are:

- Color Scene Controller
- Dimmer Switch

These switch/controller devices types must be paired for operation with lighting device types. Example applications for the paired device types are provided in the Application Note *Zigbee 3.0 Light Bulbs [JN-AN-1244]*. Two device types can be paired if they support the same cluster (Color 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 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 Color**	**Hue & Saturation**	**Color Temperature**	**Color Loop**
Dimmer Switch	Yes	Yes	No	No	No	No
Color Scene Controller	Yes	Yes	Yes	Yes	Yes	Yes
Coin Cell Switch	Yes	Yes	No	No	No	No
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 2: Switch/Controller Device Types and Controllable Attributes

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 Color**	**Hue & Saturation**	**Color Temperature**	**Color Loop**
Dimmable Light	Yes	Yes	No	No	No	No
Extended Color Light	Yes	Yes	Yes	Yes	Yes	Yes
Color 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

5 Running the Demonstration Application

This section describes how to use the supplied pre-built binaries to run the example applications on components of a DK006 kit. All the applications run on a DK006 microcontroller module on a OM15076-3 Carrier Board, fitted with a specific expansion board.

5.1 Loading the Applications

The table below lists the application binary files supplied with this Application Note and indicates the DK006 hardware kit components on which the binaries can be used. These files are located in the **Binaries** folder of the Application Note. Each device has its own folder (matching the name of the binary file), the file/folder names indicate the included functionality.

DK006 Hardware	JN5189 Binary File
OM15076-3 Carrier Board OM15082-2 Generic Expansion Board	**ColorSceneController_OM15082** **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V1**
OM15076-3 Carrier Board OM15082-2 Generic Expansion Board	**DimmerSwitch_OM15082.bin** **DimmerSwitch_NtagIcode_Ota_RamOpt_OM15082_V1**
OM15076-3 Carrier Board OM15082-2 Generic Expansion Board	**GpEhSwitch_OM15082**
DR1236 Coin-cell Switch Board (*Not included in DK006 Development Kit)*	**CoinCellSwitch_RamOpt_DR1236.bin** **CoinCellSwitch_Ota_RamOpt_DR1236.bin**

Table 4: Application Binaries and Hardware Components

A binary file can be loaded into the Flash memory of a DK006 device using the *JN518x Flash Programmer [JN-SW-4407]*. This software tool is described in the *JN518x Production Flash Programmer User Guide [JN-UG-3127]*.

Note: You can alternatively load a binary file into a DK006 device using the Flash programmer built into the relevant IDE.

To load an application binary file into a DK006 module on a Carrier Board of a kit, follow the instructions below:

1. Connect a USB port of your PC to the USB Mini B port marked “FTDI USB” on the Carrier Board (next to the 34-pin header) using a ‘USB A to Mini B’ cable. At this point, you may be prompted to install the driver for the USB virtual COM port.
2. Determine which serial communications port on your PC has been allocated to the USB connection.
3. Put the DK006 device into programming mode by holding down the ISP button while pressing and releasing the RESET button.
4. Run the batch (**.bat**) file provided alongside the binary (**.bin**) file to erase the contents of the flash memory including the persistent data stored by the PDM and program the binary file into flash memory. The batch file will prompt for the COM port number to use.
5. Once the download has successfully completed, reset the board or module to run the application.

The batch files require the installation of the DK6 Production Flash Programmer to have been completed following the instructions in the *Zigbee 3.0 Getting Started Application Note [JN-AN-1260]*.

5.2 Using the Dimmer Switch

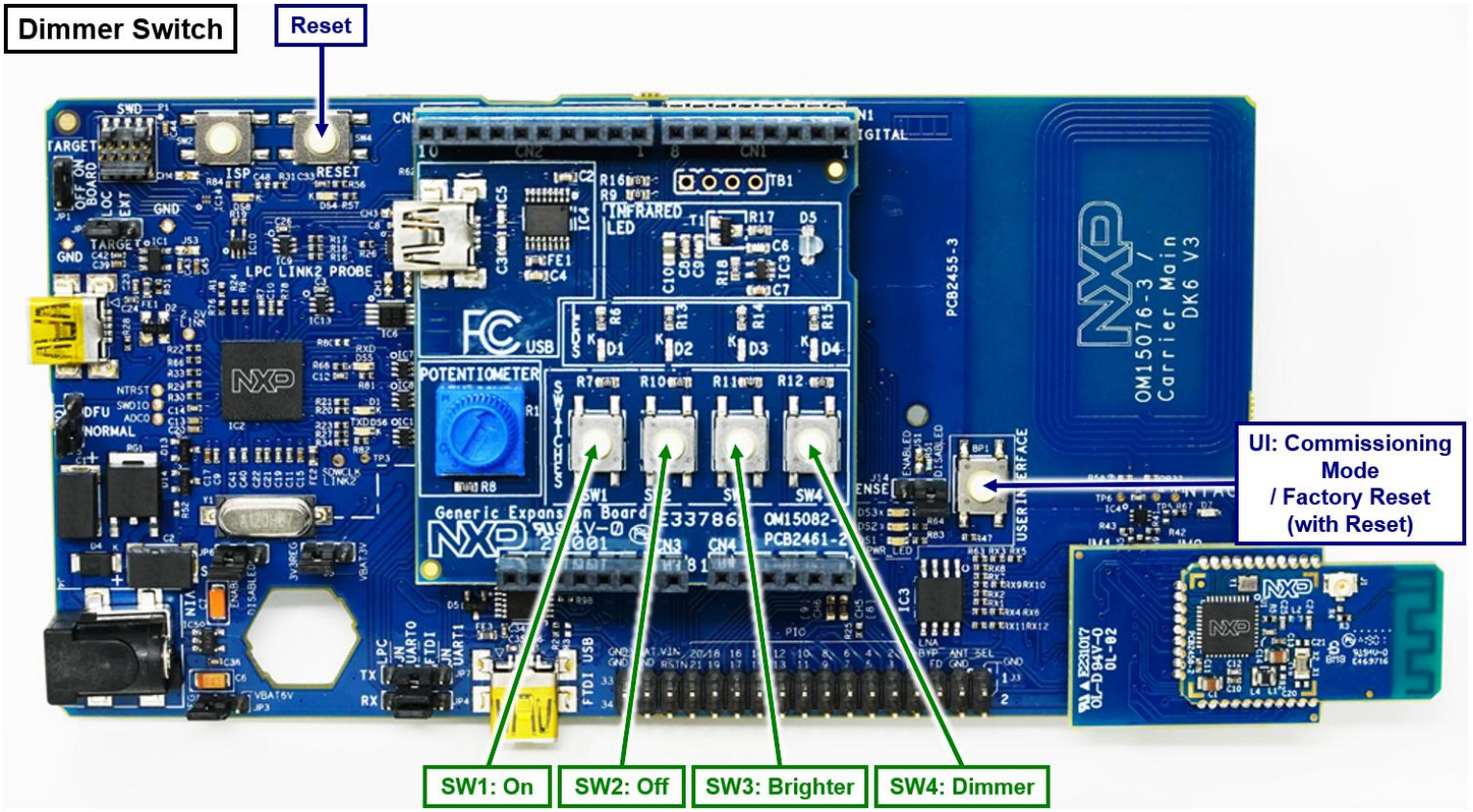
This section describes how to commission and operate the Dimmer Switch application in a Zigbee 3.0 network. To use this application, you must have programmed the relevant application binary into the DK006 wireless microcontroller module on a Carrier Board fitted with the OM15082 Generic Expansion Board, as described in Loading the Applications:

- **DimmerSwitch_OM15082.bin**
- **DimmerSwitch_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

> **Note:** To use this application which is based on the Dimmer Switch device type, you will also need to implement the paired Dimmable Light device type as described in the Application Note *Zigbee 3.0 Light Bulbs [JN-AN-1244]*.

5.2.1 Dimmer Switch Functionality

The ZLO Dimmer Switch device resides on a node (fitted with the OM15082 Generic Expansion Board) which acts as an End Device in the network. As an End Device, it can only join an existing network and cannot form a new network. This switch device can be used to perform the commissioning of the light devices and control them, using the controls indicated in the diagrams below.



LED Name	Physical LED on Board
LED1	D1 on OM15082 expansion board
LED2	D2 on OM15082 expansion board
LED3	D6 on OM15082 carrier board

Table 5: LED Hardware Mapping

5.2.2 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 required action depends on the Carrier Board used:

- On the OM15076 Carrier Board, press and release the **RESET** button while holding down the **USER INTERFACE** button (both buttons are on the Carrier Board).

A factory-new Dimmer Switch waits for user input on power-up, which is indicated by the LED pair LED1+LED3 and LED2 flashing alternately (for physical LEDs, see Dimmer Switch Functionality above).

5.2.3 Joining a Network

5.2.3.1 Classical Network Joining

The following pre-built binaries support Classical Network Joining:

- **DimmerSwitch_OM15082.bin**
- **DimmerSwitch_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

The switch device is an End Device and, as such, must join an existing Zigbee network. The network join process is as follows:

1. Start Network Steering on the switch by pressing any button on the OM15082 Generic Expansion Board. As soon as a button-press is detected, the device will search for a suitable network to join.
2. The outcome of Network Steering is now indicated as follows:
   - If a suitable open network is found (‘Permit Join’ is true), the switch node will join and indicate success by briefly illuminating LEDs LED1-LED3.
   - If a suitable network is not found, the switch node goes into deep sleep mode, which is indicated by all LEDs being switched off.

