Application Note: JN-AN-1166

Smart Lamp Drivers

This Application Note provides and describes a set of NXP software drivers for use with 'smart lamps'. These software drivers are intended to be used in applications that run on NXP's range of JN51xx wireless microcontrollers, allowing a JN51xx application to interact with the hardware driver for a lamp. The lamp can then be incorporated in a wireless network and remotely controlled via radio links. A simple example application is also provided, the Smart Lamp Driver Test Tool, which allows lamps to be controlled from a wireless remote control unit (but without requiring them to join a wireless network).

1 Overview

This Application Note is concerned with a 'smart lamp' which can be controlled via a radio link from a wireless network device, such as a remote control unit or an IP gateway/router (it may also be controlled from a pre-existing wired switch). This is illustrated in Figure 1 below.

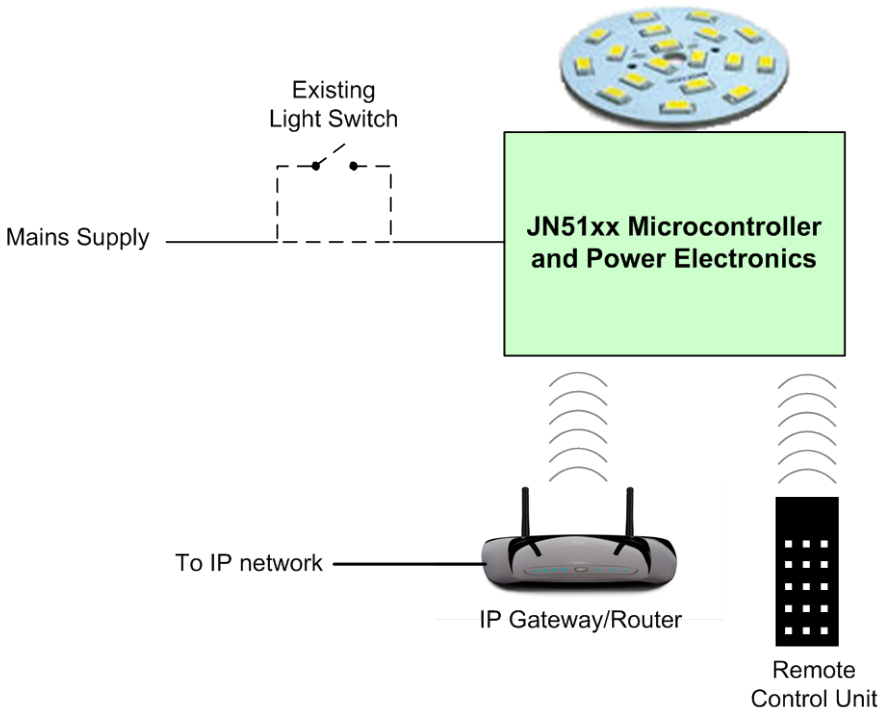


Figure 1: Smart Lamp Overview

The smart lamp contains an NXP JN51xx wireless microcontroller to handle control commands received over the radio. This device is connected to a lamp driver IC which controls the Solid State Lamp (SSL).

This Application Note provides software drivers that can be used by an application on the JN51xx device to interact with the lamp driver IC. More details of the hardware and software architectures are provided in Section 4.

2 Package Contents

This Application Note package includes the following software components:

- Software drivers (supplied as the shared header file **DriverBulb.h** and individual C source files) for the supported lamp hardware drivers (ICs):

Hardware Driver	Features	Comment
DriverBulb_DR1175	-	Supports Mono, CCTW and RGB on the DR1174 with DR1175 Lighting/Sensor Expansion Board
DriverBulb_DR1190	Asynchronous	Supports SSL2108 LED Driver
DriverBulb_DR1192	Synchronous	Supports SSL2108 LED Driver
DriverBulb_DR1221	Uses DR1192 with two PWM channels	Supports CCTW. Colour Temperature PWM is a percentage of the Level PWM period
DriverBulb_DR1223	RGBW	
DriverBulb_DR1221_DIMIC	Asynchronous	Supports SSL52xx LED Driver

- Smart Lamp Driver Test Tool (provided as two applications – see Section 5.1).

The software drivers provide C functions that can be used in an application designed to run on an NXP JN51xx microcontroller. The drivers are used in the following Application Notes:

Application Note	Description
JN-AN-1171	ZigBee Light Link Solution Application Note
JN-AN-1189	ZigBee Home Automation Demonstration Application Note
JN-AN-1162	JenNet-IP Smart Home Application Note

The principles of the lamp drivers are described in Section 4 and use of the Smart Lamp Driver Test Tool is described in Section 5.

