Application Note: JN-AN-1221

ZigBee HA Lighting with NFC

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. In conjunction with the Application Note *ZigBee IoT Gateway Host with NFC (JN-AN-1222)*, it demonstrates both the use of EZ-mode for in-band commissioning and use of Near Field Communication (NFC) for out-of-band commissioning of ZigBee end devices in a ZigBee HA 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-EK004 Evaluation Kit. The device software was developed using NXP Application Programming Interfaces (APIs).

For a broader range of features and more HA devices, please refer to the Application Note ZigBee Home Automation Demonstration (JN-AN-1189), available from the NXP Website.

1 Introduction

This Application Note provides an example ZigBee Home Automation (HA) wireless network solution which uses the NXP JN516x-EK004 Evaluation Kit. It employs HA Lighting devices, allowing the user to control lights from controller devices such as dimmer switches or via a gateway.

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 Generic Expansion Board (DR1199) is used as a Dimmer Switch
- One USB Dongle (DR1198) is used as the ZigBee Control Bridge
- One Raspberry Pi + USB Dongle (DR1198) + WiFi dongle + NFC expansion board is used as an NFC-enabled IoT gateway

The expansion boards are each fitted to a Carrier Board (DR1174_NFC). These carrier boards are equipped with a Type 2 NTAG I²C (NT3H1101) from NXP and includes a sticker NFC antenna. The evaluation kit and its components are described in the *JN516x-EK004 ZigBee Evaluation Kit User Guide (JN-UG-3108-JN516X-EK004)*.

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 6.1.

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 and Dimmer Switch. All the devices are equipped with NFC for out of band commissioning.

Default binaries provided makes all the devices ready for both NFC and EZ-mode commissioning. These two different methods are detailed in respective section 6 and 7.

Note 1: Once powered up, the devices will automatically search for an existing network via the EZ-mode process, unless you use NFC commissioning (see in Section 6.2)

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)*. The NFC commissioning mechanism and software is described in the following document, *NFC Commissioning User Guide (JN-UG-3112)*.

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.

The implemented Dimmable Light device includes the mandatory clusters defined for this device in the HA Specification.

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

- 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 5.5.1.

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 powered 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.

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.

The implemented Colour Dimmable Light device includes the mandatory clusters defined for this device in the HA Specification.

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, including commissioning, 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 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 1.1.

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

1.1.5 OTA Server

The OTA Server resides on a node (OM15020 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.

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-EK004	-
JN516x ZLL/HA SDK	JN-SW-4168	1279
'BeyondStudio for NXP' Toolchain	JN-SW-4141	1308

3 Additional Hardware Options

The supplied software can also be built for NXP's hardware Reference Design build type within the Eclipse development environment. There is a build configuration for application that run on the following NXP hardware supporting colour lights:

- HW Ref OM15008 – OM15050/OM15051 Colour Controlled Tuneable White Light

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

4 Loading the Application

Table 1 below lists the application binary files supplied with this Application Note and indicates the JN516x-EK004 Evaluation Kit components on which the binaries can be used. These files are located in the **Build** directories for the relevant devices.

The binaries are provided for JN5169, combining NFC and EZ mode commissioning. See Appendix B and `NFC_SUPPORT macro switch compiler` to disable NFC.

Application Binary	Generic	Lighting /Sensor	Raspberry Pi IoT gateway + USB Dongle
NFC-enabled IoT Gateway*			■
DimmableLight_JN5169_DR1175_LED_EXP_MONO_NFC.bin		■	
DimmableLight_JN5169_DR1175_LED_EXP_MONO_GP_NFC.bin ****		■	
ColorDimmableLight_JN5169_DR1175_LED_EXP_RGB__NFC.bin		■	
ColorDimmableLight_JN5169_DR1175_LED_EXP_RGB_GP_NFC.bin ****		■	
DimmerSwitch_JN5169_DR1199_NFC.bin	■		
OccupancySensor_JN5169_DR1199.bin	■		
EH_Switch_JN5169_DR1199.bin ****	■		
OTAserver_JN5169.bin			■

Table 1: Device Type – Evaluation Kits Compatibility Matrix

*For more information on the IoT gateway, please refer to the document JN-AN-1222 and JN-UG-3117

**These files are used for ZigBee Green Power (GP) – see Section 8.

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 JN516x Production Flash Programmer (JN-SW-4107). Note that the JN516x Production Flash Programmer must be used to load binaries.

Caution: If loading this demonstration software for the first time, the persistent data must be cleared in each device

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 9.3

Note: Occupancy Sensor, EH Switch and OTA server are not provided with NFC feature.

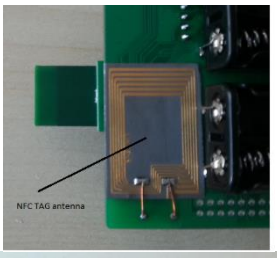
5 Device Functionality

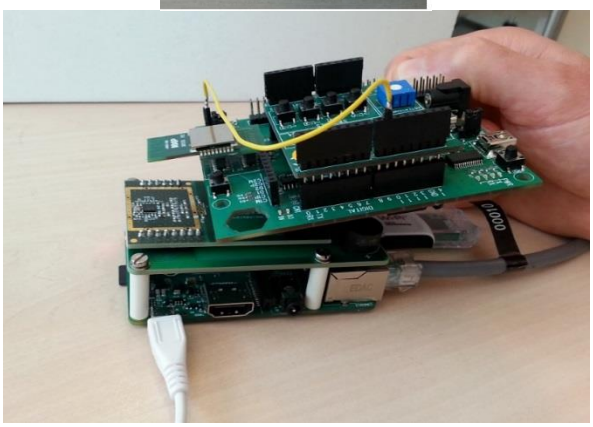
This section describes how to use the HA demonstration once the JN516x evaluation kit hardware has been programmed with the relevant binaries (as described in Section 4).

5.1 Dimmer Switch Functionality with NFC commissioning

The binary file **DimmerSwitch_JN5169_DR1199_NFC.bin** should be programmed into the JN5169 device associated with a DR1199 Generic Expansion Board to obtain the Dimmer Switch functionality with NFC. This switch allows the commissioning and control of lights.

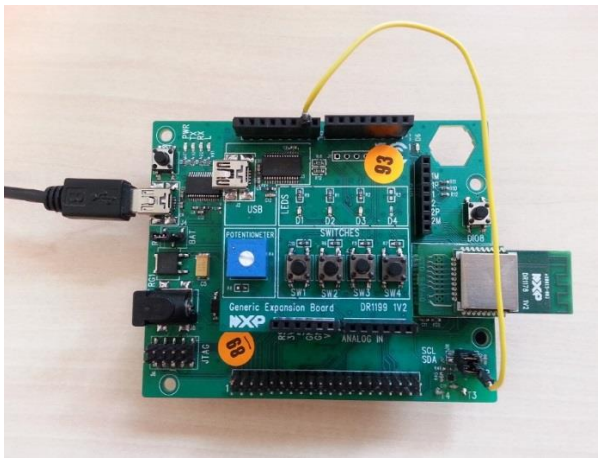
To perform the commissioning, the NFC antenna at the bottom side of the board should be put close to the top of the Gateway as shown below (see *JN-UG-3108 section 1.3.2* for detailed description of the gateway).





This action, so called “TAP”, can either be done when the:

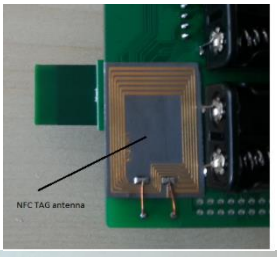
- Device is unpowered, the join of the device onto the network will then occur when power is applied.
- Device is powered, the join of the device onto the network will then occur immediately if the field detect pin is connected to DIO0.

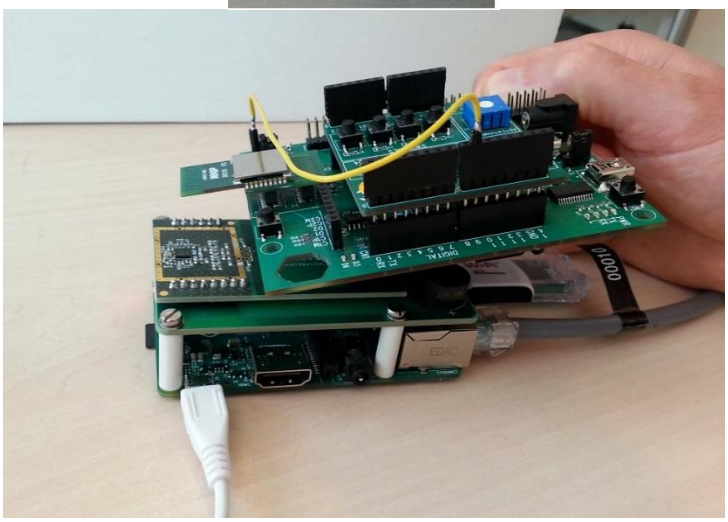


5.2 Dimmable Light Functionality with NFC commissioning

The binary file **DimmableLight_JN5169_DR1175_LED_EXP_MONO_NFC.bin** should be programmed into the JN5169 device associated with a DR1175 Lighting Expansion Board to obtain the Dimmable Light functionality with NFC.

