Application Note: JN-AN-1189

ZigBee Home Automation Demonstration

This Application Note demonstrates a typical ZigBee Home Automation (HA) network based on the NXP JN516x wireless microcontroller. The supplied demonstration application employs Lighting devices (lights, sensors and switches) from the ZigBee Home Automation Profile Specification version 1.2.2. The application optionally demonstrates the Over-The-Air (OTA) download of application images from an OTA server in the network.

The application uses HA clusters to transfer data between the devices in a wireless network in order to control lights. The hardware for the devices is implemented using components from the NXP JN516x Evaluation Kits. The device software was developed using NXP Application Programming Interfaces (APIs).

The Application Note also includes a demonstration of a typical ZigBee Home Automation (HA) network with Green Power (GP) support. This solution employs a GP device from the ZigBee Green Power Profile Specification version 1.0.

1 Introduction

This Application Note provides an example ZigBee Home Automation (HA) wireless network solution which uses the NXP JN516x-EK001 or JN516x-EK004 Evaluation Kit. This demonstration employs HA Lighting devices, allowing the user to control lights from controller devices such as dimmer switches and sensors. The demonstration also allows Over-the-Air (OTA) updates of application images on light devices to be performed.

Components from the evaluation kit can be used as HA devices and other devices as follows:

- One Lighting/Sensor Expansion Board (DR1175) is used as a Dimmable Light
- One Lighting/Sensor Expansion Board (DR1175) is used as a Colour Dimmable Light
- One Lighting/Sensor Expansion Board (DR1175) is used as a Light Sensor
- The Remote Control Unit (DR1159) can be used as either of the following HA devices:
    - Colour Dimmer Switch
    - Remote Control (must be used when Light Sensor is included in network)
- One Generic Expansion Board (DR1199) is used as a Dimmer Switch
- One Generic Expansion Board (DR1199) is used as an Occupancy Sensor
- One Generic Expansion Board (DR1199) is used to simulate a Smart Plug
- One Generic Expansion Board (DR1199) is used as a Green Power (GP) Switch
- One USB Dongle (DR1198) is used as the ZigBee Co-ordinator node for the network
- One USB Dongle (DR1198) is used as an OTA server device

The expansion boards are each fitted to a Carrier Board (DR1174). The evaluation kit and its components are described in the relevant kit's User Guide (JN-UG-3093 or JN-UG-3108).

There is no need to use every supplied HA device type in your demonstration network. You can choose which device types to use, provided that a sensible combination is selected.

The device software was developed using the following NXP Application Programming Interfaces (APIs): ZigBee PRO APIs, JenOS APIs, HA API, GP API and JN516x Integrated Peripherals API. These APIs are described in their own User Guides.

A list of useful reference documents is provided in Section 10.

1.1 System Overview

This example of an HA network consists of the following Lighting devices from the HA profile: Dimmable Light, Colour Dimmable Light, Dimmer Switch, Colour Dimmer Switch, Occupancy Sensor and Light Sensor. From the HA profile's Generic devices, the Remote Control device should be used instead of the Colour Dimmer Switch when the Light Sensor is used. There is also an OTA server device which can store and serve application images in the network. In the Green Power (GP) demonstration, a GP Switch device from the GP profile is used. The sub-sections below provide a brief introduction to these device types. Advanced user information is provided in Section 8.5.

For details of all the above HA devices and the clusters that they include, refer to the *ZigBee Home Automation User Guide (JN-UG-3076)* and the *ZigBee Cluster Library User Guide (JN-UG-3103)*. The GP device and cluster are described in the *ZigBee Green Power User Guide (JN-UG-3095)*.

1.1.1 Dimmable Light

The HA Dimmable Light device resides on a permanently-powered node (DR1175 Lighting/Sensor Expansion Board) which acts as a Router in the network. This light node will attempt to join an existing network.

The implemented Dimmable Light device includes the mandatory clusters defined for this device in the HA Specification. Special versions of the Dimmable Light device are also provided, as follows:

- A Dimmable Light device which includes the GP cluster is provided for the GP demonstration
- A Dimmable Light device which includes the Simple Metering cluster is provided in order to be eligible for Icontrol certification (see Section 11.3)

This light node allows user interaction through power-cycling in order to perform various operations, as indicated below:

- To put the light node into Identify mode, power-cycle the node CONFIG_FIND_BIND_POWER_CYCLES times - this number can be changed in the configuration makefile, but the default value is **3**.
- To clear the Binding and Group tables on the light node, power-cycle the node CONFIG_FACTORY_RESET_POWER_CYCLES times - this number can be changed in the configuration makefile, but the default value is **5**.
- To perform a 'Factory New' reset and leave the network, power-cycle the node CONFIG_FACTORY_NEW_POWER_CYCLES times - this number can be changed in the configuration makefile, but the default value is **7**. This also clears context data on the node – see Section 5.7.2.

Note 1: For the above operations to be performed successfully, the power-on logic must not reset itself before the next user interaction. The logic will be reset if the light is left on for more than 2 seconds.

Note 2: The light will briefly flash On, Off and then On again to indicate the acceptance of the user input for the above three operations.

Note 3: If the light is not in the network, there will be a 'Breathe' indication. This effect can be controlled by the compiler flag `BREATH_EFFECT` - by default, this is enabled in this application.

Note 4: By default, following a power-cycle the Dimmable Light will restore its last previous light level. If you do not want to resume from the last light level, you must remove or comment out the following line in the manufacturer configuration file **manu_config.mk** for the device:
`RESTORE_DIM_LEVEL ?= 1`

1.1.2 Colour Dimmable Light

The HA Colour Dimmable Light device resides on a permanently-powered node (DR1175 Lighting/Sensor Expansion Board) which acts as a Router in the network. This light node will attempt to join an existing network.

The implemented Colour Dimmable Light device includes the mandatory clusters defined for this device in the HA Specification. A special version of the Colour Dimmable Light device which includes the GP cluster is provided for the GP demonstration.

This light node allows user interaction through power-cycling in order to perform various operations. This is exactly the same as described for the Dimmable Light in Section 1.1.1.

1.1.3 Dimmer Switch

The HA Dimmer Switch device resides on a node (DR1199 Generic Expansion Board) which acts as an End Device in the network. This switch device can be used to perform the commissioning of the light devices and control them. For the operational details, refer to Section 5.1.

The implemented Dimmer Switch device includes the mandatory and certain optional clusters defined for this device in the HA Specification.

1.1.4 Colour Dimmer Switch

The HA Colour Dimmer Switch device resides on a node (DR1159 Remote Control Unit) which acts as an End Device in the network. This switch device can be used to perform the commissioning of the light devices and control them. For the operational details, refer to Section 5.2.

The implemented Colour Dimmer Switch device includes the mandatory and certain optional clusters defined for this device in the HA Specification.

1.1.5 Occupancy Sensor

The HA Occupancy Sensor device resides on a node (DR1199 Generic Expansion Board) which acts as an End Device in the network. This sensor device can be bound to a light device through EZ-mode Commissioning 'Find and Bind'. Once it has successfully bound, the sensor can report occupancy to the light device. For the operational details, refer to Section 5.3.

The implemented Occupancy Sensor device includes the mandatory and certain optional clusters defined for this device in the HA Specification.

1.1.6 Light Sensor

The HA Light Sensor device resides on a node (DR1175 Lighting/Sensor Expansion Board) which acts as an End Device in the network. The Light Sensor can report a light luminance-level to an application that controls light dimming based on the ambient light-level (the controlled light is located on another DR1175 board). In this demonstration, it can be used to control any HA Lighting device that supports the Level Control cluster, e.g. Dimmable Light and Colour Dimmable Light. The Light Sensor device is bound to the Remote Control device (see Section 1.1.7) which forms a group of lights to control (the binding and grouping is done through EZ-mode Commissioning). For the operational details of the Light Sensor, refer to Section 5.4.

The implemented Light Sensor device includes the mandatory and certain optional clusters defined for this device in the HA Specification.

1.1.7 Remote Control

The HA Remote Control device resides on a node (DR1159 Remote Control Unit) which acts as an End Device in the network. This device can be used to perform the commissioning of the light devices and control them. For the operational details, refer to Section 5.2.

This device is used instead of the Colour Dimmer Switch when the Light Sensor is employed in the network. In this case, it allows the DR1159 Remote Control Unit to receive a luminance-level measurement from the Light Sensor, calculate an appropriate target light-level and send a ‘Move to Level’ command to the Dimmable Light (this requires the Illuminance Measurement cluster which is not available in the Colour Dimmer Switch device).

The implemented Remote Control device includes the mandatory and certain optional clusters defined for this device in the HA Specification.

1.1.8 Smart Plug (Mains Power Outlet)

The Smart Plug device is referred to as the “Mains Power Outlet” and resides on a node (DR1199 Generic Expansion Board) which acts as an End Device in the network. In the real world, such a device would be used to monitor the instantaneous power consumption of an attached appliance (by means of the Simple Metering cluster) but in this demonstration the device can simply be switched on and off. For the operational details, refer to Section 5.6.

The implemented Smart Plug device includes the mandatory and certain optional clusters defined for this device in the HA Specification.

1.1.9 OTA Server

The OTA Server resides on a node (DR1198 USB Dongle) which acts as a Router in the network. The node joins the network and sends out an OTA Image Notify command on power-up. It responds to OTA client requests and serves application upgrade images which are stored in external Flash memory on the target client.

1.1.10 Green Power Switch

The Green Power (GP) Switch device resides on a node (DR1199 Generic Expansion Board) which acts as a ZigBee Green Power Device. It can be used to control a light in the network. In practice, it may be an Energy Harvesting (EH) node but this is not the case in this demonstration. For the operational details, refer to Section 5.5.

2 Compatibility

The software provided with this Application Note is intended for use with the following evaluation kit and SDK (Software Developer's Kit) versions:

Product Type	Part Number	Version or Build
Evaluation Kit	JN516x-EK001	-
	JN516x-EK004	-
JN516x ZLL/HA SDK	JN-SW-4168	1470
'BeyondStudio for NXP' Toolchain	JN-SW-4141	1308

3 Additional Hardware Options

The supplied software can also be built for NXP's monochrome LED bulb Reference Designs (building the application is described in Section 10.3). Use the build configuration for the appropriate hardware Reference Design build type within the Eclipse development environment.

- HW Ref DR1190 – SSL2108 Bulb
- HW Ref DR1192 – SSL2108 SYNC Bulb

Similarly, there are build configurations for applications that run on the following NXP hardware supporting colour lights:

- HW Ref DR1121 – SSL2108 SYNC Colour Controlled Tuneable White Light
- HW Ref DR1173 – RGB Colour Light

For details of these target Reference Designs and hardware, contact NXP.

4 Loading the Application

Table 1 below lists the application binary files supplied with this Application Note and indicates the JN516x Evaluation Kit components on which the binaries can be used. These files are located in the **Build** directories for the relevant devices. Most binaries (except for 'RemoteControl') are provided for JN5168 and JN5169 – in the table, <x> can be 8 or 9.

Application Binary	Expansion Board (+ Carrier Board)			Remote Control Unit	USB Dongle
	Generic	LCD	Lighting/Sensor		
Coordinator_JN516<x>.bin					■
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO.bin *			■		
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_GP.bin */**			■		
DimmableLightOpenHome_JN5169_DR1175.bin			■		
ColorDimmableLight_JN516<x>_DR1175_LED_EXP_RGB.bin			■		
ColorDimmableLight_JN516<x>_DR1175_LED_EXP_RGB_GP.bin **			■		
RemoteControl_JN5168_DR1159.bin				■	
LightSensor_JN516<x>_DR1175.bin			■		
OccupancySensor_JN516<x>_DR1199.bin	■				
DimmerSwitch_JN516<x>_DR1199.bin	■				
ColorDimmerSwitch_JN516<x>_DR1159.bin				■	
MainsPowerOutlet_JN5169_DR1199.bin	■				
EH_Switch_JN516<x>_DR1199.bin **	■				

Table 1: Device Type – Evaluation Kit Compatibility Matrix

***** Versions of the Dimmable Light binaries are also supplied for the DR1190 and DR1192 bulbs (see Section 3):
DimmableLight_JN516<x>_DR1190_MONO.bin
DimmableLight_JN516<x>_DR1190_MONO_GP.bin
DimmableLight_JN516<x>_DR1192_MONO.bin
DimmableLight_JN516<x>_DR1192_MONO_GP.bin

**These files are used for ZigBee Green Power (GP) – see Section 7.

The supplied application binaries (see Table 1 above) can be loaded into the corresponding evaluation kit components using the JN516x Flash Programmer within BeyondStudio for NXP or the JN51xx Production Flash Programmer (JN-SW-4107). Note that the JN51xx Production Flash Programmer must be used to load binaries (such as OTA upgrade images) into JN516x external Flash memory.

> **!** ***Caution:*** *If loading this demonstration software for the first time, the persistent data must be cleared in each device – refer to Section 5.7.*

> **Note:** Pre-built binary files are also supplied for the OTA Upgrade feature and are located in the **Build/OTABuild** directories for the relevant devices. OTA Upgrade and the supplied files are described in Section 8.

5 Device Functionality

This section describes how to use the HA demonstration once the JN516x evaluation kit hardware has been programmed with the relevant binaries (as described in Section 4).

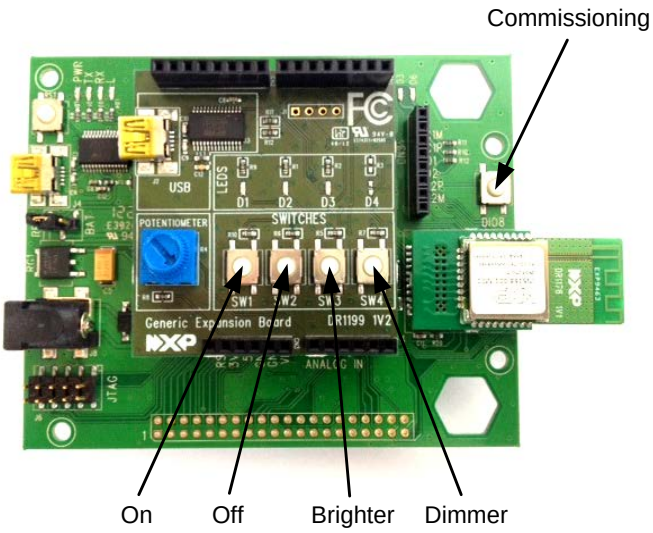
Note 1: The binary files for the JN5168 chip are specified in this section. If the JN5169 chip is to be used, the JN5169 binary versions must be used (except for the Remote Control Unit, which is JN5168 only).

Note 2: Setting up a network which contains a number of these devices is described in Section 6.

5.1 Dimmer Switch Functionality

The binary file **DimmerSwitch_JN5168_DR1199.bin** should be programmed into the JN5168 device associated with a DR1199 Generic Expansion Board to obtain the Dimmer Switch functionality. This switch allows the commissioning and control of lights.

The Dimmer Switch functionality is detailed below:



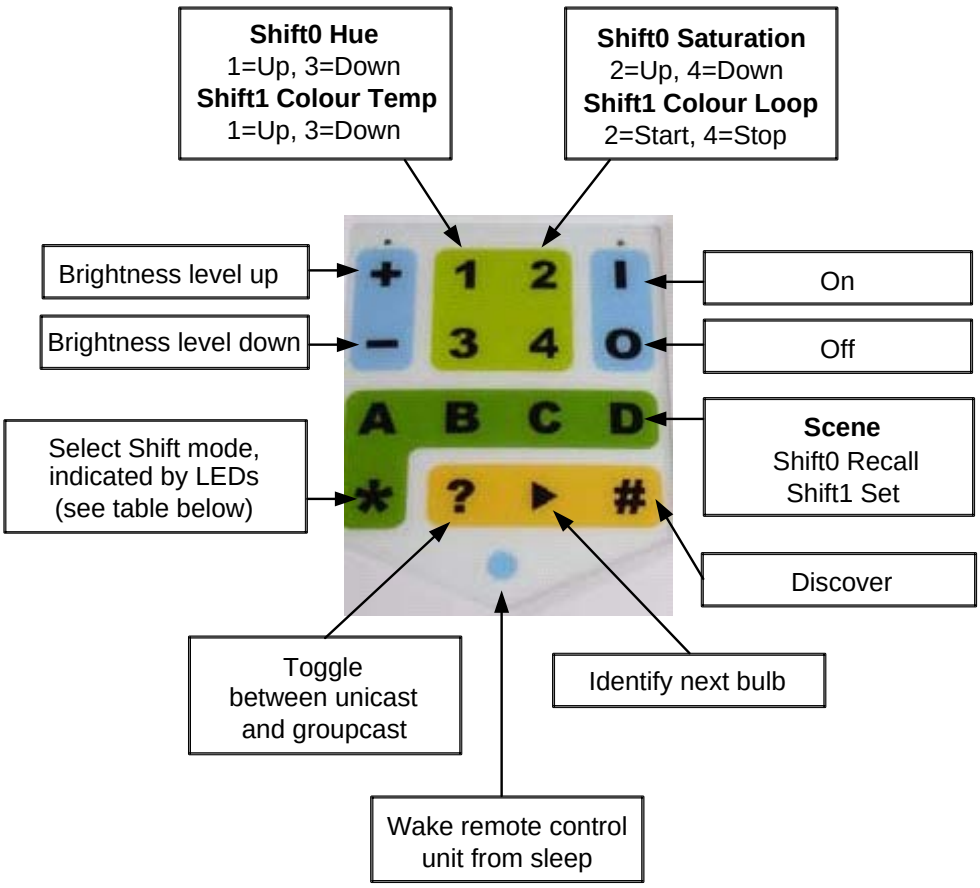
5.2 Colour Dimmer Switch and Remote Control Functionality

The JN5168 device of the DR1159 Remote Control Unit can be programmed with either of the following binary files:

- **ColorDimmerSwitch_JN5168_DR1159.bin** if the unit is to be used as a Colour Dimmer Switch device
- **RemoteControl_JN5168_DR1159.bin** if the unit is to be used as a Remote Control device (in conjunction with the Light Sensor device)

Both binary files provide the same functionality for the commissioning and control of lights. The Remote Control device provides additional functionality for use with the Light Sensor device, as indicated in Section 1.1.7.

The Remote Control Unit keypad functionality (for both device types) is detailed below:



Notes:

- When the unit is in the factory-new state, no buttons are available.
- The unit goes to sleep with RAM ON (this is the default sleep mode for the Remote Control Unit) after an inactive period of 20 seconds. The unit can wake from this sleep with a long press of any of the touch buttons. To preserve battery-life, the debounce time for button-press detection is increased to 800 ms during sleep, so during this period the user must press a button for longer than 800 ms for the unit to successfully detect the button-press.
- When deep sleep is enabled, the unit enters deep sleep mode after a minimum inactive period of 3 minutes. If the unit's LEDs do not flash on pressing a touch-button, the user should press hard on the "●" button to wake the unit from deep sleep. The unit saves the last controlled set of lights as persisted data in order to resume operation following a reset or deep sleep.
- The 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. After a power-on-reset or deep sleep, the Remote Control Unit operates in groupcast mode.

The Remote Control Unit can operate in four Shift modes (0, 1, 2 and 3) to accommodate maximum functionality – most buttons have different functions in the different Shift modes. The current Shift mode is indicated by a combination of the two LEDs on the Remote Control Unit, as shown in the table below. To move to the next Shift mode, press the "*****" button.

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

Note 1: After a power-on-reset or deep sleep, the Remote Control Unit always operates in Shift0 mode.

Note 2: The LEDs flash momentarily after each button-press to indicate that the user command has been detected by the device.

