APPLICATION NOTE

Atmel AT06482: Real Color ZLL LED Light Bulb with ATmega256RFR2 - Software User's Guide

Atmel AVR 8-bit Microcontroller

Description

This application note mainly describes the firmware architecture and the application programming interfaces (API) of Real Color ZLL LED Light Bulb with Atmel® ATmega256RFR2 reference design (hereafter the LED bulb).

Features

- ATmega256RFR2 Microcontroller - Wireless SoC with integrated IEEE® 802.15.4 transceiver
- Atmel BitCloud® Profile Suite - ZigBee® Light Link (ZLL) stack
- Production-ready Real Color LED light Bulb
- LED Color Mixing Algorithm

Figure 1. Real Color ZLL LED Light Bulb with ATmega256RFR2.

For easy usage, the LED bulb can be controlled via color scene controller mentioned in Atmel AVR2060: BitCloud ZigBee Light Link Quick Start Guide. Chapter 9 Getting Started Guide gives details about the setup and operation of the LED bulb.

For LED bulb hardware design details, refer to Atmel AT06412: Real Color ZLL LED Light Bulb with ATMEGA256RFR2 - Hardware User Guide.

For this reference design, the hardware design files (schematic, BOM, and PCB Gerber) and software source code can be downloaded from Atmel website. The provided hardware documentation can be used with no limitations to manufacture the reference hardware solution for the design.

Table of Contents

1. Overview

The Real Color ZLL LED Light Bulb with ATmega256RFR2 is a ZLL color light defined in ZigBee Light Link Profile. As a wireless solution provider, Atmel also has reference design of gateway and remote control to operate with ZLL lights and makes it easy to run a complete ZLL system.

Figure 1-1. ZigBee Light Link

The LED bulb is based on Atmel ATmega256RFR2 microcontroller - an IEEE 802.15.4 compliant single chip combines an industry-leading AVR® microcontroller and best-in-class 2.4GHz RF transceiver. The firmware is based on Atmel BitCloud ZigBee Light Link software development kit (SDK). The following ZLL clusters are supported by this LED bulb (ZLL color light).

- Basic cluster
- Identify cluster
- Groups cluster
- Scenes cluster
- On/off cluster
- Level control cluster
- Color control cluster

Besides the ZLL functions, it also provides the following features:

- LED color mixing algorithm to generate different color output from the RGB LEDs inside the LED bulb
- Power indicator
- On-board NTC sensor to detect LED bulb temperature
- Read MAC address from on-chip EEPROM

To easily download firmware and modify MAC address, separate PC SW is provided to write on-chip EEPROM, flash, FUSE bits and Lock bits.

2. Development Tools

To download or debug the firmware released together with this document, the following development toolchains are needed:

- IAR Embedded Workbench® for AVR Version: 6.11 (with IAR™ C/C++ Compiler for AVR v6.11.1.50453) - or above
- Programming and debugging tool: Atmel JTAGICE3 (recommended) or Atmel AVR JTAGICE mkII

Note: A miniature JTAG header is used on LED bulb. If using JTAGICE mkII, a 100mil to 50mil JTAG adapter from AVR ONE! (Standoff adaptor nr.2, ref: A08-0253/B) is needed.

3. Suggested Reading

To better understanding this reference design and its firmware, we suggest reading the following application notes to have a basic knowledge of Atmel BitCloud ZigBee Light Link (ZLL) stack and Atmel BitCloud ZigBee stack.