3 Abbreviations and Terminology

The following abbreviations and terms are used in this Application Note:

CCTW	Colour Changeable Tunable White (white bulb with temperature control)
SSL	Solid State Light (or LED)
SSB	Small Signal Board (contains the JN51xx device)
LSB	Large Signal Board (contains the power electronics)
LUT	Look-Up Table
PWM	Pulse Width Modulation
VBUS	High-voltage supply of the SSL driver IC after rectification and buffering
VDD	Low-voltage supply of the SSL driver IC
VADC	Threshold level for the ADC for enabling/disabling the lamp in dimmer-compatible solutions
Level	Brightness as received/calculated from brightness control commands, before any quadratic correction

4 Smart Lamp Drivers

A smart lamp contains two distinct circuit boards:

- **Large Signal Board (LSB):** This board directly controls the AC supply that powers the lamp. It contains the hardware driver IC for the particular lamp. The software provided with this Application Note can be used to interact with the following NXP lamp driver ICs: SSL2108, UBA2027, UBA3070 (see Section 1).
- **Small Signal Board (SSB):** This board employs only DC signals and contains the circuitry for the wireless operation (control) of the lamp. As such, the JN51xx wireless microcontroller is located on this board.

Thus, the JN51xx device on the SSB must interface to the lamp driver IC on the LSB in order to control the operation of the lamp (according to over-air commands received by the JN51xx device). The application on the JN51xx device incorporates the appropriate software driver in order to interact with the lamp driver IC. This is illustrated in Figure 2 below.

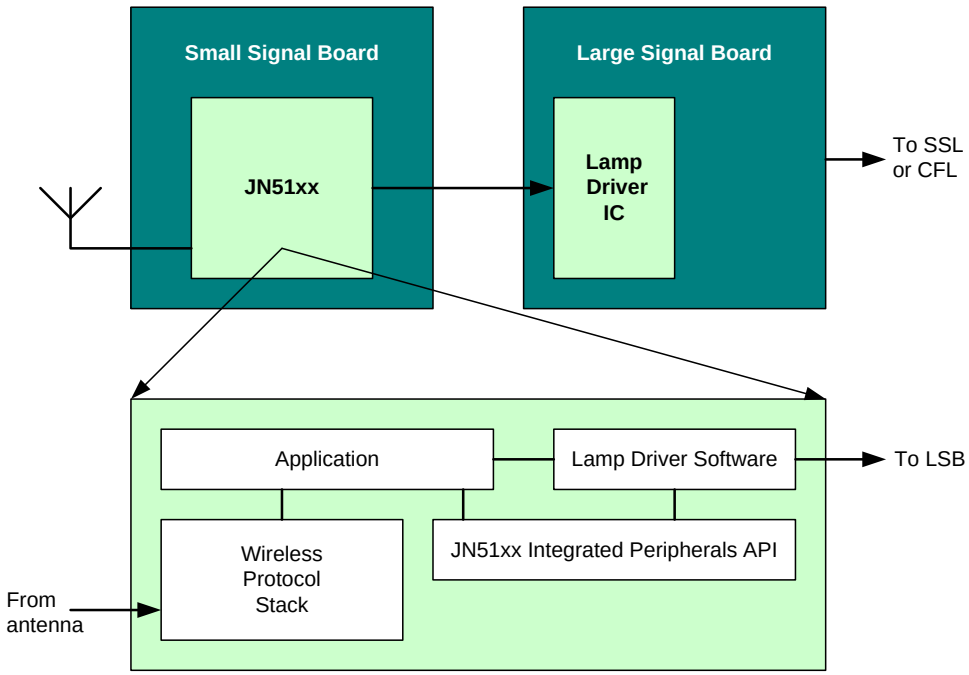


Figure 2: Hardware and Software Architecture

The purpose of the software driver is to provide the application with a standard interface to the lamp driver IC. Each of the supplied software drivers employs the same set of C functions for driver operations (and uses functions of the JN51xx Integrated Peripherals API to control the hardware interface between the JN51xx device and the lamp driver IC). The main operations of a software driver are on/off and converting the desired light level to a PWM duty-cycle. In addition, it provides a look-up table for brightness correction, needed to compensate for non-linearities in the lamp characteristics.

The sub-sections below provide more detailed driver information to help you to integrate one of the supplied software drivers into your wireless lamp application for a JN51xx device.