5.2.3.2 Joining an Existing Network using NFC

The following pre-built binaries support joining using NFC:

- **DimmerSwitch_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

A Dimmer Switch can join or move to an existing network by exchanging NFC data with either:

- The *NciIcode* build of the Zigbee IoT Control Bridge, described in the Application Note *Zigbee 3.0 IoT Control Bridge [JN-AN-1247]*
- The Zigbee IoT Gateway Host, described in the Application Note *Zigbee IoT Gateway Host with NFC [JN-AN-1222]*

This provides a fast and convenient method to introduce new devices into such a network.

Ensure the hardware is set up for NFC as described in NFC Hardware Support.

Instructions for this process are included in the above Application Notes (JN-AN-1247 or JN-AN-1222).

5.2.4 Commissioning

This section describes how to commission light nodes to be controlled from the Dimmer Switch, either as bound devices or as a group of lights – work through either Commissioning Light Nodes – Binding or Commissioning Light Nodes – Grouping for each light to be controlled by the switch.

5.2.4.1 Commissioning Light Nodes – Binding

The light nodes to be controlled by the Dimmer Switch can be bound to the device as follows:

1. Put the lights that need to be commissioned with the Dimmer Switch into 'Finding and Binding' mode (this step is light-specific). The lights will start to flash in Identify mode.
2. Press and hold down the Commissioning button on the Carrier Board of the Dimmer Switch - button **USER INTERFACE** on OM15076. The **USER INTERFACE** button will not wake the DimmerSwitch from deep sleep, press SW1 to wake it if in this sleep mode.
3. Once LED D1 starts flashing on the OM15082 Generic Expansion Board, press and release the **SW2** button.
4. Now hold down the **USER INTERFACE** button on the Carrier Board until the lights come out of Identify mode (when **USER INTERFACE** is released, the node exits Commissioning mode and returns to Individual Control mode).

5.2.4.2 Commissioning Light Nodes – Grouping

The light nodes to be controlled by the Dimmer Switch can be formed into a group as follows:

1. Put the lights that need to be commissioned with the Dimmer Switch into 'Finding and Binding' mode (this step is light-specific). The lights will start to flash in Identify mode.
2. Press and hold down the Commissioning button on the Carrier Board of the Dimmer Switch - button **USER INTERFACE** on OM15076. The **USER INTERFACE** button will not wake the DimmerSwitch from deep sleep, press SW1 to wake it if in this sleep mode.
3. Once LED1 starts flashing on the OM15082 Generic Expansion Board, press and release the **SW3** button.
4. Now hold down the **USER INTERFACE** button on the Carrier Board until the lights come out of Identify mode (when **USER INTERFACE** is released, the node exits Commissioning mode and returns to Individual Control mode).

5.2.5 Operation

The Dimmer Switch has two modes of operation:

- Commissioning mode
- Group Control mode

These modes are described below.

5.2.5.1 Commissioning Mode

In this mode, the commissioning functionality of the Zigbee Base Device can be invoked, such as Network Steering and 'Finding and Binding'. The Dimmer Switch can be put into Commissioning mode by pressing and holding down the **USER INTERFACE** button on the OM15076 Carrier Board. This mode of operation is indicated by the continuous flashing of LED1 on the OM15082 Generic Expansion Board.

In Commissioning mode, you can use the buttons **SW1-SW4** on the OM15082 Generic Expansion Board to perform various operations, as follows:

- Press **SW1** to trigger Network Steering. This will enable 'Permit Join' on the network.
- Press **SW2** to add a light node (that is in Identify mode) to the Binding table on the Dimmer Switch. This is Finding and Binding without grouping.
- Press **SW3** to add a light node (that is in Identify mode) to the group for the Dimmer Switch. This is Finding and Binding with grouping.

When **USER INTERFACE** is released, the node exits Commissioning mode and returns to Individual Control mode.

Note: The Dimmer Switch enters into the Commissioning mode only after the LED1 starts flashing. Therefore, please ensure that the LED is flashing before executing any Commissioning mode operations.

5.2.5.2 Group Control Mode

This mode is used to control a group of lights which has already been formed for the Dimmer Switch using Finding and Binding with grouping (refer to Commissioning Light Nodes – Grouping). This mode of operation can be entered by pressing either of the following switch combinations: **SW1+SW3** or **SW2+SW4.**

In Group Control mode, you can use the switches **SW1-SW4** on the OM15082 Generic Expansion Board to perform various operations, as follows:

- Press **SW1** to switch the lights on (full brightness)
- Press **SW2** to switch the lights off
- Press and hold down **SW3** to increase the brightness levels of the lights
- Press and hold down **SW4** to decrease the brightness levels of the lights

Note: On a reset, the Dimmer Switch will restart in Group Control mode.

Note: The Dimmer Switch can be woken from deep sleep by pressing any of the buttons **SW1** to **SW4**. It does not wake from deep sleep if the user presses the **USER INTERFACE** button on the Carrier Board, as this button is involved in a special sequence to erase persisted data.

5.3 Using the Color Scene Controller

This section describes how to commission and operate the Color Scene Controller application in a Zigbee 3.0 network. To use this application, you must have programmed the relevant application binary into the DK006 wireless microcontroller module on a Carrier Board fitted with the OM15082 Generic Expansion Board, as described in Loading the Applications:

- **ColorSceneController_OM15082.bin**
- **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

Note: When using an application built for a Carrier Board with OM15082 Expansion Board, a Graphical User Interface (GUI) can be run on a PC to simulate the keys of the Remote Control Unit (see Remote Control GUI). A binary serial protocol is used to send key-press data from the GUI to the device and to send LED status information from the device to the GUI. If using the application without the GUI but with a debug terminal, the LED information will look like garbage in the debug terminal, although it is valid data.

5.3.1 Remote Control GUI

To compensate for the lack of button inputs when the OM15082-based hardware is used, a Windows-based Graphical User Interface (GUI) is provided. This connects via the DK006 UART and acts as a substitute for the Remote Control Unit. The GUI is provided as source code and is a Visual Studio project.

To use the Remote Control GUI:

1. Connect the Carrier Board to a PC using a USB cable.
2. In Windows Explorer on the PC, navigate to the Application Note directory **RemoteControlGUI** and double-click on the file **Remote-Control-GUI.exe**.
3. Select the serial port used for the USB connection to the Carrier Board.
4. In the GUI, press the keys using mouse-clicks in order to issue control commands.

Button events will be passed to the Color Scene Controller hardware, resulting in the over-the-air transmission of LED control commands by the DK006 module. LED control messages are also sent back to the GUI to allow it to mirror the LED states on the hardware.

5.3.2 Color Scene Controller Functionality

The Color Scene Controller is a remote control device that is capable of controlling other devices that support the On/Off, Color Control, Level Control and Scenes clusters as servers. It supports the Touchlink cluster as both a server and client, so it is capable of adding devices to a distributed network as well as being added to a network itself. As a controller type device, it supports the Finding and Binding process as an initiator.

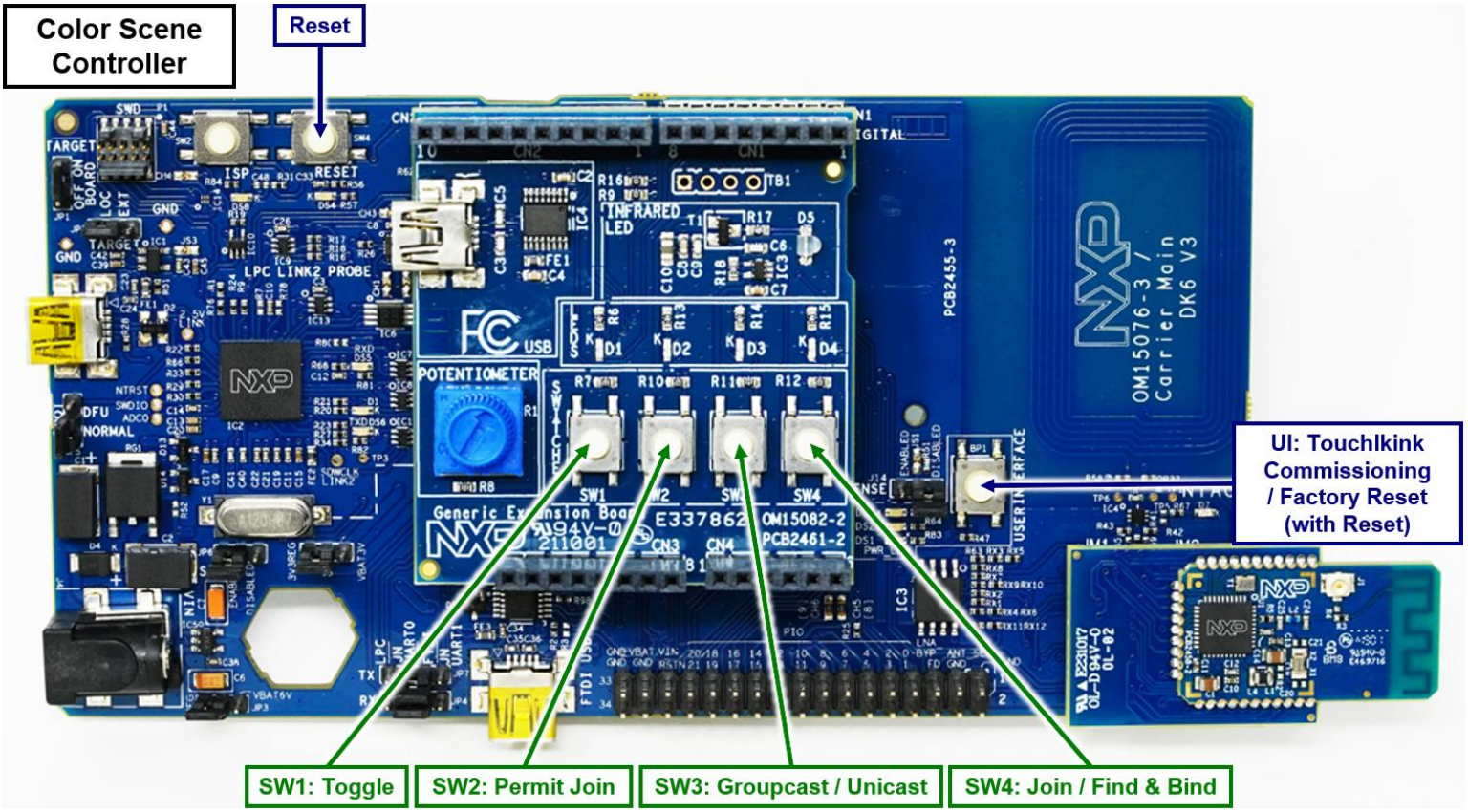
The device supports the following clusters as clients:

- **Operational clusters:** On/Off, Level Control, Color Control, Scenes
- **Support clusters:** Basic, Groups, Identify

The Zigbee Cluster Library (ZCL) provides a rich array of client commands for controlling various attributes of these clusters. This Application Note provides a sample of the most common and useful commands.

