Application Note: JN-AN-1219

ZigBee 3.0 Controller and Switch

This Application Note provides example applications for a controller and a switch in a ZigBee 3.0 network that employs the NXP JN516x and/or JN517x wireless microcontrollers. An example application can be employed as:

- **A demonstration using the supplied pre-built binaries that can be run on nodes of the JN516x/7x hardware kits**
- **A starting point for custom application development using the supplied C source files and associated project files**

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

The Application Note also includes an example of a typical ZigBee Green Power (GP) Energy Harvesting switch.

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

1 Introduction

A ZigBee 3.0 wireless network comprises a number of ZigBee software devices that are implemented on hardware platforms to form nodes. This Application Note is concerned with implementing the device types for a controller and a switch on the NXP JN516x and JN517x platforms.

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

- Color Scene Controller
- Dimmer Switch

The above device types are detailed in the *ZigBee 3.0 Devices User Guide [JN-UG-3114]* and the clusters used by the devices are detailed in the *ZigBee Cluster Library (for ZigBee 3.0) User Guide [JN-UG-3115]*.

> **Note:** If you are not familiar with ZigBee 3.0, you are advised to refer the *ZigBee 3.0 Stack User Guide [JN-UG-3113]* for a general introduction.

This Application Note also provides example implementation of Green Power switch that using the following device type from the ZigBee PRO Green Power Feature Specification:

- GP Switch

For information on the ZigBee Green Power (GP) switch provided in this Application Note, refer to Section 7.

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

2 Development Environment

2.1 Software

In order to use this Application Note, you need to install the Eclipse-based Integrated Development Environment (IDE) and Software Developer's Kit (SDK) that are appropriate for the chip family which you are using - either JN516x or JN517x:

- **JN516x:** If developing for the JN516x microprocessors, you will need:
    - 'BeyondStudio for NXP' IDE [JN-SW-4141]
    - JN516x ZigBee 3.0 SDK [JN-SW-4170]

  For installation instructions, refer to the *BeyondStudio for NXP Installation and User Guide (JN-UG-3098)*.
- **JN517x:** If developing for the JN517x microprocessors, you will need:
    - LPCXpresso IDE
    - JN517x ZigBee 3.0 SDK [JN-SW-4270]

  For installation instructions, refer to the *JN517x LPCXpresso Installation and User Guide (JN-UG-3109)*.

The LPCXpresso software can be obtained as described in the *JN517x ZigBee 3.0 SDK Release Notes*, which indicate the version that you will need.

All other resources are available via the ZigBee 3.0 page of the NXP web site.

> **Note:** The code in this Application Note can be used in either BeyondStudio or LPCXpresso and the process for importing the application into the development workspace is the same for both.
>
> **Note:** Prebuilt JN5169 and JN5179 application binaries are supplied in this Application Note package, but the applications can be rebuilt for other devices in the JN516x and JN517x families (see Section 8.3).

2.2 Hardware

Hardware kits are available from NXP to support the development of ZigBee 3.0 applications. The following kits respectively provide JN516x-based and JN517x-based platforms for running these applications:

- JN516x-EK004 Evaluation Kit, which features JN5169 devices
- JN517x-DK005 Development Kit, which features JN5179 devices

Both of these kits support the NFC commissioning of network nodes (see Section 3.1).

It is also possible to develop ZigBee 3.0 applications to run on the components of the earlier JN516x-EK001 Evaluation Kit, which features JN5168 devices, but this kit does not support NFC commissioning.

3 Application Note Overview

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

Application	Device Type
ColorSceneController	Color Scene Controller
DimmerSwitch	Dimmer Switch
EH_Switch	Green Power Switch

Table 1: Example Applications

For each application, source files and pre-built binary files are provided in the Application Note ZIP package. The pre-built binaries can be run on components of the JN516x/7x Evaluation Kits.

- To load the pre-built binaries into the evaluation kit components and run the demonstration application, refer to Section 5.
- To start developing your own applications based on the supplied source files, refer to Section 8.

3.1 NFC Hardware Support

Some NXP hardware kits for the development of ZigBee 3.0 applications provide the possibility of network commissioning through Near Field Communication (NFC). The kits and components that provide NFC support are indicated in the table below.

Hardware Kit	Hardware Components for NFC	Field Detect Connection
JN517x-DK005	NFC is built into the OM15028 Carrier Board	GPIO 17
JN516x-EK004	DR1174 Carrier Board plus OM15044 and either OM55679/NT3120 or OM5569/NT322E *Note: A 4K7 resistor should be fitted to the R1 pads on the OM15044 board to avoid unnecessary reads of the NTAG due to the FD line floating.*	DIO 0

Table 2: NFC Support in JN516x/7x Hardware Kits

The Field Detect of the NFC chip needs to be connected to an IO line of the JN516x/7x module so that an interrupt can be generated as the device is moved in or out of the field. This is achieved by fitting a jumper to the pin specified in the above table.

Note: Early samples of the JN516x-EK004 kit used a yellow wire rather than a jumper for the Field Detect connection, but the pin is the same.

3.2 NFC Data Formats

Two different NFC data formats are supported for commissioning. The Router and End Device applications can be built to support only one (or none) of these:

- **ZigBee Installation Code Format:** This is a newer format introduced with v1003 of this Application Note. The applications are built to use this format by default. This format uses a key derived from the device's ZigBee Installation Code to encrypt data in the NTAG.
- **AES Encryption Format:** This older format uses an AES key to encrypt data in the NTAG.

The selection of the data format can be made at compile-time by using makefile variables described in the Router Command Line Build Options or End Device Command Line Build Options.

Note: The Application Note JN-AN-1222, IoT Gateway Host With NFC, versions v2007 and later is able to commission either of these formats depending upon the data in the presented NTAG. Earlier versions support only AES Encryption Format.