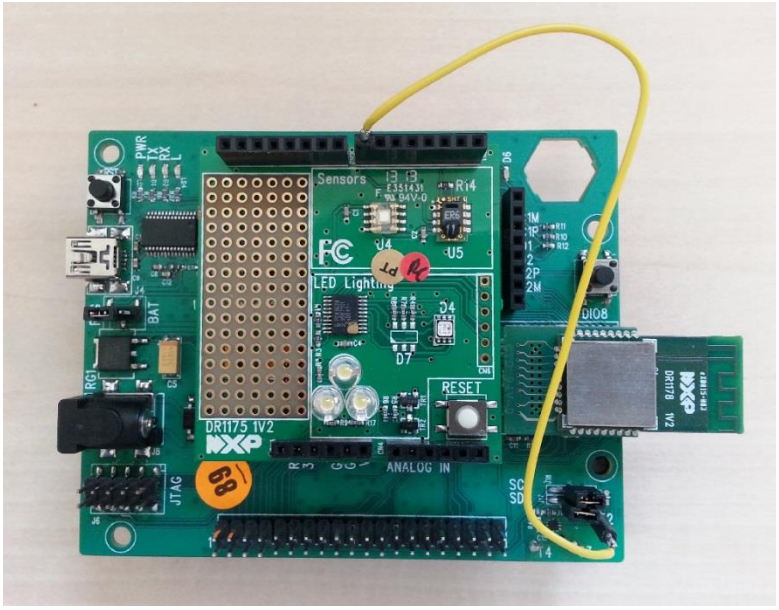
To perform the commissioning, the NFC antenna at the bottom side of the board should be put close to the top of the Gateway as shown below (see *JN-UG-3108 section 1.3.2* for detailed description of the gateway).





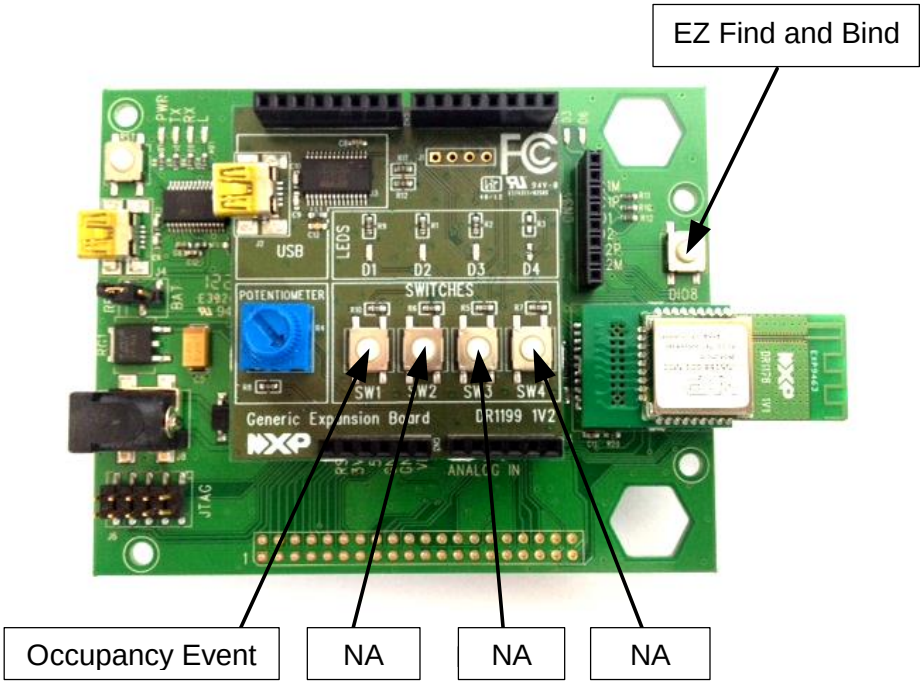
This action, so called “TAP”, can either be done when the:

- Device is unpowered, the join of the device onto the network will then occur when power is applied.
- Device is powered, the join of the device onto the network will then occur immediately if the field detect pin is connected to DIO17.



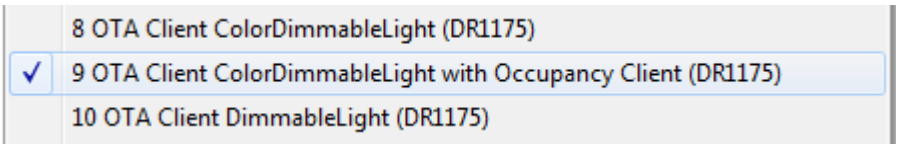
5.3 Occupancy Sensor Functionality

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 JN5169 device associated with the board must be programmed with the binary file **OccupancySensor_JN5169_DR1199.bin**. The Occupancy Sensor functionality is detailed below.



NA – Not allocated

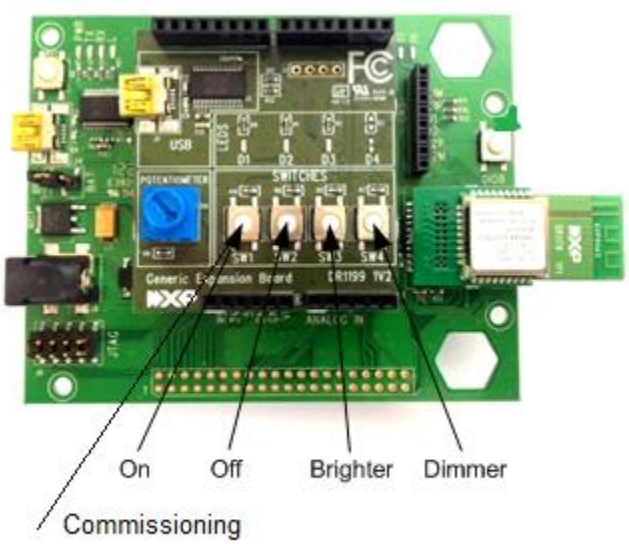
In order to use the above binary file for the Occupancy Sensor, you will need to pair this device with a light which contains the Occupancy Sensor cluster client. There is no pre-built binary file for this light but the following build configuration (9) is provided for such a light:



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 1). 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**.

5.4 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 JN5169 device associated with a Generic Expansion Board must be programmed with the binary file **EH_Switch_JN5169_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 8.2. After the GP Switch has been commissioned, it can be used to send commands to the light (On, Off, Brighter and Dimmer).

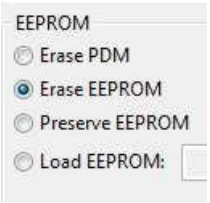
5.5 Clearing Context Data on the Devices

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

5.5.1 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 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).*



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.

5.5.2 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 Beyond Studio for NXP with the **Erase EEPROM** option, as described for the light nodes in Section 5.5.1.

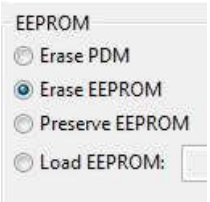
Alternatively, the context data can be cleared during normal operation of the device. This is done by pressing and releasing the '**RST**' button while holding down the '**DIO8**' button (both buttons are on the DR1174 Carrier Board).

5.5.3 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 5.5.2.

5.5.4 On OTA Server Node

The context data on OTA Server node 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).*



5.5.5 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 with NFC

This section describes how to create the demonstration network and commission light nodes using the NFC feature. For technical details on NFC Commissioning, please refer to the document *JN-UG-3112 NFC Commissioning User Guide*.

This demonstration application uses the following: NFC enabled RPi Gateway, ZigBee devices, hardware components and binary files.

- **Gateway:** The gateway includes the host application which runs on a Raspberry Pi. The PN7120 NFC reader (For more information on this part please refer to the product page). Details for the Gateway set up install is available in the document JN-AN-1222 ZigBee IoT gateway Host with NFC application.
- **Co-ordinator:** dongle plugged in USB port of the Gateway, acting as a ZigBee control bridge
- **Dimmable Light:** This is a DR1174_NFC Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board, programmed with the binary file **DimmableLight_JN5169_DR1175_LED_EXP_MONO_NFC.bin**
- **Dimmer Switch:** This is a DR1174_NFC Carrier Board fitted with a DR1199 Generic Expansion Board, programmed with the binary file **DimmerSwitch_JN5169_DR1199_NFC.bin**
- **Colour Dimmable Light:** This is a DR1174_NFC Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board, programmed with the binary file **ColorDimmableLight_JN5169_DR1175_LED_EXP_RGB_NFC.bin**

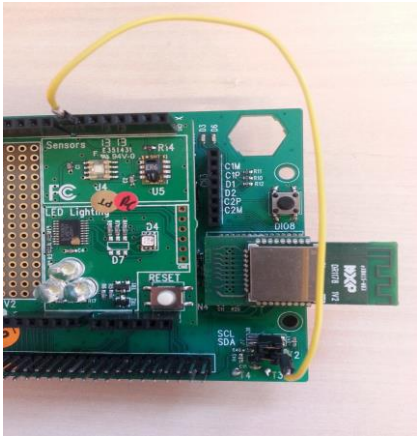
6.1 NFC application: Hardware pre-requisite

6.1.1 Field detect signal

The NTAG-I2C IC has an output pin called ‘Field Detect’ (FD) which is used as an IRQ to the wireless microcontroller. This ‘Field Detect’ signal is available via test pin T2, and must be hardwired to a DIO pin (and configured in the NFC demo firmware) of the JN516x wireless microcontroller as described in the table below.