When the OM15082-based hardware is used, a Windows-based Graphical User Interface (GUI) can be run to provide similar remote control functions from a PC (see Remote Control GUI). The functions of the Windows Remote Control GUI for the Color Scene Controller are detailed in Functionality of the Color Scene Controller.

When using the OM15082-based hardware in conjunction with the key inputs provided by the Remote Control GUI, the physical buttons on the boards also provide some functionality:



- **SW1:** On/Off toggle
- **SW2:** Initiate association joining, if already part of a network
- **SW3:** Toggle between Groupcast and Unicast modes, if part of a network
- **SW4:** Initiate association joining, if not already part of a network (factory-new)
  Initiate Finding and Binding, if part of a network (not factory-new)
- **USER INTERFACE:** Initiate Touchlink commissioning

5.3.3 Joining or Forming a Network

A factory-new Color Scene Controller can join or form a network using one of the following methods:

- Join an existing network through classical 'discovery and association'
- Join an existing network using NFC to exchange network credentials
- Join an existing network or form a new network using Touchlink

5.3.3.1 Classical Network Joining

The following pre-built binaries support Classical Network Joining:

- **ColorSceneController_OM15082.bin**
- **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

The Color Scene Controller is implemented as a sleepy, 'Rx Off When Idle' End Device. As such, it is not capable of independently forming a network or acting as a parent through which other devices can join a network. In order for the Color Scene Controller to join a network, either Centralised or Distributed, the network must be already formed and operational, and open for new devices to join.

To join the device to a network:

1. On an existing network, trigger 'Network Steering for a device on the network'.
2. On the Color Scene Controller, press the **SW4** button if using the OM15082 Generic Expansion Board.

The controller will perform network discovery across all channels in the primary and secondary sets, and attempt to associate with any discovered networks that have the 'Permit Join' bit set in their beacons. After successful association, the network key will be sent to the controller, either from the Trust Centre in a Centralised network or from the parent device in a Distributed network. The network key will be encrypted with the appropriate link key, depending on the type of network being joined. It the controller joined a Centralised Trust Centre network then the initial link key will be replaced by the Trust Centre with a new link key for all future communications between the Trust Centre and controller.

If the association was not successful or the network key was not transported successfully, the process can be repeated, as required.

5.3.3.2 Joining an Existing Network using NFC

The following pre-built binaries support joining using NFC:

- **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

A Color Scene Controller can join or move to an existing network by exchanging NFC data with either

- The *NciIcode* build of the Zigbee IoT Control Bridge, described in the Application Note *Zigbee 3.0 IoT Control Bridge [JN-AN-1247]*
- The Zigbee IoT Gateway Host, described in the Application Note *Zigbee IoT Gateway Host with NFC [JN-AN-1222]*

This provides a fast and convenient method to introduce new devices into such a network.

Ensure the hardware is set up for NFC as described in NFC Hardware Support.

Instructions for this process are included in the above Application Notes (JN-AN-1247 or JN-AN-1222).

5.3.3.3 Joining or Forming a Network using Touchlink

The following pre-built binaries support Touchlink joining or formation:

- **ColorSceneController_OM15082.bin**
- **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

Touchlink is a method of sharing network parameters with other devices in order to bring them into the network, and discovery information about their device type and capability. Touchlink is based on proximity and is carried out at low power in order to limit the range of the transmissions.

The Color Scene Controller acts as a Touchlink Initiator. The Target device must be a Router and can be either factory-new or not factory-new.

To form a new network with a factory-new Color Scene Controller:

1. Bring the controller into close proximity (about 10cm) of the Target device.
2. On the Color Scene Controller, press the **USER INTERFACE** button on the Carrier Board if using the OM15082 Generic Expansion Board.

The controller will send Touchlink Scan Requests across the primary and secondary channels. Target devices will respond with Scan Responses and, for further processing, the controller will select the responder that it considers to be closest. This Target device will then be further interrogated with Device Information Requests and asked to identify itself. The target Router will be sent a Network Start Request with the network parameters. It will start the network and then the controller will send a Network Join Request to complete the process.

Note: If the Target device is not factory-new, it may (at the application's discretion) refuse to leave its current network in order to form a new network with the controller. When a device in one network is asked to join or form a new network, this is known as 'stealing'. It is the application's responsibility to decide whether stealing is permitted.

To join a factory-new Color Scene Controller to an existing Distributed network:

1. Bring the controller into close proximity (about 10cm) of another Touchlink initiator that is already on the network.
2. Initiate Touchlink on both devices simultaneously. On the Color Scene Controller, press the **USER INTERFACE** button on the Carrier Board if using the OM15082 Generic Expansion Board.

Both Initiators will send Touchlink Scan Requests and respond to them. The factory-new controller should then abandon its scan requests and defer to the 'not factory-new' Initiator. The factory-new controller will be further interrogated and sent a Touchlink End Device Join Request with the network parameters. The controller will then send a Network Join Request to complete the process.

Once the controller is part of a Distributed network, the Touchlink process can be repeated to bring other devices into its network.

Note: Touchlink cannot be used on a Centralised network to bring other devices into the network. Discovery and association must be used instead, and the Trust Centre must confirm the association and transport the network key.

Following successful Touchlink joining, if the Target device supports clusters as servers that match the clients on the controller then bindings will be created to allow the Color Scene Controller to control the device.

Touchlink can also the used to gather device type and endpoint information from other devices that are already on the network - for example, devices brought into the network by a different Touchlink initiator. In this case, no network parameters are exchanged, but suitable bindings are created.

5.3.4 Allowing Other Devices to Join

The Color Scene Controller supports Network Steering for a device on the network. This process opens the network for other devices to join through association, but the controller must be part of the network. To do this:

1. On the Color Scene Controller, press the **SW2** button if using the OM15082 Generic Expansion Board.
2. Trigger 'Network Steering for a device not on a network' on the devices wishing to join.

A Management Permit Join Request will be broadcast to the network to open a Permit Join window for 180 seconds, to allow the new devices to join.

It is not necessary to do this in order to bring a device into the network using Touchlink.

5.3.5 Binding Nodes

'Finding and Binding' is the process of service discovery in which controller type devices search for suitable controlled type devices and create bindings for future use. The Color Scene Controller supports this process as an initiator that will look for targets to bind to. To create these bindings:

1. Trigger Finding and Binding on all the targets requiring bindings.
2. On the Color Scene Controller, press the **SW4** button if using the OM15082 Generic Expansion Board.

The controller will send Identify Query Requests to determine a list of potential targets. It will further query them to determine their capabilities and create bindings to any which have cluster servers that match to the controller's cluster clients. Once a binding is created with a target, an Identify Stop command will be sent to the target to indicate success. The controller will create bindings that can be used for either unicasts or groupcasts, as directed by the controller application.

As an alternative to Finding and Binding as an initiator, Touchlink can be used if the target devices also support the Touchlink cluster. Device capability information will be gathered from the target, and bindings will be created as required.

5.3.6 Re-Joining the Network

The Color Scene Controller is a sleepy End Device. If not factory-new, it will issue a Network Rejoin Request each time it is powered on or woken from deep sleep. The rejoin will be attempted 10 times, by default, but this number can be configured in the Zigbee Base Device **bdb_options.h** file. If all attempts fail, the controller will go into deep sleep to preserve the battery but can be woken to try again (if required).

5.3.7 Performing a Factory Reset

The Color Scene Controller can be returned to its factory-new state, erasing all persisted data except the outgoing network frame counter. To do this:

- If using the OM15082 Generic Expansion Board:
  - **a)** Press and hold down the **USER INTERFACE** button on the Carrier Board.
  - **b)** Press and release the **RESET** button on the Carrier Board.
  - **c)** Release the **USER INTERFACE** button.

The Color Scene Controller will then unicast a Leave Indication to its parent, which will re-broadcast it to the old network. The controller will then delete all persistent data, other than the outgoing network frame counter, and perform a software reset.