4.1 Control Operations

This section describes the main control operations of a lamp software driver. These operations fall into the following two areas:

- On/Off control – see Section 4.1.1
- Brightness control – see Section 4.1.2

4.1.1 On/Off Control

The on/off control is implemented using a signal named ON. Its basic purpose is to enable the VCC supply for the lamp driver IC on the LSB. The VCC supply is delivered to the IC from the SSB and is typically in the range 14-18V.

Note 1: The VCC supply powers the logic of the lamp driver IC but does not provide the power for the lamp. In standby mode, the VCC supply is disabled in order to save energy.

Note 2: The ON signal control is not used by all lamp drivers – it is used by all CFL drivers but by only some SSL drivers.

When the ON signal is being used, the lamp switches on when both:

- The ON signal has been asserted
- The PWM duty cycle has been set above PWM_MIN for the LED driver

When the ON signal is not being used, the on/off control of the lamp is performed solely via the PWM signal, as described in Section 4.1.1.4.

The sub-sections below describe different aspects of on/off control that may need to be taken into account when using one of the lamp drivers.

4.1.1.1 Power-up and Reset

Once mains power has been applied to a smart lamp, the lamp will boot. After booting, the application in the lamp should return the lamp to the ‘on ‘ or ‘off’ state that it was in before it was last powered down – this information will come from saved context data. The way that the software drivers handle this is described below.

During booting, the JN51xx device will start in the RESET state in which it keeps all its DIOs in soft pull-up - this means that there is a 50kΩ resistor between VCC and each DIO pin. As a result, the ON signal will be pulled high and VCC will be enabled, and the lamp will be immediately put in the ‘on’ state.

After booting, the application must determine from context data whether to return the lamp to the on or off state. Some time will elapse between the moment that the mains power is applied (starting the power-up) and the moment that the application determines the required state of the lamp.

This provisional ‘on’ state has the following effects:

- When the final lamp state is to be ‘on’, the switch-on time is minimised since the lamp has already been put in the ‘on’ state
- When the final light state is to be ‘off’, an initial flash of light may occur before the lamp is put in the ‘off’ state (however, power-up without the intention to switch to the ‘on’ state is rare)

4.1.1.2 VBUS Sensing

In a dimmer-compatible solution, the brightness level of the lamp is determined by the VBUS supply voltage of the lamp driver IC. For low dimmer settings, this voltage may be too low to maintain a stable light output from the lamp. The ADC on the JN51xx device is used to periodically sample this voltage for monitoring purposes. In order to prevent a flickering light output, the ADC output is compared with two voltage thresholds (Vth_on and Vth_off) and, consequently, the ON signal is set as follows:

- If ON is high and the ADC output falls below Vth_off then ON is set to low
- If ON is low and the ADC output rises above Vth_on then ON is set to high

The ADC sampling rate should be between 2 and 4 times per second (more frequent sampling brings no benefit).

During power-up (as described in Section 4.1.1.1), in order to prevent an initial flash of light when the final lamp state is to be on and to give the VBUS sensing circuit time to stabilise, the ADC sampling must start 2 seconds after power-up.

In products that are not dimmer-compatible, the ADC performs a different function.

4.1.1.3 Bleeder Control

In order to achieve a stable light output when used in with dimmable lamp, the SSL driver may need to always draw a minimum of current throughout a 50/60Hz mains cycle. This is in addition to the need to detect that the VBUS voltage is high enough, as described in Section 4.1.1.2.

4.1.1.4 Switch-off

For a switch-off initiated by a remote command, the ON signal should be pulled LOW and PWM duty-cycle must be set to its minimum value (respecting the polarity). When present in the design, the BLEEDER signal must be de-asserted.

For a switch-off due to a mains interruption, nothing useful can be done because the driver IC will stop sooner than the software can take any action (the light will switch off quickly).

4.1.2 Brightness Control

The sub-sections below describe two aspects of brightness control that may need to be taken into account when using one of the lamp drivers.

4.1.2.1 PWM Substitute for On/Off Control

As mentioned earlier, no explicit ON signal is used in some lamp drivers. In these cases, the PWM signal replaces the ON signal. At the same time, the PWM signal determines the brightness level. The lamp is switched off by setting the minimum brightness (keep PWM low for positive polarity; keep PWM high for negative logic).

4.1.2.2 Filtered PWM Brightness Control

Some drivers require a DC control voltage to set the brightness level. In this case, the PWM signal from the JN51xx DIO is first low-pass filtered, typically using a roll-off frequency in the range 4-10Hz. The polarity for all DC-controlled drivers is positive.