- Atmel AVR2060: BitCloud ZigBee Light Link Quick Start Guide
- Atmel AVR2061: BitCloud ZigBee Light Link Developer Guide
- Atmel AVR2056: BitCloud Profile Suite Developer Guide
- Atmel AVR2050: Atmel BitCloud Developer Guide

To know more about ZLL, please read the ZigBee Light Link Profile Specification from ZigBee Alliance

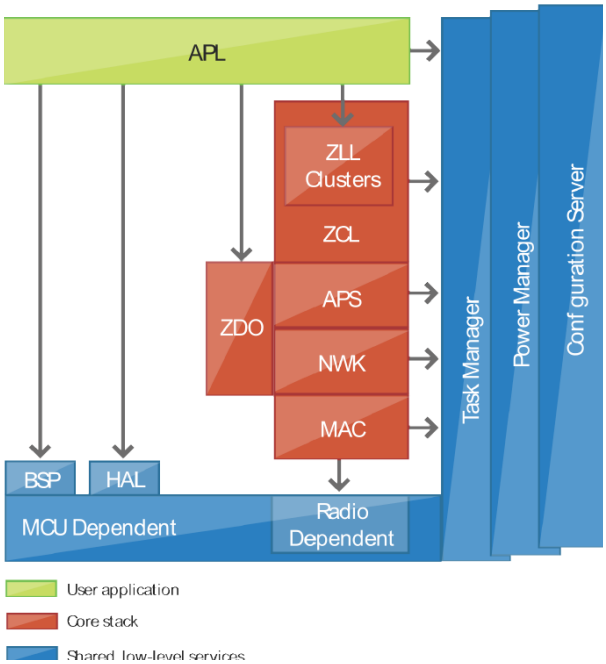
Note: BitCloud Profile Suite ZLL for megaRF v1.9.5 is used at time of writing this application note.

4. Firmware Architecture

The LED bulb firmware is based on Atmel BitCloud ZigBee Light Link SDK. The features beyond ZigBee Light Link Profile are implemented in application layer (APL).

The firmware architecture of BitCloud ZigBee Light Link stack is shown in Figure 4-1.

Figure 4-1. BitCloud ZigBee Light Link Architecture



4.1 Atmel BitCloud ZigBee Light Link stack

ZigBee Light Link is designed to set a global standard for interoperable and very easy-to-use consumer lighting and control products, allowing consumers to gain wireless control over all their LED fixtures, light bulbs, timers, remotes and switches. This standard let consumers change lighting remotely to reflect ambiance, task or season, all while managing energy use and making their homes greener.

Atmel BitCloud ZigBee Light Link stack includes implementation of the ZigBee Pro protocol and ZigBee Cluster Library with a few enhancements. In this application, BitCloud Profile Suite ZLL for megaRF v1.9.5 is used.

For more details about Atmel BitCloud ZigBee Light Link stack, please visit BitCloud Profile Suite and download the latest BitCloud ZigBee Light Link stack.

5. Inside the LED Bulb Reference Design

In this chapter, the overall LED bulb firmware structure is explained. There is also a separate section talking about the LED color mixing algorithm.

5.1 LED Bulb Application Layer

- The LED bulb application layer follows the same structure of Bit Cloud stack development suggestions in Chapter 3 User application programming of Atmel AVR2050: Atmel BitCloud Developer Guide

To know more about ZLL, please read the ZigBee Light Link Profile Specification from ZigBee Alliance.

The main() function of the LED bulb is shown below:

```
int main(void)
{
  SYS_SysInit();
  /* The main infinite operating loop */
  while (1)
  {
    SYS_RunTask();
  }
```

As said before, the application layer takes care about all the user implementations. The user implementation can be placed in `APL_TaskHandler()`. It will be called from function `SYS_RunTask()` as soon as possible.

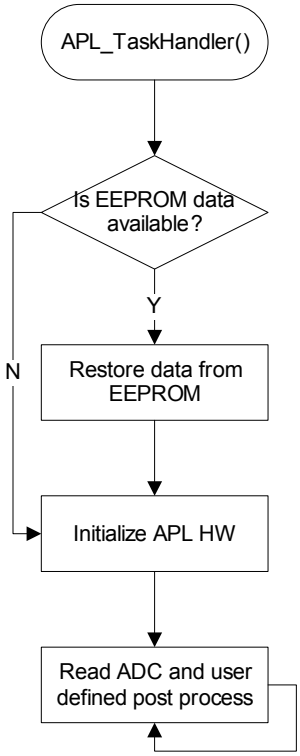
In LED bulb application, two things are done in `APL_TaskHandler()`.