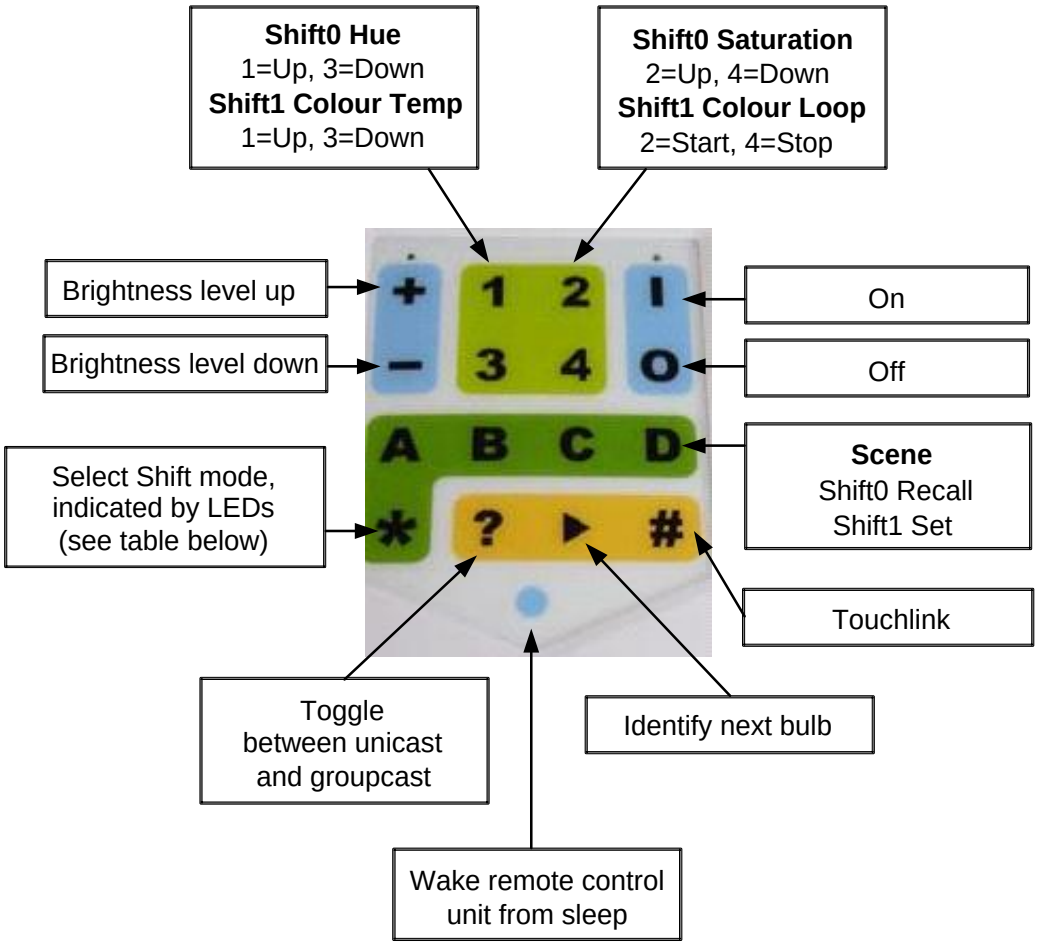
There are three supported over-the-air commands for removing a device from the network - these are:

- Network Leave Request without rejoin
- Management Network Leave Request ZDO command without rejoin
- Touchlink Factory Reset command

The Reset command of the Basic cluster will cause the ZCL to be reset to its factory-new defaults, resetting all attributes and configured reports. This will not remove the device from the network, as all network parameters, groups and bindings will remain in place.

5.3.8 Functionality of the Color Scene Controller

The button functions on the Windows-based Remote Control GUI are as shown below:



Note 1: When the controller device is in the factory-new state, only the buttons # and **+** are available (for commissioning ZLO devices into the network).

Note 2: The controller device goes to sleep after a while. So if the Remote Control Unit LEDs do not flash on pressing a touch-button, press hard on the button ● to wake the unit.

Note 3: The controller device can operate the light device(s) using one of two addressing modes – unicast or groupcast. Unicast mode allows the user to operate a single light device. Groupcast mode allows the user to operate a group of light devices simultaneously. Use the button '**?**' to toggle between these two modes.

The controller device can operate in four Shift modes (0, 1, 2 and 3) to accommodate maximum functionality. The Shift mode is indicated by a combination of two LEDs on the Remote Control Unit, as shown in the table below. You can press button ***** to move to the next Shift mode.

Shift Mode	Left LED	Right LED
Shift0	Off	Off
Shift1	On	Off
Shift2	Off	On
Shift3	On	On

Table 6: Shift Modes

The four tables below summarise the button functions in the four Shift modes.

Shift0 Mode Operation	Button
On: Send a command to switch on the light(s) The transmission mode will depend on the current mode selected.	**I**
Off with Effect: Send a command to switch off the light(s) using the 'Off with Effect' option - this has the effect of saving the current settings as the Global Scene.	**O**
Increase Brightness: Increase the brightness level of the light(s). If the light is off, this will switch on the light and then increase its level. The brightness will stop increasing when the button is released.	**+**
Decrease Brightness: Decrease the brightness level of the light(s). The brightness will stop decreasing when the button is released.	**–**
Enhanced Move Hue Up: Send a command to move the hue of the light(s) up. The movement will stop when the button is released.	**1**
Enhanced Move Hue Down: Send a command to move the hue of the light(s) down. The movement will stop when the button is released.	**3**
Increase Saturation: Send a command to move the saturation of the light(s) up. The movement will stop when the button is released.	**2**
Decrease Saturation: Send a command to move the saturation of the light(s) down. The movement will stop when the button is released.	**4**
Recall Scene 1: Groupcast a Recall Scene command to restore scene 1.	**A**
Recall Scene 2: Groupcast a Recall Scene command to restore scene 2.	**B**
Recall Scene 3: Groupcast a Recall Scene command to restore scene 3.	**C**
Recall Scene 4: Groupcast a Recall Scene command to restore scene 4.	**D**
Shift Menu: Cycle through the four Shift modes (0 → 1 → 2 → 3 → 0 etc).	*****
Groupcast/Unicast: Toggle between groupcast and unicast transmission modes. On waking from sleep, this mode will always be groupcast. After Touchlinking to a light, the mode will always be unicast with that light selected.	**?**
Select next light: Select the next light in the light database to be controlled by unicast. An Identify command will be sent to the relevant light, and unicast transmission mode will be selected.	**⏵**
Touchlink: Start Touchlink commissioning to add new devices to the network or to gather endpoint information about existing devices in the network.	**#**

Table 7: Button Functions in Shift0 Mode

Shift1 Mode Operation	Button
On: Send a command to switch on the light(s). The transmission mode will depend on the current mode selected.	**I**
Off with Effect: Send a command to switch off the light(s) using the 'Off with Effect' option - this has the effect of saving the current settings as the Global Scene.	**O**
Increase Brightness: Increase the brightness level of the light(s). If the light is off, this will switch on the light and then increase its level. The brightness will stop increasing when the button is released.	**+**
Decrease Brightness: Decrease the brightness level of the light(s). The brightness will stop decreasing when the button is released.	**–**
Increase Color Temperature: Send a command to increase the Color temperature of the light(s). The increase will stop when the button is released.	**1**
Decrease Color Temperature: Send a command to decrease the Color temperature of the light(s). The decrease will stop when the button is released.	**3**
Set the Color Loop: Send a command to the light(s) to start a Color loop. The light will start cycling through Colors.	**2**
Stop Color Loop: Send a command to the light(s) to stop the Color loop. A Color loop must be stopped before other Color control commands will be accepted by Color lights.	**4**
Store Scene 1: Groupcast a Store Scene command to save the current settings as Scene 1. Note that following the saving of a scene, the menu level will automatically revert to Shift0.	**A**
Store Scene 2: Groupcast a Store Scene command to save the current settings as Scene 2. Note that following the saving of a scene, the menu level will automatically revert to Shift0.	**B**
Store Scene 3: Groupcast a Store Scene command to save the current settings as Scene 3. Note that following the saving of a scene, the menu level will automatically revert to Shift0.	**C**
Store Scene 4: Groupcast a Store Scene command to save the current settings as Scene 4. Note that following the saving of a scene, the menu level will automatically revert to Shift0.	**D**
Shift Menu: Cycle through the four Shift modes (0 → 1 → 2 → 3 → 0 etc).	*****
Groupcast/Unicast: Toggle between groupcast and unicast transmission modes. On waking from sleep, this mode will always be groupcast. After Touchlinking to a light, the mode will always be unicast with that light selected.	**?**
Select next light: Select the next light in the light database to be controlled by unicast. An Identify command will be sent to the relevant light, and unicast transmission mode will be selected.	**⏵**
Find & Bind: Start find and bind as as a target to add new devices bindings to the binding table.	**#**

Table 8: Button Functions in Shift1 Mode

Shift2 Mode Operation	Button
On: Send a command to switch on the light(s). The transmission mode will depend on the current mode selected.	**I**
Off with Effect: Send a command to switch off the light(s) using the 'Off with Effect' option - this has the effect of saving the current settings as the Global Scene.	**O**
Increase Brightness: Increase the brightness level of the light(s). If the light is off, this will switch on the light and then increase its level. The brightness will stop increasing when the button is released.	**+**
Decrease Brightness: Decrease the brightness level of the light(s). The brightness will stop decreasing when the button is released.	**–**
Goto Hue and Saturation: Goes to a series of pre-defined enhanced hue values at maximum saturation. Steps up through the series.	**1**
Goto Hue and Saturation: Goes to a series of pre-defined enhanced hue values at maximum saturation. Stepsdown through the series.	**3**
No function assigned	**2**
No function assigned	**4**
No function assigned	**A**
No function assigned	**B**
Network steering: Trigger network steering for a device on a network, broadcast a Zigbee Management command to the network to instruct Routers to set their 'permit joining' state to TRUE for 180 seconds. This opens the network to classical joining.	**C**
Channel Change: Broadcast a Zigbee Management command to change the operational channel to one of the other ZLL primary channels, selected at random.	**D**
Shift Menu: Cycle through the four Shift modes (0 → 1 → 2 → 3 → 0 etc).	*****
Groupcast/Unicast: Toggle between groupcast and unicast transmission modes. On waking from sleep, this mode will always be groupcast. After Touchlinking to a light, the mode will always be unicast with that light selected.	**?**
Select next light: Select the next light in the light database to be controlled by unicast. An Identify command will be sent to the relevant light, and unicast transmission mode will be selected.	**⏵**
Touchlink: Start Touchlink commissioning to add new devices to the network or to gather endpoint information about existing devices in the network.	**#**