Application	DIO	Connector	Pin
Dimmable Light	17	CN1	7
Dimmer Switch	0	CN2	9
Color Dimmable Light	17	CN1	7

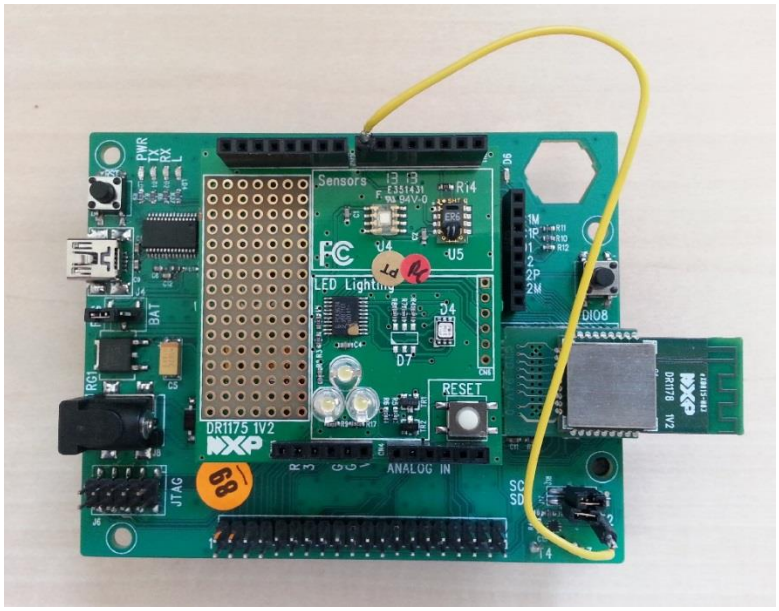
The figure below shows a picture where the ‘Field Detect’ signal is connected to DIO17 (Dimmable Light application).



6.1.2 DR1175 board – Dimmable Light / Color Dimmable Light

For the NFC demo a DR1175 board is used to show that the Dimmable Light or Color Dimmable Light NFC application, built around the JN516x wireless microcontroller, can be commissioned via NFC into a ZigBee Pro network, and that after commissioning the white LEDs D1, D2 and D3 can be controlled via the ZigBee network (for the Dimmable Light) or the RGB LED D4 (for the Color Dimmable Light).

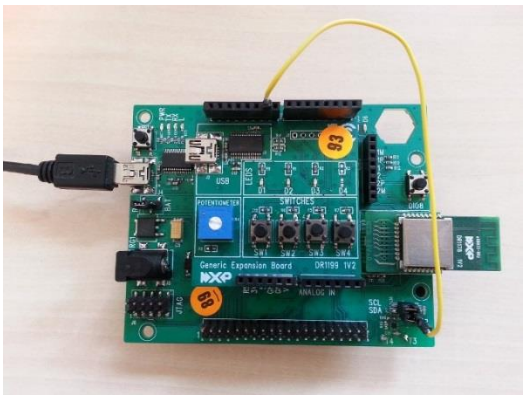
The figure below indicates how to place the LED board on top of the DR1174_NFC board.



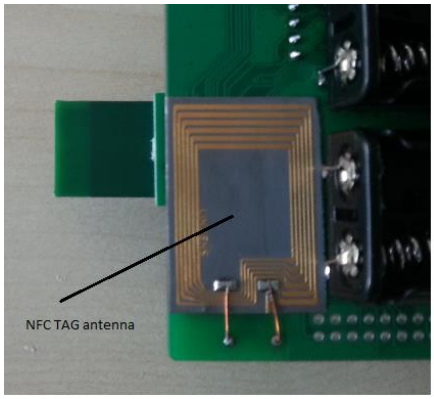
6.1.3 DR1199 board - Dimmer Switch

For the NFC demo the DR1199 board can be used to show that a Dimmer Switch NFC application, build around the JN516x wireless microcontroller, can be commissioned via NFC into a ZigBee Pro network, and that after commissioning the Dimmer Switch can be used to control the Dimmable Light (one or more) which is commissioned in the same ZigBee network.

The figure below describes how to connect the DR1199 on top of the DR1174 board, and how to connect the ‘Field Detect’ signal (pin T2) of the NTAG to the correct DIO pin of the JN516x wireless microcontroller, for the Dimmer Switch NFC application.



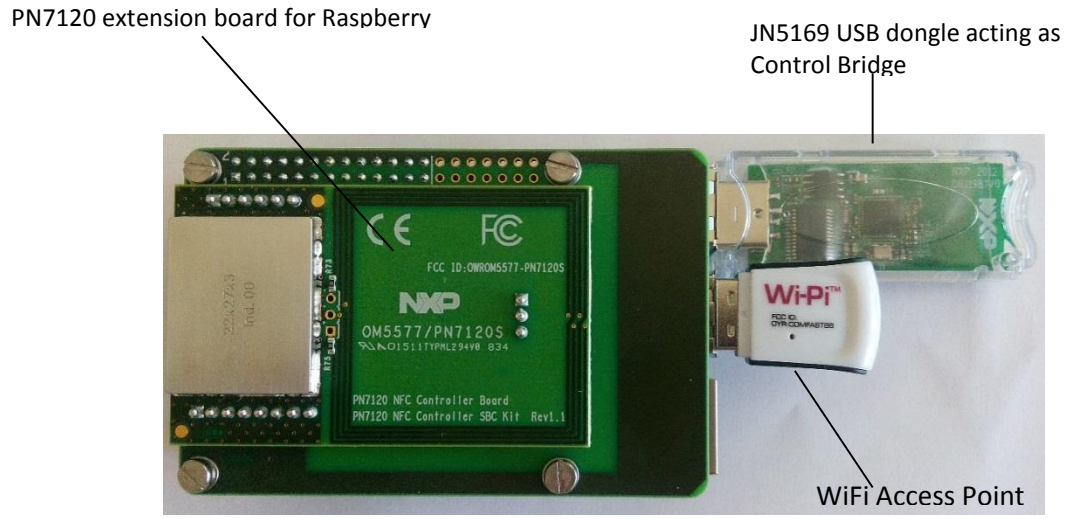
Moreover all the DR1174_NFC boards are equipped with an NTAG I²C and its sticker antenna located at the back as shown below:



6.1.4 Gateway

The gateway includes four components:

- Raspberry Pi Board with its SD card
- PN7120 NFC reader extension board
- JN5169 USB Dongle
- WiPi WiFi dongle



6.2 Forming the Network

The process described below uses NFC-mode Commissioning, which is fully detailed in the *JN-UG-3112-NFC-Commissioning-User-Guide* to form an HA network.

Plug the ZigBee Control Bridge (USB Dongle) and the WiFi dongle into the Raspberry Pi – as shown in the picture above.

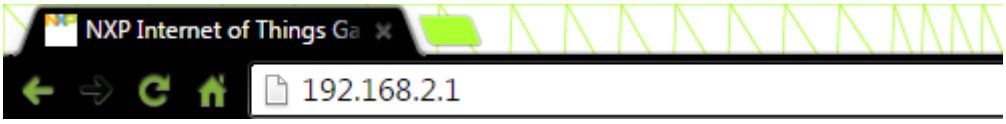
The Raspberry Pi board can be powered either using the USB port or through the 5V jack connector and the supply kit provided.

When the gateway is ready, it is signalled by a long “beep”. At this point the WiFi access point is available and can be accessed via the Wireless Network connection of a PC.

The SSID is IoT_GW_NXP

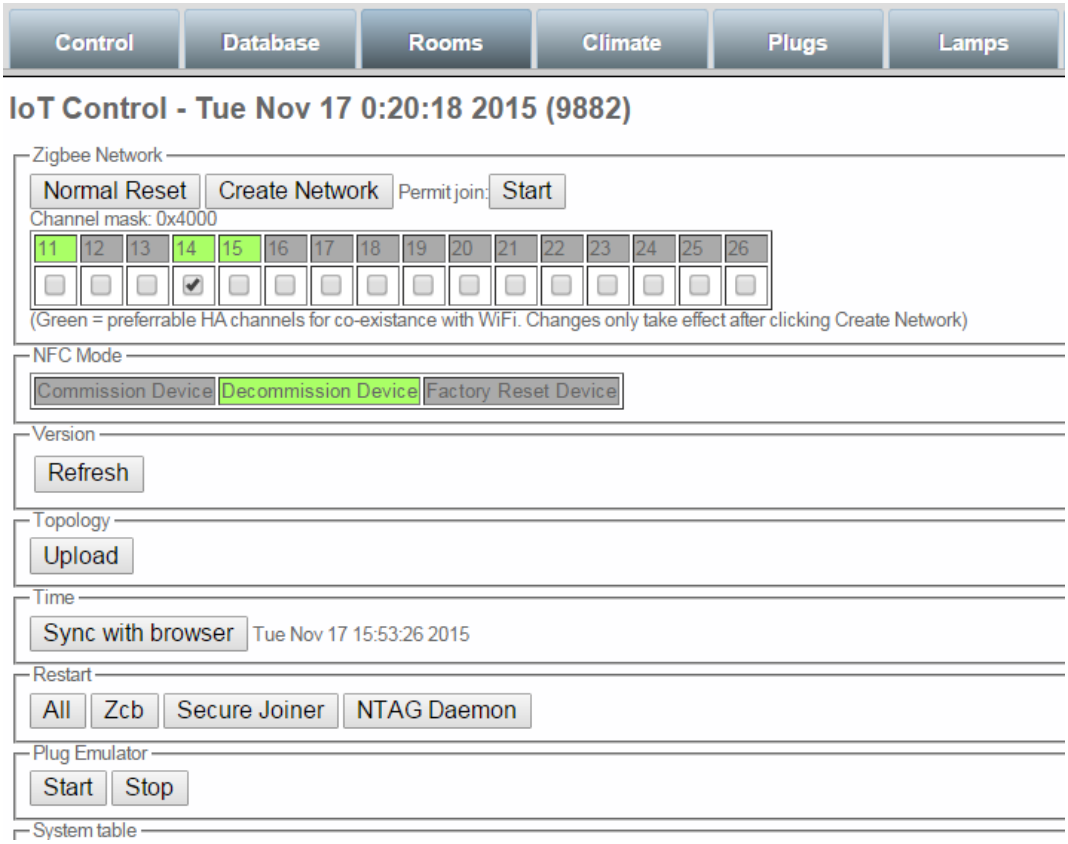
Then a browser can be opened and the gateway can be accessed at the following address:

192.168.2.1



The main window (Control) is showed in the picture below.

It includes buttons to do a Normal Reset and a Factory Reset. The factory reset will reset the gateway Host and the ZigBee Control Bridge. Note that when a factory reset is performed the Green and OrangeLEDs on the USB dongle will turn OFF during the reset and the LEDs turn back ON when the ZCB is initialized.



When using the NFC commissioning, the user must activate the “Commission Device” in the control panel:

When the gateway is in Commission mode, it is ready to accept End devices equipped with NFC Commissioning.

Note: Note that there is no need to start a permit join period (start button). This is only needed when using EZ-mode see in Section 7.1

6.2.1 Light Node NFC Commissioning

To perform the Light node commissioning, place the back of DR1174_NFC carrier board with NFC tag sticker antenna close to the Gateway reader antenna on top.

This “Tap the device” action can be done:

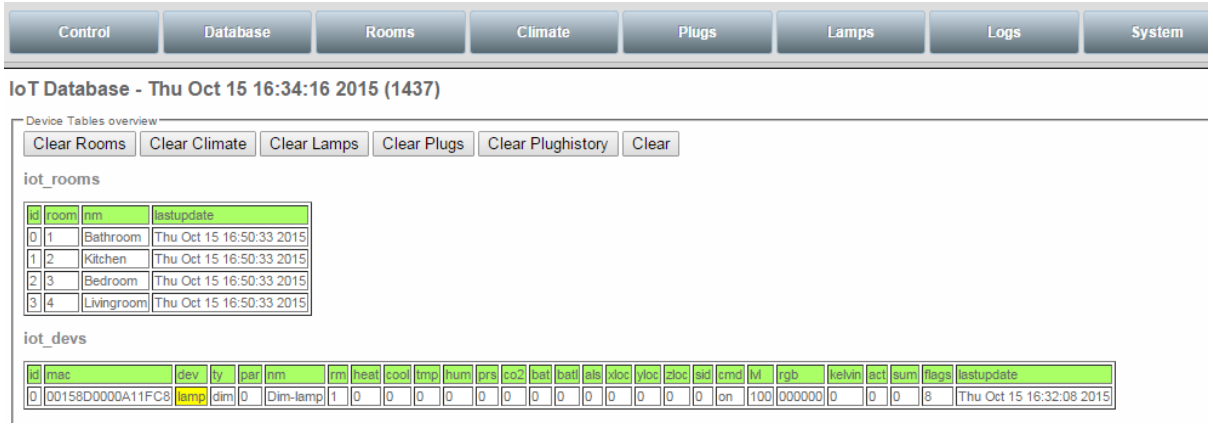
- With the light device powered off, the join of the device onto the network will then occur when power is applied.
- With the light device powered on, the join of the device onto the network will occur immediately as long as the “field detect” pin is connected.

Note: Note that this is the case for the Dimmable Light application. Without this connection, the user has to reset the device manually after the “TAP”.

The user feedback will be a single “Beep” from the gateway. Moreover the Green LED on the NFC reader board will light up. These information indicate that the commissioning has been successful.

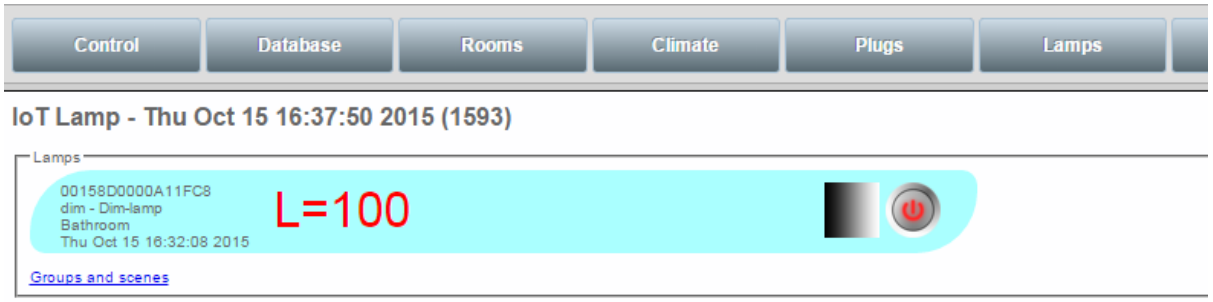
In case of failure, the red LED on the gateway will turn on and 4 “beeps” will be heard.

On the gateway interface, when looking into the “Database” tab, the lamp can be seen as being available.

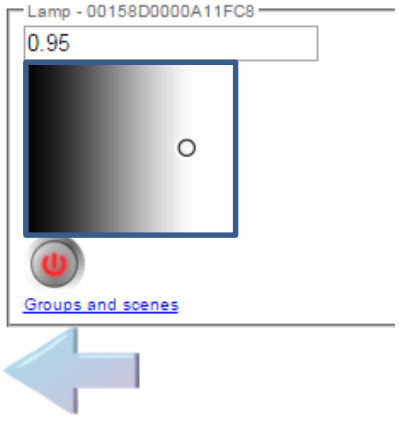


The gateway recognizes the type of bulb from data in the NTAG device.

Once the supply voltage is available, the light node can be controlled via the web interface. This is done by clicking on the “Lamps” tab.



For a dimmable light, the available functions are “On/Off” and Dim Up/Down. From thereon, the bulb can be turned “On” and “Off” using the indicated button. To control the intensity of the bulb, click on the grey scale, this will open the window below:



The intensity can be varied by dragging the circular cursor along the grey scale. The available light ratio is indicated in the text tab right above the grey scale.

6.2.2 Dimmer switch NFC Commissioning

To perform the Dimmer switch commissioning, place the back of DR1174_NFC carrier board with NFC tag sticker antenna close to the Gate way reader antenna on top.

This “Tap the device” action can be done:

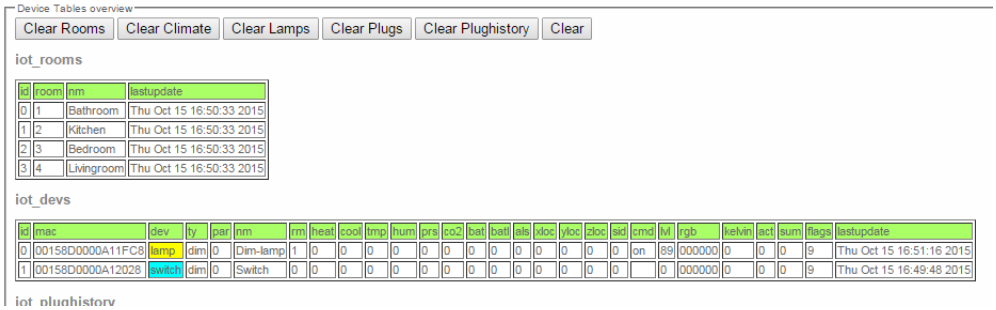
- With the light device powered off, the join of the device onto the network will then occur when power is applied.

- With the light device powered on, the join of the device onto the network will occur immediately as long as the “field detect” pin is connected.

The first user feedback will be a single “Beep” from the gateway. Moreover the Green LED on the NFC reader board will light up. These information indicate that the commissioning has been successful.

In case of failure, the red LED on the gateway will turn on and 4 “beeps” will be heard.

On the gateway interface, when looking into the “Database” tab, the switch can be seen as being available.



The switch can then be bound to the lamp as described in the next section.