- The first, after power on reset, it initializes HW used by application layer. The HW used by ZLL stack is initialized in `SYS_SysInit()` before this step. It also tries to restore ZLL stack parameters from on-chip EEPROM if they are available
- The second, a NTC sensor is read and user could define the operation if the temperature is above preset threshold

The other application layer implementations are done in callback functions. They are invoked from ZLL stack lower layer when specific events are triggered. For example, when LED bulb receives command to turn off the light, the `offInd()` will be called from lower layer. The user implementation of turn off the light should be place in this callback function.

- For more details about BitCloud Profile and ZLL application development, refer to Atmel AVR2056: BitCloud Profile Suite Developer Guide and Atmel AVR2061: BitCloud ZigBee Light Link Developer Guide

Figure 5-1. Application Task Handler Flow

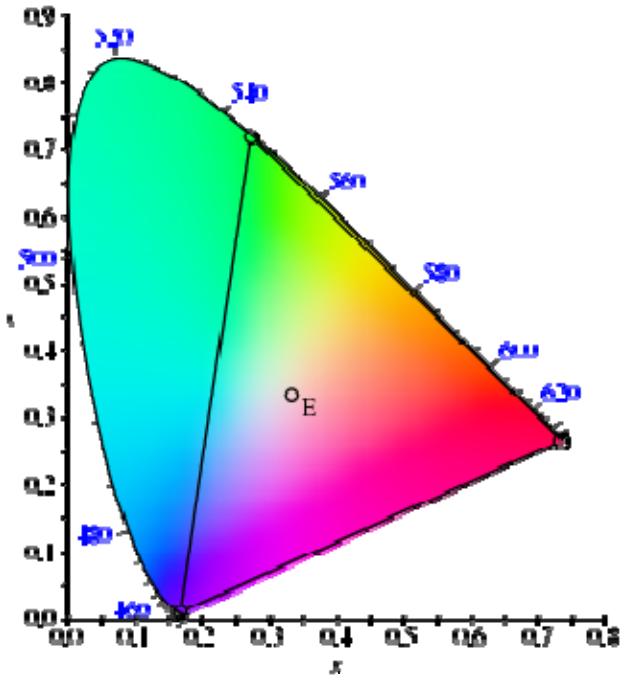


5.2 LED Color Mixing Algorithm

Thanks to BitCloud ZigBee Light Link stack, most of the ZigBee operations are done in stack lower layer. The light engineers just focus on light development itself. One of the most important features of color light is the color mixing – the ability to generate different colors and give end customer the best color experience. From engineering side, we also need to consider the LED output power, efficiency and the color space etc.

According to the CIE xy chromaticity diagram and the CIE xyY color space definition, all colors that can be formed by mixing three sources are found inside the triangle formed by the source points on the chromaticity diagram. For example, the color point E can be generated by mixing light sources located at the triangle vertices in Figure 5-2.

Figure 5-2. The CIE 1931 xy Chromaticity Diagram



In the LED bulb, there are three different kinds of color emitters. They are red (R), green (G), and blue (B) LEDs. Based on the theory above, the color space of the LED bulb is defined by the RGB LEDs used. And these LEDs are controlled by MCU PWM output. By change the RGB PWM duty cycle, we can get different colors inside the triangle formed by the RGB LEDs.

In ZLL, the color light should support x, y or HSV color mode as color input. In this LED bulb, x, y are also converted into HSV and then HSV is converted into R, G, B PWM duty cycle.