Table 9: Button Functions in Shift2 Mode

Shift3 Mode Operation	Button Sequence
On: Send a command to switch on the light(s). The transmission mode will depend on the current mode selected.	**I**
Off with Effect: Send a command to switch off the light(s) using the 'Off with Effect' option - this has the effect of saving the current settings as the Global Scene.	**O**
Factory Reset: Factory reset the Remote Control Unit, restoring the application and stack persistent data to its factory-new state.	**- + -**
No function assigned	**1**
No function assigned	**3**
Basic Reset: Send a Basic Cluster Reset Command to the light(s) to reset the attributes of the ZCL	**2**
No function assigned	**4**
Identify Effect Blink: Send a command to the light(s) to trigger the 'Blink' effect.	**A**
Identify Effect Breathe: Send a command to the light(s) to trigger the 'Breathe' effect.	**B**
Identify Effect Okay: Send a command to the light(s) to trigger the 'Okay' effect.	**C**
Identify Effect Channel Change: Send a command to the light(s) to trigger the 'Channel Change' effect.	**D**
Shift Menu: Cycle through the four Shift modes (0 → 1 → 2 → 3 → 0 etc).	*****
Groupcast/Unicast: Toggle between groupcast and unicast transmission modes. On waking from sleep, this mode will always be groupcast. After Touchlinking to a light, the mode will always be unicast with that light selected.	**?**
Select next light: Select the next light in the light database to be controlled by unicast. An Identify command will be sent to the relevant light, and unicast transmission mode will be selected.	**▸**
Touchlink: Start Touchlink to send a Factory New (reset) command to the target device. This button cannot be used to add new devices to the network.	**#**

Table 10: Button Functions in Shift3 Mode

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.

6.1 Overview

Support for the OTA Upgrade cluster as a client has been included for the Dimmer Switch and Color Scene Controller device. In order to build with these options, add `OTA=1` to the command line before building. This will add the relevant functionality to the Dimmer Switch and invoke post-build processing to create a bootable image and two upgrade images. The produced binaries will be stored in the **OTABuild** directory. By default, unencrypted binaries will be produced. In order to build encrypted binaries, add the `OTA_ENCRYPTED=1` option to the command line before building.

- If built for the JN5189 device then the internal Flash memory will be used to store the upgrade image.

The initial client binaries to be programmed into the JN5189 device are V1.bin files:

- **DimmerSwitch_NtagIcode_Ota_RamOpt_OM15082_V1.bin**
- **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V1.bin**

The Application Note *Zigbee 3.0 IoT Control Bridge [JN-AN-1247]* has OTA server functionality built into it. The included ZGWUI tool provides an easy to use interface to serve OTA images to client devices. The OTA images are V2/V3.ota files:

- **DimmerSwitch_NtagIcode_Ota_RamOpt_OM15082_V2.ota**
- **DimmerSwitch_NtagIcode_Ota_RamOpt_OM15082_V3.ota**
- **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V2.ota**
- **ColorSceneController_NtagIcode_Ota_RamOpt_OM15082_V3.ota**

6.2 OTA Upgrade Operation

Follow the instructions in *Zigbee 3.0 IoT Control Bridge (JN-AN-1247)* to serve a .ota upgrade to a client device.

Any devices with OTA clients in the network will periodically send Match Descriptor Requests in order to find an OTA server. Once a server responds, it will then be sent an IEEE Address Request in order to confirm its address details. After this, the clients will periodically send OTA Image Requests to determine whether the server is hosting an image for that client device. In response to the Image Request, the server will return details of the image that it is currently hosting - Manufacturer Code, Image Tag and Version Number. The client will check these credentials and decide whether it requires this image. If it does not, it will query the server again at the next query interval. If the client does require the image, it will start to issue Block Requests to the server to get the new image. Once all blocks of the new image have been requested and received, the new image will be verified, the old one invalidated, and the device will reboot and run the new image. The client will resume periodically querying the server for new images.

6.3 Image Credentials

There are four main elements of the OTA header that are used to identify the image, so that the OTA client is able to decide whether it should download the image. These are Manufacturer Code, Image Type, File Version and OTA Header String:

- **Manufacturer Code:** This is a 16-bit number that is a Zigbee-assigned identifier for each member company. In this application, this number has been set to 0x1037, which is the identifier for NXP. In the final product, this should be changed to the identifier of the manufacturer. The OTA client will compare the Manufacturer Code in the advertised image with its own and the image will be downloaded only if they match.
- **Image Type:** This is a manufacturer-specific 16-bit number in the range 0x000 to 0xFFBF. Its use is for the manufacturer to distinguish between devices. In this application, the Image Type is set to the Zigbee Device Type of the switch - for example, 0x0104 for a Dimmer Switch or 0x1104 if the image is transferred in an encrypted format. The OTA client will compare the advertised Image Type with its own and only download the image if they match. The product designer is entirely free to implement an identification scheme of their own.
- **File Version:** This is a 32-bit number representing the version of the image. The OTA client will compare the advertised version with its current version before deciding whether to download the image.
- **OTA Header String:** This is a 32-byte character string and its use is manufacturer-specific. In this application, it is possible (through a build option) for the OTA client to compare the string in the advertised image with its own string before accepting an image for download. If the strings match then the image will be accepted. In this way, the string can be used to provide extra detail for identifying images, such as hardware sub-types.

6.4 Encrypted and Unencrypted Images

OTA images can be provided to the OTA server in either encrypted or unencrypted form. Encrypting the image will protect sensitive data in the image while it is being transferred from the manufacturer to the OTA server. Regardless of whether the image itself is encrypted, the actual transfer over-air will always be encrypted in the same way as any other on-air message. The encryption key is stored in protected e-fuse and is set by the manufacturer.

6.5 Upgrade and Downgrade

The decision to accept an image following a query response is under the control of the application. The code, as supplied, will accept an upgrade or a downgrade. As long as the notified image has the right credentials and a version number which is different from the current version number, the image will be downloaded. For example, if a client is running a v3 image and a server is loaded with a v2 image then the v2 image will be downloaded. If it is required that the client should only accept upgrade images (v2 -> v3 -> v5), or only accept sequential upgrade images (v2 -> v3 -> v4 -> v5) then the application callback function that handles the Image Notifications in the OTA client will need to be modified.

7 Zigbee Green Power (GP) Energy Harvesting Switch

This Application Note also provides an example software implementation of a Zigbee Green Power (GP) energy harvesting switch device based on the following switches:

- Level Control switch
- On/Off switch

For information on the NXP implementation of Zigbee Green Power, refer to the *Zigbee 3.0 Green Power User Guide [JN-UG-3134]*.

> **Note:** This GP device could be implemented on Energy Harvesting (EH) hardware, although in this demonstration EH hardware is not used.

7.1 Using GpEhSwitch

This section describes how to commission and operate the GP Switch in a Zigbee 3.0 network with GP 'infrastructure devices'. The network can be set up using:

- Control Bridge application from the Application Note JN-AN-1247
- One of the following devices with GP support (GP infrastructure devices) from the Application Note JN-AN-1244: Dimmable Light, Extended Color Light or Color Temperature Light

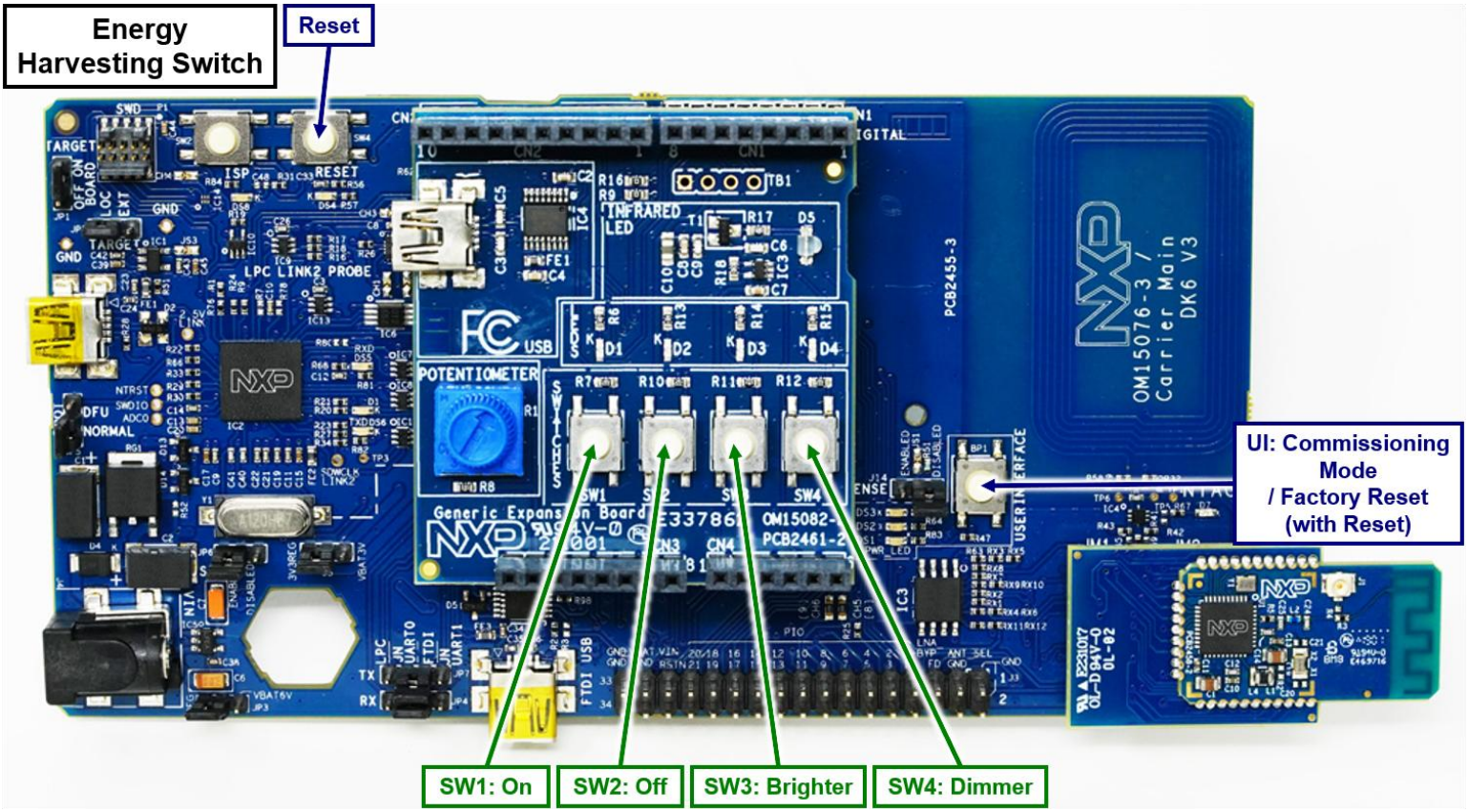
To use this application, you must have programmed the application **GpEhSwitch_OM15082.bin** into the DK006 module on a Carrier Board fitted with the OM15082 Generic Expansion Board.