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).	**I**
Off: Send a command to switch off the light(s).	**O**
Increase Brightness: Increase the brightness level of the light(s) at the rate of LEVEL_CHANGE_STEPS_PER_SEC_FAST. 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) at the rate of LEVEL_CHANGE_STEPS_PER_SEC_FAST. The brightness will stop decreasing when the button is released.	**–**
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**
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 commissioning 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.	**⏵**
EZ mode commissioning: Start commissioning to gather endpoint information about existing devices in the network.	**#**

Table 1: Button Functions in Shift0 Mode

Shift1 Mode Operation	Button
On: Send a command to switch on the light(s).	**I**
Off: Send a command to switch off the light(s).	**O**
Increase Brightness: Increase the brightness level of the light(s) at the rate of LEVEL_CHANGE_STEPS_PER_SEC_MED. 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) at the rate of LEVEL_CHANGE_STEPS_PER_SEC_MED. The brightness will stop decreasing when the button is released.	**–**
Increase Colour temperature: Send a command to increase the colour temperature of the light(s). The increase will stop when the button is released.	**1**
Decrease Colour temperature: Send a command to decrease the colour temperature of the light(s). The decrease will stop when the button is released.	**3**
Start Move Hue: Send a Move Hue command to the light(s) to start a continuous colour movement. The light will start cycling through colours.	**2**
Stop Move Hue: Send a Move Hue command to the light(s) to stop the continuous colour movement.	**4**
Store Scene 1: Groupcast a Store Scene command to save the current settings as Scene 1.	**A**
Store Scene 2: Groupcast a Store Scene command to save the current settings as Scene 2.	**B**
Store Scene 3: Groupcast a Store Scene command to save the current settings as Scene 3.	**C**
Store Scene 4: Groupcast a Store Scene command to save the current settings as 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 commissioning 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.	**⏵**
EZ mode commissioning: Start commissioning to gather endpoint information about existing devices in the network.	#

Table 2: Button Functions in Shift1 Mode

Shift2 Mode Operation	Button
On: Send a command to switch on the light(s).	**I**
Off: Send a command to switch off the light(s).	**O**
Increase Brightness: Increase the brightness level of the light(s) at the rate of LEVEL_CHANGE_STEPS_PER_SEC_SLOW. 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) at the rate of LEVEL_CHANGE_STEPS_PER_SEC_SLOW. The brightness will stop decreasing when the button is released.	**–**
Increase Colour Temperature: Increase the colour temperature using Move to Colour Temperature at pre-defined temperature values.	**1**
Decrease Colour Temperature: Decrease the colour temperature using Move to Colour Temperature at pre-defined temperature values.	**3**
Add Group: Add group with group ID set to its network address (always unicast).	**2**
Remove Group: Remove group (always unicast).	**4**
Set Colour Point Next: Step to next fixed colour point (of 7) using Move to Hue and Saturation command.	**A**
Set Colour Point Previous: Step to previous fixed colour point (of 7) using Move to Hue and Saturation command.	**B**

Permit Join: 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 HA 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 commissioning 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.	**⏵**
EZ-mode Commissioning: Start commissioning to gather endpoint information about existing devices in the network.	#

Table 3: 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: Send a command to switch off the light(s).	**O**
Factory Reset: Factory reset the Remote Control Unit, restoring the application and stack persistent data to its factory-new state.	**- + -**
Colour Dimmer Switch: No function assigned **Remote Control:** Initiate EZ-mode Commissioning with 'Find and Bind' – this is used in the first part of the Light Sensor demonstration set-up described in Section 6.2.3 (this mode can be exited by pressing any key)	**1**
No function assigned	**3**
Colour Dimmer Switch: No function assigned **Remote Control:** Initiate EZ-mode Commissioning with Grouping, to add identifying nodes to a group and then to stop them from identifying – this is used in the second part of the Light Sensor demonstration set-up described in Section 6.2.3 (this mode can be exited by pressing the ***** key or by the device entering sleep mode)	**2**
No function assigned	**4**
No function assigned	**A**
No function assigned	**B**
No function assigned	**C**
No function assigned	**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 commissioning 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.	**⏵**
EZ-mode Commissioning: Start commissioning to gather endpoint information about existing devices in the network.	#

Table 4: Button Functions in Shift3 Mode

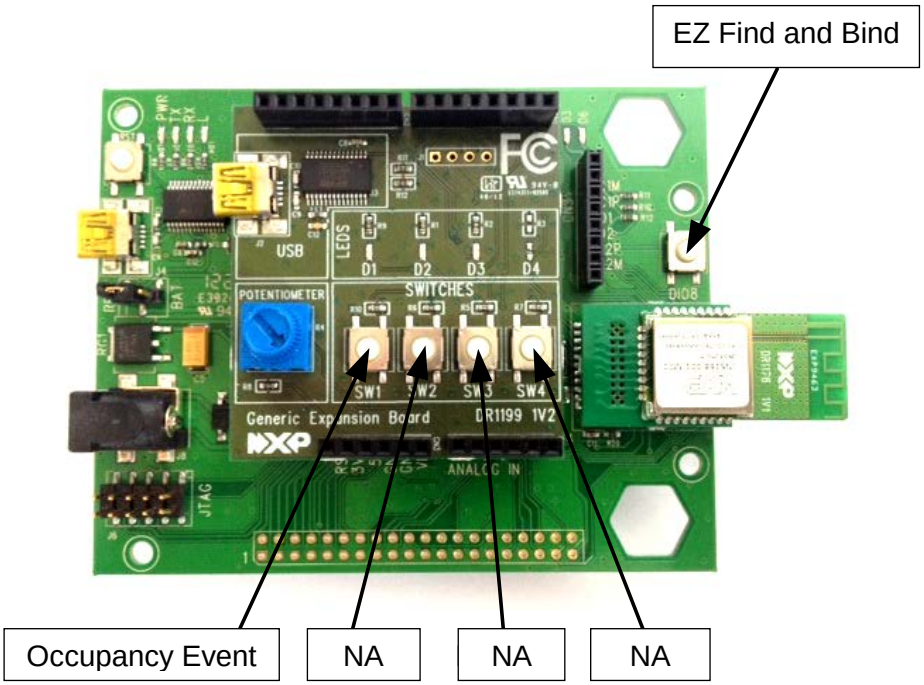
The sleep configuration of the Remote Control Unit is dependent on the value provided to the `KEEPALIVETIME` flag through the build configuration:

- If KEEPALIVETIME = 0, sleep is disabled
- If KEEPALIVETIME = 20 (default configuration), the unit is awake for 20 seconds and sleeps for SLEEP_TIME_MS (defined in **zha_remote_node.c**)
- If 0 < KEEPALIVETIME < 255, the unit will be awake for KEEPALIVETIME and then sleeps for SLEEP_TIME_MS - in this case, the unit will never enter deep sleep mode

- If KEEPALIVETIME = 255, the unit is awake for 20 seconds and sleeps for SLEEP_TIME_MS (defined in **zha_remote_node.c**). After 3 minutes of inactivity (or DEEP_SLEEP_TIME, as defined in **zha_remote_node.c**), the unit will enter deep sleep mode.

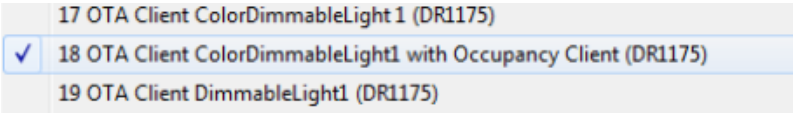
5.3 Occupancy Sensor Functionality

The EZ-mode Commissioning 'Find and Bind' of the lights and the control of the Occupancy attribute (of the Occupancy Sensing cluster) are driven from the Occupancy Sensor which resides on a DR1199 Generic Expansion Board. The JN5168 device associated with the board must be programmed with the binary file **OccupancySensor_JN5168_DR1199.bin**. The Occupancy Sensor functionality is detailed below.



NA – Not allocated

In order to use the above binary file for the Occupancy Sensor, you will need to pair this device with a light which contains the Occupancy Sensor cluster client. There is no pre-built binary file for this light but the following build configuration (19) is provided for such a light:



This is a Colour Dimmable Light which should be used with a DR1175 Lighting/Sensor Expansion Board (the 'OTA Client' feature of this build configuration can be ignored unless an OTA upgrade is to be performed as described in Section 8). Building and loading an application are described in Section 10.3. Once built, the resulting application binary file will appear in the directory **OTAColorDimmableLightWithOccupancy/Build/OTABuild** and the filename will be prefixed with **OTAColorDimmableLightWithOccupancy**.

5.3.1 Virtual PIR Simulation

Occupancy events are simulated by pressing a button on the board - button SW1 is assigned as the output of a virtual PIR detector. There are two types of virtual occupancy sensor: Open Collector and PWM. They are defined in the makefile under "PIR Sensor Type". The different sensor types and how to use them are detailed below.

Note that for both types of sensor, LED D1 on the Generic Expansion Board is used as an indicator of the state of the Occupancy attribute:

- If the Occupancy attribute is 0 (i.e. Unoccupied), the LED is OFF
- If the Occupancy attribute is 1 (i.e. Occupied), the LED is ON

Also note the following uses of LEDs D2 and D3 on the Generic Expansion Board:

- LED D3 will flash during a 'Find and Bind' operation started using the button DIO8 - if it continues to flash after completing the 'Find and Bind' then press the button again.
- The LEDs D2 and D3 may flash intermittently to indicate the operational state when the JN5168 module is awake for sampling or rejoining.

Open Collector Sensor

An Open Collector sensor outputs a constant digital high/low signal for occupancy. In this demonstration, 'occupied' is represented by digital low and 'unoccupied' is represented by digital high.

To define a sensor as an Open Collector, uncomment the compile flag `PIR_TYPE_OPEN_COLLECTOR` in the makefile.

Simulating Unoccupied to Occupied Event:

To move the sensor from the unoccupied to occupied state, press and hold SW1.

Simulating Occupied to Unoccupied Event:

Once in the occupied state, to simulate an 'unoccupied' event, release SW1. If no further occupancy event is simulated by pressing the SW1 button within a certain timeout period (180 seconds, by default), the sensor will automatically move to the unoccupied state.

Note: 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 SW1 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.

PWM Sensor

An PWM sensor toggles its digital output between high and low while occupied.

Simulating Unoccupied to Occupied Event:

To move the sensor from the unoccupied to occupied state, press SW1 a certain number of times (5, by default) within a certain timeout period (10 seconds, by default).

Note 1: The number of button-presses required can be customised using the `APP_OCCUPANCY_SENSOR_UNOCCUPIED_TO_OCCUPIED_THRESHOLD` macro.

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

Simulating Occupied to Unoccupied Event:

Once in the occupied state, if no further occupancy event is simulated by pressing the SW1 button within a certain timeout period (180 seconds, by default), the sensor will automatically move to the unoccupied state.

Note: 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 by repeating the simulated occupancy event. This will reset the timer that keeps track of the timeout defined by APP_OCCUPANCY_SENSOR_OCCUPIED_TO_UNOCCUPIED_DELAY.

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 'Find and Bind'

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 'Find and Bind'

Attribute Reporting

Attribute Reporting can be configured optionally by defining the macro HA_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 HA_SYSTEM_MIN_REPORT_INTERVAL) if there is any change in the attribute or, otherwise, every 60 seconds (or HA_SYSTEM_MAX_REPORT_INTERVAL). On receiving the attribute report:

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

5.3.2 Sleep Options

The Occupancy Sensor is a sleeping End Device and will, by default, attempt to sleep with RAM on and the oscillator on, whenever possible. It will wake up every `HA_SYSTEM_MAX_REPORT_INTERVAL`, which by default is 60 seconds.

Enabling Deep Sleep

The Occupancy Sensor can be configured to enter deep sleep by setting the `HA_SYSTEM_MAX_REPORT_INTERVAL` to zero, which will disable periodic reporting. Since periodic reporting is disabled, when in the unoccupied state the device does not need to keep track of time, meaning it can go into deep sleep and wait for the virtual sensor to trigger an occupancy event.

Note: When the Occupancy Sensor is in the occupied state, the device needs to start the 'occupied to unoccupied' timer, which means the device will sleep with RAM held and the oscillator on.

Note: Since the Occupancy Sensor is a sleepy End Device, there may be a delay in sending out the attribute reports.

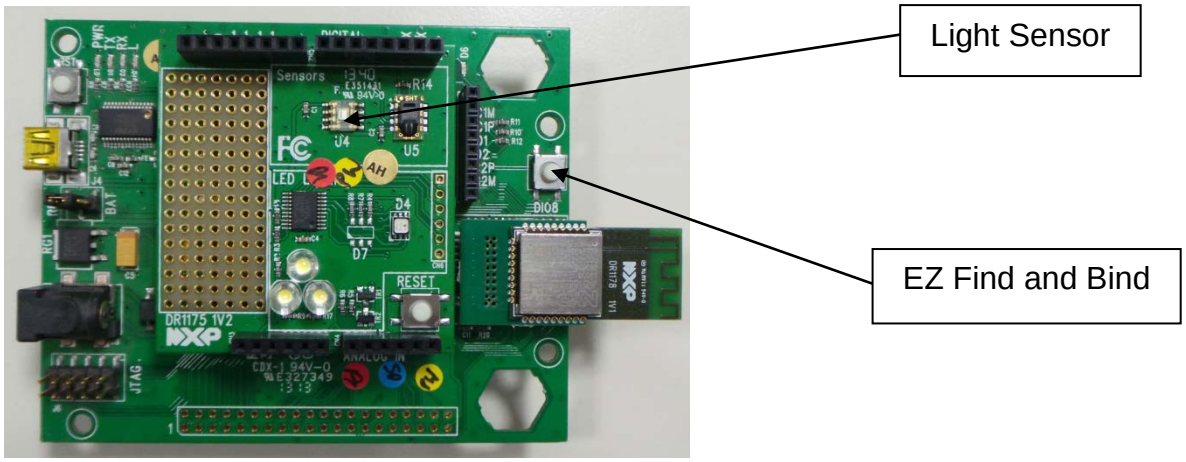
Enabling Sleep Prevention

The Occupancy Sensor can be kept awake by pressing the SW2 button, which causes the LED to start flashing 250ms on, 250ms off. The device is put back into normal operational mode by pressing the SW3 button, which stops the LED from flashing.

5.4 Light Sensor 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 HA Lighting devices that support the Level Control cluster. The control application resides on the Remote Control device (see Section 1.1.7), to which the Light Sensor is bound, and the Remote Control device forms the lights into a group for synchronous control. The binding and grouping are performed via EZ-mode Commissioning (see Section 6.2.3).

The Light Sensor resides on a DR1175 Lighting/Sensor Expansion Board. The JN5168 device associated with the board must be programmed with the binary file **LightSensor_JN5168_DR1175.bin**. The resulting board functionality is indicated below.



The Light Sensor can but put into 'keep-alive' mode by power-cycling 3 times. The two modes of operation are as follows:

- **Normal Mode:** The Light Sensor sleeps and wakes up every `HA_SYSTEM_MAX_REPORT_INTERVAL`, when it obtains a new light reading, updates the `u16MeasureValue` attribute with the new reading and sends a periodic report. This is done to reduce the current consumption, which increases the battery life.
- **Keep-alive Mode:** The Light Sensor is permanently active and obtains a new light reading every second. If the light level changes since the last reading by at least `LIGHT_SENSOR_MINIMUM_REPORTABLE_CHANGE` then the `u16MeasuredValue` attribute is updated and the Light Sensor sends out an attribute report after `HA_SYSTEM_MIN_REPORT_INTERVAL` (default 1 second). If there is no change, it will send a report every `HA_SYSTEM_MAX_REPORT_INTERVAL` (default 60 seconds).

Note 1: 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 2: 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 (JN-UG-3103)*.

The (sleepy) Remote Control device polls its parent for messages once per second. After receiving the attribute report, the Remote Control device (hosting the Illuminance Measurement cluster client) will adjust the light level on the (grouped) light nodes based on the reported illuminance measurement. The required calculation is detailed below, which determines the light level that should be produced by the light for a given illuminance measured at the Light Sensor.

At the Light Sensor, the maximum lux level (as measured by the ALS driver on the DR1175 board) is 4015 and minimum lux level is 1.

At the light, the light level to be produced is represented as a value in the range 0 to 255 (this is the `u8CurrentLevel` attribute of the Level Control cluster).

A divisor is defined which is used (later) in calculating the produced light level from the measured Lux value:

ILLUMINANCE_LUX_LEVEL_DIVISOR = 4015/254 ~ 16

(i.e. ILLUMINANCE_MAXIMUM_LUX_LEVEL/CLD_LEVELCONTROL_MAX_LEVEL)

The value of the `u8CurrentLevel` attribute (of the Level Control cluster) for the light is then calculated as follows:

CLD_LEVELCONTROL_MAX_LEVEL - (u16MeasuredLux/ILLUMINANCE_LUX_LEVEL_DIVISOR)

where `u16MeasuredLux` is the attribute of the Illuminance Measurement cluster reported to the light.

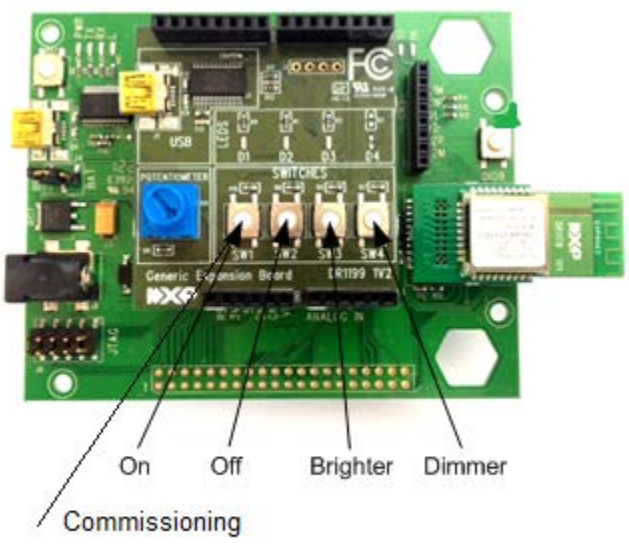
Therefore:

- When the measured illuminance is at its maximum value of 4015 lux, the light level to be produced by the light is 4
- When the measured illuminance is at its minimum value of 1 lux, the light level to be produced by the light is 254

Note: The Light Sensor needs a light source to measure the correct thresholds for proper operation (sensitivity increases based on the amount of light falling on sensor).

5.5 GP Switch Functionality

The Green Power (GP) Switch device employs a DR1199 Generic Expansion Board, which features four switches SW1-SW4. The switch functionality is detailed below.



If a GP Switch is required, the JN5168 device associated with a Generic Expansion Board must be programmed with the binary file **EH_Switch_JN5168_DR1199.bin**.

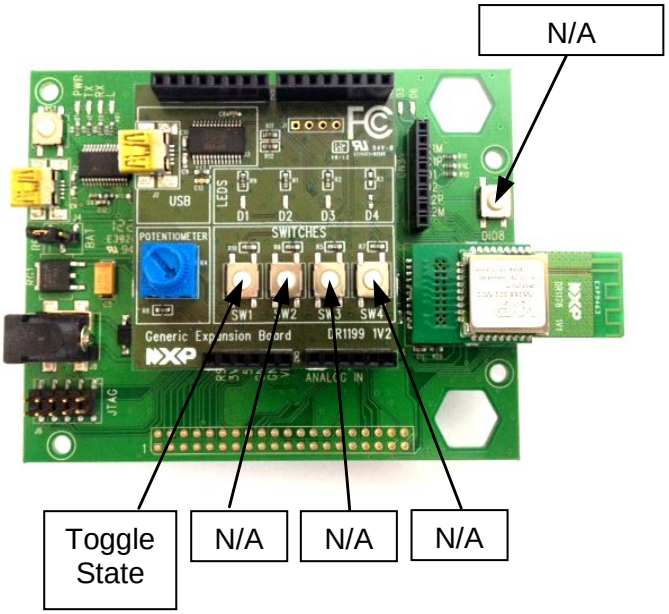
A ‘factory new’ GP Switch should be commissioned using Commissioning button (SW1). The light node to be commissioned should first be put into GP self-commissioning mode. Then

the Commissioning button must be repeatedly pressed at an interval of one second until a visual indication is observed on the light node, indicating successful commissioning. For full details of GP Switch commissioning, refer to Section 7.2. After the GP Switch has been commissioned, it can be used to send commands to the light (On, Off, Brighter and Dimmer).

5.6 Smart Plug (Mains Power Outlet) Functionality

A Smart Plug or “Mains Power Outlet” can be used to monitor the instantaneous power consumption of an attached appliance (by means of the Simple Metering cluster). However, in this demonstration the device can simply be switched on and off, either using a button on the unit or remotely from a switch node, such as a Dimmer Switch. LEDs on the unit indicate whether the device is on or off.

The binary file **MainsPowerOutlet_JN5169_DR1199.bin** should be programmed into the JN5169 device associated with a DR1199 Generic Expansion Board to obtain the Smart Plug functionality. The device functionality is detailed below:



The button **SW1** allows the user to manually toggle the on/off state of the device.

The device can also be controlled remotely from a switch device in the network, such as the Dimmer Switch (see Section 5.1). Note that only the On and Off buttons on the Dimmer Switch node can be used to control the Smart Plug.

The current state of the device is indicated by the LEDs D1 and D2, with D1 illuminated when the device is off and D2 illuminated when the device is on.

For information on the commissioning the Smart Plug device, refer to Section 6.3.

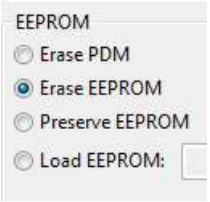
5.7 Clearing Context Data on the Devices

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

5.7.1 On Co-ordinator

The Co-ordinator in this application resides on a DR1198 USB Dongle and has no switch inputs.

The context data on the Co-ordinator is stored in EEPROM and can be cleared using the JN516x Flash Programmer within BeyondStudio for NXP by selecting the **Erase EEPROM** option in the **Program serial device** dialogue box (see image below). For information on using this Flash programmer, refer to the *BeyondStudio for NXP Installation and User Guide (JN-UG-3098).*



5.7.2 On All Light Nodes

The context data on a light node is stored in EEPROM and can be cleared using the JN516x Flash Programmer within BeyondStudio for NXP with the **Erase EEPROM** option, as described for the Co-ordinator in Section 5.7.1.