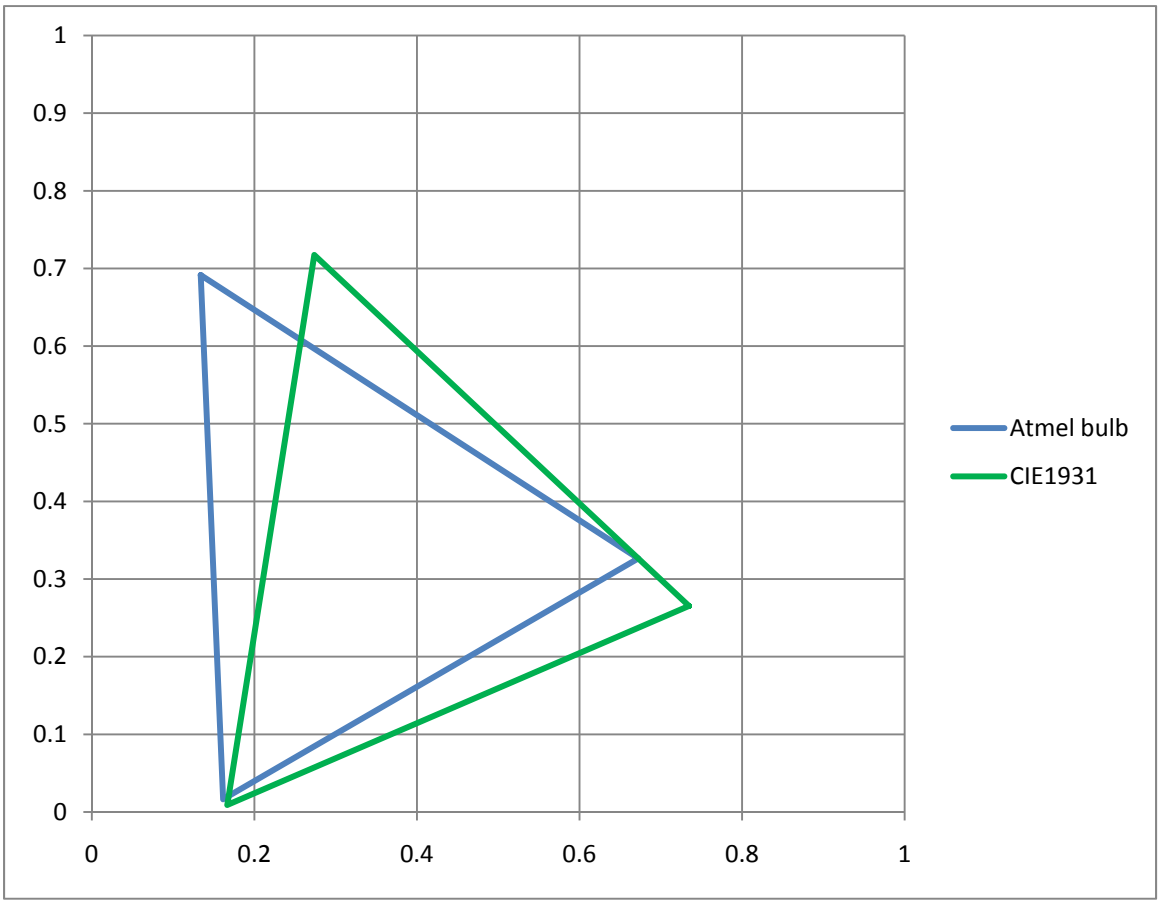
To get the correct PWM duty cycle, we also need take the following factors into consideration.

- The numbers of R, G, B LEDs
- The efficacy of different color LEDs
- The flux of different color LEDs

The other thing for a light is to generate white light which is more generally used. As mentioned above, as far as the white point is located inside the triangle, it can also be generated by mixing the RGB LEDs. However, depending on the application, different definitions of white are needed to give acceptable results. In this LED bulb, white point of illuminant E is used.

Figure 5-3 shows the color space of this LED bulb and the CIE1931 color space.

Figure 5-3. The Color Space of Atmel LED Bulb and CIE1931



6. Main API Introduction

The main APIs used in LED bulb are introduced in the following two parts: BitCloud ZigBee Light Link stack API and LED color mixing algorithm API.

6.1 BitCloud ZigBee Light Link Stack API

Some of Atmel BitCloud ZigBee Light Link stack API is similar to Atmel BitCloud ZigBee PRO stack API.

The similar APIs are as below.