> **Note:** To use this application which is based on the GP Switch device, you will also need to implement a paired light device with GP support, as described in the Application Note *Zigbee 3.0 Light Bulbs [JN-AN-1244]*.

7.1.1 GP Switch Functionality

The GP Switch application resides on a node fitted with the OM15082 Generic Expansion Board which acts as a GP Energy Harvesting switch. This switch device can be used to perform the commissioning of the light devices and control them.

The GP Switch functionality is implemented on the OM15082 Generic Expansion Board as shown in the following diagram:



7.1.2 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 pressing and releasing the **RST** button while holding down the **USER INTERFACE** button (both buttons are on the Carrier Board).

7.1.3 Commissioning

To commission the GP Switch to operate with one or more lights in a Zigbee 3.0 network that support the GP Combo Basic device (to get the light into the network, refer to the Application Note JN-AN-1244 document), follow the procedure below:

1. Put a light into Commissioning mode (this step is light-specific). The light will start flashing to identify itself.
2. On the GP Switch node, press the **SW1** button on the OM15082 Generic Expansion Board repeatedly at one-second intervals until the light stops flashing and returns to its original state (the GP Switch sends commissioning packets to the light node on each button-press).

Note: The **SW1** button should not be pressed too fast nor too slow. It should be pressed at approximately one-second intervals. A light node will clear the buffered packets for the GP Switch after 5 seconds, so the switch should receive the packets within 5 seconds.

The GP Switch is now paired with the light node and can be used to control the light. The **SW1-SW4** buttons on the OM15082 Generic Expansion Board of the GP Switch can be respectively used to send On, Off, Brighter and Dimmer commands to the light node.

3. To commission the GP Switch with another light, press the **USER INTERFACE** Commissioning button on the Carrier Board of the GP Switch and then repeat Steps 1-2 for the new light.

Note: To decommission the GP Switch from a network, press the **USER INTERFACE** Commissioning button on the Carrier Board of the GP Switch followed by the **SW2** button on the OM15082 Generic Expansion Board. This will send out a decommissioning command and cause the switch to be removed from network.

Note: After decommissioning, the frame counters are not reset on the lights. Therefore, a switch cannot be factory-reset and used in a previously commissioned network without clearing Sink/Proxy table entries on the lights. Decommissioning is typically done when the switch needs to operate in another network.

7.2 GpEhSwitch Application Code

The code required to build the GP Switch application can be found within the Application Note package in the **GpEhSwitch/Source** directory.

Note the following:

- The button initialisation and button functionality mapping is present in the files **EH_Button.c/h**
- The MicroMAC related functionality is included in **EH_IEEE_802154_Switch.c/h**
- The Green Power command construction is in **EH_IEEE_Commands.c/h**
- The Green Power features are in **EH_IEEE_Features.c/h**
- The Flash reads/writes for persistence storage are in **app_nvm.c/h**

7.3 Configuration Setting

The GP Switch configuration is specified in **EH_Switch_Configurations.h**.

The following macros are used for the configuration settings:

Macro	Description
GPD_DEFAULT_CHANNEL	This macro should contain the default operating channel of the device. This channel may be over-ridden during commissioning. This will be the default operating channel if channel requests and channel configurations are not supported.
GPD_FIXED	This macro should be defined for a fixed, non-movable GP Switch.
GPD_NO_OF_COMMANDS_IN_ OPERATIONAL_CHANNEL	The number of commands to send in a channel on each button-press.
GPD_SUPPORT_PERSISTENT_DATA	If data needs to be stored in persistent memory, this macro must be configured.
GPD_SOURCE_ID	4-byte GPD source address of the GP Switch.
GPD_WITH_SECURITY	If data needs to be secured, this macro must be configured.
GPD_DEFAULT_PANID	This macro should contain the initial PAN ID of the device. This PAN ID may be over-ridden during commissioning.
GPD_MAX_PAYLOAD	The maximum payload size for the GP Switch.
GPD_TYPE	The GP source node type. The possible values are: GP_LEVEL_CONTROL_SWITCH GP_ON_OFF_SWITCH
GPD_SEND_CHANNEL_REQUEST	To support channel request and channel configuration commands, this macro must be defined.
GPD_NO_OF_CHANNEL_PER_COMM_ ATTEMPT	The number of channel request commands sent in a channel on each button-press.
PRIMARY_CHANNELS	A list of the channel numbers that will be considered for commissioning.
SECONDARY_CHANNELS	A list of the channel numbers that will be considered for commissioning. Note that enabling all channels will increase the commissioning time. This list should be enabled only if required.
GPD_KEY_TYPE	The key type supported by the switch.
GPD_KEY	The key that is configured on the switch.
GPD_RX_ENABLE	Indicates that the switch is Rx capable.
GPD_RX_AFTER_TX	Indicates that the switch is capable of receiving for a short time after transmitting a request
GPD_NO_OF_REQ_BEFORE_RX	The number of commands transmitted in a channel during commissioning before going into receive mode.
GPD_REQ_PANID	Enables a request for a PAN ID during commissioning.
MOVE_RATE	The rate to be specified in the move up/move down commands.

Apart from **EH_Switch_Configurations.h**, the following configuration can be done in the makefile in the **GpEhSwitch/Build** directory:

- SWITCH_TYPE_OM15082: This macro must be set to 1 when using the OM15082 Generic Expansion Board from the DK006 (this is the default and is set in the makefile).
- TRACE: Debug prints can be enabled by setting this macro to 1.

8 Zigbee 3.0 Coin Cell Switch

This Application Node also provides an example of a software implementation for a low power energy switch which can be powered from a coin cell. The current device is based on the following switches:

- Level Control switch
- On/Off switch

> **Note:** This low power device could be implemented on DR1236 hardware or on OM15076-3 Carrier Board with DK006 module and OM15082-2 Generic Expansion Board, although in this demonstration only DR1236 hardware is used.

For information about the DR1236 hardware refer to the *JN518x Coin-cell Switch Reference Design [JN-RD-6057].*

8.1 Loading the application

The table below indicates the low power coin cell switch binary files supplied with this Application Note as well as the hardware components on which the binary can be used. Those files are located in the **Build** directory of the Coin Cell Switch application.

Hardware Components	Binary File
DR1236 Coin-cell Switch Board	**CoinCellSwitch_MinRetention_DR1236.bin** **CoinCellSwitch_Ota_MinRetention_DR1236_V1.bin**

The binary file can be loaded into the Flash memory of the DR1236 Coin-cell Switch Board using the JN518x Flash Programmer [JN-SW-4407], as described in Loading the Applications with some changes. Additional hardware is needed to connect the DR1236 hardware with the USB port of your PC. DR1236 has a header which is compatible with DR1128 USB Programming dongle. For information on how to connect the USB Programming dongle and the DR1236 hardware, please refer to *JN518x Coin-cell Switch Reference Design [JN-RD-6057].*

The USB Programming Dongle will put automatically the Coin Cell Switch in programming mode, so skip the 5th step from the Loading the Applications section.