4 Supported Device Types

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

- Color Scene Controller
- Dimmer Switch

These switch/controller devices types must be paired for operation with lighting device types. Example applications for the paired device types are provided in the Application Note *ZigBee 3.0 Light Bulbs [JN-AN-1218]*. Two device types can be paired if they support the same cluster (Color Control, Level Control or On/Off cluster) such that the cluster client on the switch/controller device type can access/control attributes of the cluster server on the lighting device type.

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

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

Table 3: Switch/Controller Device Types and Controllable Attributes

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

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

Table 4: Lighting Device Types and Accessible Attributes

5 Running the Demonstration Application

This section describes how to use the supplied pre-built binaries to run the example applications on components of the JN516x-EK004 or JN517x-DK005 kit. Both applications run on a JN5169 module on a DR1174 Carrier Board or a JN5179 module on an OM15028 Carrier Board, fitted with a specific expansion board. The Color Scene Controller application can also be used on the DR1159 Remote Control Unit of the JN516x-EK001 kit.

5.1 Loading the Applications

The table below lists the application binary files supplied with this Application Note and indicates the JN516x/7x hardware kit components on which the binaries can be used. These files are located in the **Build** directories for the relevant applications.

Application	**JN5168 Binary File**	**JN516x-EK001 Hardware**
ColorSceneController	**ColorSceneController_JN5168_DR1159.bin**	DR1159 Remote Control Unit
Application	**JN5169 Binary File**	**JN516x-EK004 Hardware**
ColorSceneController	**ColorSceneController_JN5169_DR1199.bin**	DR1174 Carrier Board with JN5169 module DR1199 Generic Expansion Board
DimmerSwitch	**DimmerSwitch_Ntaglcode_JN5169_DR1199.bin**	DR1174 Carrier Board with JN5169 module DR1199 Generic Expansion Board OM15044 NTAG Adaptor Board OM5569/NT322E NTAG Board
GPSwitch	**EH_Switch_JN5169_DR1199.bin**	DR1174 Carrier Board with JN5169 module DR1199 Generic Expansion Board
Application	**JN5179 Binary File**	**JN517x-DK005 Hardware**
ColorSceneController	**ColorSceneController_JN5179_DR1199.bin**	OM15028 Carrier Board with JN5179 module DR1199 Generic Expansion Board
DimmerSwitch	**DimmerSwitch_Ntaglcode_JN5179_DR1199.bin**	OM15028 Carrier Board with JN5179 module DR1199 Generic Expansion Board
GPSwitch	**EH_Switch_JN5179_DR1199.bin**	OM15028 Carrier Board with JN5179 module DR1199 Generic Expansion Board

Table 5: Application Binaries and Hardware Components

A binary file can be loaded into the Flash memory of a JN516x/7x device using the JN51xx Flash Programmer [JN-SW-4107], available via the NXP web site. This software tool is described in the *JN51xx Production Flash Programmer User Guide [JN-UG-3099]*.

Note: You can alternatively load a binary file into a JN516x/7x device using the Flash programmer built into the relevant IDE.

To load an application binary file into a JN516x/7x module on a Carrier Board of a kit, follow the instructions below:

1. Connect a USB port of your PC to the USB Mini B port on the Carrier Board using a 'USB A to Mini B' cable. At this point, you may be prompted to install the driver for the cable.
2. Determine which serial communications port on your PC has been allocated to the USB connection.
3. On your PC, open a command window.
4. In the command window, navigate to the Flash Programmer directory:

   C:\NXP\ProductionFlashProgrammer

5. Run the Flash programmer to download your binary file to JN516x/7x Flash memory by entering a command with the following format at the command prompt:

   ```
   JN51xxProgrammer.exe -s <comport> -f <path to .bin file>
   ```

   where `<comport>` is the number of the serial communications port.

6. Once the download has successfully completed, disconnect the USB cable and, if required, reset the board or module to run the application.

To load an application binary file into the DR1159 Remote Control Unit supplied in the JN516x-EK001 kit, refer to the firmware re-programming instructions in the *JN516x-EK001 Evaluation Kit User Guide (JN-UG-3093)*.

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

5.2 Using DimmerSwitch

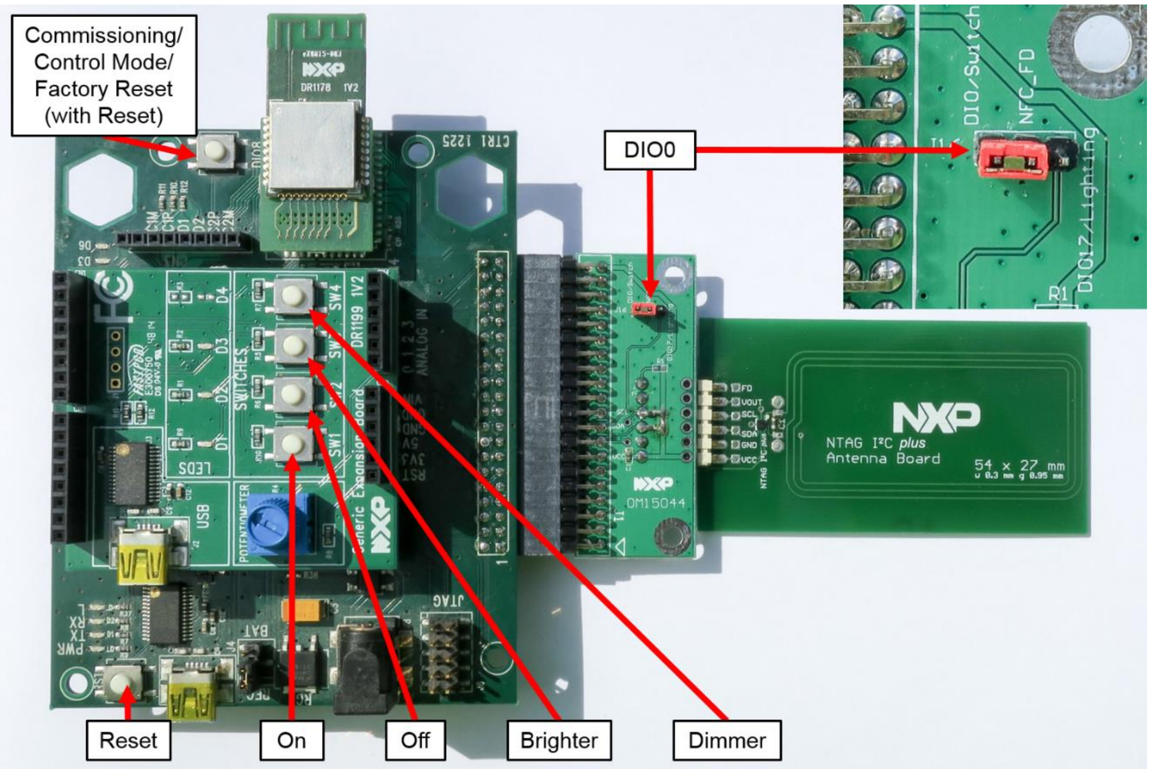
This section describes how to commission and operate the DimmerSwitch application in a ZigBee 3.0 network. To use this application, you must have programmed the relevant application binary into the JN5169/79 module on a Carrier Board fitted with the DR1199 Generic Expansion Board, as described in Section 5.1:

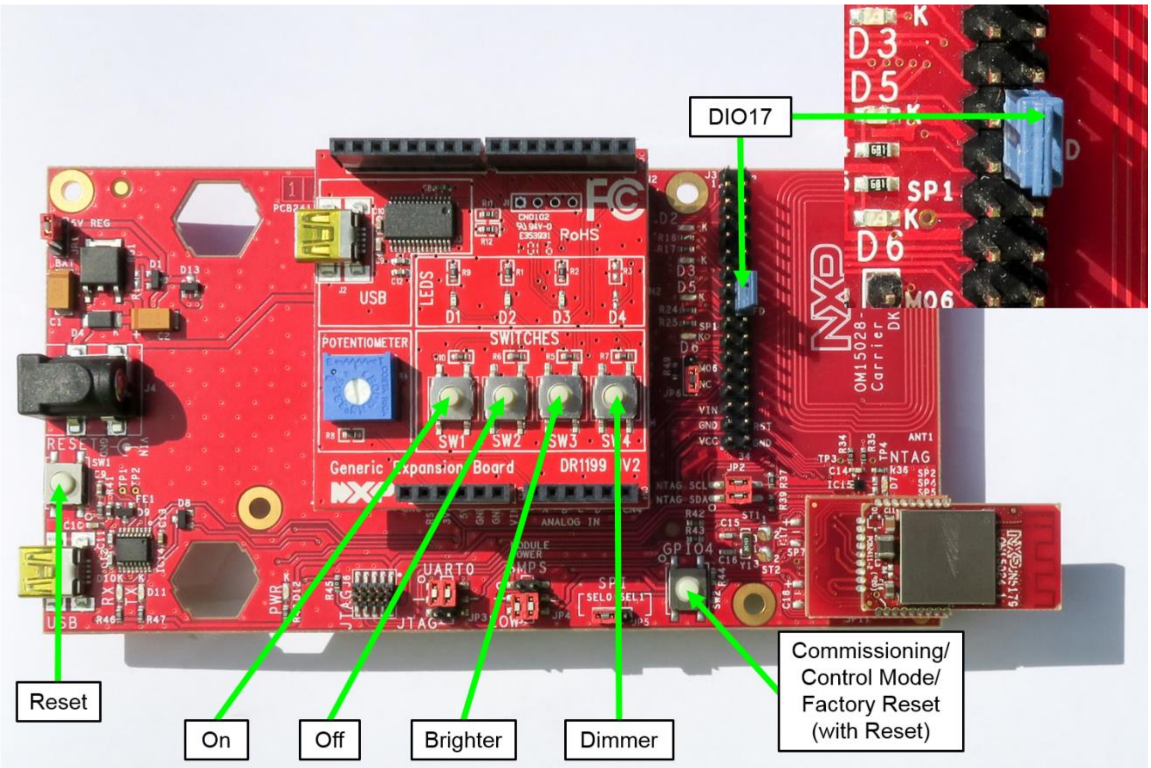
- **DimmerSwitch_NtagIcode_JN5169_DR1199.bin** for a JN5169 module
- **DimmerSwitch_NtagIcode_JN5179_DR1199.bin** for a JN5179 module

Note: To use this application which is based on the Dimmer Switch device type, you will also need to implement the paired Dimmable Light device type as described in the Application Note *ZigBee 3.0 Light Bulbs [JN-AN-1218]*.

5.2.1 Dimmer Switch Functionality

The ZLO Dimmer Switch device resides on a node (fitted with the DR1199 Generic Expansion Board) which acts as an End Device in the network. As an End Device, it can only join an existing network and cannot form a new network. This switch device can be used to perform the commissioning of the light devices and control them, using the controls indicated in the diagrams below.





LED Name	Physical LED on Board
LED1	D1 on DR1199 expansion board
LED2	D2 on DR1199 expansion board
LED3	D6 on DR1174 carrier board D2 on OM15028 carrier board

Table 6: LED Hardware Mapping

5.2.2 Clearing Context Data on the Device

When loading the application for the first time, any persistent context data must be cleared in the device – the required action depends on the Carrier Board used:

- On the DR1174 Carrier Board (green), press and release the **RST** button while holding down the **DIO8** button (both buttons are on the Carrier Board).
- On the OM15028 Carrier Board (red), press and release the **RESET** button while holding down the **GPIO4** button (both buttons are on the Carrier Board).

A factory-new Dimmer Switch waits for user input on power-up, which is indicated by the LED pair LED1+LED3 and LED2 flashing alternately (for physical LEDs, see Table 6 above).

5.2.3 Joining a Network

5.2.3.1 Classical Network Joining

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

1. Start Network Steering on the switch by pressing any button on the DR1199 Generic Expansion Board. As soon as a button-press is detected, the device will search for a suitable network to join. The LED pair LED1+LED3 and LED2 will flash alternately during this state.