- `SYS_SysInit()`
  It initializes the stack HAL layer.
- `SYS_RunTask()`
  This function is called from main() function to process tasks. It's the core API of Atmel BitCloud ZigBee Light Link stack. Both application layer and ZLL stack lower layer task handlers are called in this API as soon as possible.
- APL_TaskHandler()
  It's the application layer task handler.

It also has dedicated APIs for ZLL. Usually these APIs are identified by ZLL in the function names.

- `init()`
  It's the application layer initialization function. MAC address is read and set into stack in this function. LED PWM driver pins, ADC input pin and ZLL stack clusters are also initialized.
- `reborn()`
  Start inter-PAN operation or join last network depending on factory new status.
- `interpan()`
  Prepare for inter-PAN communication by calling `ZCL_ZllInterPanStartReq()`.
- `ZCL_ZllIdentifyInd()`
  Notification of receiving a ZLL identify command. Usually it means touchlink is in progress.
- `ZCL_ZllStartInd()`
  Notification of ZLL start network command. Usually it means touchlink is completed and ZLL network should be restarted.

For the ZLL clusters, they have separate callback function for each cluster. For example, some of the color control clusters API.

- `moveToHueInd()`
  The callback to handle move to hue command in color control cluster.
- `moveHueInd()`
  The callback to handle move hue command in color control cluster.
- `moveToSaturationInd()`
  The callback to handle move to saturation command in color control cluster.
- `moveSaturationInd()`
  The callback to handle move saturation command in color control cluster.

Note: The APIs described here are focusing on application layer. For complete API descriptions, refer to BitCloud_ZLL_API_Reference.chm from the link mentioned in Section 4.1.

6.2 LED Color Mixing Algorithm API

The color mixing algorithm used in LED bulb is implemented in rgb.c and rgb.h. The main purpose of this part is to implement and showcase the features of ZLL on a real LED bulb.

- `rgbUpdatePwmChannels()`

  It is the API converting HSV into RGB LED PWM output. First HSV is converted into rgb signals. And then rgb signals are compensated to output all the colors in the color space (triangle mentioned in Figure 5-3) formed by R, G, and B LEDs. At last, the compensated rgb signals are converted into PWM duty cycle and written into TC module registers. The PWM output will be generated by MCU TC module.
  Also the RGB LED PWM of white point (illuminant E) is calculated in this API.

- `rgbColorToHueSaturation()`

  It is the API converting x, y color mode to HSV color mode.

The following APIs are used to realize the LED color control by ZLL clusters.

- `APP_RgbMoveToHue()`

  The implementation of move to hue on RGB LEDs.

- `APP_RgbMoveToSaturation()`

  The implementation of move to saturation on RGB LEDs.

- `APP_RgbMoveToLevel()`

  The implementation of move to level on RGB LEDs.

- `APP_RgbMoveToColor()`

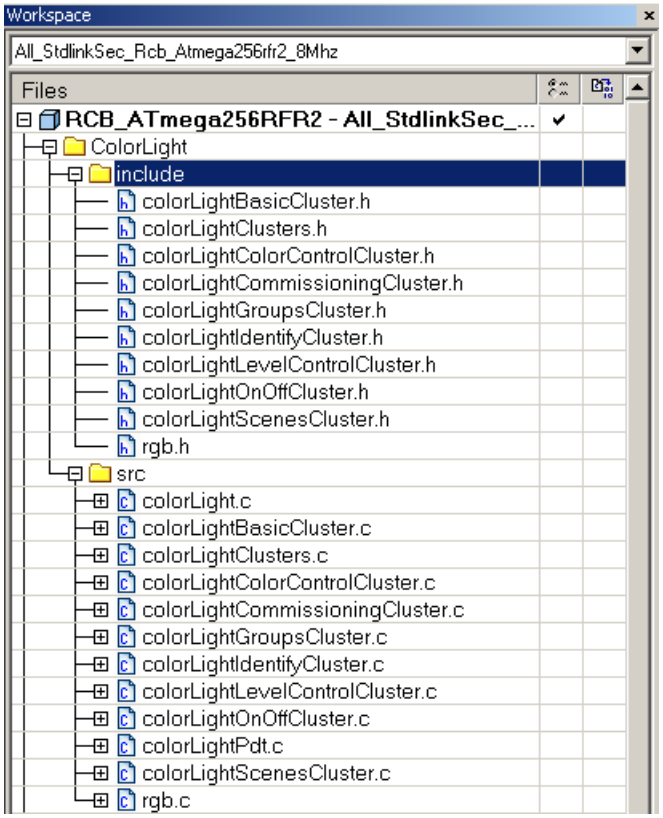
  The implementation of move to color on RGB LEDs.