8.2 Using the Coin Cell Switch

This section describes how to commission and operate the Coin Cell Switch application in a Zigbee 3.0 network. To use this application, you must have programmed the relevant application binary into the wireless microcontroller on the DR1236 board:

- **CoinCellSwitch_MinRetention_DR1236.bin**
- **CoinCellSwitch_Ota_MinRetention_DR1236_V1.bin**

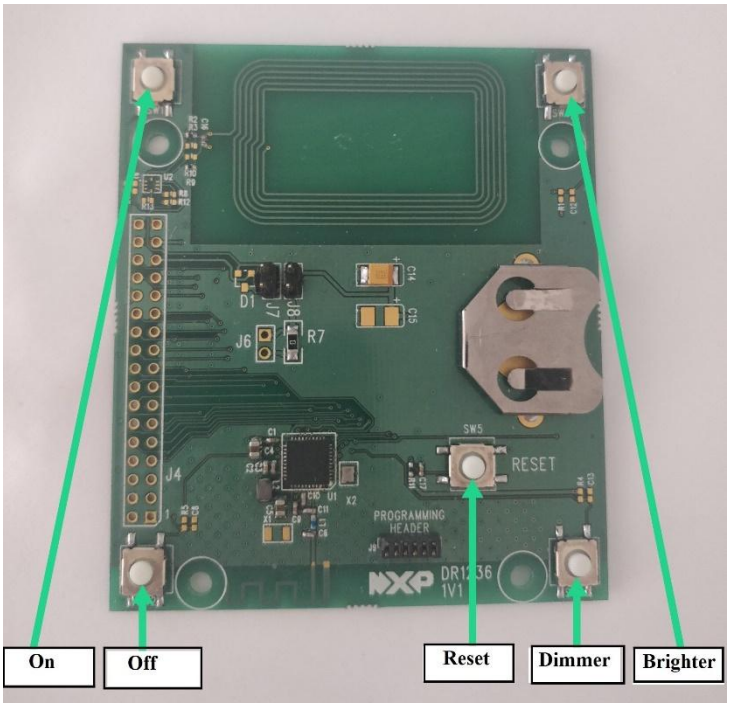
> **Note:** To use this application which includes the On/Off and Level Control Clusters, you will also need to implement the paired Dimmable Light device type as described in the Application Note *Zigbee 3.0 Light Bulbs [JN-AN-1244]*.

8.2.1 Coin Cell Switch Functionality

The low power Coin Cell Switch device resides on a node (fitted with the DR1236 board) which acts as a sleepy End Device in the network. As an End Device, it can only join an existing network and cannot form a new network. This switch device is capable of controlling other devices that supports the On/Off and/or Level Control clusters as servers. The device supports the following clusters as clients:

- Operational clusters: On/Off, Level Control
- Support clusters: Basic, Identify

The physical buttons on the board provide the functionality indicated in the diagram below:



8.2.2 Joining a Network

The Coin Cell Switch device is a sleepy End Device and, as such, must join an existing Zigbee network. The network join process is as follows:

1. The Network Steering process starts automatically after the node is programmed.
2. The coin cell switch will perform network discovery across all channels in the primary and secondary sets and attempts to associate with any discovered networks that have the 'Permit Join' bit set in their beacons. If it does not find an open network on the current channel, the coin cell switch goes to sleep for one second before it starts to perform network discovery on the next channel.
3. The outcome of Network Steering is indicated as follows:
   - If a suitable open network is found ('Permit Join' is true), the switch will attempt to join into network and after a successful join it will enter into 'Fast Poll' mode for 5 minutes.
   - The Coin Cell Switch scans for 3 times all channels in the attempt to find a suitable network. If no suitable network is found, the coin cell switch goes into deep sleep. It can be woken up from deep sleep by pressing the **RESET** button and the Network Steering process will start again.

8.2.3 Binding Nodes

8.2.3.1 Set bindings with light nodes

The light nodes can be controlled by the Coin Cell Switch only if they are bound to the node. The procedure to bind the switch with the light nodes is described below:

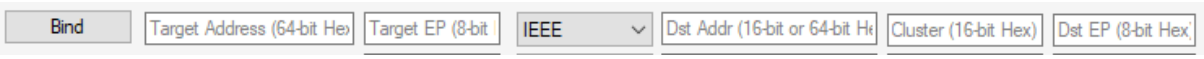
1. Put the lights that need to be commissioned with the Coin Cell Switch into 'Finding and Binding' mode (this step is light-specific). The lights will start to flash in Identify mode.
2. Press and hold down the **SW4** button on the DR1236 board.
3. Now press and hold down the **SW3** button on the DR1236 board. The Coin Cell Switch supports 'Finding and Binding' as an Initiator that tries to find targets to bind to. It will send Identify Query Requests to determine a list of potential targets. It will further query them to determine their capabilities and create bindings to any which have cluster servers that match to the coin cell switch's cluster clients (On/Off and/or Level Control Clusters). Once a binding with a target is created, an Identify Stop command will be sent to the target to indicate success.
4. After the target stops to identify itself, release **SW3** button followed by the release of **SW4** button.

8.2.3.2 Set bindings for background events

By default, the Coin Cell Switch wakes up every 5 minutes to perform some background events. The user must specify whom the Coin Cell Switch have to address the events or it will pass over them. For every background task, the Coin Cell Switch has to set bindings with another device from the network which supports the cluster to which the event belongs as client or server depending on the cluster implementation on the Coin Cell Switch side.

Zigbee Gateway User Interface have to be used to set the bindings for the background events. It is located under the Tools folder in the *Zigbee 3.0 IoT Control Bridge Application Note [JN-AN-1247]*.

The bind command is found in the Management tab. The Target Address is the IEEE address of the Coin Cell Switch and the Destination Address is the IEEE address of the other device.



The clusters supported by the Coin Cell Switch which need bindings with other devices from the network are presented below:

- Poll Control Cluster – Cluster ID 0x0020
- Power Configuration Cluster – Cluster ID 0x0001
- OTA Cluster – Cluster ID 0x0019 (only for the bin file which supports the OTA cluster)

By default, the Coin Cell Switch sleeps for 5 minutes and then wake up to verify if it has received any data while it was asleep. If many packets are expected over a short period of time, the end device should retrieve these packets as quick as possible. It periodically checks whether the poll mode requires to be changed. The automatic 'check-ins' are conducted with the device endpoint to which the local endpoint is bound.

The Power Configuration Cluster allows to obtain information about the power source of a device. By default, the Coin Cell Switch is configured to send a report which contains the battery voltage and battery percentage remaining details. The report will be sent every hour when the local endpoint is bound with the parent's endpoint.

As the Coin Cell Switch is a sleeping End Device, the OTA server cannot notify it of the availability of the new image. The Coin Cell Switch must poll the server periodically in order to establish whether new software is available. It will send Query Next Image Requests every hour if binding between the Coin Cell Switch and the OTA sever was previously established.

For more details about the above clusters, refer to *JN518x-Zigbee3-Cluster-Library [JN-UG-3132].*

8.2.4 Re-Joining the Network

The Coin Cell Switch is a sleepy End Device. If not factory-new, it will issue a Network Rejoin Request each time it is powered on or woken from deep sleep. The rejoin will be attempted 3 times, by default, but this number can be configured in the Zigbee Base Device **bdb_options.h** file. If all attempts fail, the controller will go into deep sleep to preserve the battery but can be woken to try again (if required).

8.2.5 Performing a factory reset

The Coin Cell Switch can be returned to its factory-new state, erasing all persisted data except the outgoing network frame counter. To do this using the DR1236 board:

a) Press and hold down the **RESET** button.

b) Press and hold down **SW1** button.

c) Release the **RESET** button.

d) Release the **SW1** button.

The End Device controller will then unicast a Leave Indication to its parent, which will re-broadcast it to the old network. The switch will then delete all persistent data, other than the outgoing network frame counter, and perform a software reset.

8.2.6 Functionality of the Coin Cell Switch

The Coin Cell Switch is a sleeping End Device which wakes up to perform periodic background events. The following section gives an overview of the switch behaviour.

The device begins in Network Steering mode where it scans all channels to attempt to join any open network. Once joined to a network, the device will be in fast poll mode for 5 minutes before performing the preconfigured reporting. Once completed, the switch will go to sleep for 5 another minutes.

During fast poll mode it is recommended to set bindings with the devices to whom the background events will be addressed, see section Set bindings for background events, otherwise you have to wait for a wakeup event.

By default, the Coin Cell Switch will wake up every 5 minutes to verify if it has received data while it was asleep and if it should change its poll mode in order to retrieve all those packets as soon as possible. If it hasn't received any data and it does not have other tasks to do, it will go back to sleep.

During the wakeup cycle, the Coin Cell Switch will also verify if it is the time to send the Power Configuration Report with both reportable attributes, Battery Voltage and Battery Percentage Remaining and if it is the time to query the OTA server for a new image.

Coin Cell Switch is designed to control the light bulbs which already have bound with the switch using the finding and binding mechanism (refer to Set bindings with light nodes section) .

The **SW1-SW4** buttons on the DR1236 board can be used to perform various operations, as follow:

- Press **SW1** button to send On command
- Press **SW2** button to send Off command
- Press **SW3** button to increase the light brightness
- Release **SW3** button when the light brightness has reached the desired brightness
- Press **SW4** button to decrease the light brightness
- Release **SW4** button when the light has reached the desired brightness

8.3 Coin Cell Switch Application Code

The code required to build the low power switch application can be found within the Application Note package in the **CoinCellSwitch, Common_CoinCellSwitch** and **Common** folders.