4.1.3 Mains Synchronization

The DR1192 and DR1221 (not DIMIC) drivers employ a mains synchronisation scheme. A potential divider provides a control voltage (half-wave rectified sine) on the comparator's positive input. When the voltage falls below Vref, an interrupt is generated. The timing difference between successive interrupts is measured by a hardware counter and is used to calculate the value of the PWM timer period. In effect, a frequency-multiplier PLL is implemented whereby the PWM frequency is eight times the mains frequency.

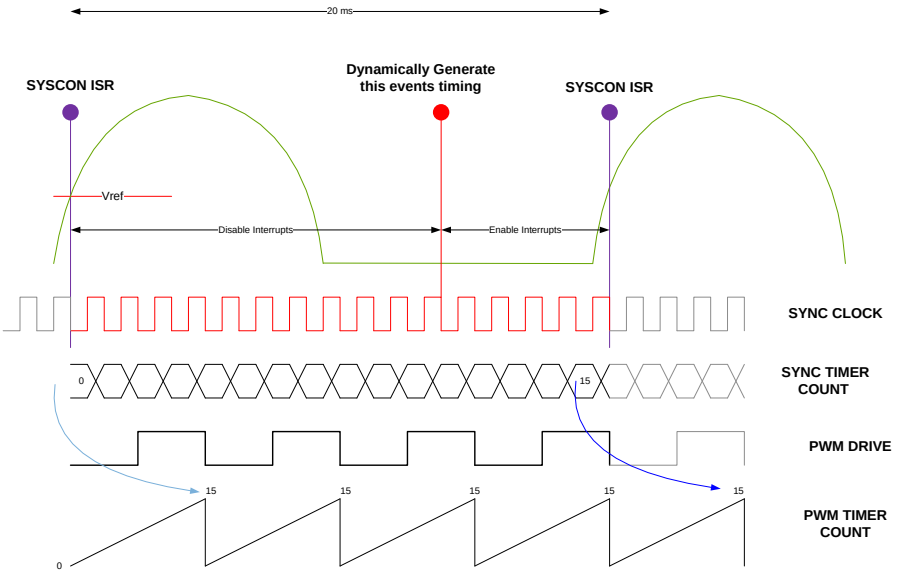


Figure 3: Synchronisation Technique

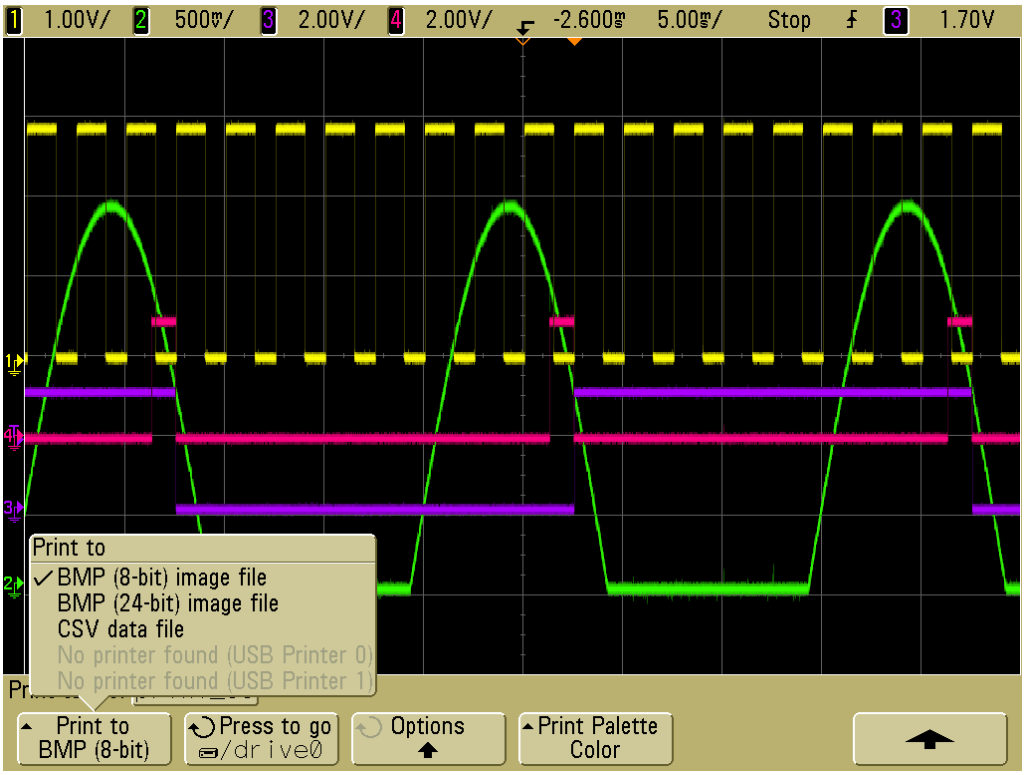


Figure 4: Synchronisation Performance

4.1.4 Anti-Flicker

The supply load on the JN51xx device is variable as the device changes between Receive and Transmit (i.e. rebroadcasting). In certain SSB/LSB system configurations, these current drops cause flicker. To compensate, four resistors are driven from outputs which make the load constant. The following two illustrations show the current profile before and after the implementation of the anti-flicker function.