2. The outcome of Network Steering is now indicated as follows:

   - If a suitable open network is found (‘Permit Join’ is true), the switch node will join and indicate success by briefly illuminating LEDs LED1-LED3.
   - If a suitable network is not found, the switch node goes into deep sleep mode, which is indicated by all LEDs being switched off.

5.2.3.2 Joining an Existing Network using NFC

A Dimmer Switch node can join or move to an existing network by exchanging NFC data with a ZigBee IoT Gateway Host, described in the Application Note *ZigBee IoT Gateway Host with NFC (JN-AN-1222)*. This provides a fast and convenient method to introduce new devices into such a network.

Ensure the hardware is set up for NFC as described in Section 3.1.

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

5.2.4 Commissioning

This section describes how to commission light nodes to be controlled from the Dimmer Switch, either as bound devices or as a group of lights – work through either Section 5.2.4.1 or Section 5.2.4.2. At least one light should be commissioned by each method - the same light can be used for both methods.

5.2.4.1 Commissioning Light Nodes – Binding

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

1. Put the lights that need to be commissioned with the Dimmer Switch into 'Finding and Binding' mode (this step is light-specific). The lights will start to flash in Identify mode.
2. Press and hold down the Commissioning button on the Carrier Board of the Dimmer Switch - button **DIO8** on DR1174 or button **GPIO4** on OM15028.
3. Once LED D1 starts flashing on the DR1199 Generic Expansion Board, press and release the **SW2** button.
4. Now hold down the **DIO8** or **GPIO4** button on the Carrier Board until the lights come out of Identify mode (when **DIO8/GPIO4** is released, the node exits Commissioning mode and returns to Individual Control mode).

5.2.4.2 Commissioning Light Nodes – Grouping

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

1. Put the lights that need to be commissioned with the Dimmer Switch into 'Finding and Binding' mode (this step is light-specific). The lights will start to flash in Identify mode.
2. Press and hold down the Commissioning button on the Carrier Board of the Dimmer Switch - button **DIO8** on DR1174 or button **GPIO4** on OM15028.
3. Once LED1 starts flashing on the DR1199 Generic Expansion Board, press and release the **SW3** button.
4. Now hold down the **DIO8/GPIO4** button on the Carrier Board until the lights come out of Identify mode (when **DIO8/GPIO4** is released, the node exits Commissioning mode and returns to Individual Control mode).

5.2.5 Operation

The Dimmer Switch has three modes of operation:

- Commissioning mode
- Individual Control mode
- Group Control mode

These modes are described below.

5.2.5.1 Commissioning Mode

In this mode, the commissioning functionality of the ZigBee Base Device can be invoked, such as Network Steering and 'Finding and Binding'. The Dimmer Switch can be put into Commissioning mode by pressing and holding down the **DIO8** button on the DR1174 Carrier Board or the **GPIO4** button on the OM15028 Carrier Board. This mode of operation is indicated by the continuous flashing of LED1 on the DR1199 Generic Expansion Board.

In Commissioning mode, you can use the buttons **SW1-SW4** on the DR1199 Generic Expansion Board to perform various operations, as follows:

- Press **SW1** to trigger Network Steering. This will enable ‘Permit Join’ on the network.
- Press **SW2** to add a light node (that is in Identify mode) to the Binding table on the Dimmer Switch. This is Finding and Binding without grouping.
- Press **SW3** to add a light node (that is in Identify mode) to the group for the Dimmer Switch. This is Finding and Binding with grouping.
- Press **SW4** to move to another bound light (to control in Individual Control mode). This is indicated by identification of the light node.

When **DIO8/GPIO4** is released, the node exits Commissioning mode and returns to Individual Control mode.

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

5.2.5.2 Individual Control Mode

This mode is used to control individual lights which have already been bound to the Dimmer Switch using Finding and Binding without grouping (refer to Section 5.2.4.1). This mode of operation can be entered by pressing and releasing **DIO8/GPIO4** button on the Carrier Board. The next bound light can be selected by pressing the **SW4** button on the DR1199 Generic Expansion Board while in Commissioning mode (refer to Section 5.2.5.1).

In Individual Control mode, you can use the switches **SW1-SW4** on the DR1199 Generic Expansion Board to perform various operations, as follows:

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

5.2.5.3 Group Control Mode

This mode is used to control a group of lights which has already been formed for the Dimmer Switch using Finding and Binding with grouping (refer to Section 5.2.4.2). This mode of operation can be entered by pressing either of the following switch combinations: **SW1+SW3** or **SW2+SW4.**

In Group Control mode, you can use the switches **SW1-SW4** on the DR1199 Generic Expansion Board to perform various operations, as follows:

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

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

Note: The Dimmer Switch can be woken from deep sleep by pressing any of the buttons **SW1** to **SW4**. It does not wake from deep sleep if the user presses the **DIO8/GPIO4** button on the Carrier Board, as this button is involved in a special sequence to erase persisted data.

5.3 Using the Color Scene Controller

This section describes how to commission and operate the Color Scene Controller application in a ZigBee 3.0 network. The application can be built for use on any one of the following hardware platforms:

- DR1159 Remote Control Unit supplied in the JN516x-EK001 Evaluation Kit
- DR1174 Carrier Board plus DR1199 Generic Expansion Board supplied in the JN516x-EK004 Evaluation Kit
- OM15028 Carrier Board plus DR1199 Generic Expansion Board supplied in the JN517x-DK005 Development Kit

Note the DR1159 Remote Control Unit contains the JN5168 device and requires the application to be built for this chip. The DR1199-based version of the application can be built for the JN5168, JN5169 or JN5179 device, depending on the JN516x/7x module fitted. The built binaries are tagged with both the chip and hardware platform being used.

To use this application, you must have programmed the relevant application binary into the appropriate hardware for the build, as indicated in the table below.