Note the following:

- The button initialisation and button functionality mapping is present in the files **app_coin_cell_switch_buttons.c/h**
- The tasks that the Coin Cell Switch will perform at eve

8.4 Configuration settings

The following macros are used for the configuration settings:

Macro	Description
APP_POLL_CONTROL_CHECKIN_INTERVAL_IN_QUARTER_SEC	This macro should contain the number of seconds, in quarter-seconds, after which the coin cell switch will ask if it has to change its poll mode in order to receive data. By default it is 1200 - meaning 5 minutes.
APP_POLL_CONTROL_SHORT_POLL_IN_QUARTER_SEC	This macro indicates the time between consecutive polls of the parent for data when the end device is in fast poll mode. It is defined in quarter-seconds. By default, it is 4 – 1 second.
APP_POLL_CONTROL_LONG_POLL_IN_QUARTER_SEC	This macro indicates the time between two consecutives polls of the parent for data when the end device is in normal poll mode. It is also specified in quarter-seconds. By default it is 1200 – 5 minutes.
APP_POLL_CONTROL_FAST_POLL_TIMEOUT_IN_QUARTER_SEC	This macro defines the time-interval, in quarter-seconds, for which the end device should stay in fast poll mode(if it is not overridden by another command).
ZLO_SYSTEM_MAX_REPORT_INTERVAL	This macro defines how often the reports which contains information about the end device battery is sent. It is in seconds and ,by default, it is 3600.
APP_SLEEP_BETWEEN_SCANS	This macro should be defined when sleeping after every channel scan is desired. By default, it is defined.
APP_SCAN_CHANNELS_ATTEMPTS	This macro indicates how many times all channels are scanned in order to find an open network, before the coin cell switch goes into deep sleep.
OTA_QUERY_INTERVAL_IN_SECS	This macro defines how often the coin cell switch will ask if a new image is available. It is defined in seconds.
SEND_TO_DEFAULT_IF_BINDING_ENTRY_EMPTY	When this macro is defined, the background events will be sent to the network coordinator if there is no entry in the binding table for the required cluster.

9 Developing with the Application Note

The example applications provided in this Application Note were developed using the:

- MCUXpresso IDE
- JN518x Zigbee 3.0 SDK
- K32W061/K32W041 Zigbee 3.0/Bluetooth SDK

These are the resources that you should use to develop JN518x Zigbee 3.0 applications, respectively. They are available free-of-charge to authorised users via the MCUXpresso web site.

Throughout your Zigbee 3.0 application development, you should refer to the documentation listed in Related Documents.

9.1 Compilation for Specific Chips/SDKs

The Application Notes are provided ready for compilation for a single chip on a single SDK, the default configuration is specified in the Release Notes for each Application Note. To alter the compilation for a different chip/SDK use comments near the top of the makefile to select the appropriate chip using the JENNIC_CHIP variable which will also select the appropriate SDK. This assumes that MCUXpresso and the SDK has been installed following the instructions in the *Zigbee 3.0 Getting Started Application Note [JN-AN-1260]*. The example below selects the K32W061 chip and the appropriate SDK:

```
# Set specific chip   (choose one)
JENNIC_CHIP            ?= K32W061
#JENNIC_CHIP           ?= K32W041
#JENNIC_CHIP           ?= JN5189
#JENNIC_CHIP           ?= JN5188
```


9.2 DimmerSwitch Application Code

This section describes the DimmerSwitch application code, which is located in the **DimmerSwitch**, **Common_Switch** and **Common** folders of the Application Note. You may wish to use this code as a basis for your own application development. You can rebuild your customised application as described in Rebuilding the Applications.

9.2.1 Operational State Machine

The operational state machine (**sDeviceDesc.eNodeState**) is located within **app_zlo_switch_node.c** for the Dimmer Switch. Additional states must be added to this switch statement if further operational modes are required. All the network and Zigbee Base Device related events are also handled inside **app_zlo_switch_node.c**.

9.2.2 Handling a Button Press and Release

Buttons are debounced in **APP_cbTimerButtonScan()** contained in **app_buttons.c**. Any button events are then passed into the **APP_msgAppEvents** queue for processing in **APP_ZLO_vSwitch_Task()** in **app_zlo_switch_node.c** where the button event is eventually handled in **vApp_ProcessKeyCombination()**. Any alteration to the key map to allow different functionality should go in this function.

Similarly, the **vApp_ProcessKeyCombination()** function can be modified to add a function that is called upon release of a key.

The function that handles a button press and release for different operational modes is located in **app_switch_state_machine.c**

9.2.3 Handling NFC

app_ntag_icode.c contains the code that drives the NFC commissioning data exchange and initiates the joining process when valid data is read from the NTAG. This code uses the newer NTAG data format that employs Zigbee Installation Code.

9.2.4 Handling of Sleep

The Dimmer Switch has the following sleep configuration - it is awake for 6 seconds and then sleeps for 6 seconds. The Dimmer Switch goes into deep sleep after one minute, if no activity is detected.

The following macros control the sleep configurations:

- KEEP_ALIVETIME – defined inside the makefile to determine for how long the switch should be awake between soft sleeps.
- APP_LONG_SLEEP_DURATION_IN_SEC – defined inside **app_zlo_switch_node.c** to decide for how long the Dimmer Switch should remain in soft sleep.
- DEEP_SLEEP_TIME – defined inside **app_zlo_switch_node.h** determines after how many iterations of soft sleep the Dimmer Switch should go into deep sleep.

9.2.5 Handling of OTA Upgrade

During the upgrade process the following behaviour is expected:

- The Dimmer Switch will not enter ‘deep sleep’ mode once an OTA upgrade has been started, but soft/warm sleep cycles will continue.
- For OTA upgrade, the sleep cycle for the Dimmer Switch is configured in the makefile as ‘keep alive’ for 20 seconds and then sleep for 6 seconds.

9.3 Color Scene Controller Application Code

The code required to build the Color Scene Controller is held in directories within the Application Note.

- Code in the **Common\Source** directory is common between all the devices in this Application Note, such as event type definitions and definitions of persistent data record identifiers.
- Code in the **Common_Controller\Source** directory contains code that would be common to all types of controller device defined in the ZLO Specification. This provides the basic functionality of all these devices and could be reused if a new controller type were to be developed.
- Code in **ColorSceneController\Source** holds the files specific to the Color Scene Controller, such as endpoint registration and constructor, initialisation of Device IDs and Basic cluster attributes. The files to configure the Zigbee Base Device and Zigbee Cluster Library are also held here.

9.3.1 Touchlink Preconfigured Link Key

The Touchlink Preconfigured Link Key is used to encrypt the network key when passed to joining devices as part of Touchlink operations. The key that should be used in real world applications is a secret key and is supplied by the Zigbee Organisation upon application after successfully completing the ZLO Certification process. This key is not supplied in this Application Note.

The application has been set up to use the Certification Key as defined in the Base Device Behavior (BDB) Specification. Once in possession of the real Touchlink Preconfigured Link Key, the following changes need to be made in order to use this key rather than the Certification Key.

1. In the **zcl_options.h** file, add the following definition:

```
#define TL_SUPPORTED_KEYS ( TL_MASTER_KEY_MASK )
```

2. Copy the Zigbee-supplied key into the definition of *sTLMasterKey* in the file **bdb_link_keys.c** (located in **BDB\Source\Common**). This will make the key available to all applications built using the BDB component.

   Alternatively, you can over-ride the definition in the BDB component by including a definition in the device application (in **App_ColorSceneController.c**) and then including the following definition in **bdb_options.h**:

```
#define BDB_APPLICATION_DEFINED_TL_MASTER_KEY
```

9.3.2 Handling NFC

app_ntag_icode.c contains the code that drives the NFC commissioning data exchange and initiates the joining process when valid data is read from the NTAG. This code uses the newer NTAG data format that employs Zigbee Installation Code encryption and is used in the default builds.

9.3.3 Handling of OTA Upgrade

During the upgrade process the following behaviour is expected:

- The Dimmer Switch will not enter ‘deep sleep’ mode once an OTA upgrade has been started, but soft/warm sleep cycles will continue.

9.3.4 Common Controller Code

The following are the main application files and are common to all controller type devices.

app_start_controller.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_controller_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 which are passed to the menu event handler. Sleep scheduling and polling for data are also handled here.

app_menu_handler_OM15082.c contains a menu handler for the OM15082-based hardware. It interprets key-presses from the GUI and button-presses from the switches, and initiates the ZCL to send the appropriate ZCL commands to devices on the network.

app_Color_commands.c contains a collection of functions that interact with the Color Control cluster of the ZCL to send Color control commands.

app_general_commands.c contains a set of utility functions for the menu handler.

app_group_commands.c contains a collection of functions that interact with the Groups cluster of the ZCL to send group control commands.

app_identify_commands.c contains a collection of functions that interact with the Identify cluster of the ZCL to send identify control commands.

app_level_commands.c a collection of functions that interact with the Level Control Cluster of the ZCL to send level control commands.

app_on_off_commands.c contains a collection of functions that interact with the On/Off cluster of the ZCL to send on/off control commands.

app_scenes_commands.c contains a collection of functions that interact with the Scenes cluster of the ZCL to send scene recall and save commands.

zlo_zcl_controller_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_led_control.c contains the driver software to control the two LEDs on the DR1159 Remote Control Unit.

app_serial_interface.c contains functions to process a character received from the UART, manage the serial interface protocol, validate messages and pass key events to the application event queue. It also manages the protocol to send LED messages to the GUI. This file is only used for builds for the OM15082-based hardware.

uart.c contains an interrupt handler and deals with UART management for the serial interface. This file is only used for builds for the OM15082-based hardware.

9.3.5 Code Specific to Color Scene Controller

App_ColorSceneController.c contains code specific to the Color Scene Controller, dealing with endpoint registration and constructor, and initialisation of the Basic cluster attributes.

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.

10 Related Documents

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

- JN518x Production Flash Programmer User Guide [JN-UG-3127]
- JN518x Zigbee 3.0 Stack User Guide [JN-UG-3130]
- JN518x Zigbee 3.0 Device User Guide [JN-UG-3131]
- JN518x Zigbee 3.0 Cluster Library User Guide [JN-UG-3132]
- JN518x Zigbee 3.0 Green Power User Guide [JN-UG-3134]
- JN518x Encryption Tool [JN-UG-3135]

Important Notice

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V.

All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 21-Aug-2020

Document identifier: JN-AN-1245