6.2.3 Light Discovery from Switch

On Dimmer Switch node:

1. Enter identify mode by pressing and holding down the Commissioning button (DIO8) on the node (on the DR1174_NFC Carrier Board).
2. Identify a light.
3. 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).
4. If the light is initially off, it will switch on and identify itself
5. Press **SW1** to join the light to the switch

On Dimmer Switch node:

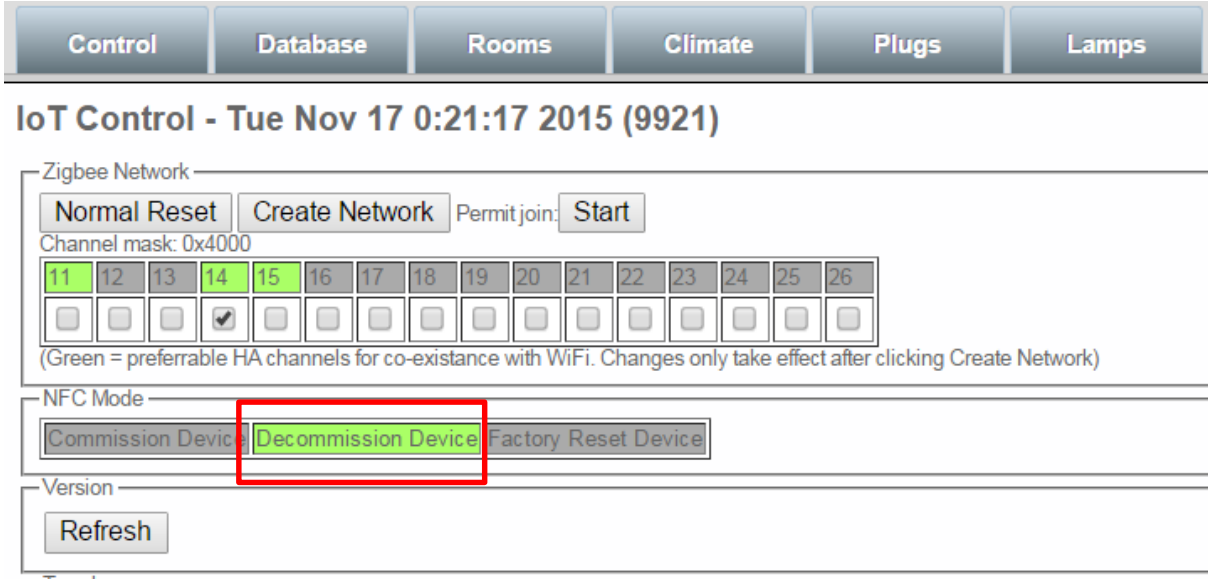
The Dimmer Switch now enters control mode and can be used to control the light as follows:

- 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

Note that when the light is controlled via the switch board, the intensity of the light is reported periodically to the gateway, and therefore the gateway interface is updated regularly to reflect the exact state of the light.

6.2.4 NFC De-commissioning a device

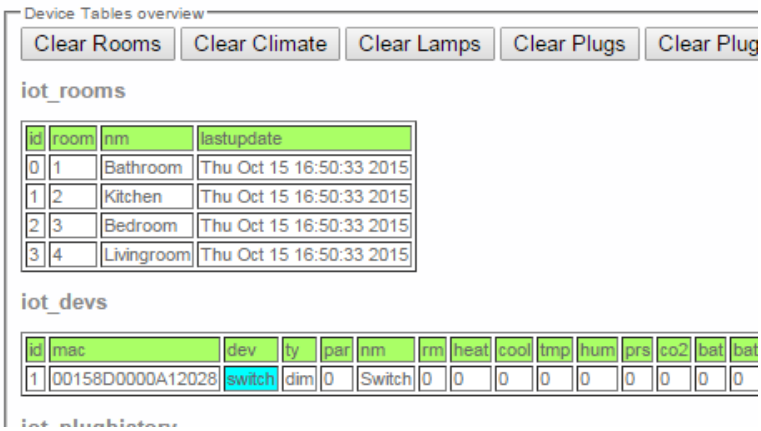
De-commissioning a device using NFC is as simple as the commissioning part. On the gateway interface, click on the decommission button:

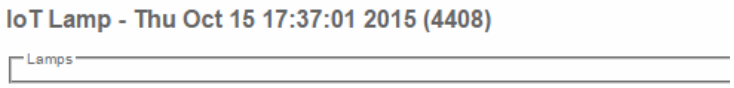


Then tap the device to be de-commissioned, on the gateway. For example tap the dimmable light on the gateway.

The user feedback from the gateway is a double “beep” and the green LED turning ON. This indicates that the de-commissioning is successful.

After this step, the light is not available anymore in the database and on the “lamp” tab the lamp device will disappear.





The same thing can then be done with the switch.

6.2.5 NFC Factory reset a device

Device factory reset using NFC is as simple as the commissioning part. On the gateway interface, click on the factory reset:

Then tap the device to be reset, on the gateway. For example tap the dimmable light on the gateway.

The user feedback from the gateway is a triple “beep” and the green LED turning ON. This indicates that the factory reset is successful.

After this step, the light is not available anymore in the database and on the “lamp” tab the lamp device will disappear. The device will then restart with context data in EEPROM cleared.

7 Setting up the network with EZ-mode

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 6.

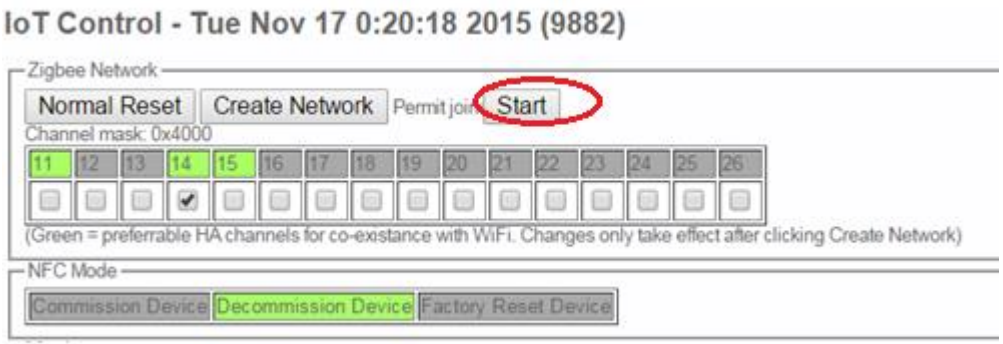
The network set-up procedure is presented in two stages, as follows:

- First the network is formed, in which the various nodes join the network – this is described in Section 7.1
- 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 7.2.1
    - The binding (Find and Bind) method is described in Section 7.2.2

7.1 Forming the network

The procedure below describes how to form the demonstration network using EZ-mode Commissioning. This network formation process is the same as NFC method described in 6.1.4, irrespective of whether 'Find and Bind' or Grouping will be later used to establish control links to the light nodes.

1. Once the gateway is up and the network created, a permit join period has to be started using the "Start" button as below :



2. 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 after 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 5.5.1.

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 top side of the 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.

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 at a 1Hz frequency.

   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.

8. Perform the rest of the network set-up to establish control links between the controller nodes and light nodes – refer to Section 7.2.

7.2 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 7.2.1) or binding (Section 7.2.2)
- **Occupancy Sensor:** Use binding (Section 7.2.2)

The commissioning of light nodes is performed per controller node.

The procedures assume that the network has been formed as described in Section 7.1 .

7.2.1 Commissioning Light Nodes - Groups and Scenes

In the following procedure, you will collect light nodes into a group to be controlled by the 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.

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

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

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.

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.

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

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

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

> **!** ***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.

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

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

7.2.2 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.

8 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:

- **Gateway:** The gateway includes the host application which runs on a Raspberry Pi. The PN7120 NFC reader (For more information on this part please refer to the product page). Details for the Gateway set up install is available in the document JN-AN-1222 ZigBee IoT gateway Host with NFC application.
- **Co-ordinator:** dongle plugged in USB port of the Gateway, acting as a ZigBee control bridge
  - **Dimmable Light:** This is a DR1174 Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board, programmed with the binary file **DimmableLight_JN5169_DR1175_LED_EXP_MONO_GP_NFC.bin**
  - **Colour Dimmable Light:** This is a DR1174 Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board, programmed with the binary file **ColorDimmableLight_JN5169_DR1175_LED_EXP_RGB_GP_NFC.bin**
  - **GP Switch:** This is a DR1174 Carrier Board fitted with a DR1199 Generic Expansion Board, programmed with the binary file **EH_Switch_JN5169_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.

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

- First the ZigBee PRO network is formed – this is described in Section 1.
- Then the GP Switch is commissioned – this is described in Section 1.

GP Switch decommissioning is also described in Section 1.

8.1 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 5.5.1.

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).

8.2 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 5.5.5) 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.

8.3 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.