Alternatively, the context data can be cleared by leaving the light on for at least 2 seconds and then power-cycling seven times - in this sequence, the light must be ON for less than 2 seconds following an OFF. This alternative method is the same for all the Light builds.

5.7.3 On Dimmer Switch Node

The context data on the Dimmer Switch node is stored in EEPROM and can be cleared using the JN516x Flash Programmer within BeyondStudio for NXP with the **Erase EEPROM** option, as described for the Co-ordinator in Section 5.7.1.

Alternatively, the context data can be cleared during normal operation of the device. This is done by pressing and releasing the '**RST**' button while holding down the '**DIO8**' button (both buttons are on the DR1174 Carrier Board).

5.7.4 On Remote Control Node

The context data on the Remote Control node is stored in EEPROM and can be cleared using the JN516x Flash Programmer within BeyondStudio for NXP with the **Erase EEPROM** option, as described for the Co-ordinator in Section 5.7.1.

Alternatively, the context data can be cleared as follows:

1. Press the button ***** repeatedly until both of the LEDs illuminate (Shift3 mode).
2. Now enter the button sequence **–**, **+**, **–**.

5.7.5 On Occupancy Sensor Node

The context data on the Occupancy Sensor node can be cleared using exactly same methods as described for the Dimmer Switch in Section 5.7.3.

5.7.6 On Light Sensor Node

The context data on the Light Sensor node can be cleared using exactly same methods as described for the Dimmer Switch in Section 5.7.3.

5.7.7 On Smart Plug Node

The context data on the Smart Plug (Mains Power Outlet) node is stored in EEPROM and can be cleared using the JN516x Flash Programmer within BeyondStudio for NXP with the **Erase EEPROM** option, as described for the Co-ordinator in Section 5.7.1.

Alternatively, the context data can be cleared during normal operation of the device. This is done by pressing and holding down the **SW1** button on the DR1199 Generic Expansion Board.

5.7.8 On OTA Server Node

The context data on the OTA Server node is stored in EEPROM and can be cleared using the JN516x Flash Programmer within BeyondStudio for NXP with the **Erase EEPROM** option, as described for the Co-ordinator in Section 5.7.1.

5.7.9 On GP Switch

The context data on a GP Switch node can be cleared by power-cycling the node. This is done by pressing and releasing the '**RST**' button while holding down the '**DIO8**' button (both buttons are on the Carrier Board). Alternatively, the context can be cleared by power-cycling the node 13 consecutive times. The required number of power-cycles can be modified using the configuration macro CLEAR_PERSISTENT_SHORT_PRESS in the header file **EH_Switch_Configurations.h**.

6 Setting Up the Network

This section describes how to create the demonstration network and commission light nodes. This set-up process employs EZ-mode Commissioning, which is fully detailed in the *ZigBee Cluster Library User Guide (JN-UG-3103)*. The Green Power demonstration set-up is described separately in Section 7.

The network set-up procedure is presented in two stages, as follows:

- First the network is formed, in which the various nodes join the network – this is described in Section 6.1.
- Then links are established between the controller nodes and light nodes, through either binding or grouping, depending on the type of controller device:
    - The grouping method is described in Section 6.2.1
    - The binding (Find and Bind) method is described in Section 6.2.2

Note 1: When the Light Sensor and Remote Control devices are to be included in the network, both binding and grouping must be used for these devices – this special case is described in Section 6.2.3.

Note 2: Introducing a Smart Plug (Mains Power Outlet) into the network is described in Section 6.3.

6.1 Forming the Network

The procedure below describes how to form the demonstration network using EZ-mode Commissioning. This network formation process is the same irrespective of whether 'Find and Bind' or Grouping will be later used to establish control links to the light nodes.

1. Plug the Co-ordinator node (USB Dongle) into a PC or an USB power supply.

The Co-ordinator will create a network - the green LED on the dongle will start to flash when the network has been successfully created.

The Co-ordinator opens the network for other HA devices to join for a period of 180 seconds after start-up or reset - the orange LED on the dongle will start to flash indicating the network is open for joining.

Note that the network will also be opened for joining in the following circumstances:

- When a light node joins the network, it broadcasts a 'permit join' message containing an EZ-mode Commissioning time of 180 seconds, allowing other nodes to join it during this time (that is, the network will remain open to joining for a further 180 seconds from that point).
- If the Network Steering phase of EZ-mode Commissioning is invoked on a light node or switch node that is already part of the network, this will open the network for joining for a period of 180 seconds from the point at which the steering was invoked.

Note 1: Once a node has joined the network, it will remain in the network, even through a power-cycle.

Note 2: If you wish to add a node after the 180-second period has expired, start network steering on any of the nodes or power-cycle the Co-ordinator node.

2. Power-up the light nodes (Lighting/Sensor Expansion Boards).

A light node will start searching for a suitable HA network with joining allowed. During this search, the light implements a "breathe" effect (a gradual change in light level between minimum and maximum values).When a light node has joined the Co-ordinator, the light stops the breathe effect (when the breathe effect is not enabled on a light, the light flashes once to indicate that joining is complete).

Note: To reset a light node to the 'Factory New' state, power-cycle the node 7 times to clear context data, as described in Section 5.7.2.

3. Power-up the Dimmer Switch node (Generic Expansion Board).

On power-up, a Dimmer Switch node will search for a suitable network - the LED pair D1+D3 and LED D2 will flash alternately during the search.

When a suitable open network is found ('permit join' is true), the switch node will join and indicate success by briefly illuminating LEDs D1-D3.

Note: To reset the Dimmer Switch node to the 'Factory New' state, press and hold down the button '**DIO8**' and then press the switch '**RST**', both of which are on the underlying DR1174 Carrier Board.

4. Plug the OTA Server node (USB Dongle) into a PC.

   The OTA Server will join the network and then the green LED on the USB Dongle will light up.

5. Power-up the Colour Dimmer Switch or Remote Control device (Remote Control Unit) by inserting batteries.

   Note that the Remote Control Unit should have been programmed as an HA Remote Control device only if the Light Sensor is to be used (Step 7) in the network, otherwise it should have been programmed as an HA Colour Dimmer Switch device.

   On power-up, this node will search for a suitable network - the LED pair (left and right) will continuously flash during the search.

   When a suitable open network is found ('permit join' is true), the node will join and indicate success by briefly illuminating both LEDs, and then move to operational mode Shift0 (when both LEDs will be off).

Note: To reset the Remote Control Unit node to the 'Factory New' state, press the button * repeatedly until both LEDs illuminate to indicate Shift3 mode, and then enter the button sequence **–**, **+**, **–**.

6. Power-up the Occupancy Sensor node (Generic Expansion Board).

   On power-up, an Occupancy Sensor node will search for a suitable network – during the search, the LED D3 will flash one second On and one second Off.

   When a suitable open network is found ('permit join' is true), the node will join and indicate success by switching off the LED D3.

Note: To reset an Occupancy Sensor node to the 'Factory New' state, press and hold down the button '**DIO8**' and then press the switch '**RST**', both of which are on the underlying DR1174 Carrier Board.

7. Power-up the Light Sensor node (Lighting/Sensor Expansion Board), if required.

   Note that this step is optional and the Light Sensor can only be used if the Remote Control Unit has been programmed as an HA Remote Control device.

   On power-up, a Light Sensor node will search for a suitable network – during the search, the red LED D4 will flash one second On and one second Off.

   When a suitable open network is found ('permit join' is true), the sensor node will join and indicate success by turning off the LED.

Note: To reset a Light Sensor node to the 'Factory New' state, press and hold down the button '**DIO8**' and then press the switch '**RST**', both of which are on the underlying DR1174 Carrier Board.

8. Perform the rest of the network set-up to establish control links between the controller nodes and light nodes – refer to Section 6.2.

6.2 Commissioning

This section describes how to commission light nodes to be controlled from a controller node either as a group of lights or as bound devices. You must use the method appropriate for each controller device, as follows:

- **Dimmer Switch:** Use grouping (Section 6.2.1) or binding (Section 6.2.2)
- **Colour Dimmer Switch:** Use grouping (Section 6.2.1)
- **Occupancy Sensor:** Use binding (Section 6.2.2)
- **Light Sensor and Remote Control:** Use both binding and grouping (Section 6.2.3)

The commissioning of light nodes is performed per controller node.

The procedures assume that the network has been formed as described in Section 6.1.

6.2.1 Commissioning Light Nodes - Groups and Scenes

In the following procedure, you will collect light nodes into a group to be controlled by either of the following controller devices:

- Dimmer Switch
- Colour Dimmer Switch

In this demonstration, there is only one group on these devices. You will also configure and save a scene for this group of lights.

1. Start the discovery of lights from a switch.

   On Dimmer Switch node:

   Enter identify mode by pressing and holding down the Commissioning button (**DIO8**) on the node (on the DR1174 Carrier Board).

   On Colour Dimmer Switch node:

   Enter identify mode by pressing the Commissioning button (#) on the node (on the DR1159 Remote Control Unit).

2. Identify a light and add it to a group.

   After Step 1 above, wait for a minimum of 3 seconds for a discovered light node to identify itself (which it does by toggling the state of its three white LEDs in the case of a Dimmable Light, or turning the multi-colour LED to red in the case of a Colour Dimmable Light).

   If the light is initially off, it will switch on and identify itself - only one light node will identify itself at a time.

During this state (for one light node), you can use the switches SW1-SW4 on the Dimmer Switch node as follows:

- Press **SW1** to join the light to the group
- Press **SW2** to remove the light from the group (if it is already a member)
- Press **SW3** to move on to another discovered light (not yet in the group)
- Press **SW4** to select the previous light from the group

Similarly, during this state (for one light node), you can use the buttons in Shift2 mode on the Colour Dimmer Switch node (Remote Control Unit) as follows:

- Press **2** to join the light to the group
- Press **3** to remove the light from the group (if it is already a member)
- Press **>** to move on to another discovered light (not yet in the group).

Once the light node is added to a group, it can be controlled as part of the group (in group mode).

3. Enter individual control mode.

On Dimmer Switch node:

Upon releasing the Commissioning button (**DIO8**) after Step 2 above, the Dimmer Switch node will enter individual control mode for the currently selected light node.

On Colour Dimmer Switch node:

On the Colour Dimmer Switch, individual control mode can be entered by pressing the "**?**" button.

4. Configure the brightness of the currently selected light for a scene.

On Dimmer Switch node:

On the Dimmer Switch node, you can now configure the currently selected light for a scene by setting its brightness level using the switches SW1-SW4 as follows:

- Press **SW1** to switch the light on (full brightness)
- Press **SW2** to switch the light off
- Press **SW3** to increase its brightness level
- Press **SW4** to decrease its brightness level

On Colour Dimmer Switch node:

For adding scenes from the Colour Dimmer Switch, refer to Section 5.2.

5. Repeat Steps 1 to 4 for each light node to be included in the group and scene.

6. After adding all the required light nodes to a group and configuring them for a scene, you can save the settings to one of two possible scenes.

On Dimmer Switch node:

On the Dimmer Switch, this is done by pressing either of the following switch combinations: **SW1+SW3** or **SW2+SW4**.

On Colour Dimmer Switch node:

For saving scenes from the Colour Dimmer Switch, refer to Section 5.2.

> **!** ***Caution:*** *To exit the commissioning process from the Dimmer Switch as described above, a scene is saved and this will over-write any previously saved scene corresponding to the switch combination used. To exit without saving a scene, leave the switch in individual control mode for 30 seconds, which will return you to control mode and will not write a scene.*

7. Enter control mode.

On Dimmer Switch node:

The Dimmer Switch now enters control mode and can be used to control the group of lights as follows.

- Press **SW1** to switch the lights on
- Press **SW2** to switch the lights off
- Press **SW3** to increase their brightness level
- Press **SW4** to decrease their brightness level
- Press switch combination **SW1**+**SW3** or **SW2**+**SW4** to recall a scene

On Colour Dimmer Switch node:

For the control of lights from the Colour Dimmer Switch, refer to Section 5.2.

> **Note:** To save light settings for the other scene, repeat the above procedure (but be sure to form the same group of lights).

6.2.2 Commissioning Light Nodes - Binding

In the following procedure, a controller node is bound to one or more light nodes (if bound to multiple light nodes, the lights will be controlled synchronously). The controller node can be either of the following devices:

- Dimmer Switch
- Occupancy Sensor

This method of control does not allow the use of scenes.

1. Put the light node(s) into identify mode by power-cycling the node three times. The light will then enter identify mode and remain in identify mode for up to `EZ_MODE_TIME*60` seconds, where `EZ_MODE_TIME` is in minutes and set to 3 minutes in the application. The light will indicate this by flashing.

2. Entering the Commissioning mode from the different devices.

On the Dimmer Switch node:

Make sure the Dimmer Switch node is in group control mode, as follows:

a) Press and hold down the button **SW3**.

b) Press and hold down the Commissioning button, **DIO8**.

c) Release **SW3**.

Following this sequence, the Dimmer Switch node will be in EZ-mode Commissioning and LED1 will start to flash (when **DIO8** is released, the node exits EZ-mode Commissioning and returns to group control mode).

On the Occupancy Sensor node:

Press and hold down the Commissioning button, **DIO8**, to enter EZ-mode Commissioning 'Find and Bind' mode (when **DIO8** is released, the node will exit EZ-mode Commissioning).

The Occupancy Sensor node will now start the 'Find and Bind' phase and LED D3 will start to flash (500ms on and 500ms off).

3. The Dimmer Switch node can now be used to perform the following actions.

Press **SW1** to bind the Dimmer Switch node to the light node(s) in identify mode.

While the Dimmer Switch node is in EZ-mode Commissioning, you can use the switches SW1-SW4 to perform various operations, as follows:

- Press **SW1** to bind clusters
- Press **SW2** to 'factory reset' the switch node without leaving the network
- Press **SW3** to add the light node (in identify mode) to the switch's group
- Press **SW4** to initiate a change of channel (all network nodes must be active)

4. Release button DIO8 to complete the binding process. You can then use switches SW1-SW4 on the Dimmer Switch node to control the lights, as indicated in **Error! Reference source not found.**.

While controlling the lights through bound transmissions, group transmissions will be ignored by the switch. Therefore, at any one time, the Dimmer Switch can be used to control bound or grouped devices, but not both.

6.2.3 Commissioning Light Sensor, Remote Control and Light Nodes

When the Light Sensor and Remote Control devices are used, both binding and grouping must be used in the network:

- The Light Sensor is bound to the Remote Control device, allowing the Light Sensor to send regular luminance measurements to the Remote Control device, where this data is interpreted and translated into control instructions for the lights nodes.
- The light nodes are grouped by the Remote Control device, allowing the Remote Control device to synchronously control the grouped lights.

This arrangement is illustrated in Figure 1 below.

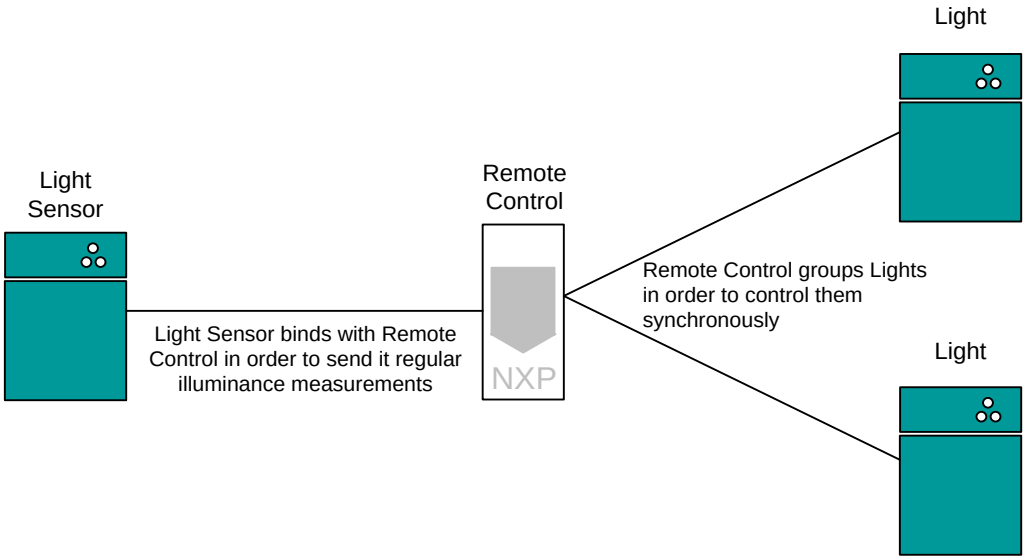


Figure 1: Light Sensor/Remote Control Set-up

Note 1: A light node in the above network can be any HA Lighting device that supports the Level Control cluster – for example, a Dimmable Light or a Colour Dimmable Light.

Note 2: In this network, the Light Sensor and Remote Control are both sleepy End Devices. The light nodes are Routers. The Co-ordinator resides on a USB Dongle (from the evaluation kit). Therefore, messages from the Light Sensor will be sent to the Remote Control via its parent (which may be one of the light nodes or the dongle) and, when awake, the Remote Control must poll its parent for these messages.

The following procedure describes how to set up the required binding and group – you must perform both parts of the procedure in quick succession.

Part 1 – Bind Light Sensor to Remote Control

1. Put the Light Sensor into the EZ-mode Commissioning 'Find and Bind' state by pressing and holding down the Commissioning button, **DIO8** (when **DIO8** is released, the node will exit EZ-mode Commissioning). The Light Sensor will also identify itself in this mode by flashing LED D4 once per second (only while the button is held down).
2. Put the Remote Control device into the EZ-mode Commissioning 'Find and Bind' state by pressing the "**1**" key in Shift3 mode. This also puts the device into identify mode by flashing both LEDs once per second.

   You can exit EZ-mode Commissioning 'Find and Bind' on the Remote Control device at any time by pressing any key (in Shift3 mode).
3. Wait for the Light Sensor to bind to the Remote Control device - when this binding completes, the Light Sensor and Remote Control device will both stop identifying themselves (LEDs will stop flashing).
4. Exit the 'Find and Bind' state on both devices as follows:
   - **a)** On the Light Sensor device, release button **DIO8**.
   - **b)** On the Remote Control device, press any key (in Shift3 mode) once only (if you accidently press the key two or more times, you will exit EZ-mode Commissioning and will need to restart the commissioning process from the beginning).

> **!** ***Important:*** *Part 2 of the procedure, below, must now be executed within 20 seconds, before the Remote Control device goes to sleep.*

Part 2 – Group Lights for Remote Control

5. Put the light nodes to be controlled into identify mode by power-cycling each node three times. The light will then enter identify mode and remain in identify mode for up to `EZ_MODE_TIME*60` seconds, where `EZ_MODE_TIME` is in minutes and set to 3 minutes in the application. The light will indicate this by flashing.
6. Put the Remote Control device into the EZ-mode Commissioning 'Grouping' state by pressing the "**2**" key in Shift3 mode.

   You can exit EZ-mode Commissioning 'Grouping' on the Remote Control device at any time by pressing the "*****" key (in Shift3 mode). This mode will also be automatically exited if the device goes to sleep (due to inactivity).
7. Wait for the Remote Control device to add a light to the group - when this occurs, the light stops flashing.
8. If there are any lights still to be grouped (still identifying themselves) then repeat Steps 6 and 7 for the next light.

   When all the lights have stopped identifying themselves, they are all in the group.

There is a known issue relating to the operation of this network – refer to Appendix E.

6.3 Commissioning Smart Plug (Mains Power Outlet)

The Smart Plug (Mains Power Outlet) device can be switched on and off from a switch device in the network. This section describes how the Smart Plug can be introduced into the network and bound to a Dimmer Switch that will be used to control the Smart Plug. For information on the Dimmer Switch and Smart Plug functionality, refer to Section 5.1 and Section 5.6.

6.3.1 Introducing Smart Plug to Network

The Dimmer Switch and Smart Plug nodes are introduced into the network as described in Section 6.1, where the Smart Plug is used instead of a light node. The network joining behavior of the Smart Plug is as described below.

When the Smart Plug node is powered on, it starts to search for a suitable HA network with joining allowed. During this search, LED D2 on the DR1999 Generic Expansion Board flashes every 200ms. When the node has joined the network, D2 stops flashing and illuminates permanently, indicating that the Smart Plug is in the 'on' state.

6.3.2 Binding Dimmer Switch and Smart Plug

Once the Smart Plug is in the network, it must be bound to a Dimmer Switch as follows:

1. Power cycle the Smart Plug node **3 times** to enter EZ-mode Commissioning. The Smart Plug device will now start the 'Find and Bind' phase and LED D2 will start to flash (500ms on and 500ms off).
2. On the Dimmer Switch node, press button **SW1** to bind the Dimmer Switch node to the Smart Plug.