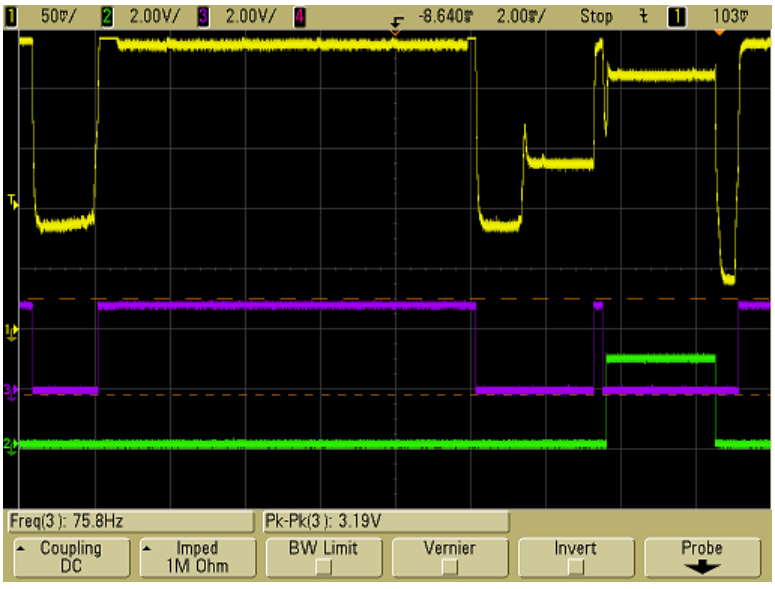


Figure 5: Current Profile without Anti-Flicker (Yellow Trace)

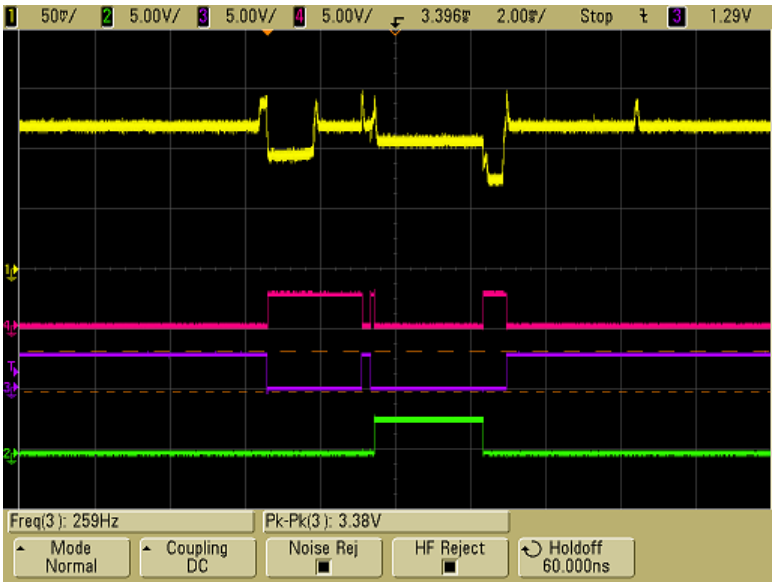


Figure 6: Current Profile with Anti-Flicker

The anti-flicker system runs the bulb in radio high-power mode so that the Tx/Rx state is available on DIO2 and DIO3. These pins are configured to generate interrupts, allowing a 'Software NOR' logic function to be realised. The above oscilloscope trace illustrates this; the red trace is high only when both the purple and green traces are low. The radio is off during this period, resulting in a current drop. However, by driving 3mA on each of four DIOs, the drop is almost cancelled out, thereby eliminating rebroadcast flicker.

4.1.5 Level Correction

Some drivers contain a non-linear conversion function. This is to compensate for the non-linear nature of the eye and the LEDs. In the DR1192 and DIMIC drivers, this can be ignored by defining LINEAR_MODE=TRUE. The DIMIC device can support the correction curve in hardware and so the software correction may not be required.