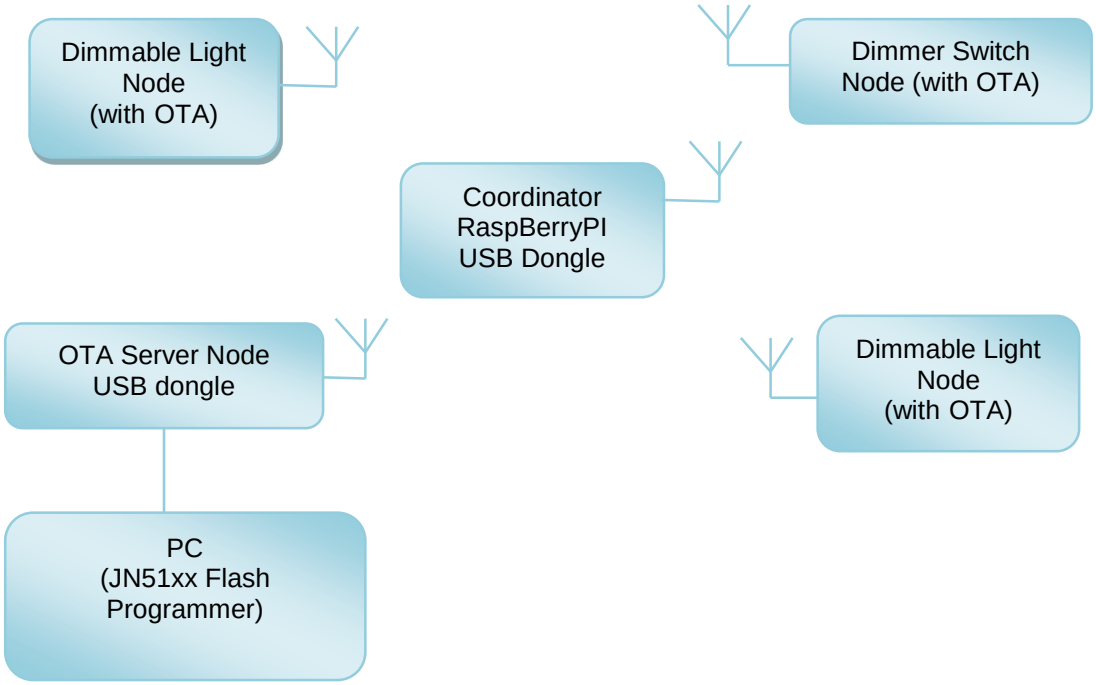
Note 1: Once started, decommissioning can be aborted by pressing any button other than SW1.

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 5.5.5) 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.

9 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.

9.1 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)*.

9.2 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.

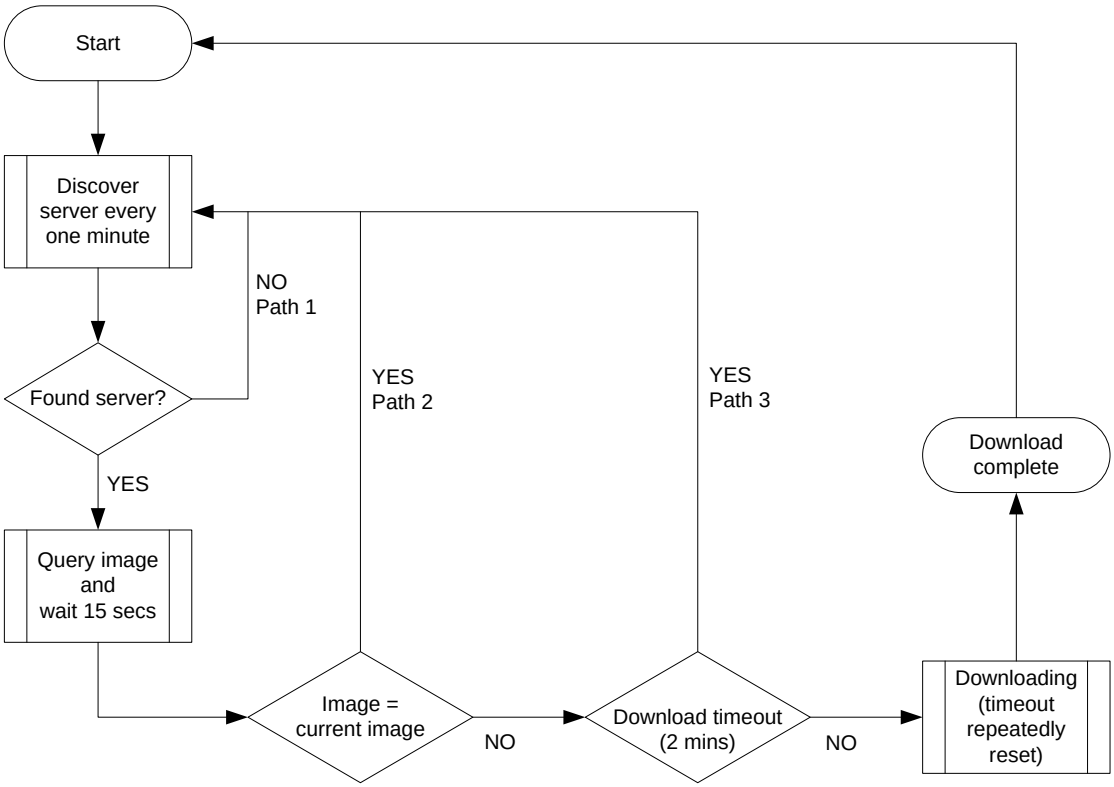


Figure 1: OTA Upgrade Discovery Process on Client

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.

9.3 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 1).

The table below lists and describes the supplied pre-built binary files for the above cases. The binaries are provided for JN5169 in the table below.

OTA Binary	Notes
OTA Upgrade Server	
OTAServer_JN5169.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_JN5169_DR1199_NFC_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_JN5169_DR1199_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_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_JN5169_DR1175_LED_EXP_MONO_NFC_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_JN5169_DR1175_LED_EXP_RGB_NFC_OTA_<xxx>	Eight images as described above for Dimmable Light
Colour Dimmable Light with Occupancy Sensor	
OTAColorDimmableLightWithOccupancy_JN5169_DR1175_LED_EXP_RGB_NFC_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).

9.4 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 the JN516x-EK004 Evaluation Kit (the named files are for the JN5169 device but the procedure is the same for the JN5168 device with the corresponding JN5168 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 1). Therefore, you can omit the build steps and use the pre-built files, if you wish.

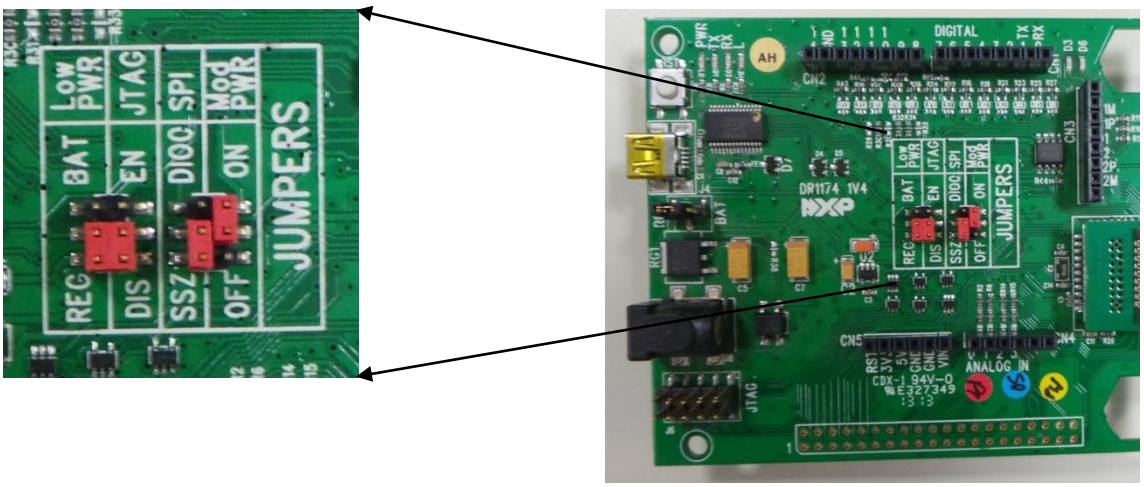
Note 2: When building an upgrade image, both a **.bin** and a **.ota** file will be produced containing the image. The **.ota** file is for use with the current JN51xx Production Flash Programmer (JN-SW-4107) and the **.bin** file is for use with the former JN51xx Flash Programmer (JN-SW-4007) – see Section 1. 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

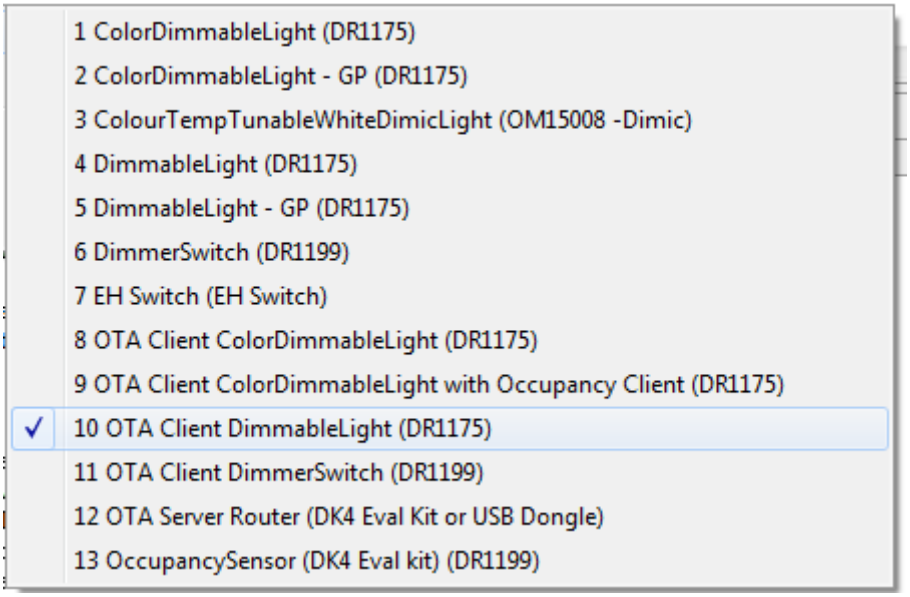
The evaluation kit hardware that is used for the Dimmable Light node is a DR1174 Carrier Board fitted with a DR1175 Lighting/Sensor Expansion Board. The Carrier Board has a SPI Flash device that can be used as external storage during the OTA download process. To make use of this Flash device, the SPI 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)*.