   While the Dimmer Switch node is in EZ-mode Commissioning, you can use the switches **SW1-SW4** to perform various operations, as follows:

   - Press **SW1** to bind clusters
   - Press **SW2** to 'factory reset' the switch node without leaving the network
   - Press **SW3** to add a node to the switch's group
   - Press **SW4** to initiate a change of channel (all network nodes must be active)

   The Smart Plug node will automatically exit EZ-mode Commissioning once 'Find and Bind' is complete. However, you can exit at any time by power cycling the Smart Plug.
3. You can now use the Dimmer Switch to control the Smart Plug (see Section 5.6).

Note: While controlling the Smart Plug through bound transmissions, group transmissions are ignored by the device. Therefore, at any one time, the Dimmer Switch can be used to control bound or grouped devices, but not both.

7 Setting Up the Green Power (GP) Network

This section describes how to create the Green Power (GP) demonstration network. This network employs the following ZigBee devices, hardware components and binary files:

- **Co-ordinator:** This is the DR1198 USB Dongle, programmed with the binary file **Coordinator_JN5168.bin**
- **Dimmable Light:** This is a DR1174 Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board, programmed with the binary file **DimmableLight_JN5168_DR1175_LED_EXP_MONO_GP.bin**
- **Colour Dimmable Light:** This is a DR1174 Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board, programmed with the binary file **ColorDimmableLight_JN5168_DR1175_LED_EXP_RGB_GP.bin**
- **GP Switch:** This is a DR1174 Carrier Board fitted with a DR1199 Generic Expansion Board, programmed with the binary file **EH_Switch_JN5168_DR1199.bin**

The GP Switch is used to control a light node, which can be a Dimmable Light or Colour Dimmable Light. You may use more than one light node in the network and the GP Switch can be commissioned to control all of them.

Note 1: The Dimmable Light and Colour Dimmable Light devices can each act as both a GP Proxy node and Sink node in this network.

Note 2: Alternative Dimmable Light binary files are provided for the DR1190 and DR1192 bulbs (see Section 3 and Section 4).

The network set-up procedure is presented in two stages, as follows:

- First the ZigBee PRO network is formed – this is described in Section 7.1.
- Then the GP Switch is commissioned – this is described in Section 7.2.

GP Switch decommissioning is also described in Section 7.3.

7.1 Forming the Network

The process described below uses EZ-mode Commissioning, which is fully detailed in the *ZigBee Cluster Library User Guide (JN-UG-3103)*, to form an HA network.

1. Plug the Co-ordinator node (USB Dongle) into a PC.

The Co-ordinator will create a network - the green LED on the dongle will start to flash when the network has been created.

The Co-ordinator opens the network for other HA devices to join for a period of 180 seconds after start-up or reset - the orange LED on the dongle will start to flash when the network is open for joining.

Note that the network will also be opened for joining in the following circumstances:

- When a light node joins the network, it broadcasts a 'permit join' message containing an EZ-mode Commissioning time of 180 seconds, allowing other nodes to join it during this time (that is, the network will remain open to joining for a further 180 seconds).
- If the Network Steering phase of EZ-mode Commissioning is invoked on a light node or switch node that is already part of the network, this will open the network for joining for a duration of 180 seconds.

2. Power up the light nodes (Lighting/Sensor Expansion Boards).

A light node will start searching for a suitable HA network with joining allowed. During this search, the light implements a "breathe" effect (a gradual change in light level between minimum and maximum values).When a light node has joined the Co-ordinator, the light stops the breathe effect (when the breathe effect is not enabled on a light, the light flashes once to indicate that joining is complete).

Note: To reset a light node to the 'Factory New' state, power-cycle the node 7 times to clear context data, as described in Section 5.7.2.

Note: If you wish to add a node after the 180-second period has expired, re-initiate EZ-mode Commissioning by power-cycling the USB Dongle.

The network will be formed in less than 10 seconds. The nodes will then remain in this network, even through a power cycle.

3. Power up the GP Switch node (Generic Expansion Board).

7.2 GP Switch Commissioning

In the following commissioning procedure, a GP Switch node is paired with one or more light nodes (Dimmable Light or Colour Dimmable Light devices).

1. Put each of the light nodes to be commissioned into GP self-commissioning mode by power-cycling the light node 3 times. The light will then enter commissioning mode and indicate this by flashing.
2. On the GP Switch node, press the Commissioning button (SW1) repeatedly at an interval of one second until all the lights stop flashing and return to their original states (the GP Switch sends commissioning packets to the light nodes on each button-press).

Note: The switch SW1 should not be pressed too fast nor too slow. It should be pressed at approximately a one-second interval. 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(s) and can be used to control the light(s). The switches SW1-SW4 on the GP Switch can be used to send On, Off, Brighter and Dimmer commands to the light node(s).

Note 1: When a commissioned light is in normal operational mode (not in commissioning or decommissioning mode), it will not process any commissioning or decommissioning packets that it receives. So this light will not be affected by commissioning packets sent when adding another light to those controlled by the GP Switch (apart from updating its internal tables). In addition, a commissioned light in normal operational mode will ignore any received On/Off commands when another light is in commissioning mode.

Note 2: Once the GP Switch is commissioned with light nodes, repeatedly pressing button SW1 will result in decommissioning packets being transmitted. Therefore, before attempting to commission new light nodes, the GP Switch must be factory-reset so that context data on the switch will be cleared (see Section 5.7.9) and the switch will subsequently transmit commissioning packets when SW1 is repeatedly pressed. This will commission the switch only with new lights that are in commissioning mode without disturbing the lights that are already commissioned.

Note 3: There is actually no difference at the GP cluster level between the commissioning and decommissioning modes of a light - a light can accept both commissioning and decommissioning commands in either mode. For practical reasons, the two modes have been distinguished in their visual indications and, to avoid confusion, you are advised to only commission in commissioning mode and only decommission in decommissioning mode.

7.3 GP Switch Decommissioning

Once commissioned and in operational mode, a light node can be decommissioned from the GP Switch as follows:

1. Put the light node into decommissioning mode by power-cycling the light node 4 times. Following this power-cycling, the light will be in the ON state, but a little time later the light will go into the OFF state when the node enters decommissioning mode.
2. On the GP Switch node, press the Commissioning button (SW1) repeatedly at an interval of one second until the light goes into the ON state – this state indicates that the node has been successfully decommissioned.

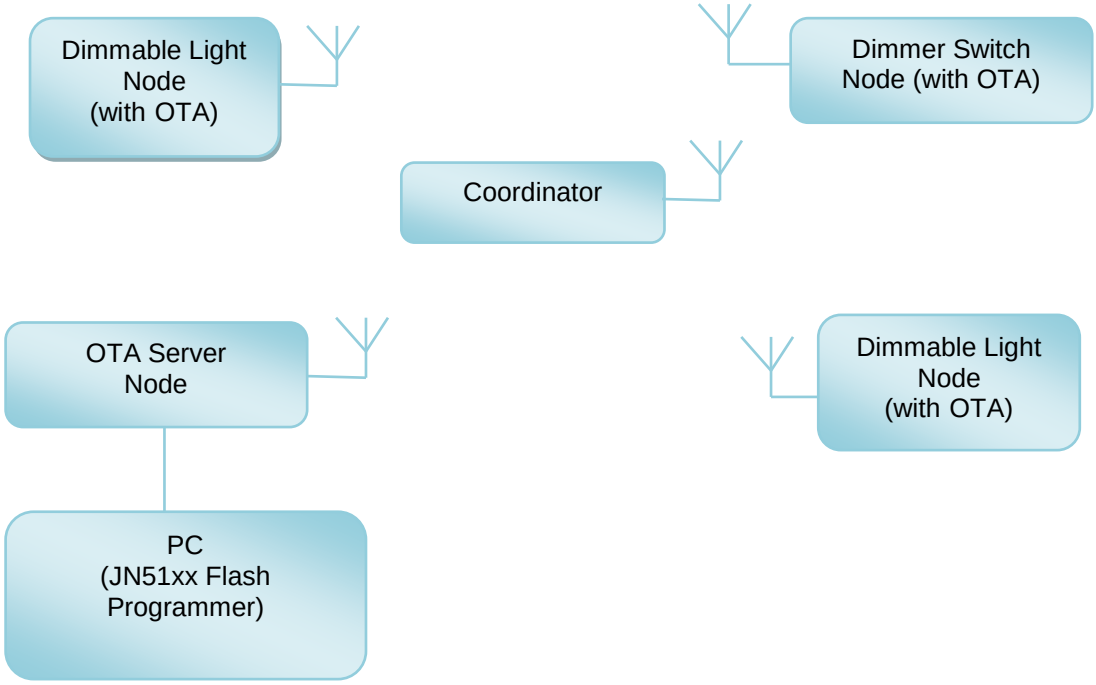
Note 1: Once started, decommissioning can be aborted by pressing any button other than SW1.

Note 2: On repeatedly pressing button SW1, the GP Switch will transmit decommissioning packets. If a commissioned light is in normal operational mode (not in commissioning or decommissioning mode), it will not process any commissioning or decommissioning packets that it receives. So this light will not be affected by decommissioning packets sent when removing another light from those controlled by the GP Switch (apart from updating its internal tables). In addition, a commissioned light in normal operational mode will ignore any received On/Off commands when another light is in decommissioning mode.

Note 3: If button SW1 on the GP Switch is pressed a certain number of times (13 by default, see Section 5.7.9) then it will be reset and lose context data about the network. In this case, the GP Switch will need to be re-commissioned in order to control the lights in the network.

8 OTA Upgrade of Devices

This section describes the OTA upgrade of nodes in an HA lighting network.



All the nodes are contained in a single HA network. The network includes a node hosting the OTA Upgrade server and one or more nodes hosting the OTA Upgrade client. The OTA clients reside on the target nodes.

8.1 OTA Upgrade Server

On power-up, the OTA server checks whether there is a valid OTA upgrade image in its external Flash memory, starting at Sector 0. If it finds a valid image, it transmits an Image Notify broadcast to the entire network.

If the OTA server receives a Query Next Image Request from an OTA client (see below), it responds with the details of the available image. The upgrade starts if the upgrade image has a different version from that of the current image running on the client node.

Note: An upgrade image can be loaded into the OTA server's external Flash memory using the JN51xx Production Flash Programmer (JN-SW-4107), described in the *JN51xx Production Flash Programmer User Guide (JN-UG-3099)*.

8.2 OTA Upgrade Client

OTA Upgrade images are downloaded to Flash memory on the OTA client nodes - internal Flash memory (with 32Kbyte sectors) for JN5169 and external Flash memory for JN5168.

A node containing the OTA client periodically sends a Query Next Image Request (by calling the function **eOTA_ClientQueryNextImageRequest()**) to the OTA server.

The following state machine diagram illustrates the OTA discovery process implemented in the client node.

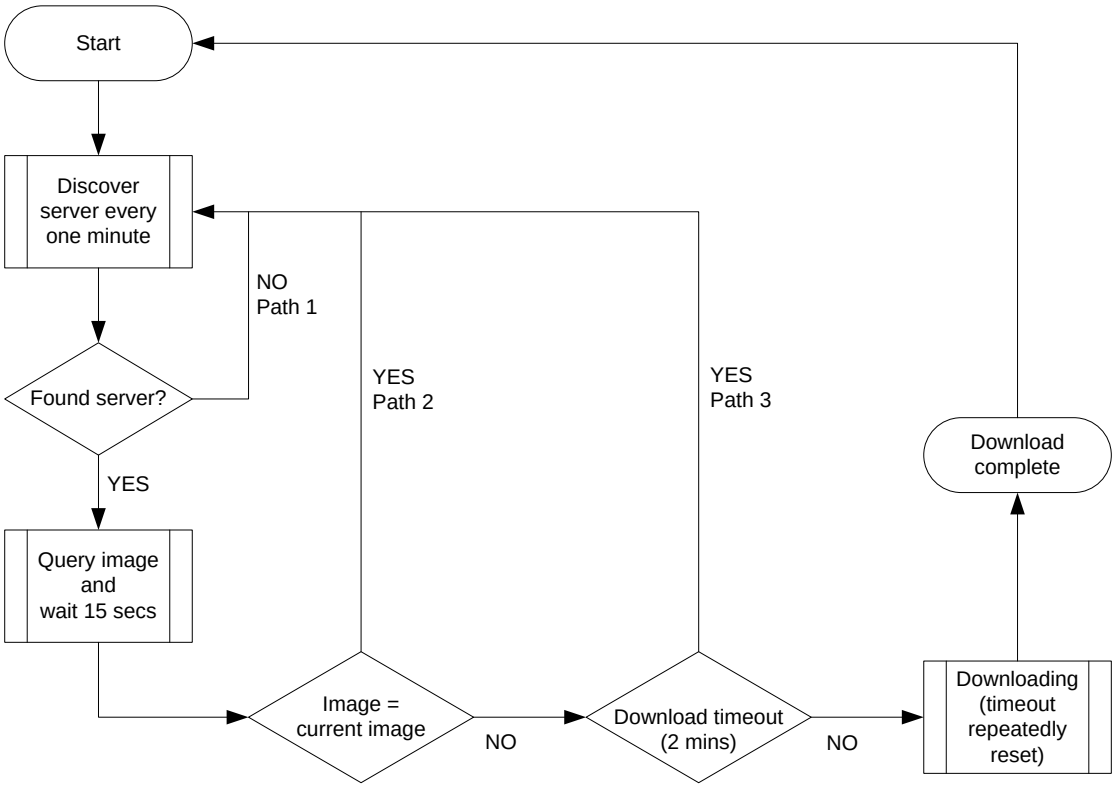


Figure 2: OTA Upgrade Discovery Process on Client

Note: The timings mentioned in the above diagram are those that are currently present in the demonstration code, but can be tuned via the appropriate macro definitions - please refer to Macros appendix of this document.

The OTA client in the above flowchart attempts to discover the OTA server and, if successful, queries the server for an upgrade image.

The following scenarios (corresponding to Paths 1, 2 and 3 in the above flowchart) do not lead to a successful download.

Path 1 – No server found:

In this case, another query is performed after 1 minute.

In the demonstration the periodicity of the server search is one minute, but in a real implementation it can be in terms of days. A macro is defined for this period.

For one minute, the macro definition is:

```
#define OTA_QUERY_TIME_IN_SEC 60
```

For one day, this becomes:

```
#define OTA_QUERY_TIME_IN_SEC 86400
```


Path 2 – Server is found but new image is same as current image:

In this case, another query for a new (different) image is performed after one minute plus a small delay to wait for a query response.

It is possible for the client to request an OTA download of an image on the server that has the same version as the client's current image. To enable this functionality, you must include the following line in the **zcl_options.h** file on the client:

```
#define OTA_ENABLE_IMAGE_RE_INSTALL
```

It is important to note that if this option is enabled, the client will continually request the same image while this image still remains on the server. It is the responsibility of the application to deal with this behaviour.

Path 3 - Server is found and image is available but download timeout occurs:

In this case, a valid image is available but the image download does not start within a timeout period (measured from the query response), where this period is defined as:

(2 * `OTA_DL_IN_PROGRESS_TIME_IN_SEC`) + random value between 1 and 2 seconds

The result is typically 3+ minutes.

8.3 OTA Upgrade Demonstration Files

Pre-built binary files for OTA upgrades (and downgrades) of devices are supplied with this Application Note. For each device type, the files are located in the corresponding **Build/OTABuild** directory. The files for a device type cover one or more of the following:

- Image upgrade (v1 to v2)
- Image upgrade (v1 to v2) on a device which uses encryption
- Image downgrade (v2 to v1)
- Image downgrade (v2 to v1) on a device which uses encryption

In the case of encrypted files, the target node must contain the same encryption key as the one in the image file (see Section 8.5).

The table below lists and describes the supplied pre-built binary files for the above cases. The binaries are provided for JN5168 and JN5169 – in the table below, <x> can be 8 or 9.

OTA Binary	Notes
OTA Upgrade Server	
OTAServer_JN516<x>.bin	To be loaded into JN516x internal Flash memory of DR1198 USB Dongle (which also acts as the network Co-ordinator)
Dimmer Switch – image upgrade	
DimmerSwitch_JN516<x>_DR1199_OTA_Client_v1.bin	Bootable v1 image to be loaded into JN516x internal Flash memory on the switch node when demonstrating an image upgrade
DimmerSwitch_JN516<x>_DR1199_OTA_v2.ota/.bin *	OTA v2 image to be loaded into JN516x external Flash memory on the OTA server when demonstrating an image upgrade
Dimmable Light – image upgrade	
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_Client_v1.bin	Bootable v1 image to be loaded into JN516x internal Flash memory on the light node when demonstrating an image upgrade
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_v2.ota/.bin *	OTA v2 image to be loaded into JN516x external Flash memory on the OTA server when demonstrating an image upgrade
Dimmable Light – image upgrade with encryption	
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_Client_v1_Enc.bin	Encrypted bootable v1 image to be loaded into JN516x internal Flash memory on the light node when demonstrating an image upgrade on a node which uses encryption
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_v2_Enc.ota/.bin *	Encrypted OTA v2 image to be loaded into JN516x external Flash memory on the OTA server when demonstrating an image upgrade on a light node which uses encryption
Dimmable Light – image downgrade	
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_Client_v2.bin	Bootable v2 image to be loaded into JN516x internal Flash memory on the light node when demonstrating an image downgrade
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_v1.ota/.bin *	OTA v1 image to be loaded into JN516x external Flash memory on the OTA server when demonstrating an image

	downgrade
Dimmable Light – image downgrade with encryption	
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_Client_v2_Enc.bin	Encrypted bootable v2 image to be loaded into JN516x internal Flash memory on the light node when demonstrating an image downgrade on a node which uses encryption
DimmableLight_JN516<x>_DR1175_LED_EXP_MONO_OTA_v1_Enc.ota/.bin *	Encrypted OTA v1 image to be loaded into JN516x external Flash memory on the OTA server when demonstrating an image downgrade on a light node which uses encryption
Colour Dimmable Light	
ColorDimmableLight_JN516<x>_DR1175_LED_EXP_RGB_OTA_<xxx>	Eight images as described above for Dimmable Light
Colour Dimmable Light with Occupancy Sensor	
OTAColorDimmableLightWithOccupancy_JN516<x>_DR1175_LED_EXP_RGB_OTA_<xxx>	Eight images as described above for Dimmable Light

Table 2: Pre-built Binary Files for OTA Upgrade/Downgrade Demonstrations

***** Each OTA upgrade binary is provided in two versions, with **.ota** and **.bin** extensions. The **.bin** file contains a 4-byte chip-specific header for use with the former JN51xx Flash Programmer v1.8.9. The **.ota** file does not contain this header and is for use with the current JN51xx Production Flash Programmer (JN-SW-4107).

8.4 OTA Upgrade Procedure

The following procedure demonstrates the OTA upgrade of a Dimmable Light application. You can implement the procedure using the NXP resources provided in this Application Note and a JN516x Evaluation Kit (the named files are for the JN5168 device but the procedure is the same for the JN5169 device with the corresponding JN5169 binary files).

Note 1: This procedure explains how to build the relevant binary files using BeyondStudio for NXP - detailed build instructions are provided in Section 10.3. However, pre-built files for OTA upgrade demonstrations are supplied with this Application Note (see Section 8.3). Therefore, you can omit the build steps and use the pre-built files, if you wish.

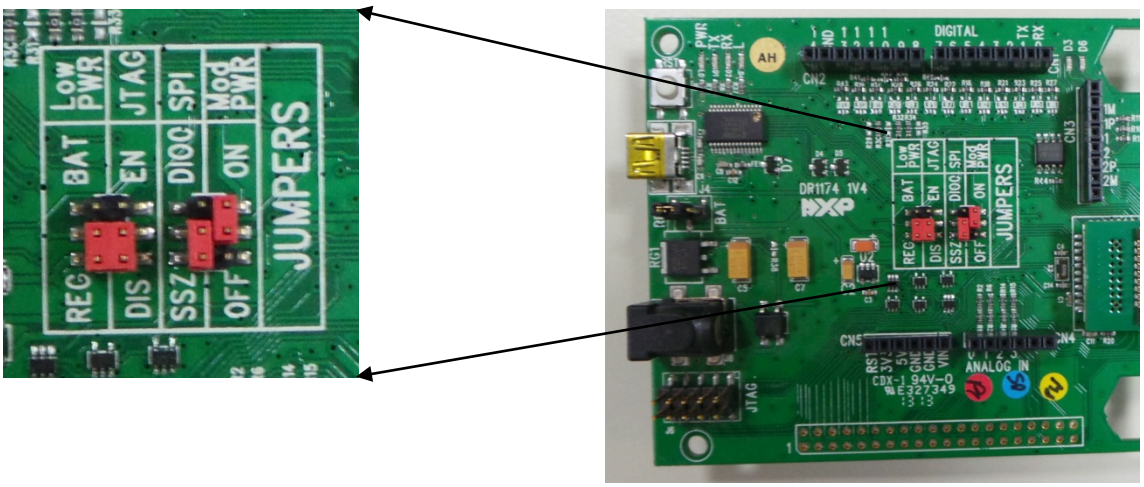
Note 2: When building an upgrade image, both a **.bin** and a **.ota** file will be produced containing the image. The **.ota** file is for use with the current JN51xx Production Flash Programmer (JN-SW-4107) and the **.bin** file is for use with the former JN51xx Flash Programmer (JN-SW-4007) – see Section 8.3. Both files are also supplied pre-built.