Binary File	Hardware Component
ColorSceneController_JN5168_DR1159.bin	DR1159 Remote Control Unit
ColorSceneController_JN5169_DR1199.bin	DR1199 Generic Expansion Board on DR1174 Carrier Board fitted with JN5169 module
ColorSceneController_JN5179_DR1199.bin	DR1199 Generic Expansion Board on OM15028 Carrier Board fitted with JN5179 module

Note: When using an application built for a Carrier Board with DR1199 Expansion Board, a Graphical User Interface (GUI) can be run on a PC to simulate the keys of the Remote Control Unit (see Section 5.3.1). A binary serial protocol is used to send key-press data from the GUI to the device and to send LED status information from the device to the GUI. If using the application without the GUI but with a debug terminal, the LED information will look like garbage in the debug terminal, although it is valid data.

5.3.1 Remote Control GUI

To compensate for the lack of button inputs when the DR1199-based hardware is used, a Windows-based Graphical User Interface (GUI) is provided. This connects via the JN516x/7x UART and acts as a substitute for the Remote Control Unit. The GUI is provided as source code and is a Visual Studio project.

To use the Remote Control GUI:

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

Button events will be passed to the Color Scene Controller hardware, resulting in the over-the-air transmission of LED control commands by the JN516x/7x module. LED control messages are also sent back to the GUI to allow it to mirror the LED states on the hardware.

5.3.2 Color Scene Controller Functionality

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

The device supports the following clusters as clients:

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

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

The DR1159 Remote Control Unit provides the full control functionality of the application. When the DR1199-based hardware is used instead, a Windows-based Graphical User Interface (GUI) can be run to provide similar remote control functions from a PC (see Section). The functions of the keys on the DR1159 Remote Control Unit and the keys from the Windows Remote Control GUI for the Color Scene Controller are detailed in Section 5.3.8.

When using the DR1199-based hardware in conjunction with the key inputs provided by the Remote Control GUI, the physical buttons on the boards also provide some functionality:

- **SW1:** On/Off toggle
- **SW3:** Toggle between Groupcast and Unicast modes, if part of a network
- **SW4:** Initiate association joining, if not already part of a network (factory-new)
- **SW4:** Initiate Finding and Binding, if part of a network (not factory-new)
- **DIO8/GPIO4:** Initiate Touchlink commissioning

5.3.3 Joining or Forming a Network

A factory-new Color Scene Controller can join an existing network through classical ‘discovery and association’, or it can join an existing network or form a new network using Touchlink.

5.3.3.1 Joining a Network using Discovery and Association

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

To join the device to a network:

1. On an existing network, trigger 'Network Steering for a device on the network'.

2. On the Color Scene Controller, press the **+** key if using the DR1159 Remote Control Unit or the **SW4** button if using the DR1199 Generic Expansion Board.

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

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

5.3.3.2 Joining or Forming a Network using Touchlink

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

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

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

1. Bring the controller into close proximity (about 10cm) of the Target device.

2. On the Color Scene Controller, press the # key if using the DR1159 Remote Control Unit or the **DIO8/GPIO4** button on the Carrier Board if using the DR1199 Generic Expansion Board.

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

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

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

1. Bring the controller into close proximity (about 10cm) of another Touchlink initiator that is already on the network.
2. Initiate Touchlink on both devices simultaneously. On the Color Scene Controller, press the **#** key if using the DR1159 Remote Control Unit or the **DIO8/GPIO4** button on the Carrier Board if using the DR1199 Generic Expansion Board.

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

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

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

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

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

5.3.4 Allowing Other Devices to Join

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

1. On the Color Scene Controller, press the *** * C** key sequence if using the DR1159 Remote Control Unit or the **SW2** button if using the DR1199 Generic Expansion Board.
2. Trigger 'Network Steering for a device not on a network' on the devices wishing to join.

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

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

5.3.5 Binding Nodes

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

1. Trigger Finding and Binding on all the targets requiring bindings.
2. On the Color Scene Controller, press the *** #** key sequence if using the DR1159 Remote Control Unit or the **SW4** button if using the DR1199 Generic Expansion Board.

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

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

5.3.6 Re-Joining the Network

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

5.3.7 Performing a Factory Reset

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

- If using the DR1159 Remote Control Unit:
  a) Press the *** * *** key sequence.
  b) Press the **- + -** key sequence.
- If using the DR1199 Generic Expansion Board:
  a) Press and hold down the **DIO8/GPIO4** button on the Carrier Board.
  b) Press and release the **RST** button on the Carrier Board.
  c) Release the **DIO8/GPIO4** button.