There are also more functions to handle LED bulb color and level transitions. For more details, refer to the source files.

7. Software Package Content

The software package structure is the same as BitCloud ZigBee Light Link SDK. For details, refer to documents mentioned in Chapter 3. Mainly two files are added in the SDK to for this reference design. They are rgb.h and rgb.c shown in Figure 7-1:

Figure 7-1. The LED Bulb Application Source Files



8. Footprint

Figure 8-1 and Figure 8-2 illustrate the CODE and RAM spaces that each module used in the firmware of the LED bulb.

Figure 8-1. Real Color ZLL LED Light Bulb with ATmega256RFR2 CODE Footprint [KB]

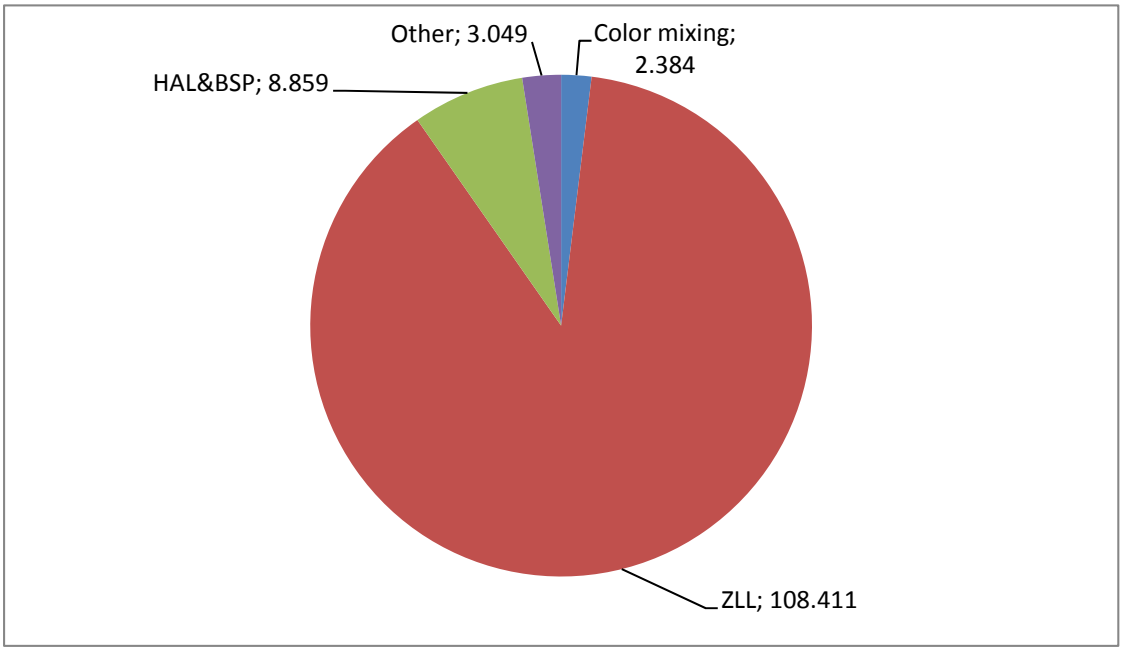
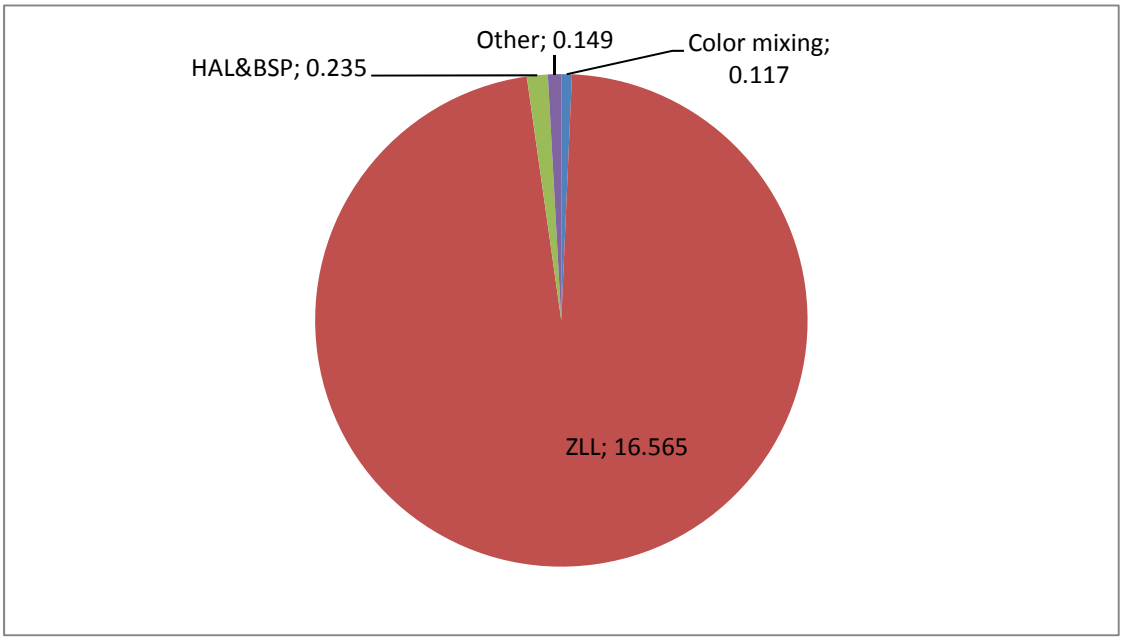