Note 3: This procedure assumes that you will use the JN51xx Production Flash Programmer (JN-SW-4107) to program the external Flash memory of the JN5168 device on the USB Dongle. In this case, you will need the upgrade images in the form of **.ota** files (rather than **.bin** files).

Note 4: For the JN5169 device, the macro OTA_MAX_BLOCK_SIZE should be defined as 48 (bytes) in the **zcl_options.h** file.

1. Configure hardware jumper setting for OTA upgradable light

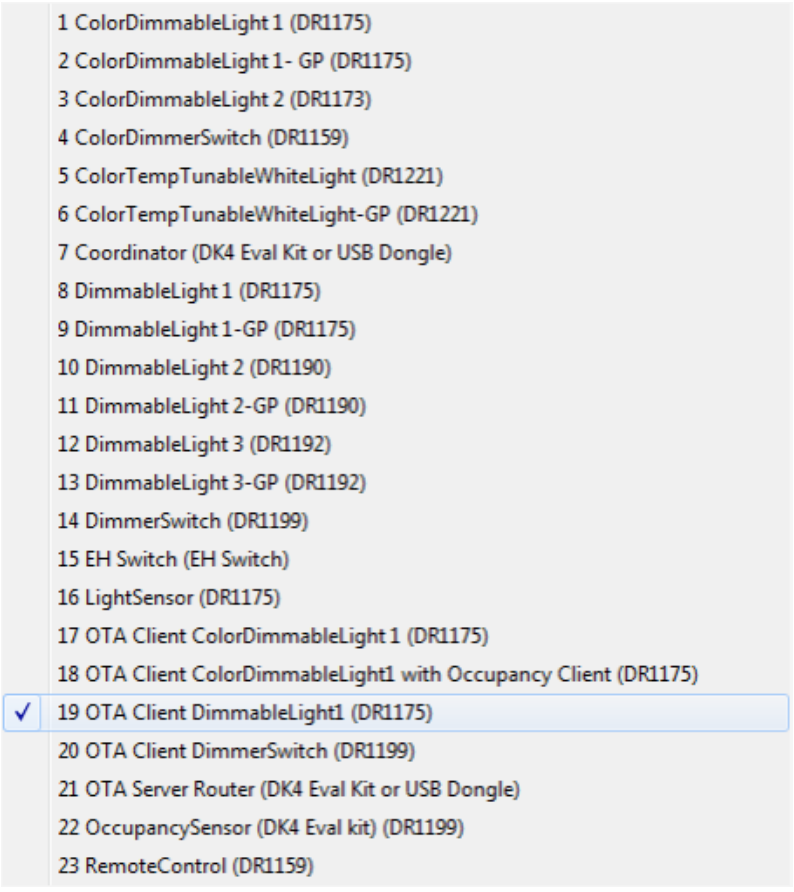
The evaluation kit hardware that is used for the Dimmable Light node is a DR1174 Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board. The Carrier Board has a SPI Flash device that can be used as external storage during the OTA download process. To make use of this Flash device, the SPI jumper must be set to “SSZ” on the Carrier Board, as shown below.



2. Compile the images for the Dimmable Light node

An OTA upgrade image has certain special properties and a special header in the binary file. This Application Note provides a build configuration to compile the Dimmable Light application code for an OTA upgrade image using the NXP JN51xx Encryption Tool (JET). The build configurations are listed below, in which the “OTA Client DimmableLight1” configuration is selected.

Note: The JN51xx Encryption Tool (JET) is provided in the JN516x ZigBee Home Automation SDK (JN-SW-4168). It must be placed in the directory **Tools/OTAUtils**. The makefile for the above configuration runs a batch script. JET is described in the *JET User Guide (JN-UG-3081)*.



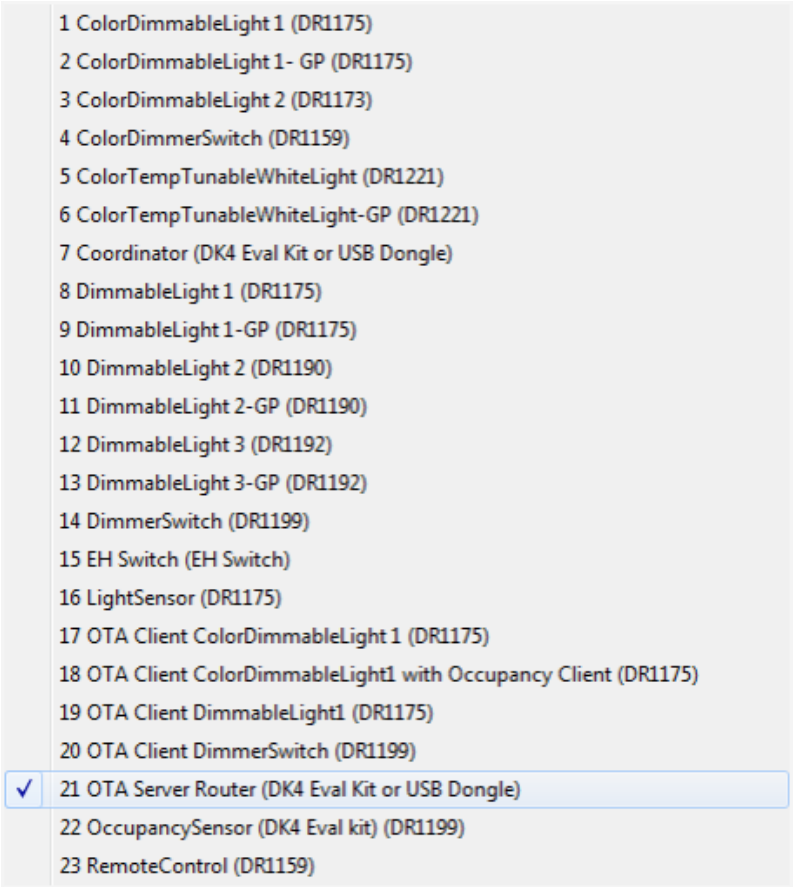
The build configuration “OTA Client DimmableLight1” will result in the following two binary files, both located in the **/DimmableLight/Build/OTABuild** directory:

- **DimmableLight_JN5168_DR1175_LED_EXP_MONO_OTA_Client_v1.bin**: This binary file must be programmed into the JN5168 module of the Dimmable Light device. This file has been coded as version 1 of the Dimmable Light image.
- **DimmableLight_JN5168_DR1175_LED_EXP_MONO_OTA_v2.ota**: This file is version 2 of the Dimmable Light image and has the OTA headers required by the OTA server. This image is required to be distributed over-air from the OTA server. Hence, this image needs to be programmed into the external Flash device of the OTA server node.

Device programming is covered in Step 4.

3. Compile the OTA server image

An OTA server node must be introduced to the network and a special build configuration “OTA Server Router” is provided to build the appropriate binary files. This build configuration is selected in the list below.



The “OTA Server Router” build configuration will create a binary filed called **OTAServer_JN5168.bin** in the **/OTAServer/Build/OTABuild** directory. This binary must be programmed into the DR1198 USB Dongle that will act as the OTA server - see Step 4.

4. Program the OTA nodes

The binary files of the OTA demonstration must now be programmed into the hardware. This involves programming binary files into JN516x internal and external Flash memory. Since external Flash devices cannot be programmed from BeyondStudio for NXP, you should use the JN51xx Production Flash Programmer (JN-SW-4107) which can program both JN516x internal and external Flash memory, and is described in the *JN51xx Production Flash Programmer User Guide (JN-SW-3099)*.

a) **OTAServer_JN5168.bin:** This image provides the OTA server functionality and needs to be programmed into the 'internal' Flash device of the DR1198 USB Dongle that is to be used as the OTA server hardware.

b) **DimmableLight_JN5168_DR1175_LED_EXP_MONO_OTA_v2.ota:** This is the Dimmable Light upgrade image (v2) and needs to be programmed into the 'external' Flash device of the USB Dongle used as the OTA server

c) **DimmableLight_JN5168_DR1175_LED_EXP_MONO_OTA_Client_v1.bin:** This is the Dimmable Light image (v1) that has OTA client capability and needs to be programmed into the 'internal' Flash device of the Dimmable Light node (DR1174 Carrier Board with external Flash device enabled and fitted with DR1175 Lighting/Sensor Expansion Board).

5. Distribute the OTA Upgrade image

The transfer of the OTA upgrade image from OTA server to client starts automatically when either of the following conditions is met:

- The OTA client in the Dimmable Light node receives an Image Notify command from the OTA server
- The OTA client receives a response to a Query Next Image Request with a file version which is different from that of the currently running image

The transfer can alternatively be initiated manually (before it is initiated automatically) by pressing the Reset (RST) button on the OTA client node.

Whenever there is OTA activity, the orange LED on the OTA server dongle lights up.

Note: To upgrade to another version (v3) of the OTA image, prepare a **DimmableLight_JN5168_DR1175_LED_EXP_MONO_OTA_v3.ota** file using the JET utility. Repeat the programming step 4b to store this upgrade image in the OTA server. Power cycle the OTA server node and the download will start when either of the conditions mentioned in Step 5 is met.

8.5 Serving OTA Encrypted Images

The OTA server can serve application images to clients that implement encryption. The encrypted OTA upgrade image for the client needs to be stored in the server's external Flash memory.

In this application, an encrypted OTA image is indicated by the most significant bit of the `ImageType` parameter in the OTA header. For example:

Unencrypted Dimmable Light is represented by `ImageType` = 0x0101

Encrypted Dimmable Light is represented by `ImageType` = 0x1101

8.5.1 Encrypting an Image

In this application, an encrypted image is prepared using the JET tool. The JET commands are invoked through a batch file named **<Light or Switch >CreatOtaEncClient.bat** located in the **/OTABuild** folder. This batch file is called from the respective Makefile when building the target with the OTA option enabled. The image needs to be encrypted using the same encryption key as is present in the target device (see Section 8.5.2 below). For details of the encryption options using the JET tool, refer to the *JET User Guide (JN-UG-3081)*.

As part of the existing build of the OTA versions of the Dimmable Light and Dimmer Switch applications, the encrypted binary and OTA upgrade images are generated in the **/OTABuild** folder.

For JN5168, the above build process will produce an encrypted image and a non-encrypted image, but the latter may not be needed. For JN5169, a macro is provided which allows just one image to be built, encrypted or non-encrypted. This macro, OTA_ENCRYPTED, must be included in the make command for the build:

- OTA_ENCRYPTED=1 to generate an encrypted image
- OTA_ENCRYPTED=0 to generate an non-encrypted image

For example, the following make command builds an encrypted Colour Dimmable Light application image for the JN5169 device:

```
make LIGHT=ColorDimmableLight DR=DR1175 REV=r1v1 TYPE=RGB OTA=1
JENNIC_CHIP=JN5169 OTA_ENCRYPTED=1
```

Note that you must perform a clean build every time an encrypted or non-encrypted image is re-built for JN5169.

8.5.2 Programming an Encryption Key into a Device

In order to implement encryption/decryption, a JN516x module must be programmed with an 128-bit AES encryption key. This is stored in the One Time Programmable (OTP) memory within the Index Sector of the chip's internal Flash memory - see the datasheet for your chip.

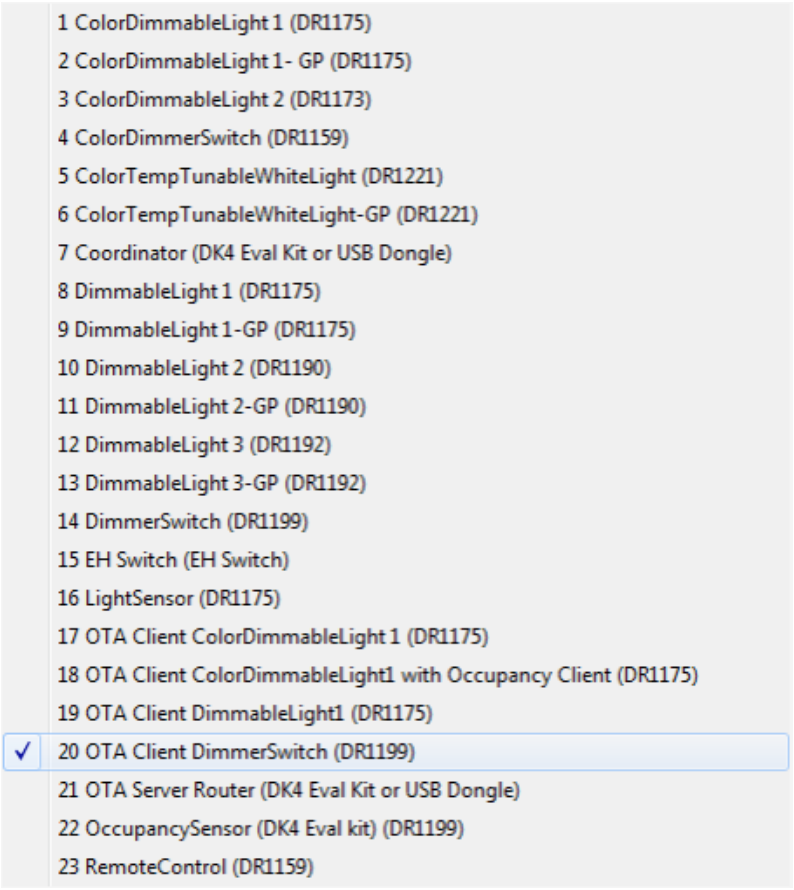
The AES encryption key can be programmed into a JN516x device using the JN51xx Production Flash Programmer (JN-SW-4107) with the `--deviceconfig=` option, as described in the *JN51xx Production Flash Programmer User Guide (JN-UG-3099)*.

8.6 OTA Support for a Battery-Powered End Device (Target)

An End Device is often battery-powered and may also be sleepy (goes through sleep/wake cycles) to conserve power. The OTA support for battery-powered End Devices in this demonstration includes constantly monitoring the battery voltage, since an OTA upgrade should not continue when the supply voltage becomes low.

Note: A sleepy End Device receives data by polling its parent for buffered data packets while awake (it cannot receive data while asleep) and this is how it receives OTA upgrade images. The sleep period does not change during an OTA upgrade.

In this demonstration, the only End Device that supports OTA upgrades is the Dimmer Switch. A build configuration for a Dimmer Switch OTA client is provided and is called "OTA Client DimmerSwitch" (this is the same as the OTA build configuration for the Dimmable Light device).



An encrypted OTA upgrade image can be produced for the Dimmer Switch as described in Section 8.5.1.

To enable the voltage check, the compile-time option CHECK_VBO_FOR_OTA_ACTIVITY must be defined. In this demonstration, it is enabled in the Dimmer Switch makefile for the OTA build.

When the voltage check is enabled, the following behaviour is expected:

- The Dimmer Switch will not enter ‘deep sleep’ mode once an OTA upgrade has been started, but ‘warm sleep’ cycles will continue.
- Upon reset or wake-up, the voltage monitoring is initialised to check for the battery voltage falling below 2.4V (defined by the macro APP_OTA_VBATT_LOW_THRES).
- OTA upgrade activities such as server discovery and block download requests continue while the battery voltage is at or above 2.4V but stop when it falls below this threshold.
- When OTA upgrade activity stops, the device enters ‘deep sleep’ mode when there is no further user activity (such as a button-press) before the expiry of the ‘deep sleep’ counter.
- If the device is non-sleepy (KEEPALIVETIME is equal to 0 in the build configuration) then a suspended OTA upgrade will resume only if the voltage rises back above 2.7V (defined by the macro APP_OTA_VBATT_HI_THRES).

9 Advanced User Information

9.1 Saving Network Context

All device types are protected from losing their network configuration during a power outage by means of context saving. The required network parameters are automatically preserved in non-volatile memory by the ZigBee PRO Stack (ZPS). On restart, the radio channel, Extended PAN ID (EPID) and security keys are restored.

Application-specific information can also be preserved in the non-volatile memory, which is most commonly used to preserve the application's operating state.

9.2 Security Key

The HA profile uses a public pre-configured link key. This link key can be obtained from the *ZigBee HA Profile Specification*.

9.3 Adding More Devices to the Network

The maximum number of devices in the network is set to 10 in the ZigBee PRO Stack (ZPS) Configuration Editor. However, this value can be increased to have more lights and control units in the network. This is done by modifying the ZigBee Network Parameters in the ZPS Configuration Editor. These parameters and the editor are described in dedicated chapters of the *ZigBee PRO Stack User Guide (JN-UG-3101).*

The maximum number of GP Switches in the network is set to 2. This can be increased by increasing the sizes of the Translation table, Sink table, GP Transmit Queue and GP Security table. The Translation table and Sink table sizes can be configured in **zcl_options.h** - these parameters are described in the *ZigBee Green Power User Guide (JN-UG-3095)*. The GP Transmit Queue and GP Security table sizes can be configured in the ZPS Configuration Editor.

9.4 Adding More Groups to a Light

Each light is configured with a group addressing table of size 8. This allows each light to be a member of 8 different groups. To increase this group addressing table size, increase the value of the "Group Addressing Table Size" in the ZPS Configuration Editor. Also update the CLD_GROUPS_MAX_NUMBER_OF_GROUPS macro in the **zcl_options.h** file to match the updated value of "Group Addressing Table Size" in the ZPS Configuration Editor.

9.5 Adding More Scenes to a Light

Each light is configured to have 16 scenes. This allows 16 scenes to be added in the device. To increase the number of scenes, edit the CLD_SCENES_MAX_NUMBER_OF_SCENES macro in the **zcl_options.h** file.

9.6 OTA Upgrade Image Validation

Normally during an OTA upgrade, the entire upgrade image is downloaded to the OTA client before the image is validated by checking the embedded link key - if the link key in the image does not match the one in the target device then the image is rejected as an invalid image.

This is an inefficient way of validating a new application image - since the link key is contained in the image header, the validation can be performed once the first Kilobyte of the image has been downloaded. The embedded link key can then be extracted and compared with the link key in the device. If there is no match, the download can be aborted. One

Kilobyte is chosen to ensure that the link key in the upgrade image is written to Flash memory on the target device before the validation begins.

For an encrypted image, the link key must be decrypted before the comparison can be performed. If the encryption key on the target device (used to decrypt the image) is not the same as the key used to encrypt the image then the validation will fail (even if the embedded link key is the correct one).

9.7 EZ-mode Commissioning

EZ-mode Commissioning is used in the demonstration to a join a node to the network and to establish one or more pairings between the joined node and other nodes in the network. For such a pairing, there are two types of device from a commissioning perceptive:

- **Initiator:** The device that will act as a source for operational transactions
- **Target:** The device that will need to receive operational transactions

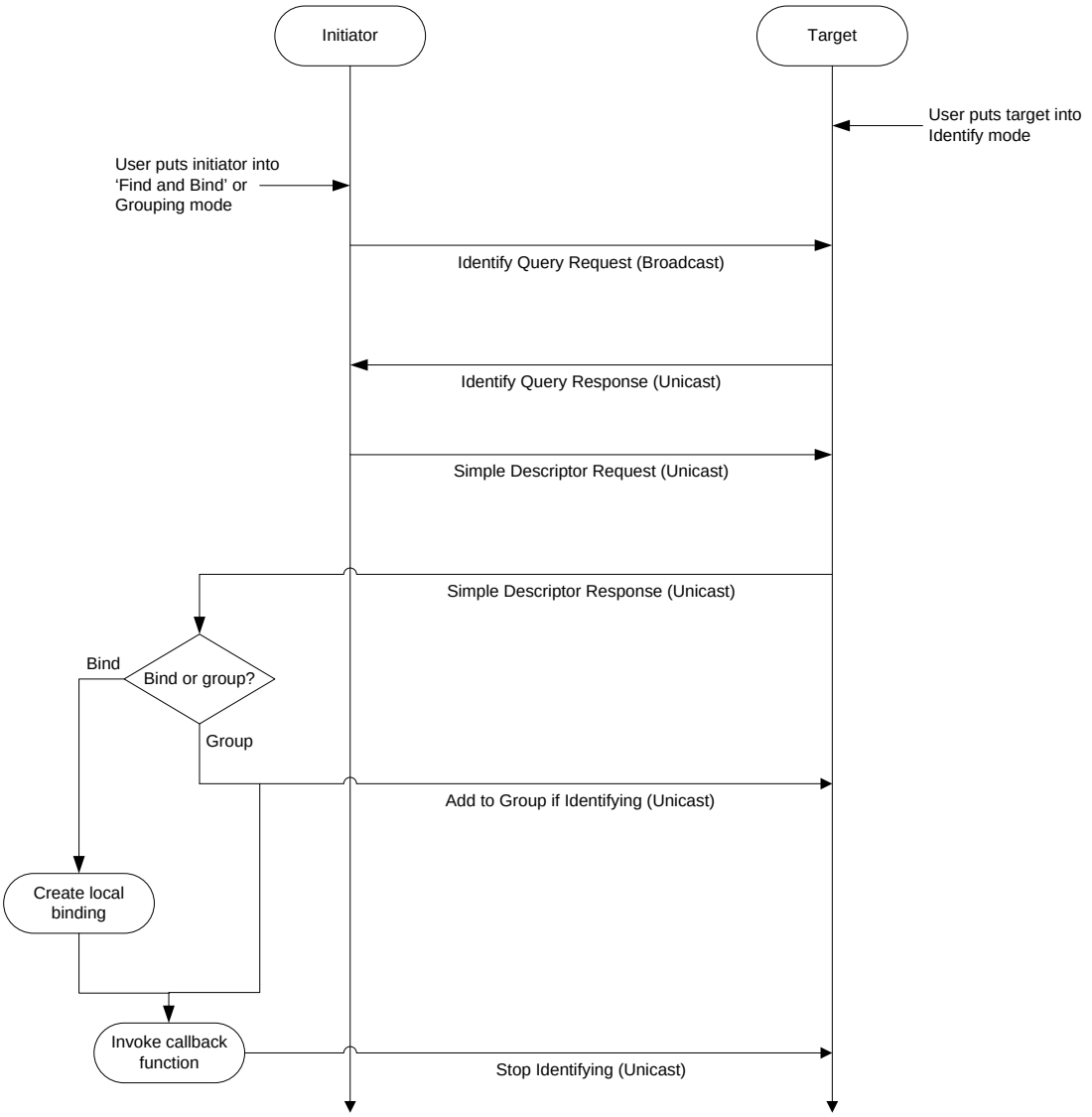
For a complete introduction to EZ-mode Commissioning, refer to the *ZigBee Cluster Library User Guide (JN-UG-3103)*.