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

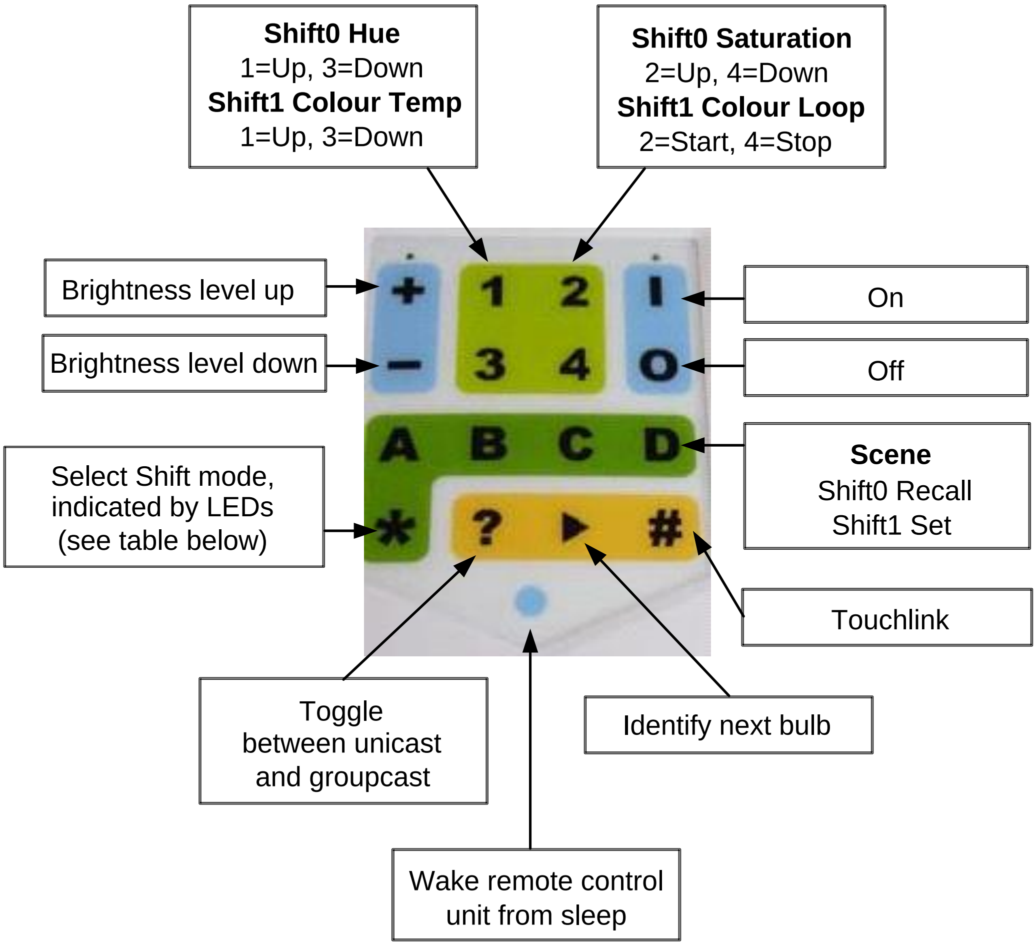
There are three supported over-the-air commands for removing a device from the network - these are:

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

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

5.3.8 Functionality of the Color Scene Controller

The button functions on the DR1159 Remote Control Unit for the Color Scene Controller are as shown below. The Windows-based Remote Control GUI provides exactly the same buttons and works in the same way.



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

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

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

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

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

Table 7: Shift Modes

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

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

Table 8: Button Functions in Shift0 Mode

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

Table 9: Button Functions in Shift1 Mode

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

Table 10: Button Functions in Shift2 Mode

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

Table 11: Button Functions in Shift3 Mode

6 Over-The-Air (OTA) Upgrade

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

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

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

6.1 Overview

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

- If built for the JN5168 device then external Flash memory will be used to store the upgrade image before replacing the old one.
- If built for the JN5169 or JN5179 device then the internal Flash memory will be used to store the upgrade image.

The Application Note *ZigBee 3.0 IoT Control Bridge (JN-AN-1216)* has OTA server functionality built into it. A device called OTA_server is provided to host the upgrade images that the clients will request.

6.2 OTA Upgrade Operation

To implement an OTA upgrade:

1. Build the switch application with `OTA=1` in the makefile to enable OTA upgrade (this option is not enabled by default). There is an OTA debug flag defined in the makefile: `CFLAGS += -DDEBUG_APP_OTA`. Uncomment this line if the OTA debug is required.

The binary files for the Dimmer Switch are created in the **OTABuild** folder – bootable binaries have the extension **.bin** and no version suffix, and upgrade binaries have the extension **.ota** or **.bin** and a version suffix. The upgrade image is intended to be loaded into external Flash memory of the OTA Upgrade server using the JN51xx Production Flash Programmer (JN-SW-4107), as described in Step 4 below. Encrypted upgrade binary images will have a **_ENC** suffix. There are two binaries in a set, with the files having different versions with different headers so that the upgrading of the switch can be tested - a bootable image, version 1 (v1), and one upgrade image, version 2 (v2).

2. Program the bootable binary file from the **OTABuild** folder into the internal Flash memory of the JN516x/7x device on the Carrier Board of the switch node - for example, **DimmerSwitch_JN5169_DR1199_OTA_Client_v1.bin**. You can do this using the Flash Programmer within the IDE, as described in the appropriate IDE *Installation and User Guide (JN-UG-3098* or *JN-UG-3109)*. Alternatively, you can use the JN51xx Production Flash Programmer (JN-SW-4107) described in the *JN51xx Production Flash Programmer User Guide (JN-UG-3099).*

3. Form a network with a Dimmer Switch node and the Control Bridge in the normal way using centralised commissioning.

4. Load a **.ota** upgrade image (v2) into the external Flash memory of the Control Bridge using the JN51xx Production Flash Programmer (JN-SW-4107) - the required command line will be similar to the following:

```
Jn51xxProgrammer –S external –s COM<port> -f <filename>
```

5. When viewing the UART output from the Dimmer Switch node, the upgraded image should be found and the switch node upgraded.

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

6.3 Image Credentials

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

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

6.4 Encrypted and Unencrypted Images

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

For JN5169 and JN5179 builds, to use encrypted images the following define must be included as a build option in the **zcl_options.h** file:

```
#define INTERNAL_ENCRYPTED
```

6.5 Upgrade and Downgrade

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

7 ZigBee Green Power (GP) Switch

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

- Level Control switch
- On/Off switch

For information on the NXP implementation of ZigBee Green Power, refer to the *ZigBee Green Power User Guide (JN-UG-3119)*.

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

7.1 Loading the Application

The table below indicates the GP application binary file supplied with this Application Note as well as the JN516x/7x kit hardware components on which the binary can be used. This file is located in the **Build** directory for the application.

Application	Binary File	Hardware Components
EH_SWITCH	**EH_Switch_JN5169_DR1199.bin**	DR1199 Generic Expansion Board on DR1174 Carrier Board fitted with JN5169 module
EH_SWITCH	**EH_Switch_JN5179_DR1199.bin**	DR1199 Generic Expansion Board on OM15028 Carrier Board fitted with JN5179 module

The binary file can be loaded into the Flash memory of a JN516x/7x device using the JN51xx Flash Programmer [JN-SW-4107], as described in Section 5.1.