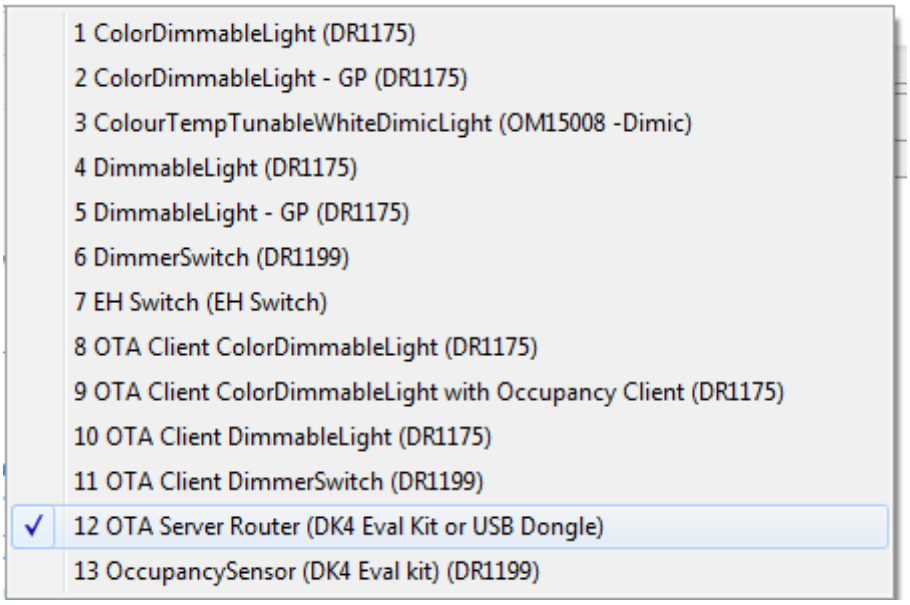
The build configuration “OTA Client DimmableLight1” will result in the following two binary files, both located in the **/DimmableLight/Build/OTABuild** directory:

- **DimmableLight_JN5169_DR1175_LED_EXP_MONO_ NFC_OTA_Client_v1.bin**: This binary file must be programmed into the JN5169 module of the Dimmable Light device. This file has been coded as version 1 of the Dimmable Light image.
- **DimmableLight_JN5169_DR1175_LED_EXP_MONO_ NFC_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_JN5169.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_JN5169.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_JN5169_DR1175_LED_EXP_MONO_NFC_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_JN5169_DR1175_LED_EXP_MONO_ NFC_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_JN5169_DR1175_LED_EXP_MONO_ NFC_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.

9.5 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

9.5.1 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 1 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.

9.5.2 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.

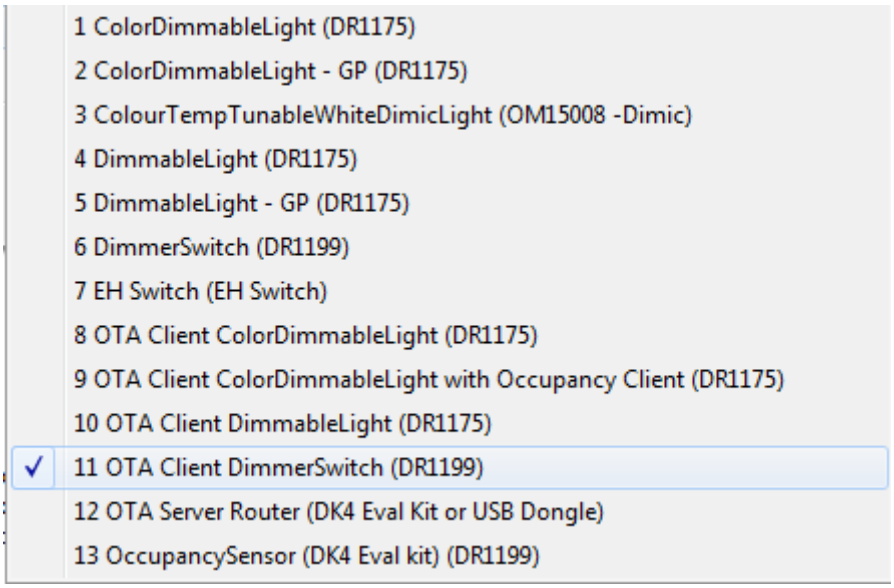
The AES encryption key can be programmed into a JN516x device using the JN51xx Production Flash Programmer (JN-SW-4107) with the `--deviceconfig=` option, as described in the *JN51xx Production Flash Programmer User Guide (JN-UG-3099)*.

9.6 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 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).

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] - JN-UG-3112-NFC-Commissioning-User-Guide
[R8] - ZigBee HA Profile Specification
[R9] - ZigBee Cluster Library (ZCL) Specification
[R10] - ZigBee Green Power Profile Specification
[R11] - docs-13-0553-37-00ha-ha-1-2-errata-document

Documents [R1] to [R7] can be obtained from the NXP Wireless Connectivity TechZone, while documents [R8] to [R11] 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, the pre-built binaries are without TRACE_START activated.

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-1221) 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-1221-Zigbee-HA-with-NFC

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.

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 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 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 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.

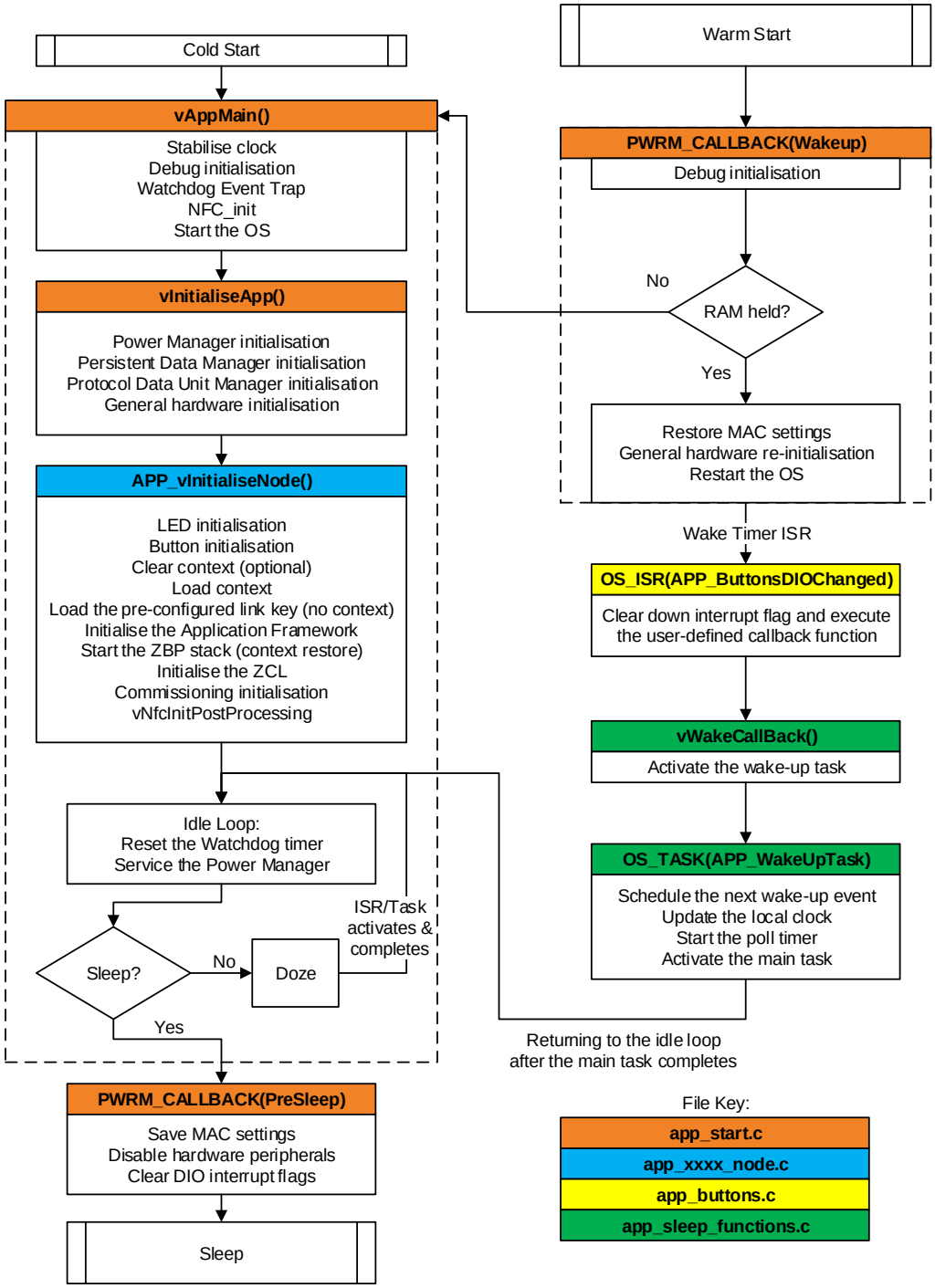


Figure 2: Typical Start-up Flow

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 reads the NTAG I²C NDEF structure to see if network information and Join command is available. If these information are not available the light is in idle until an NFC event is detected.