Figure 8-2. Real Color ZLL LED Light Bulb with ATmega256RFR2 RAM Footprint [KB]



9. Getting Started Guide

This chapter gives a step-by-step guide to set up a typical ZLL system and run the preprogrammed firmware. It includes LED bulbs and color scene controller.

9.1 Code Image

Along with this document, two hex codes are provided. One is for LED bulb (LEDBULB.hex) and the other is for color scene controller (ZLLDemo_ATmega128RFA1_Remote.hex). The color scene controller hex file is the same as the one in BitCloud ZigBee Light Link SDK folder: BitCloud_PS_MEGARF_ZLL_1_9_5\Evaluation Tools\ZLLDemo.

For safety, the LED bulbs shipped to customer are injected with glue. There is no chance to reprogramming the LED bulb in this case. In case of prototype development, the tools mentioned in Chapter 2 can be used.

9.2 System Setup

The LED bulb is designed to follow E27 standard lamp interface. It is easy to have it installed in an E27 lamp holder. For the electrical characteristics, refer to Atmel AT06412: Real Color ZLL LED Light Bulb with ATMEGA256RFR2 - Hardware User Guide.

The color scene controller is based on an RCB board attached to a Key Remote Control board. To know more about RCB board and Key Remote Control board, refer to AVR2104: RF4CE Remote Control Evaluation Kit - User Guide.

9.2.1 Default LED Bulb Status

After power on, the LED bulb will be reset to the following status in regardless of its previous working status.