7.2 Using EH_SWITCH

This section describes how to commission and operate the GP Switch in a ZigBee3.0 network with GP ‘infrastructure devices’. The network can be set up using:

- Coordinator application from the Application Note JN-AN-1217
- One of the following devices with GP support (GP infrastructure devices) from the Application Note JN-AN-1218: Dimmable Light, Extended Color Light or Color Temperature Light

To use this application, you must have programmed the application **EH_Switch_JN51xx_DR1199.bin** into the JN51xx module on a Carrier Board fitted with the DR1199 Generic Expansion Board, as described in Section 5.1.

ⓘ **Note:** To use this application which is based on the GP Switch device, you will also need to implement a paired light device with GP support, as described in the Application Note *ZigBee 3.0 Light Bulbs (JN-AN-1218)*.

7.2.1 GP Switch Functionality

The GP Switch application resides on a node fitted with the DR1199 Generic Expansion Board which acts as a GP Energy Harvesting switch. This switch device can be used to perform the commissioning of the light devices and control them.

The GP Switch functionality is implemented on the DR1199 Generic Expansion Board as shown in the diagram for the Dimmer Switch in Section 5.2.1.

7.2.2 Clearing Context Data on the Device

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

The context data can be cleared by pressing and releasing the **RST** button while holding down the **DIO8/GPIO4** button (both buttons are on the Carrier Board).

7.2.3 Commissioning

To commission the GP Switch to operate with one or more lights in a ZigBee 3.0 network that support the GP Combo Basic device (to get the light into the network, refer to the Application Note JN-AN-1218 document), follow the procedure below:

1. Put a light into Commissioning mode (this step is light-specific). The light will start flashing to identify itself.
2. On the GP Switch node, press the **SW1** button on the DR1199 Generic Expansion Board repeatedly at one-second intervals until the light stops flashing and returns to its original state (the GP Switch sends commissioning packets to the light node on each button-press).

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

   The GP Switch is now paired with the light node and can be used to control the light. The **SW1-SW4** buttons on the DR1199 Generic Expansion Board of the GP Switch can be respectively used to send On, Off, Brighter and Dimmer commands to the light node, as illustrated in the diagram in Section 5.2.1.
3. To commission the GP Switch with another light, press the **DIO8/GPIO4** Commissioning button on the Carrier Board of the GP Switch and then repeat Steps 1-2 for the new light.

> **Note:** To decommission the GP Switch from a network, press the **DIO8/GPIO4** Commissioning button on the Carrier Board of the GP Switch followed by the **SW2** button on the DR1199 Generic Expansion Board. This will send out a decommissioning command and cause the switch to be removed from network.

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

7.3 EH_SWITCH Application Code

The code required to build the GP Switch application can be found within the Application Note package in the **EH_Switch/Source** directory. The GP Switch also uses the button handling file from the **Common/Source** directory.

Note the following:

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

7.4 Configuration Setting

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

The following macros are used for the configuration settings:

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

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

- SWITCH_TYPE_DR1199: This macro must be set to 1 when using the DR1199 Generic Expansion Board from the JN517x-DK005, JN516x-EK004 or JN516x-EK001 kit.
- SWITCH_TYPE_DR1216: This macro is used for the DR1216 carrier board. It must be set to 0 when using the DR1199 Generic Expansion Board on a carrier board from the JN517x-DK005, JN516x-EK004 or JN516x-EK001 kit.
- TRACE: Debug prints can be enabled by setting this macro to 1.

7.5 Application Code Size

The GP Switch applications has the following memory footprints on a JN5169 device.

Application	Text Size (Bytes)	Data Size (Bytes)	BSS Size (Bytes)
EH_Switch_JN5169_DR1199	12671	100	5987

The GP Switch applications has the following memory footprints on a JN5179 device.

Application	Text Size (Bytes)	Data Size (Bytes)	BSS Size (Bytes)
EH_Switch_JN5179_DR1199	13968	460	6003

8 Developing with the Application Note

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

- JN516x ZigBee 3.0 SDK [JN-SW-4170] and the 'BeyondStudio for NXP' IDE [JN-SW-4141]
- JN517x ZigBee 3.0 SDK [JN-SW-4270] and the LPCXpresso IDE

These are the resources that you should use to develop JN516x and JN517x ZigBee 3.0 applications, respectively. They are available free-of-charge via the ZigBee 3.0 page of the NXP web site.

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

8.1 DimmerSwitch Application Code

This section describes the DimmerSwitch application code, which is provided in the **Common_Switch** or Common directory for the application. You may wish to use this code as a basis for your own application development. You can rebuild your customised application as described in Section 8.3.

8.1.1 Operational State Machine

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

8.1.2 Handling a Button Press and Release

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

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

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

8.1.3 Handling NFC

app_ntag_aes.c contains the code that drives the NFC commissioning data exchange and initiates the joining process when valid data is read from the NTAG. This code uses the older NTAG data format that employs AES encryption and is not used in the default builds.

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

8.1.4 NTAG Folder (AES Format)

The NTAG library and header files containing the public APIs for NFC are held in the **NTAG** directory. This code uses the older NTAG data format that employs AES encryption and is not used in the default builds.

8.1.5 NFC Folder (ZigBee Installation Code Format)

The NFC libraries and header files containing the public APIs for NFC are held in the **NFC** directory. This code uses the newer NTAG data format that employs ZigBee Installation Code encryption and is used in the default builds.

Documentation for these APIs and the **app_ntag_icode.c/h** APIs can be found in the **NFC.chm** help file in the **Doc** directory of this Application Note.