4.2 Driver Software Functions

The following functions are available in all the lamp software drivers and can be used by the JN51xx application to interface to the lamp driver IC:

- **DriverBulb_vInit()**
- **DriverBulb_vOn()**
- **DriverBulb_vOff()**
- **DriverBulb_vSetLevel()**
- **DriverBulb_bReady()**
- **DriverBulb_bOn()**
- **DriverBulb_i16Analogue()**
- **DriverBulb_vTick()**
- **DriverBulb_bFailed()**

The above functions are provided in the shared header file **DriverBulb.h**, which must be included in the JN51xx application. The functions are described in the sub-sections below and their relationship to the hardware interface is illustrated in Figure 7.

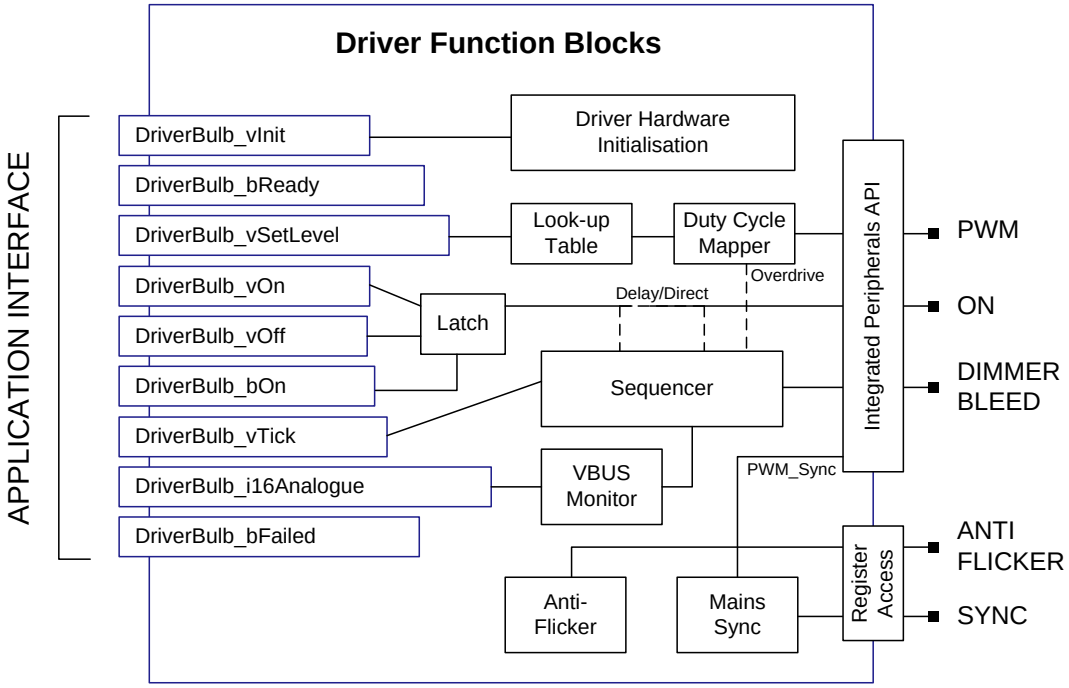


Figure 7: Software and Hardware Interfaces

The above diagram shows all possible driver components. Dependent on the particular driver, some or all of the components are used to control the lamp. For instance, the synchronous SSL driver does not use VBUS monitoring but does use the ‘mains sync’ functionality.

4.2.1 DriverBulb_vInit

void DriverBulb_vInit(void);

This function initialises the DIOs used by the driver and switches on the lamp. The JN51xx ADC is enabled and the function waits until the bus voltage is sufficiently high before switching on the lamp.

4.2.2 DriverBulb_vOn

void DriverBulb_vOn(void);

This function switches on the lamp. The way that the function does this is dependent on the LSB and driver implementation. It will do either of the following, depending on whether the ON signal is used:

- If the ON signal is used, it will assert the DIO
- Otherwise, it will restore the previous PWM duty-cycle

4.2.3 DriverBulb_vOff

void DriverBulb_vOff(void);

This function switches off the lamp. The way that the function does this is dependent on the LSB and driver implementation. It will do either of the following, depending on whether the ON signal is used:

- If the ON signal is used, it will de-assert the DIO
- Otherwise, it will set the PWM duty-cycle to 0% (positive PWM logic) or 100% (negative PWM logic), which is a special value representing the ‘off’ state

4.2.4 DriverBulb_vSetLevel

void DriverBulb_vSetLevel(uint8 *u8Level***);**

This function sets the brightness level of the lamp to the specified value (*u8Level*) in the range 0-255, which is translated into a PWM duty-cycle:

- Some drivers divide the specified value by two and then index into a quadratic look-up table which remaps the value
- Other drivers have 128 linear steps between the #defines PWM_MIN and PWM_MAX, and maps the specified level to one of these values

In each case, the new value is scaled to a duty-cycle percentage and written to a JN51xx timer.