Once a node has joined the network, if it needs to perform one-to-many operational transactions with other devices in the network then it should be grouped or bound to the relevant devices. Grouping and binding both allow this one-to-many communication but there are factors that should be taken into consideration.

- **Binding:** This is best confined to situations with a small number of target devices (up to four). In this case, the initiator needs to store all the target information in a Binding table, requiring a table entry for each relevant cluster on each relevant endpoint on each target node. Therefore, if the number of target devices is large then the Binding table will be large and consume memory. Binding is suitable when a device needs to report regularly to a small number of target devices.
- **Grouping:** This is appropriate when an operation is to be performed with a large number of network devices (more than four) - for example, for an On/Off Switch which can send an On or Off command to a large set of lights that should all receive and execute the same command.

The following diagram illustrates the ‘Find and Bind/Group’ process within EZ-mode Commissioning. This requires a button to be pressed on a target to put it in Identify mode and a button to be pressed on the initiator to start the ‘Find and Bind/Group’ phase.

Note: In the case of binding, depending on whether the initiator has the target’s IEEE address available locally in the device (Address Map entry), it may need to perform an IEEE address look-up before it can perform the binding.



By a similar process to the above, a node can be removed from a group.

In this application, groups and bindings can be reset by performing an EZ-mode Commissioning reset invoked by a user-defined button. This causes the devices to remove their binding and group entries allowing the device to be re-commissioned with another set of devices.

This application provides methods to perform the grouping and binding of targets through functions contained in **haEzFindAndBind.h.** Details of the functions are provided in the EZ-mode Commissioning chapter of the *ZCL User Guide (JN-UG-3103)*. However, for a quick reference, the functions are listed below with brief descriptions.

eEZ_FindAndBind(): This function can be called by both the initiator and target devices at any point in time to start the 'Find and Bind' phase. It requires the local endpoint number and the type of the calling device.

- On a target device, this call will cause the device to enter Identify mode for a time defined by EZ_MODE_TIME (180 seconds, be default).
- On an initiator device, this call will result in the transmission of an Identify Query Request to discover the target devices that are currently in Identify mode. This discovery process is repeated with a period defined by EZ_RESPONSE_TIME

(10 seconds, by default) until the timeout defined by EZ_MODE_TIME (180 seconds, by default) expires or the user exits this mode by taking an action which results in a call to **vEZ_Exit()**.

When a target is discovered by responding to an Identify Query Request, the initiator performs a further Simple Descriptor Request to discover the matching clusters and, if required, an IEEE address look-up after which it binds to the target via its local Binding table.

eEZ_Group(): This function is similar to the above function except this one starts the Grouping process at the initiator (instead of binding).

vEZ_SetGroupId(): This function sets the Group ID that will be used by the initiator for grouping (and must be called before **eEZ_Group()**).

u16EZ_GetGroupId(): This function returns the Group ID currently in use by the initiator.

vEZ_Exit(): This function causes the device to exit the 'Find and Bind' or Grouping process.

vEZ_FactoryReset(): This function performs a reset of the Group and Binding tables on the local endpoint. It can be used by both initiator and target.

- When called on a target, it removes all the local Group table entries
- When called on the initiator, it removes all the local Binding table entries. If there is a group server, it removes all the local Group table entries and sends a Remove Group command as a groupcast to allow the targets in the group to be freed up.

eEZ_GetFindAndBindState(): This function returns the current state of the EZ-mode Commissioning process on the local endpoint.

eEZ_ExcludeClusterFromEZBinding(): This function can be used to exclude clusters that are not intended to be bound, in order to save Binding table space on the initiator. By default, all the clusters on the endpoint will be considered for binding.

vEZ_EZModeNWKFindAndBindHandler(): This function needs to be called on the initiator when a stack event occurs. It handles transactions such as the Simple Descriptor Response and IEEE Address Response from the target during EZ-mode Commissioning.

vEZ_EPCallBackHandler(): This function needs to be called on the initiator from the endpoint callback function. It handles the Identify cluster commands such as Query Identify Response from potential targets during EZ-mode Commissioning.

vEZModeCb(): This callback function is invoked by the EZ-mode Commissioning process when a grouping or binding is created on the initiator. The developer should add further actions within this function based on the possible events.

9.8 Report Manager

A Report Manager is available that is responsible for generating reports (as configured) on a node. Depending on the usage of system resources, the Report Manager may sometimes need to retransmit a report. The Report Manager is an optional feature that, if required, must be enabled (see below). The feature is normally required for periodic reporting, such as from a sensor. However, if a sensor does not require use of the Report Manager due to power saving preferences, the feature can be left unused in order to achieve a smaller code size.

If required, the Report Manager must be enabled in the ZCL using the following macros in the **zcl_options.h** file:

- Define CLD_BIND_SERVER
- Define MAX_NUM_BIND_QUEUE_BUFFERS to an appropriate value
- Define MAX_PDU_BIND_QUEUE_PAYLOAD_SIZE to an appropriate value

For more information on the above macros, refer to the *ZigBee Cluster Library User Guide (JN-UG-3103)*.

10 Developing with the Application Note

This section provides additional information that may be useful when developing with this Application Note.

10.1 Useful Documents

Before commencing a ZigBee Home Automation development, you are recommended to familiarise yourself with the following documents:

[R1] - JN-UG-3101 ZigBee PRO User Guide
[R2] - JN-UG-3075 JenOS User Guide
[R3] - JN-UG-3076 ZigBee Home Automation User Guide
[R4] - JN-UG-3103 ZigBee Cluster Library User Guide
[R5] - JN-UG-3087 JN516x Integrated Peripherals API User Guide
[R6] - JN-UG-3095 ZigBee Green Power User Guide
[R7] - ZigBee HA Profile Specification
[R8] - ZigBee Cluster Library (ZCL) Specification
[R9] - ZigBee Green Power Profile Specification
[R10] - docs-13-0553-37-00ha-ha-1-2-errata-document

Documents [R1] to [R6] can be obtained from the Wireless Connectivity area of the NXP web site, while documents [R7] to [R10] can be obtained from the ZigBee Alliance web site.

10.2 Debugging the Demonstration Application

10.2.1 Serial Debug

Each node in the demonstration prints out debug information via the UART port based on the debug flags set in the Makefile. This debug information can be viewed using terminal emulator software, e.g. Tera Term. Connect the node of interest to a PC using the Mini-USB cable (supplied in the evaluation kit) and configure the terminal emulator's COM port as follows:

BAUD Rate	115200
Data	8 bits
Parity	None
Stop bit	1 bit
Flow Control	None

Debug can be disabled for production by setting the 'Trace' flag in the relevant node's Makefile to zero. The Makefile also defines a subset of debug flags that allows localised debug statements to be collectively enabled or disabled, e.g. TRACE_START.

By default, there are certain debug print lines left in the Application Note code to trace any issues.

10.2.2 JTAG Debug

The application on a node can be debugged from BeyondStudio for NXP via a JTAG connection. This method requires additional hardware to form the JTAG interface on the node, including a JTAG expansion board and JTAG adaptor/dongle. JTAG debugging is fully described in the Application Note *JN516x JTAG Debugging in BeyondStudio (JN-AN-1203)*.

10.3 Building and Loading the Application

This section provides application build instructions. If you simply wish to use the supplied application binaries, refer to Section 4.

10.3.1 Pre-requisites and Installation

Before you start to build and load the application, please ensure that you have following installed on your development PC:

- BeyondStudio for NXP (JN-SW-4141)
- JN516x ZigBee Home Automation SDK (JN-SW-4168)

Note: For the installation instructions, please refer to *BeyondStudio for NXP Installation and User Guide (JN-UG-3098)* and the Release Notes supplied with the JN516x ZigBee Home Automation SDK (JN-SW-4168).

In order to build the application, this Application Note (JN-AN-1189) must be unzipped into the directory:

<BeyondStudio for NXP installation root>\workspace

where **<BeyondStudio for NXP Installation root>** is the path into which BeyondStudio for NXP was installed (by default, this is **C:\NXP\bstudio_nxp**). The **workspace** directory is automatically created when you start BeyondStudio for NXP.

All files should then be located in the directory:

…\workspace\JN-AN-1189-ZigBee-HA-Demo

There is a sub-directory for each application, each having **Source** and **Build** sub-directories.

10.3.2 Build Instructions

The software provided with this Application Note can be built for the JN5168 or JN5169 device.

The applications can be built from the command line using the makefiles or from BeyondStudio for NXP – makefiles and Eclipse-based project files are supplied.

- To build using makefiles, refer to Section 10.3.2.1.
- To build using BeyondStudio for NXP, refer to Section 10.3.2.2.

Note: An OTA upgrade build can be enabled by uncommenting the line `OTA=1` in the manufacturer configuration file **manu_config.mk**. Alternatively, when building from the command line using makefiles (see Section 10.3.2.1), the option `OTA=1` can be added to the build command.

10.3.2.1 Using Makefiles

This section describes how to use the supplied makefiles to build the applications. Each application has its own **Build** directory, which contains the makefiles for the application.

To build an application and load it into a JN5168 or JN5169 device, follow the instructions below.

> **Note:** The make commands given below will build the application according to the default build options in the makefile (e.g. device type). To use alternative build options, these must be specified in the make command. The required options for different builds can be obtained from the build configurations provided in BeyondStudio for NXP.

1. Ensure that the project directory is located in

   <BeyondStudio for NXP installation root>\workspace

2. Start an MSYS shell by following the Windows Start menu path:
   All Programs > NXP > MSYS Shell

3. Navigate to the **Build** directory for the application to be built and follow the instructions below for your chip type:

   For JN5168:

   At the command prompt, enter:

   ```
   make clean all
   ```

   Note that for the JN5168, you can alternatively enter the above command from the top level of the project directory, which will build the binaries for all applications.

   For JN5169:

   At the command prompt, enter:

   ```
   make JENNIC_CHIP=JN5169 clean all
   ```

   In both of the above cases, the binary file will be created in the **Build** directory, the resulting filename indicating the chip type (e.g. **5168**) for which the application was built.

4. Load the resulting binary file into the device. You can do this from the command line using the JN51xx Production Flash Programmer (JN-4107), described in the *JN51xx Production Flash Programmer User Guide (JN-UG-3099)*.

10.3.2.2 Using BeyondStudio for NXP

This section describes how to use BeyondStudio for NXP to build the demonstration application.

To build the application and load it into JN5168 or JN5169 devices, follow the instructions below:

1. Ensure that the project directory is located in

   <BeyondStudio for NXP installation root>\workspace

2. Start the BeyondStudio for NXP and import the relevant project as follows:

   a) In BeyondStudio, follow the menu path **File>Import** to display the **Import** dialogue box.

   b) In the dialogue box, expand **General**, select **Existing Projects into Workspace** and click **Next**.

   c) Enable **Select root directory** and browse to the **workspace** directory.

   d) In the **Projects** box, select the project to be imported and click **Finish**.

3. In the makefile(s) for application(s) to be built, ensure that the JN516x chip on which the application is to run is correctly specified in the line beginning JENNIC_CHIP. For example, in the case of the JN5169 device, this line should be:

   ```
   JENNIC_CHIP=JN5169
   ```

4. Build an application. To do this, ensure that the project is highlighted in the left panel of BeyondStudio and use the drop-down list associated with the hammer icon  in the toolbar to select the relevant build configuration – once selected, the application will automatically build. Repeat this to build the other applications.

   The binary files will be created in the relevant **Build** directories for the applications.

5. Load the resulting binary files into the devices. You can do this using the integrated Flash programmer, as described in the *BeyondStudio for NXP Installation and User Guide (JN-UG-3098)*.

Note: To program a binary file into JN516x external Flash memory, you will need to use the JN51xx Production Flash Programmer (JN-4107), described in the *JN51xx Production Flash Programmer User Guide (JN-UG-3099)*. This tool can be used to program JN516x internal or external Flash memory.

10.4 Application Start-up

This section describes the typical start-up flow of an NXP ZigBee PRO device. Note that not all devices sleep, hence the 'Warm Start' path is not always applicable.

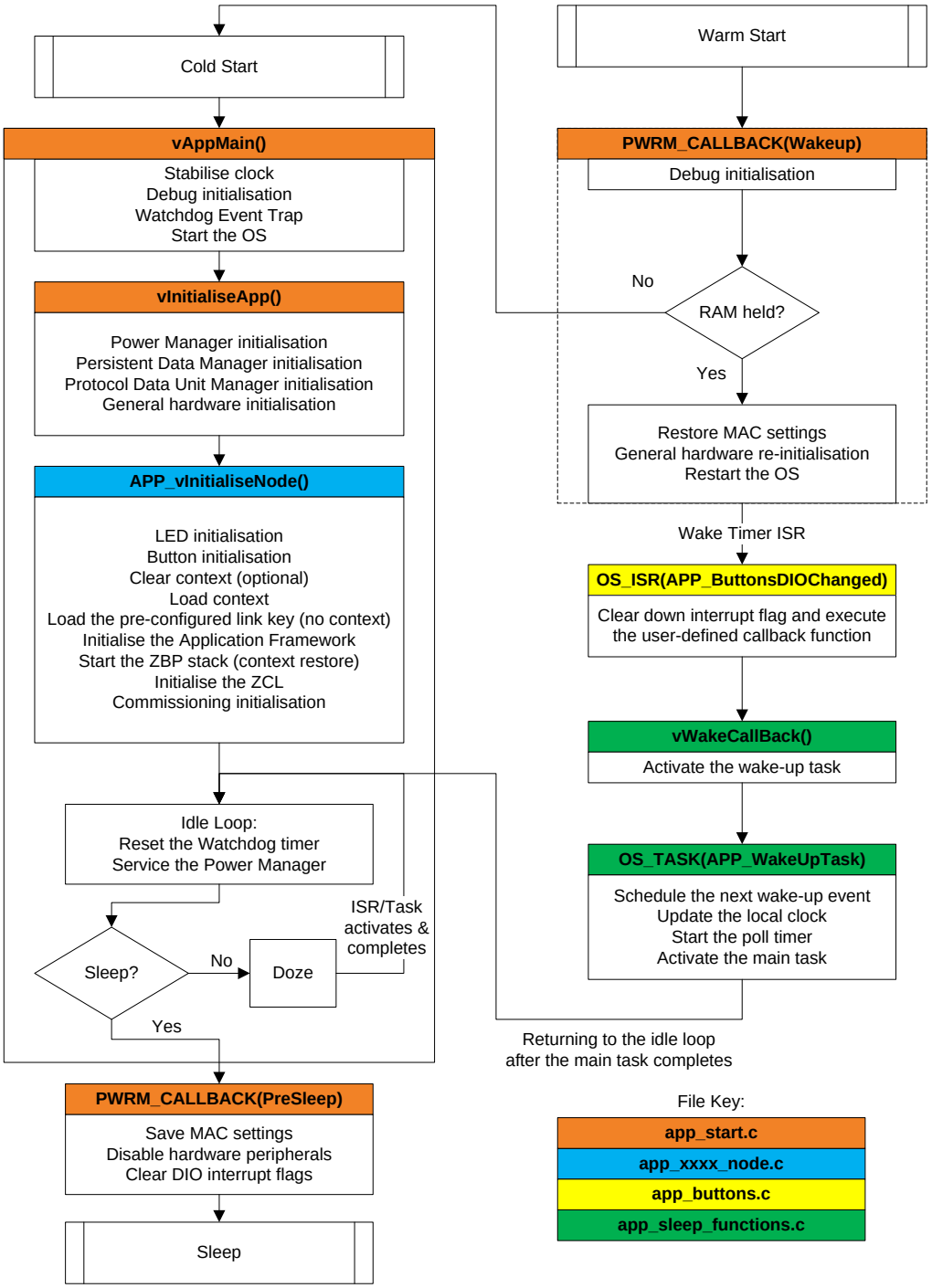


Figure 3: Typical Start-up Flow

10.5 HA Device Start-up

The start-up flows for a light node (Dimmable Light or Colour Dimmable Light) and switch node (Dimmer Switch or Colour Dimmer Switch) with respect to different states (Factory New or Non-Factory New), are described in the sub-sections below.

10.5.1 Factory New Light

The light node acts as a Router in the ZigBee network. On power-up, a 'Factory New' light performs a network discovery on each channel selected in the ZPS configuration. Once the scan is complete, the light node will attempt MAC association with the discovered networks.

10.5.2 Non-Factory New Light

Once the light node has joined a network, the ZigBee PRO stack and the application save the network context and commissioning parameters to the on-chip EEPROM. On subsequent power cycles, the light node restores these parameters and then functions the same as before the reset.

10.5.3 Factory New Switch

The switch node acts as an End Device in the ZigBee network. On power-up, a 'Factory New' switch performs a network discovery on each channel selected in the ZPS configuration. Once the scan is complete, the switch node will attempt MAC association with the discovered networks.

10.5.4 Non-Factory New Switch

Once the switch node has joined a network, the ZigBee PRO stack and the application save the network context and commissioning parameters to the on-chip EEPROM. On subsequent power cycles, the switch node restores these parameters and will function the same as before the reset.

10.6 Guidelines for Modifying the Switch

This section highlights the key areas of interest within the code, in case the developer wishes to alter the switch node's functional states or move to a different user interface.

10.6.1 Operational State Machine

The operational state machine (**sDeviceDesc.eNodeState**) is located within **zha_switch_node.c** for the Dimmer switch and within **zha_remote_node.c** for the Remote Control device. Additional states must be added to this switch statement if further operational modes are required.

10.6.2 Handling a Key Press and Release

For Dimmer Switch:

The function that handles a key press and release is located in **app_switch_state_machine.c** as part of APP_ZHA_Remote_Task. The events related to key input are processed in this task with the **sAppEvent.eType** event as a parameter. Any changes to the key handling should be made within the corresponding switch statement. The file also contains the key-press handler function, **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.

For Colour Dimmer Switch or Remote Control:

The DR1159 Remote Control Unit is used to implement a Colour Dimmer Switch device or a Remote Control device. The unit contains a capacitive-touch screen which senses key touches - the driver files for this functionality are **DriverCapTouch.c** and **app_captouch_buttons.c**. When a touch is detected, an application event is raised for the remote task with **sAppEvent.eType** as a parameter.

This event is subsequently translated to the appropriate commands - the translation is located in the file **zha_remote_node.c**. Any changes to the key map should be made within this file.

10.7 Joining and Re-joining

This application implements a common joining state machine in the **haEzJoin.c** file. All of the join and re-join implementation is included in this file.

The function **vEZ_EZModeNWKJoinHandler()** is used for both joining and re-joining purposes by specifying the appropriate action in the function parameters. This function is a wrapper on top of the underlying stack function and is managed by a state machine. The function is called from the application when a stack event occurs.