8.1.6 Handling of Sleep

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

The following macros control the sleep configurations:

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

8.1.7 Handling of OTA Upgrade

During the upgrade process the following behaviour is expected:

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

8.2 Color Scene Controller Application Code

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

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

8.2.1 Touchlink Preconfigured Link Key

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

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

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

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

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

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

   ```
   #define BDB_APPLICATION_DEFINED_TL_MASTER_KEY
   ```

8.2.2 Common Controller Code

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

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

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

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

app_menu_handler_DR1159.c contains a menu handler for the DR1159-based hardware. It interprets key-presses and initiates the ZCL to send the appropriate ZCL commands to devices on the network.

app_menu_handler_DR1199.c contains a menu handler for the DR1199-based hardware. It interprets key-presses from the GUI and button-presses from the switches, and initiates the ZCL to send the appropriate ZCL commands to devices on the network.

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

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

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

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

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

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

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

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

app_captouch_buttons.c handles button-presses from the CapTouch driver (for the DR1159 Remote Control Unit), packages them into application events and posts them to the queue.

DriverCapTouch.c contains the driver software to run the ‘capacitance touch’ algorithm to detect key-presses on the DR1159 Remote Control Unit.

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

Irq_JN516x.S defines which of the hardware interrupts are supported, serviced and at which priority. This is defined by two tables - an interrupt priority table and a table of handler functions. This file in only required for builds for JN516x devices. For the JN517x devices, interrupt set-up is handled in the **app_main** file.

app_serial_interface.c contains functions to process a character received from the UART, manage the serial interface protocol, validate messages and pass key events to the application event queue. It also manages the protocol to send LED messages to the GUI. This file is only used for builds for the DR1199-based hardware.

uart.c contains an interrupt handler and deals with UART management for the serial interface. This file is only used for builds for the DR1199-based hardware.

8.2.3 Code Specific to Color Scene Controller

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

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

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

8.3 Rebuilding the Applications

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

8.3.1 Pre-requisites

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

In order to build the application, this Application Note [JN-AN-1219] must be unzipped into the directory:

<IDE installation root>\workspace

where **<IDE Installation root>** is the path in which the IDE was installed. By default, this is:

- **C:\NXP\bstudio_nxp** for BeyondStudio
- **C:\NXP\LPCXpresso_<version>_<build>\lpcxpresso** for LPCXpresso

The **workspace** directory is automatically created when you start the IDE.

All files should then be located in the directory:

…\workspace\ JN-AN-1219-Zigbee-3-0-Controller-and-Switch

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

There will also be sub-directories **JN516x** and **JN517x** containing the project definition files.

8.3.2 Build Instructions

The software provided with this Application Note can be built for JN516x and JN517x.

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

- To build using makefiles, refer to Section 8.3.2.1.
- To build using the IDE, refer to Section 8.3.2.2.

8.3.2.1 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.

The following command line options can be used to configure the built devices:

- `JENNIC_CHIP_FAMILY=JN516x` to build for a JN516x microcontrollers
- `JENNIC_CHIP_FAMILY=JN517x` to build for a JN517x microcontrollers
- `JENNIC_CHIP=JN5169` to build for a JN5169 microcontroller
- `JENNIC_CHIP=JN5168` to build for a JN5168 microcontroller
- `JENNIC_CHIP=JN5164` to build for a JN5164 microcontroller
- `JENNIC_CHIP=JN5179` to build for a JN5179 microcontroller
- `JENNIC_CHIP=JN5178` to build for a JN5178 microcontroller
- `JENNIC_CHIP=JN5174` to build for a JN5174 microcontroller
- `OTA=0` to build without OTA client
- `OTA=1` to build with OTA client

- `OTA_ENCRYPTED=0` to build OTA images without encryption
- `OTA_ENCRYPTED=1` to build OTA images with encryption
- `APP_NTAG_ICODE=0` to build without NTAG/NFC (ZigBee Installation Code format) support
- `APP_NTAG_ICODE=1` to build with NTAG/NFC (ZigBee Installation Code format) support (this is the default option)
- `APP_NTAG_AES=0` to build without NTAG/NFC (AES Encryption format) support (this is the default option)
- `APP_NTAG_AES=1` to build with NTAG/NFC (AES Encryption format) support

To build an application and load it into a JN516x/7x board, follow the instructions below:

1. Ensure that the project directory is located in

   <IDE 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 at the command prompt enter an appropriate `make` command for your chip type, as illustrated below.

   For example, for JN5169:

   ```
   make JENNIC_CHIP_FAMILY=JN516x JENNIC_CHIP=JN5169 clean all
   ```

   For example, for JN5179:

   ```
   make JENNIC_CHIP_FAMILY=JN517x JENNIC_CHIP=JN5179 clean all
   ```

   The binary file will be created in the **Build** directory, the resulting filename indicating the chip type (e.g. **5179**) for which the application was built.

4. Load the resulting binary file into the board. You can do this from the command line using the JN51xx Production Flash Programmer, as described in Section 5.1.

8.3.2.2 Using the IDE (BeyondStudio or LPCXpresso)

This section describes how to use the IDE to build the demonstration application.

To build the application and load it into JN516x/7x boards, follow the instructions below:

1. Ensure that the project directory is located in

   <IDE installation root>\workspace

2. Start the IDE and import the relevant project as follows:

   a) In the IDE, 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, only select the project file appropriate for the chip family and IDE you are using, and click **Finish**.

3. Build an application. To do this, ensure that the project is highlighted in the left panel of the IDE 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.

4. Load the resulting binary files into the board. You can do this using the integrated Flash programmer, as described in the *User Guide for the IDE that you are using.*

9 Related Documents

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

- ZigBee 3.0 Stack User Guide [JN-UG-3113]
- ZigBee Device User Guide [JN-UG-3114]
- ZigBee Cluster Library (for ZigBee 3.0) User Guide [JN-UG-3115]
- ZigBee Green Power (for ZigBee 3.0) User Guide [JN-UG-3119]
- JN51xx Core Utilities User Guide [JN-UG-3116]
- BeyondStudio for NXP Installation and User Guide [JN-UG-3098]
- JN517x LPCXpresso User Guide [JN-UG-3109]
- JN51xx Production Flash Programmer User Guide [JN-UG-3099]

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

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