4.2.5 DriverBulb_bReady

bool_t DriverBulb_bReady(void);

This function determines whether the supply voltage is sufficiently high to allow the lamp to be switched on. It can only be used with dimmer-compatible drivers (e.g. SSL2108) and can be optionally used by the application to determine whether the lamp is ready to be switched on.

The function returns TRUE if the bus voltage is greater than the pre-defined value or FALSE otherwise. The non-dimmer-compatible drivers always return TRUE.

4.2.6 DriverBulb_bOn

bool_t DriverBulb_bOn(void);

This function determines whether the lamp is 'on'. It can be optionally used by the application to check whether an 'on' command was successful.

4.2.7 DriverBulb_i16Analogue

int16 DriverBulb_i16Analogue(uint8 *u8Adc*, **uint16** *u16AdcRead*);

This function is used by the application to provide an ADC value to the driver and should be called every 250ms. The supplied value represents the VBUS voltage level. The ADC value (*u16AdcRead*) and the identifier of the relevant ADC input (*u8Adc*) must be specified. The ADC input must be one of the following (depending on the driver):

- E_AHI_ADC_SRC_ADC_1 (ADC1 input)
- E_AHI_ADC_SRC_ADC_4 (ADC4 input)

The function returns the measured voltage in Volts.

4.2.8 DriverBulb_vTick

void DriverBulb_vTick(void);

This function is used by the application to provide a periodic 10ms tick to any driver that requires a background task to be performed. The function must be called for all drivers but may not be used.

4.2.9 DriverBulb_bFailed

```
bool_t   DriverBulb_bFailed(void);
```

This function is used by the application to determine if a lamp has failed. This functionality is not currently implemented and the function always returns FALSE.

5 Smart Lamp Driver Test Tool

The Smart Lamp Driver Test Tool can be used to test the supplied software drivers by controlling smart lamps from a wireless remote control unit without forming a wireless network. The required hardware is as follows:

- From the NXP JN516x-EK001 Evaluation Kit, a Carrier Board (DR1174) fitted with an LCD Expansion Board (DR1215) is used as the remote control unit. The Carrier Board is also fitted with a JN5168 module.
- A smart lamp can be any lamp which incorporates a JN516x device, and which requires one of the supplied NXP lamp drivers:.

The remote control unit is used to control all lamps within radio range of the unit. The user can switch the lamps on/off as well as vary their brightness up/down. If testing only one lamp, the temperature of the lamp is displayed on the LCD screen of the remote control unit.

5.1 Software Overview and Programming

The Smart Lamp Driver Test Tool is supplied as pre-built binary files, which must be programmed into Flash memory using the BeyondStudio for NXP Installation and User Guide (JN-UG-3098).

5.1.1 Remote Control Unit Software

The file **Smart_Remote_JN5168_DEVKIT4.bin** must be programmed into the Flash memory of a JN5168 device on the Carrier Board of the remote control unit.

5.1.2 Lamp Software

There are several binary files for the lamps and the correct one must be used, according to the relevant microcontroller and driver types. The lamp binary filenames are prefixed with **Smart_Lamp** followed by the microcontroller type and then the driver type - for example:

Smart_Lamp_JN5168_DR1175.bin

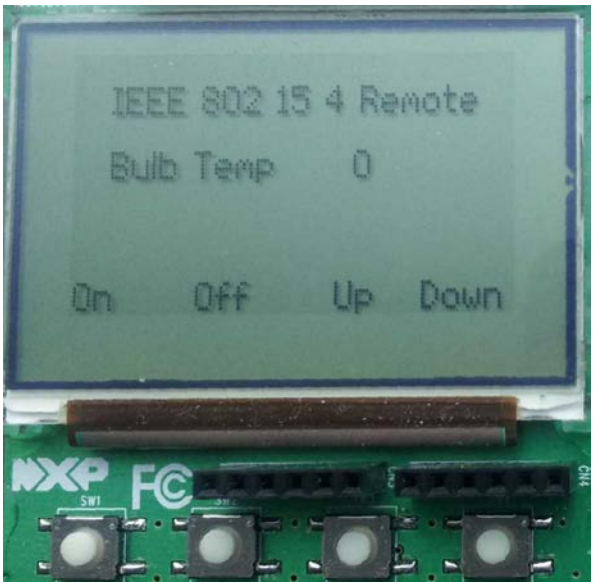
The above binary file is built for the JN5168 microcontroller and for the DR1175 platform.