- Brightness is set to max value
- Color mode is set up use Hue and Saturation
- Saturation is set to 0. The LED bulb will be in white color in this case
- Hue is set to 0 (Red). To show color on LED bulb, set the saturation to none-zero value

9.2.2 Setup Steps

1. Install LED bulb in E27 lamp holder and power on.
2. Insert two AAA batteries in RCB board battery holder and switch on color scene controller.
3. With color scene controller held close to LED bulb A, press and hold “PWR” button on the Key Remote Control board for at least 3s to initiate touchlink.
4. Wait until the touchlink completes successfully by holding “PWR” button.
5. Repeat touchlink to other LED bulbs. (For example, LED bulb B, C, D, …)
   If touchlink to all the LED bulbs are completed successfully, the LED bulbs can be controlled by color scene controller remotely. Below is the table of buttons for most frequency commands supported by LED bulbs. To know complete button operation of color scene controller, refer to Atmel AVR2060: BitCloud ZigBee Light Link Quick Start Guide.

Table 9-1. Buttons for Executing Most Frequent Commands for LED Bulb

Button(s)	Effect
PWR	Press and hold for three seconds to perform touch link
PWR + R	Press and hold for three seconds to reset device to factory new state
PWR + R+ + R-	Reset a color light to the factory new state. The target color light should be selected using the SEL button; otherwise all color lights present in the network will be reset to the factory new state
L+/L-	Color light’s on/off
Up/Down	Increase/decrease light level

Left/Right	Increase/decrease saturation
Colored buttons	Set the corresponding color using the following values: Red: hue = 60000 or x = 40000, y = 20000 Green: hue = 30000 or x = 10000, y = 40000 Yellow: hue = 15000 or x = 30000, y = 30000 Blue: hue = 45000 or x = 10000, y = 10000
SEL	Select the next bound device and requests it to identify itself. This allows sending unicast commands to a single device. Groupcast mode will be entered automatically after five seconds of inactivity.
1/2/3	Store Scene if pressed for more than three seconds and Recall scene if pressed for less than three seconds
7/8/9	Set minimum/middle/maximum light level

Appendix A. Additional Information

A.1 BitCloud ZigBee Light Link Stack Configuration

Table A-1 lists the BitCloud ZigBee Light Link stack configuration used in this reference design and this configuration can be modified in BitCloud ZigBee Light Link SDK folder Applications\ZLLDemo\configuration.h.

Table A-1. BitCloud ZigBee Light Link Stack Options

Option	Value	Description
APP_DEVICE_TYPE_COLOR_LIGHT	Enabled	Device type. Enable this macro define for ZLL light.
APP_SCAN_ON_STARTUP	0	Don't activate channel scanning on startup for FN Light devices
APP_ZLL_CHANNELS	11, 15, 20, 25	ZLL working channels
APP_ZLL_DEFAULT_WORKING_CHANNEL	11	ZLL default working channel

There are more configurations in the source file. Most of them are the same as used in BitCloud ZigBee PRO stack.

A.2 LED Color Mixing Configuration

Table A-2 lists the LED color mixing configuration in this reference design, and these configurations can be modified in Applications\ZLLDemo\ColorLight\include\rgb.h.

Table A-2. QTouch Library Options

Option	Value	Description
_CIE_RGB_	Enabled	Enable this macro define to use CIE rgb white point.
_LED_N_R_	2	Number of red LED used in LED bulb
_LED_N_G_	2	Number of green LED used in LED bulb
_LED_N_B_	1	Number of blue LED used in LED bulb
_FLUX_T_R_	72.0f	Red LED rated flux (Can be found in LED datasheet)
_FLUX_T_G_	102.0f	Green LED rated flux (Can be found in LED datasheet)
_FLUX_T_B_	41.0f	Blue LED rated flux (Can be found in LED datasheet)
_WP_FLUX_R_	0. 3457f	Red LED flux to generate white color
_WP_FLUX_G_	0.6377f	Green LED flux to generate white color
_WP_FLUX_B_	0.0166f	Blue LED flux to generate white color
PWM_TOP	0x01F4	Timer top value to generate 1kHz PWM
PWM_BOTTOM	0x0000	Timer top value at start up

There are more configurations in the source file. Please see the comments inside.

Appendix B. Revision History

Doc. Rev.	Date	Comments
42240A	01/2014	Initial document release

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370