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 reads the NTAG I²C NDEF structure to see if network information and Join command is available. If these information are not available the switch is in idle until an NFC event is detected.

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. Additional states must be added to this switch statement if further operational modes are required.

10.6.2 Handling a Key Press and Release

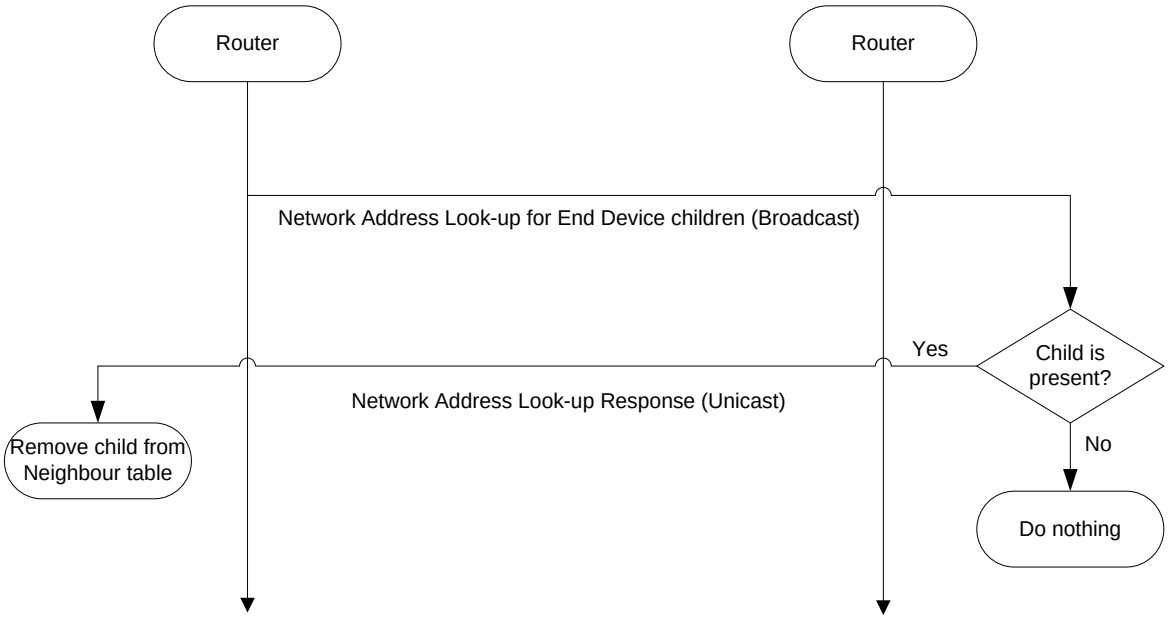
For Dimmer Switch:

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.

10.7 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.

10.8 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
Light Node	**Single Flash:** User input accepted during power-cycling
	Constantly ON: Light node has joined the network
Dimmer Switch	**LEDs D1, D2 flash alternately:** Joining or re-joining the network
	LEDs D1,D2 and D3 constantly OFF: Node in the network

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.*

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.
ndef.c/h	Implement the NDEF layer driver to read / write NDEF messages
nfc.c / h	Implement the NFC loop task using the NFC module
ntag.c / h	Implement the I2C NTAG driver

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_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_ZHA_Light_JN516x_mono_nfc.oscfgdiag	This is the JenOS configuration diagram file as described above for mono including an additional NFC task. 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_nfc.oscfgdiag	This is the JenOS configuration diagram file as described above for RGB including an additional NFC task.
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)*.

App_ZHA_Controller_JN516x_NFC.oscfgdiag	This is the JenOS configuration diagram file as described above including an additional NFC task.

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.

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.
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.

The compile-time macros to configure NFC as follows:

Macro	Description
NFC_SUPPORT	Enable use of NFC
NTAG_FD_PIN	NTAG field detector DIO pin number
CONFIG_NUMBER_OFF_APP_NDEF_MSG	Number of NDEF messages registered by the application
CONFIG_NDEF_MAX_TYPE_LENGTH	Max of bytes of the NDEF message payload

Appendix C - Build File Descriptions

Common Build Files

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

Filename	Description
Makefile	This 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_JN516x.ld	Linker command file defining the default application stack size without Green power option . Can adjust _stack_size for the desired stack size.

Filename	Description
APP_stack_size_ JN516x_GP.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.

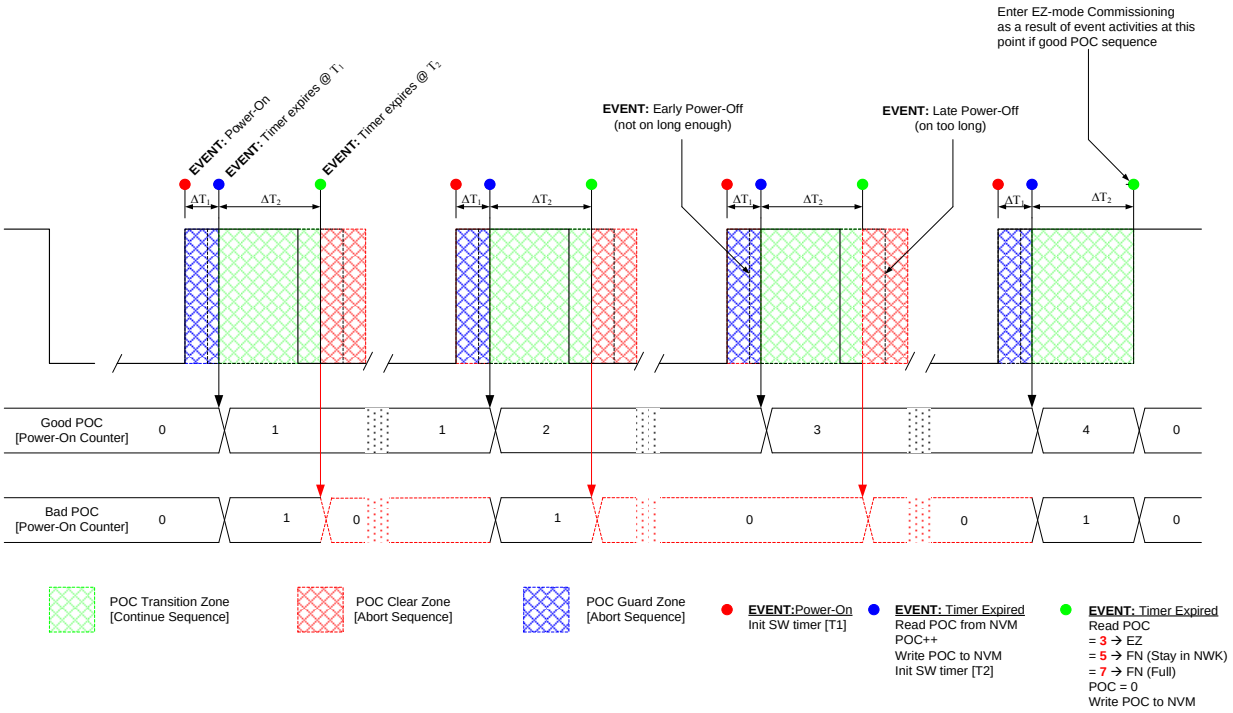


Figure 3: Dimmable Light Power-Cycling Operations

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.

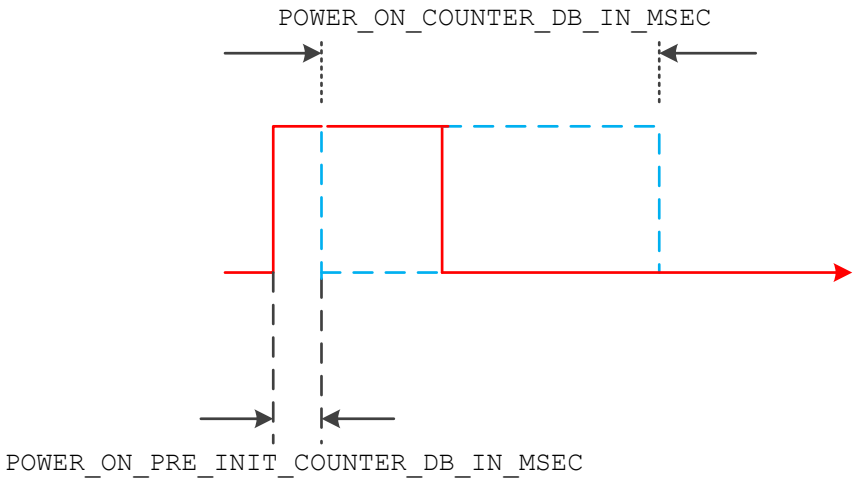


Figure 4: Dimmable Light Compile-time Constants

Revision History

Version	Notes
1.0	First release
1.1	Occupancy, OTA and GP features added, combined with NFC Additional HW CCTW

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