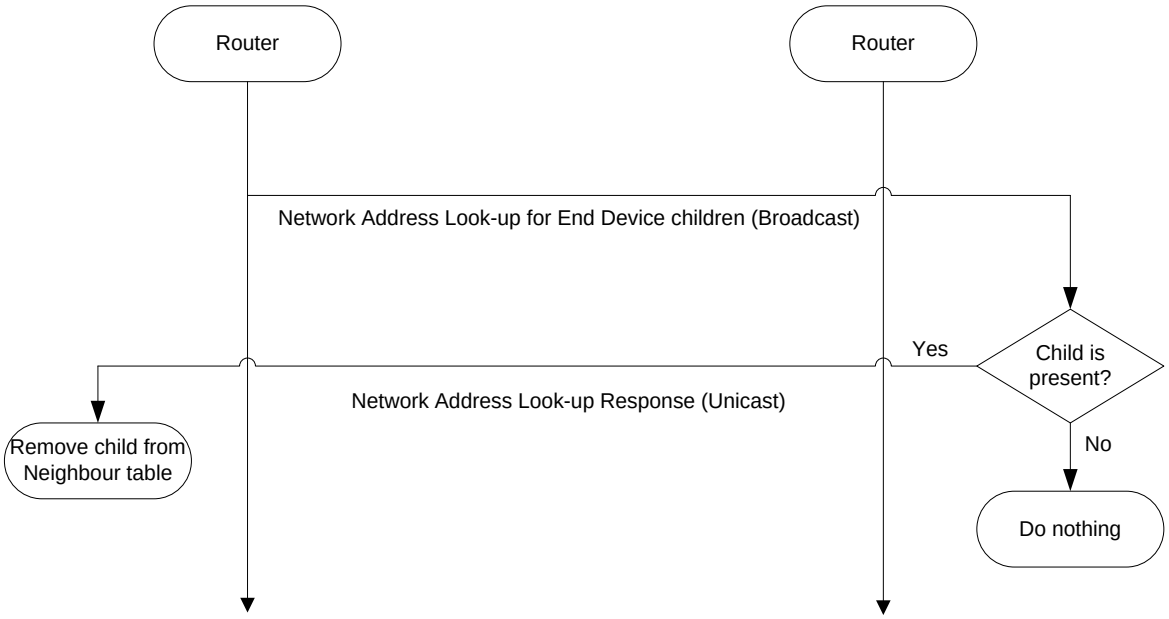
When an End Device loses its parent, the above function is called with an action flag to start a re-join. The re-join behaviour of the End Device is summarised below for each relevant HA Lighting device type used:

- **Dimmer Switch**
    - Detects parent loss when a poll or ping fails, and initiates joining process
    - Can continue its sleep/wake cycle during the joining process
- **Light Sensor**
    - Detects parent loss when a poll or ping fails, and initiates joining process
    - Can continue its sleep/wake cycle during the joining process
- **Occupancy Sensor**
    - Detects parent loss when a poll or ping fails, and initiates joining process
    - Can continue to sleep for 50 seconds and stay awake for 10 seconds in order to attempt a re-join once per minute
- **Colour Dimmer Switch or Remote Control**
    - Detects parent loss when a poll or ping fails, and initiates joining process
    - If no parent found, continues to sleep and scan the keypad - as soon as the user interacts via the keypad, a re-join is initiated and the unit resumes normal operation once the re-join is complete
    - Re-join process involves scanning all channels for the parent – during this process, a user interaction via the keypad brings the re-joining process back to the last known operating channel of the device to make the joining faster

The above function and other associated functions are detailed in the EZ-mode Commissioning chapter of the *ZCL User Guide (JN-UG-3103)*.

10.8 Child Ageing

The application takes into account child ageing for Routers and the Coordinator upon a power-on reset. The following diagram illustrates the child ageing implementations.



On a power-on reset of a Router, the device broadcasts a 'Network Address Look-up' request containing the IEEE addresses of the sleepy End Devices from its Neighbour table (NT) that were previously children of the Router (before the reset). On receiving the request, any other Router that has a relevant address in its NT will respond. On receiving this response, the initiating Router assumes that the specified End Device child has moved to the parent which has just responded and it removes the relevant entry from its own NT.

10.9 Pinging Parent

In the above mechanism for ageing children, there is one possible scenario where the Router parent has removed an End Device child from its NT while the child is asleep but, on waking, the child still thinks this Router is its parent that it should be polling for data. To avoid this scenario, in this application there is a mechanism for the sleepy End Device to ping its parent at 60-second intervals. If there is no response from the parent, the End Device then re-joins the network.

11 Useful Device Information

11.1 Light and LED Indication of Different States

The light and LED indications on each of the network devices in the demonstration are summarised below:

Device	Light/LED Indication
All lights	**Breathe:** Light node is joining the network
	Single Flash: User input accepted during power-cycling
	Constantly ON: Light node has joined the network
Dimmer Switch	**LEDs D1, D2 and D3 flash alternately:** Joining or re-joining the network
	LEDs D1,D2 and D3 constantly OFF: Node in the network
Colour Dimmer Switch (Remote Control Unit)	**Both LEDs flash 1 second ON, 1 second OFF:** Joining or re-joining the network
	One or both LEDs constantly ON or OFF (no flashing): Node in the network
	Both LEDs flash quickly: Low battery
	Right LED flashes 800ms OFF, 200ms ON: Unit is sleeping with key scanning
Light Sensor	**LED D4 flashes 1 second ON, 1 second OFF:** Joining or re-joining the network
	LED D4 flashes 250ms ON, 250ms OFF: Keep-alive mode
	LED D4 OFF: Node is in the network or sleeping
	LED D4 flashes 500ms ON, 500ms OFF: Initiator mode active
Occupancy Sensor	**LED D1 (Sensor State):** ON – Occupied, OFF - Unoccupied
	LED D2 flashes 1 second ON, 1 second OFF: Initiator mode active
	LED D3 flashes 1 second ON, 1 second OFF: Joining or re-joining the network
	LED D3 flashes 250ms ON, 250ms OFF: Keep-alive mode
	During sleep, LED D1 indicates the sensor state but LED D3 is OFF
Co-ordinator (USB Dongle)	**Green LED flashes:** Formed the network
	Amber LED flashes: 'Permit Join' enabled
OTA Server (USB Dongle)	**Green LED On :** Joined the network
	Amber LED flashes: OTA activity

11.2 Sleep Cycles in the End Devices

In this application, the different End Devices employ different sleep and wake periods, as summarised below.

Note: In this section, references are made to 'warm sleep'. This is sleep with RAM held – that is, the contents of RAM are preserved during sleep.

11.2.1 Dimmer Switch

Once the Dimmer Switch is in the network, the device will cycle through 6 seconds of being awake and 6 seconds of 'warm sleep' (sleep with RAM held). During this sleep, the device can be woken by pressing any of the buttons SW1-SW4 or DIO8.

A 'deep sleep' timer is also started which, by default, is set to 1 minute. This timer is reset when there is user activity. If the timer expires (due to no user activity), the device enters deep sleep mode. The device can be woken from deep sleep by pressing any of the buttons SW1-SW4.

> **!** ***Caution:*** *The button DIO8 should not be pressed to wake the device from deep sleep, as this button is used in clearing persisted context data.*

11.2.2 Colour Dimmer Switch or Remote Control

Once the Remote Control Unit (Colour Dimmer Switch or Remote Control device) is in the network, it remains awake for 20 seconds of inactivity and then starts a 1-second sleep/wake cycle in which it is asleep (warm sleep) for 800 ms and is awake for 200 ms. During the wake period, the device scans the keypad for user input and also sends a poll request to its parent to detect parent loss.

A 'deep sleep' mode is also available, but is not enabled by default. If used, a deep sleep timer is started on first entering warm sleep. This timer is reset when there is user activity. If the timer expires (due to no user activity), the device enters deep sleep mode.

11.2.3 Light Sensor

The Light Sensor goes through a 12-second sleep/wake cycle in which it is asleep (warm sleep) for 6 seconds and awake for 6 seconds. During the wake periods, the device listens for its parent.

11.2.4 Occupancy Sensor

The Occupancy Sensor goes through a sleep/wake cycle of 1 minute, which is equal to the maximum attribute reporting interval (HA_SYSTEM_MAX_REPORT_INTERVAL). The sensor wakes:

- Every minute to send an attribute report and poll its parent to check for pending data
- As the result of a button-press to simulate the motion detection pulses

After dealing with the above processing, the device returns to warm sleep.

In every sleep/wake cycle, the device spends approximately 57 seconds asleep and 3 seconds awake.

11.3 Dimmable Light with iControl Support

A special version of the Dimmable Light device application is provided to support the requirements for Icontrol certification. Icontrol provides a framework for consistent operational behavior between devices from different manufacturers (and is in addition to ZigBee certification). The Dimmable Light device with Icontrol support includes the Simple Metering cluster, which is not included in the standard Dimmable Light device. Otherwise, the functionality is the same for the two versions of the device.

The application binary file for the Icontrol version of the Dimmable Light device is:

DimmableLightOpenHome_JN5169_DR1175.bin

Therefore, a pre-built application is available for JN5169 only.

To rebuild this application, the **zcl_options.h** file must include the following line to enable the Simple Metering cluster:

```
#define CLD_SIMPLE_METERING
```

Note that Icontrol requires that the Dimmable Light restores its last previous light level following a power-cycle. This feature is enabled by default in the device application.

12 Release Details

12.1 New Features

ID	Feature	Description
Version 1.12		
lpsw6960	OTA Image Stamp Support	Added app support for OTA client image stamp
lpsw6958	Support for Simple Metering Server Cluster	Added app support for Simple Metering server cluster update
lpsw6954	Reset of Basic cluster support added	Add app support for Basic Cluster 'Reset Defaults' command
lpsw6951	Diagnostic cluster added	Diagnostic Cluster server update
Version 1.11		
lpsw7071	Dimmable Light for Icontrol	Dimmable Light device suitable for Icontrol certification
lpsw7073	Smart Plug	Smart Plug (Mains Power Outlet) device
lpsw7536	ZCL Factory New	Support for basic reset to ZCL factory new
Version 1.10		
N/A	OTA download of same image	Same version of an OTA image can now be downloaded from the OTA server to a client.

12.2 Known Issues

ID	Severity	Description
Version 1.12		
Version 1.11		
Version 1.10		
lpap550	Low	The light flashes momentarily when the following sequence of operations is performed: dim the light down, then switch off the light and then send an identify command to the light - when the light switches back on, it flashes briefly.
-	Medium	If a Light Sensor is re-allocated a different network address after being commissioned to a Remote Control, commissioning must be restarted as described in Section 6.2.3.

12.3 Bug Fixes

ID	Description
Version 1.12	
Version 1.11	
lpsw5462	The previous dim level of a bulb was not restored following a power cycle. The level attribute is now preserved across resets.
lpsw5585	When an OTA image is created using the appropriate build target, it contains 4 extra bytes at the start of the file - these are a version number used by the Flash programmer. However, these bytes cause the NTS test harness to reject the image when certifying the OTA cluster.
lpsw5595	Need to add a build option that allows a developer to easily change the number of power cycles required to put a lamp into the 'factory new' state, to clear PDM and to identify itself. This option should allow an (or all) of these to be disabled by setting the number of power cycles to zero.
lpsw5642	When the value of CLD_LEVELCONTROL_MAX_LEVEL is changed in the zcl_options.h file, this value is not applied when the lamp is power cycled. The value applied following a reset is hardcoded to 0xFE, but it should actually use the CLD_LEVELCONTROL_MAX_LEVEL value.
lpsw6961	The OTA client attribute Manufacturer ID is not initialized until an OTA is performed. Before that it reads back as 0.
lpsw7320	OTA needs to take into account the remapping of Flash by the bootloader.
lpsw7445	Touchlink handling of scan response fails if number of endpoints is not 1.
lpsw7526	Header file app_exceptions.h contained wrong copperplate. This has been corrected.
lpsw7527	Include the Mains Power Outlet example.
lpsw7528	The ZigBee OTA cluster specification (Revision 23, version 1.1, document number 095264r23, section 6.10.5.2.4) states that the version value may be the same as the client.
lpsw7529	OTAColorDimmableLightWithOccupancy wrong Device ID setting.
Version 1.10	
lpap558	Two versions of OTA binaries provided - OTA version and bin version.
lpap578	bAHI_APRegulatorEnabled is checked before any ADC conversion.
lpap579	Colour control name is now defined correctly.
lpap583/ lpap606	Management, remove and APS leave functionality can be changed in the application.
lpap592	Same version of OTA image can now be downloaded.
lpap600	Sensor now sends report if the sensor re-joins via another device in the network.
lpap607	Missing LD files for JN5164 device now added.
lpap632	APS acks enabled by default.

Appendix A - Source File Descriptions

Automatically Generated Files

Each device has several files that are automatically generated at build-time from the JenOS and ZPS configurations. These files are not generally used by the developer but are located in the respective device's **\Source** folders, in case they are of interest.

Common Files

A number of common files are used across all device types and are located within the **\Common\Source** folder. The following table gives a brief description of each of the common files.

Filename	Description
app_common.h	Contains the common macro definitions used in the project. It contains the conditional inclusion of the appropriate device headers.
app_events.h	Contains global definitions for the events in the HA application.
os_msg_types.h	Contains all the include files required.
PDM_ID.h	Contains the PDM_IDs that are used in PDM load and restore.
app.zpscfg and app_GP.zpscfg	This is the ZPS configuration file, which is used to configure generic network and node parameters. This includes profile, cluster, endpoint and RF channel configurations. For more information, refer to the "ZPS Configuration Editor" chapter of the *ZigBee PRO Stack User Guide (JN-UG-3101)*. There is a separate ZPS configuration supporting ZigBee Green Power for lights and switches – the filename is app_GP. A Green Power build of an application must include this configuration.
haEzJoin.c/.h	These are the EZ-mode Commissioning files. For details of EZ-mode Commissioning, refer to the relevant chapter of the *ZigBee Cluster Library User Guide (JN-UG-3103).*
haEzFindAndBind.c/.h	These are commissioning files for binding a cluster based on the Simple Descriptor request/response.
app_zbp_utilities.c/.h	Contain the debug functions for ZigBee PRO function calls in the application.
app_buttons.c/.h	Contain button sampling/de-bouncing, interrupt routines and button task to post messages under application events.
AgeChildren.c/.h	Implement the child table. The ageing is done at the start-up of a Router node in order to clean up the child table in case a child has moved to a different parent while the node was off.
haKeys.c/.h	Contain the pre-defined link keys for the HA profile
PingParent.c/.h	Implement a ping mechanism for querying the network address of the parent from an End Device.
app_exceptions.c/.h	Contain the exception routines.
App_GreenPower.c/.h	Contain the ZigBee Green Power related functionality to be included for a node which supports GP.
App_pdm.c/.h	Contain PDM functions, including the callback function for error debug.
app_ota_client.c/h	Contain application routines for OTA initialisation, OTA server discovery and application retries.
app_scenes.c/h	Contain scenes store, load and management routines.

Dimmable Light Source Files

The following table gives a brief description of the files used by the Dimmable Light device, which are located in the **\Common_Light\Source** folder.

Filename	Description
app_manage_temperature.c/.h	Manages the radio recalibration temperature corrections.
app_start_light.c	Start-up module with **vAppMain** and sleep/wake-up callback functions. All the initialisation for start-up and wake-up is available in this module.
app_light_effect.c/.h	Implement the breathe effect and other identify effects.
app_zcl_light_task.c/h	ZCL event handler for the node - the endpoint callback function is part of this module. This also contains the node task and functional state machine.
zha_light_node.c/h	Contains HA application command send/receive functions. It also handles the initialisation of HA states by calling appropriate PDM store and retrieve descriptors.
App_ZHA_Light_JN516x_mono.oscfgdiag	This is the JenOS configuration diagram file, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*. This is used for the lights without “SYNC” in the build target, such as a DR1174 Carrier Board with DR1175 Lighting/Sensor Expansion Board.
App_ZHA_Light_JN516x_rgb.oscfgdiag	This is the JenOS configuration diagram file, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*. This is used for the lights with “SYNC” in the build target, such as an SSL SYNC bulb.
app_power_on_counter.c/h	Application power ON counter management to change the state to identify/factory reset or a full reset
app_reporting.c/h	Default reporting and loading values upon power on.

Dimmer Switch Source Files

The following table gives a brief description of the files used by the Dimmer Switch device, which are located in the **\Common_Switch\Source** folder.

Filename	Description
app_start_switch.c	Start-up module with **vAppMain** and sleep/wake-up callback functions. All the initialisation for start-up and wake-up is available in this module.
app_switch_state_machine.c/.h	Contain logic for button-presses and resulting actions/commands
app_zcl_switch_task.c/h	ZCL event handler for the node - the endpoint callback function is part of this module. This also has the node task and functional state machine.
zha_switch_node.c/h	Contain the HA application command send/receive functions for the switch. It also handles the initialisation of HA states by calling the appropriate PDM store and retrieve descriptors.
App_ZHA_Controller_JN516x.oscfgdiag	This is the JenOS configuration diagram file, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

Colour Dimmer Switch Source Files

The following table gives a brief description of the files used by all the Colour Dimmer Switch device, and are located in the **\Common_Controller\Source** folder

Filename	Description
app_captouch_buttons.c/h	Capacitive-touch remote button task and event handler. It also defines the DIO ISR that is used for capacitive touch.
app_led_control	Defines LED initialisation and how to set/reset LEDs. It also defines the callback function for LED flashes.
app_start_remote.c	Start-up module with **vAppMain** and sleep/wake-up callbacks. All the initialisation for start-up and wake-up is available in this module.
app_zha_remote_task.c/h	ZCL event handler for the node - the endpoint callback function is part of this module. This also contains the node task and functional state machine.
DriverCapTouch.c/h	Capacitive-touch driver module.
zha_remote_node.c/h	Contains the HA application command send/receive functions and key map for the Remote Control Unit. It also handles the initialisation of HA states by calling the appropriate PDM store and retrieve descriptors.
App_ZHA_Controller_JN516x.oscfgdiag	This is the JenOS configuration diagram, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

Light Sensor Source Files

The following table gives a brief description of the files used by the Light Sensor device, which are located in the **\LightSensor\Source** folder.

Filename	Description
app_start_sensor.c	Start-up module with **vAppMain** and sleep/wake-up callback functions. All the initialisation for start-up and wake-up is available in this module.
app_sensor_state_machine.c/.h	Handles the ZigBee PRO stack events for both running and start-up states.
app_zcl_sensor_task.c/h	ZCL event handler for the node - the endpoint callback function is part of this module. This also contains the node task and ZCL tick task.
zha_sensor_node.c/h	Handles the initialisation of HA states by calling the appropriate PDM store and retrieve descriptors. It processes all queues and sends them to the relevant module.
app_blink_led.c/h	Handles the LED blink rate based on the API functions called.
app_event_handler.c/h	Handles all application events passed via the queue. This includes DIO changes, report sending and wake timer expiry. Contains 'find and bind' and polling keep-alive functionality.
app_nwk_event_handler.c/h	Handles all network-related stack events, including network join, leave and poll request/response.
app_reporting.c/h	Creates and restores saved reports, and defines which attributes are reportable.
app_sleep_handler.c/h	Determines at which point the device will go to sleep.
app_zcl_tick_handler.c/h	Holds the number of ticks remaining until a periodic report will be sent out. This needs to be updated when the device comes out of a timed sleep.
app_power_on_counter.c/h	Keeps track of the number of power-cycles that have occurred. When a certain number of power-cycles have occurred, user-defined functionality is executed.
app_light_sensor_buttons.c/h	Implements a Light Sensor-specific de-bounce algorithm and ISR handler.
App_ZHA_LightSensor_JN516x.oscfgdiag	This is the JenOS configuration diagram file, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

Occupancy Sensor Source Files

The following table gives a brief description of the files used by the Occupancy Sensor device, which are located in the **\OccupancySensor\Source** folder.

Filename	Description
app_start_sensor.c	Start-up module with **vAppMain** and sleep/wake-up callback functions. All the initialisation for start-up and wake-up is available in this module.
app_sensor_state_machine.c/.h	Handles the ZigBee PRO stack events for both running and start-up states.
app_zcl_sensor_task.c/h	ZCL event handler for the node - the endpoint callback function is part of this module. This also contains the node task and ZCL tick task.
zha_sensor_node.c/h	Handles the initialisation of HA states by calling the appropriate PDM store and retrieve descriptors. It processes all queues and sends them to the relevant module.
PIR/app_PIR_events.h	Uses an interface pattern which declares the functions that the PIR drivers should implement. It also defines the different timer events within the driver.
PIR/app_PIR_OpenCollector_events.c	Defines the interface pattern declared in **app_PIR_events.h**. This is a single on/off PIR driver.
PIR/app_PIR_pwm_events.c	Defines the interface pattern declared in **app_PIR_events.h**. This reacts to an occupied event based on a number of rising and falling edges (APP_OCCUPANCY_SENSOR_TRIGGER_THRESHOLD), defined in the file **App_OccupancySensor.h,** within a time-frame (APP_OCCUPANCY_SENSOR_UNOCCUPIED_TO_OCCUPIED_DELAY)
app_blink_led.c/h	Handles the LED blink rate based on the API functions called.
app_event_handler.c/h	Handles all application events passed via the queue. This includes DIO changes, report sending and wake timer expiry. Contains 'find and bind' and polling keep-alive functionality.
app_nwk_event_handler.c/h	Handles all network-related stack events, including network join, leave and poll request/response.
app_reporting.c/h	Creates and restores saved reports, and defines which attributes are reportable.
app_sleep_handler.c/h	Determines at which point the device will go to sleep.
app_zcl_tick_handler.c/h	Holds the number of ticks remaining until a periodic report will be sent out. This needs to be updated when the device comes out of a timed sleep.
app_occupancy_buttons.c/h	Implements an Occupancy Sensor-specific de-bounce algorithm and ISR handler.
App_ZHA_OccupancySensor_JN516x.oscfgdiag	This is the JenOS configuration diagram file, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

GP Switch Source Files

The following table gives a brief description of the files used by the GP Switch, which are located in the **\EH_Switch\Source** folder.