5.2 Using the Smart Lamp Driver Test Tool

To use the Smart Lamp Driver Test Tool:

1. First cycle the power to each lamp to restart the microcontroller. The lamp is now ready to receive commands from the remote control unit.
2. Now power up the remote control unit (e.g. fit batteries to the Carrier Board).

Once the remote control unit is up and running, the LCD screen will display labels that indicate the functionality of the four switches SW1-SW4 under the screen. This is illustrated in the photograph below.



The four switches have the following functionality:

- SW1 switches on all lamps within radio range
- SW2 switches off all lamps within radio range
- SW3 increases the brightness of all lamps within radio range until the button is released or the maximum brightness is achieved
- SW4 decreases the brightness of all lamps within radio range until the button is released or the minimum brightness is achieved

The LCD screen also displays bulb (lamp) temperature, which is refreshed every 5 seconds. If controlling multiple lamps, it is not possible to know to which lamp this temperature corresponds. Each lamp being tested will report its temperature but only one temperature will be displayed (and not identified).

Therefore, if controlling multiple lamps, you may wish to remove the temperature feature and re-build the applications. To do this, comment out or delete the following line in the remote control and lamp makefiles:

```
CFLAGS+=-DDEBUG_TCL
```

You can re-build the applications as described in Section 5.5.

5.3 Development Resources and Installation

In order to develop/modify the software provided with this Application Note, build it and load it into a JN516x device, you will need the JN516x software tools and libraries described in this section. This developer's software is available free-of-charge via the NXP Wireless Connectivity TechZone and is provided in two installers.

BeyondStudio for NXP (JN-SW-4141)

This installer contains the toolchain that you will use in creating an application, including:

- 'Beyond Studio for NXP' IDE (Integrated Development Environment)
- Integrated JN51xx compiler
- Integrated JN516x Flash Programmer

JN516x Software Developer's Kit (JN-SW-41xx)

This installer includes the following software and APIs:

- Stack software for relevant wireless network protocol (e.g. ZigBee PRO)
- Application Programming Interfaces (APIs) for relevant wireless network protocol
- JN516x Integrated Peripherals API

This application can be used with any of the NXP JN516x SDKs with a part number of the form JN-SW-41xx. The makefiles in the project relate to the JN516x ZigBee Home Automation/Light Link SDK (JN-SW-4168).

For full details of the toolchain and installation instructions for the toolchain and SDK, refer to the *BeyondStudio for NXP Installation and User Guide (JN-UG-3098)*.

5.4 Compatibility

The software provided with this Application Note is intended to be used with the following evaluation kits and SDK (Software Developer's Kit) versions:

Product Type	Part Number	Version
Evaluation Kits	JN516x-EK001	-
SDK Libraries	JN-SW-4168 *or* JN-SW-4163	v1270 v1168
SDK Toolchain	JN-SW-4141	v1217

5.5 Building and Loading the Application

A source (**.c**) file is also supplied corresponding to each of the application binaries for the Smart Lamp Driver Test Tool. This allows the applications to be modified and re-built using the 'BeyondStudio for NXP' IDE or the supplied makefiles.

- The remote control unit application can be built for the JN5168 device only
- The lamp application is built for the JN5168 device by default

In order to build the supplied software, the application's folder must be placed in the **workspace** folder of the SDK installation:

C:\NXP\bstudio_nxp\workspace

The **workspace** folder is automatically created when you first start BeyondStudio.

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 5.5.1.
- To build using BeyondStudio for NXP, refer to Section 5.5.2.

5.5.1 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 JN516x board, follow the instructions below:

1. Ensure that the project directory is located in

<BeyondStudio for NXP installation root>\workspace

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

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

For JN5168:

At the command prompt, enter:

```
make clean all
```

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

For JN5164:

At the command prompt, enter:

```
make JENNIC_CHIP=JN5164 clean all
```

For JN5161:

At the command prompt, enter:

```
make JENNIC_CHIP=JN5161 clean all
```

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

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

5.5.2 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 JN516x boards, 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. 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.

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

Revision History

Version	Notes
1.0	First release
1.1	Title of Application Note changed and lamp driver information added, including synchronisation and anti-flicker features
1.2	Updated for JN516x SDKs based on 'BeyondStudio for NXP' and latest software organisation

Important Notice

NXP Semiconductors

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

www.nxp.com