Filename	Description
AHI_EEPROM.c/.h	Contain EEPROM access functions for persistent data storage.
EH_Button.c/.h	Contain functionality for button interrupt handling and invocation of mapped commands.
EH_IEEE_802154_Switch.c/.h	Provide functionality for the GP Switch including start-up, the switch state machine and IEEE 802.15.4 MAC functionality.
EH_IEEE_Commands.c/.h	Contain the GP Switch command encoding and decoding functionality.
EH_IEEE_Features.c/.h	Contain functionality for persistent data storage and state machine handling of the switch.
EH_Timer.c/.h	Contain timer functionality.
EH_Switch_Configurations.h	Contains the configuration settings for the GP Switch. The configuration settings are detailed in Appendix B.

Device-specific Source Files

The following table gives a brief description of the device-specific source files, located in the **\<DeviceType>\Source** folder.

Filename	Description
App_<DeviceType>.c/h	Device-specific module that has the device definition and registration functions for the device type (e.g. Dimmer Switch, Dimmable Light). It also handles device-specific initialisation.

OTA Server Source Files

The following table gives a brief description of the files used by the OTA Server device, which are located in the **\OTAServer\Source** folder.

Filename	Description
App_endpoint.c	Contains a template endpoint (number 1).
app_start_upgrade_server.c	Start-up module with **vAppMain** and sleep/wake-up callback functions. All the initialisation for start-up and wake-up is available in this module.
app_exception.c/.h	Implements all the exception handlers.
app_zcl_server_node_task.c/h	ZCL event handler for the node - the endpoint callback function is part of this module. This also contains the node task and functional state machine.
zha_upgrade_server_node.c/h	Contains HA application command send/receive functions. It also handles the initialisation of HA states by calling appropriate PDM store and retrieve descriptors.
App_OTAServer_JN516x.oscfgdiag	This is the JenOS configuration diagram file, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.
App_ota_server.c/h	Application routines for OTA initialisation.
App_pdm.c/h	Application state and debug routines for PDM.

Appendix B – Preprocessing Macro Descriptions

Compile-time macros to manipulate ZigBee Cluster Library functionality are defined in the **zcl_options.h** file for the respective device. Other than these ZCL-specific macros, the demonstration application uses the following macros for ease of development and testing.

Macro	Description
HALT_ON_EXCEPTION	Stops execution in the case of an exception. Otherwise, allows the application to continue after a reset following an exception.
POLL_TIME	1-second standard poll-time
POLL_TIME_FAST	100-ms fast poll-time
TEN_HZ_TICK_TIME	100-ms timer to provide tick for HA clusters
NUMBER_DEVICE_TO_BE_DISCOVERED	The higher this value, the more service discoveries are triggered
OS_STRICT_CHECKS	OS strict check for task handlers
SLEEP_ENABLE	Enables sleep for a switch node
BUTTON_MAP_DR1175	Button mapping for DR1175 (Lighting/Sensor Expansion Board)
SSL_2108	Build code for different target - SSL_2108 for an NXP LED lamp
DEEP_SLEEP_ENABLE	Enables deep sleep for a switch node
KEEP_ALIVETIME	Time, in seconds, before the switch node goes to sleep
DEEP_SLEEP_TIME	Counter for the number of sleeps after which the switch node will enter deep sleep
APP_LONG_SLEEP_DURATION_IN_SEC	Time, in seconds, for which the switch node will be in the sleep state before it starts polling
MAX_REJOIN_TIME	Time, in seconds, before a node will try to join again after an unsuccessful join attempt (when no deep sleep enabled)
BACK_OFF_TIME	Time, in seconds, for which a node will back off before it attempts a joining (when no deep sleep enabled)
OTA_MAX_BLOCK_SIZE	Maximum image block size, in bytes, for OTA transfer. Should be set to 48 for JN5169 (and this value will also work for JN5168)
OTA_QUERY_TIME_IN_SEC	The time interval, in seconds, for which the OTA client will try to discover the OTA server
OTA_DISCOVERY_TIMEOUT_IN_SEC	Timeout, in seconds, before the OTA client declares the timeout without any servers discovered
OTA_DL_IN_PROGRESS_TIME_IN_SEC	Timeout, in seconds, for detecting the download has been stopped
MAX_SERVER_EPs	Maximum number of OTA server instances on a node
MAX_SERVER_NODES	Maximum number of OTA server nodes in the network
BREATH_EFFECT	Adds “Breathe” effect to the lights during network discovery
EZ_MODE_TIME	EZ-mode Commissioning time, in minutes. Should be fixed at 3.
EZ_RESPONSE_TIME	Time, in seconds, between requests to allow any response to be serviced. Defined as 10.
EZ_MAX_TARGET_DEVICES	Maximum number of nodes in the network that can be discovered per single query. Set to 10.
EZ_NUMBER_OF_ENDPOINTS	Number of endpoints on a node that can invoke EZ-mode Commissioning. This is same as HA_NUMBER_OF_ENDPOINTS because each HA endpoint on a device should also be able to perform commissioning.
EZ_MAX_CLUSTER_EXCLUSION_SIZE	The maximum number of clusters that need to be excluded during a ‘Find and Bind’ operation. Based on the ‘use case’ scenario, a user may want to change the cluster IDs so that the ‘Find and Bind’ or Grouping will not form a group or binding for this cluster.
EZ_MODE_TARGET	Defines the EZ-mode Commissioning target.
EZ_MODE_INITIATOR	Defines the EZ-mode Commissioning initiator.
COLOUR_TEMP_CHANGE_STEPS_PER_SEC	Rate of movement in Move Colour Temperature command in Colour Dimmer Switch.
MOVE_COLOUR_TEMPERATURE_MAX	Maximum colour temperature in Move Colour Temperature command in Colour Dimmer Switch or Remote Control.

MOVE_COLOUR_TEMPERATURE_MIN	Minimum colour temperature in Move Colour Temperature command in Colour Dimmer Switch or Remote Control.
MOVE_TO_COLOUR_TEMP_IN_ TRANSITION_TIME	Transition time for Move to Colour Temperature command in Colour Dimmer Switch or Remote Control.
LEVEL_CHANGE_STEPS_PER_SEC_SLOW	Rate of movement in Move Level command in Colour Dimmer Switch in Shift2 mode.
LEVEL_CHANGE_STEPS_PER_SEC_MED	Rate of movement in Move Level command in Colour Dimmer Switch or Remote Control in Shift1 mode.
LEVEL_CHANGE_STEPS_PER_SEC_FAST	Rate of movement in Move Level command in Colour Dimmer Switch or Remote Control in Shift0 mode.
HUE_CHANGE_STEPS_PER_SEC	Rate of movement in Move Hue command in Colour Dimmer Switch or Remote Control.
SATURATION_CHANGE_STEPS_PER_SEC	Rate of movement in Move Saturation command in Colour Dimmer Switch or Remote Control.
LIGHT_SENSOR_MINIMUM_MEASURED_ VALUE	Default value for Illuminance Measurement cluster attribute u16MinMeasuredValue in Light Sensor.
LIGHT_SENSOR_MAXIMUM_MEASURED_ VALUE	Default value for Illuminance Measurement cluster attribute u16MaxMeasuredValue in Light Sensor.
LIGHT_SENSOR_MINIMUM_ REPORTABLE_CHANGE	Minimum reportable change for which reports should be sent from Light Sensor.
LIGHT_SENSOR_SAMPLING_TIME_IN_ SECONDS	Rate at which samples are read from Light sensor driver.
LIGHT_SENSOR_NUMBER_OF_REPORTS	Number of attributes to be reported by Light Sensor and to be saved in EEPROM.
OCCUPANCY_SENSOR_UNOCCUPIED_TO_ OCCUPIED_DELAY	Default value for the Occupancy Sensing cluster attribute u8PIRUnoccupiedtoOccupiedDelay in the Occupancy Sensor.
OCCUPANCY_SENSOR_OCCUPIED_TO_ UNOCCUPIED_DELAY	Default value for the Occupancy Sensing cluster attribute u8PIROccupiedtoUnccupiedDelay in the Occupancy Sensor.
OCCUPANCY_SENSOR_UNOCCUPIED_TO_ OCCUPIED_THRESHOLD	Default value for the Occupancy Sensing cluster attribute u8PIRUnoccupiedtoOccupiedThreshold in the Occupancy Sensor.
OCCUPANCY_NUMBER_OF_REPORTS	Number of attributes to be reported by Occupancy Sensor and to be saved in EEPROM
ILLUMINANCE_MAXIMUM_LUX_LEVEL	Maximum level of Lux measured by Light Sensor for which light has to be completely dimmed off (defined at Dimmable Light with Illuminance Measurement cluster).
ILLUMINANCE_LUX_LEVEL_DIVISOR	The divisor used for converting Lux measurement to a value of Level Control cluster attribute u8CurrentLevel, as described in Section 5.4
POWER_ON_PRE_INIT_COUNTER_DB_ IN_MSEC	Time, in milliseconds, after which the Power-On Counter (POC) sequence will be validated on a Dimmable Light and the POC will be incremented and written to Non-Volatile Memory (NVM). The default value is 250 ms.
POWER_ON_COUNTER_DB_IN_MSEC	Time, in milliseconds, during which a Dimmable Light must be switched off after the POC is validated in order for the power-on sequence to be considered valid. On expiry of this time, the Power-On Counter (POC) will be reset to 0 and saved to NVM, then the EZ-mode Commissioning actions will be invoked. The default value is 1750 ms.
GP_SECURITY	This macro should be defined if GP security should be defined.
GPD_SEC_PRECONFIG_MODE	This macro should be defined if the light has a security key for the GP Switch. The key should be defined in GP_SHARED_KEY
GP_SECURITY_LEVEL	The GP security level to be used by the light node.
GP_SHARED_KEY	The GP shared key. The key should be defined using this macro when GPD_SEC_PRECONFIG_MODE is also defined
GP_DISABLE_SECURITY_FOR_ CERTIFICATION	This macro can be defined to disable security for non-secured joining and receiving /transmitting unsecured GP cluster packets. This can be used for GP certification.

The compile-time macros to manipulate the configuration settings on the GP Switch node are as follows:

Macro	Description
GPD_DEFAULT_CHANNEL	This macro should contain the default operating channel of device. This channel may be overridden during commissioning. This will be the default operating channel if channel request 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_SEND_DECOMM_CMD	To support the decommissioning command, this macro must be defined.
GPD_DEFAULT_PANID	This macro should contain the initial PAN ID of the device. This PAN ID may be overridden during commissioning.
GPD_MAX_PAYLOAD	The maximum payload size for the GP Switch.
DECOMMISSIONING_SHORT_PRESS	The number of short button-presses required to send a decommissioning command.
CLEAR_PERSISTENT_SHORT_PRESS	The number of short button-presses required to clear persistent data.
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_WITH_SECURITY	Enables GP security on switch.
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.

Appendix C - Build File Descriptions

Common Build Files

The following table gives a brief description of the build files, located in the **\Common_<Light>\Build** and **\Common_<Controller>\Build** folders.

Filename	Description
Makefile	This common Makefile is used to build the binary file for the specific HA device based on input
manu_config	This is the configuration file for the light devices located in the respective light device **Build** folders. The file contains configurations to control certain features in the light at build-level based on manufacturers' preferences.

Device-specific Linker Files

The following table gives a brief description of the device-specific linker files, located in the **\ <DeviceType>\Build** folders.

Filename	Description
APP_stack_size_JN5168.ld	Linker command file defining the default application stack size. Can adjust _stack_size for the desired stack size.

Appendix D – Dimmable Light Power-Cycle Operations

The Dimmable Light device allows certain operations to be performed by power-cycling the device a certain number of times – see Section 1.1.1. The diagram below illustrates the timings and events involved in these operations.

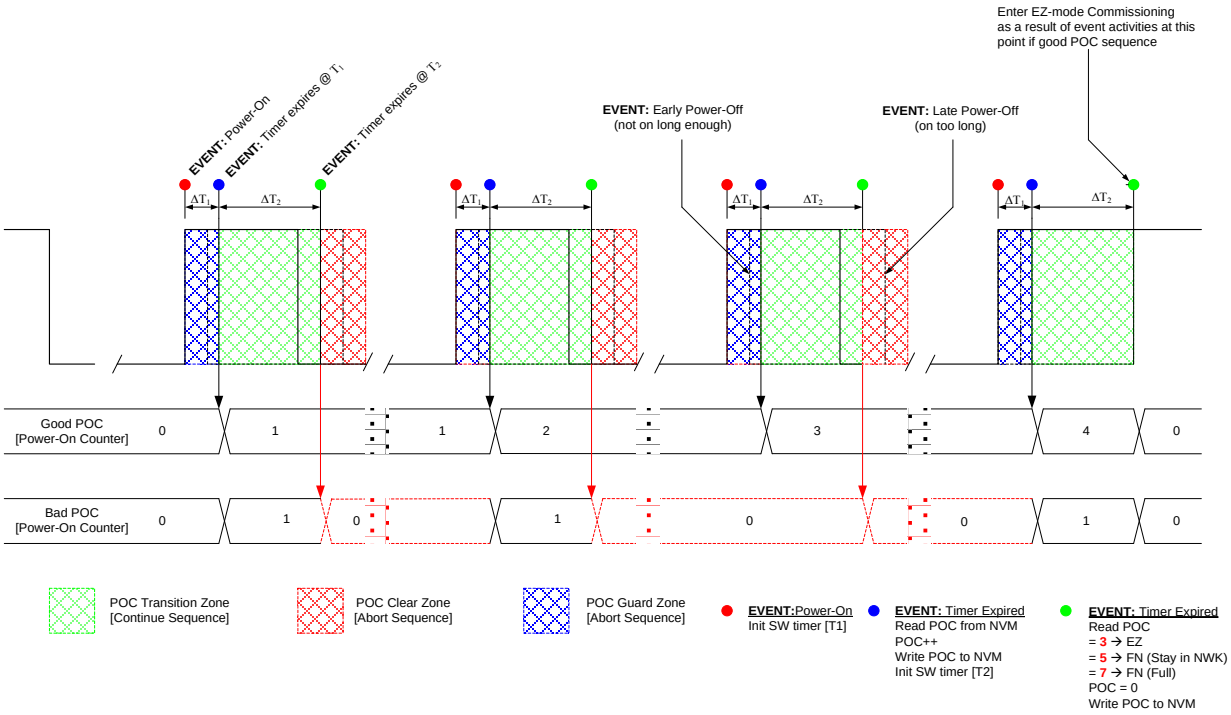


Figure 4: Dimmable Light Power-Cycling Operations

The values of ΔT_1 and ΔT_2 in the above diagram are defined by compile-time constants:

ΔT_1 = POWER_ON_PRE_INIT_COUNTER_DB_IN_MSEC

ΔT_2 = POWER_ON_COUNTER_DB_IN_MSEC

These values are described in the table in Appendix B and illustrated below.

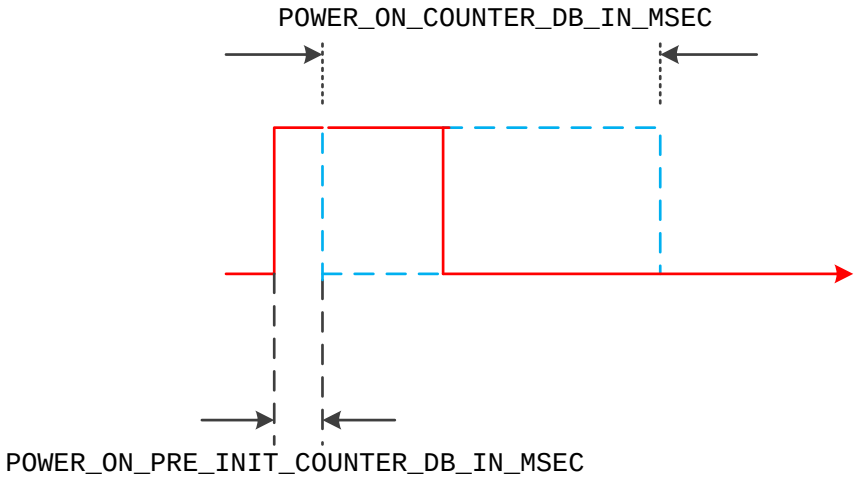


Figure 5: Dimmable Light Compile-time Constants

Appendix F - Application Code Size Statistics

The demonstration application of this Application Note has the following memory footprint on the JN5168 device, using the JN516x ZigBee Home Automation SDK (JN-SW-4168).

Components	Text Size (Bytes)	Data Size (Bytes)	BSS Size (Bytes)
ColorDimmableLight_JN5168_DR1175_LED_EXP_RGB.bin	154169	2336	21325
ColorDimmableLight_JN5168_DR1175_LED_EXP_RGB_GP.bin	187436	2712	27581
ColorDimmableLight_JN5168_DR1175_LED_EXP_RGB_OTA.bin	166749	2448	22193
ColorDimmableLight_JN5168_DR1173_LED_EXP_RGB.bin	154273	2336	21229
ColorDimmerSwitch_JN5168_DR1159.bin	111822	2056	24477
RemoteControl_JN5168_DR1159.bin	113658	2104	24725
ColorTempTunableWhiteLight_JN5168_DR1221_CCTW.bin	146666	2364	21417
ColorTempTunableWhiteLight_JN5168_DR1221_CCTW_GP.bin	180370	2748	27409
Coordinator_JN5168.bin	106165	1780	17549
DimmableLight_JN5168_DR1175_LED_EXP_MONO.bin	135503	1984	20261
DimmableLight_JN5168_DR1175_LED_EXP_MONO_GP.bin	168606	2360	26533
DimmableLight_JN5168_DR1175_LED_EXP_MONO_OTA.bin	148017	2100	21133
DimmableLight_JN5168_DR1190_MONO.bin	135231	1984	20245
DimmableLight_JN5168_DR1190_MONO_GP.bin	168314	2356	26521
DimmableLight_JN5168_DR1192_MONO.bin	136271	2016	20261
DimmableLight_JN5168_DR1192_MONO_GP.bin	169402	2388	26529
DimmerSwitch_JN5168_DR1199.bin	111914	2120	23621
DimmerSwitch_JN5168_DR1199_OTA.bin	124890	2248	24501
LightSensor_JN5168_DR1175.bin	114418	2016	20397
OccupancySensor_JN5168_DR1199.bin	119086	1912	20109
OTAServer_JN5168.bin	125626	1904	19813
EH_Switch_JN5168_DR1199.bin	4809	92	5887

Note: The above filenames are as produced by the compiler. Some files (e.g. OTA files) are then processed using the JN51xx Encryption Tool (JET) which extends the filenames (e.g. by adding a version number).

Revision History

Version	Notes
1.0	First release
1.1	Build warnings resolved, devices renamed, SSL2108/Green Power information added to text, ZigBee keys removed
1.2	ZPS Configuration Editor issue resolved in software
1.3	Channel masks updated to support preferred channels. Internal EEPROM usage optimised for scenes. Switch state-machine modified for commissioning.
1.4	Updated with the OTA Upgrade demonstration
1.5	Updated for the HA 1.2.1 release and Green Power demonstration included
1.6	Fixed recall scenes for driver DR1221
1.7	Updated for the ZigBee Home Automation Specification v1.2.2 and for the 'BeyondStudio for NXP' toolchain. Revised functionality for Occupancy Sensor and other minor updates also made.
1.8	Updated to support JN5169 device
1.9	Binaries rebuilt on JN-SW-4168 SDK v1279 for improved radio settings
1.10	Updated with the new features and bug fixes detailed in Section 12
1.11	Updated with the new features and bug fixes detailed in Section 12. Binaries rebuilt on JN-SW-4168 SDK v1461.
1.12	Updated with the new features and bug fixes detailed in Section 12. Binaries rebuilt on JN-SW-4168 SDK v1470.

Important Notice

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

All trademarks are the property of their respective owners.

NXP Semiconductors

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

www.nxp.com