参考資料

# *TI Designs: TIDA-050011*
# *各種LEDのリング照明パターンのリファレンス・デザイン*

## 概要

このリファレンス・デザインを採用すると、2個のLP5024 LEDドライバを使用して鮮明な照明パターンを映し出すLEDリング 搭載のヒューマン・マシン・インターフェイス（HMI）を実現できます。このデザインは、複雑な照明パターンと定電圧電源を使用する最終機器に共通する課題の解決に役立ちます。また、このデザインはアナログ調光を使用してLP5024の輝度を調整することもできます。動的な周辺光センス機能を搭載しているので、照明パターンのコントラスト比に影響を及ぼすことはありません。

## リソース

| | |
|---|---|
| TIDA-050011 | デザイン・フォルダ |
| LP5024 | プロダクト・フォルダ |
| OPT3001 | プロダクト・フォルダ |
| SimpleLink™ MSP432P401R LaunchPad™開発キット | ツール・フォルダ |

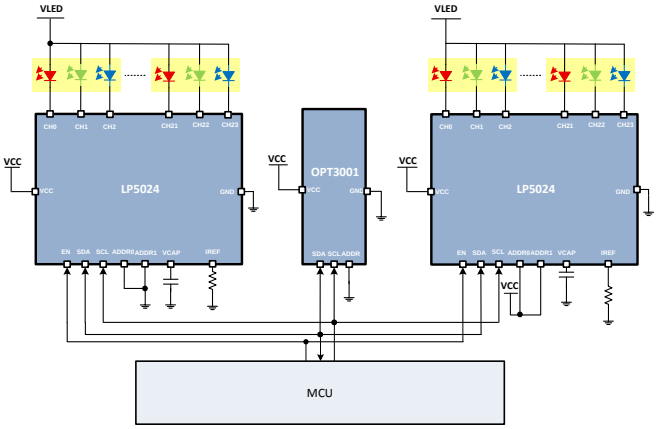
TI E2E™ Community　E2E™ エキスパートに質問

## 特長

- I²Cインターフェイス経由で12ビット、29kHzのPWM調光を実施して鮮明な照明パターンを指定
- 最小10µAの超低スタンバイ電流で、自動的な省電力モードも搭載
- 各カラーのソフトウェア制御を容易にする、プログラマブル・バンク（R / G / B）
- 周辺光センスに基づいて輝度を動的に調整

## アプリケーション

- スマート・スピーカー（音声アシスタント）
- 家電製品向けユーザー・インターフェイスとコネクティビティ・モジュール
- ビデオ・ドアベル
- 電子スマート・ロック
- STBおよびDVR
- WLAN/WiFiアクセス・ポイント
- 仮想および拡張現実用のヘッドセットおよび眼鏡
- プリンタ
- ビデオ・ゲーム機
- ホーム・シアター・イン・ア・ボックス(HTiB)




# 1 System Description

This is the design guide for the Various LED Ring Lighting Patterns Reference Design, which makes the vivid lightings pattern on the LED ring with 2 LP5024 devices. In this reference design, LP5024 linear LED drivers are used to drive 16 RGB LED modules with constant current control. OPT3001 senses the ambient light dynamically. The MSP432P401R LaunchPad sends the control signal to adjust LP5024 analog current and generate the various lighting patterns.

## *1.1 Key System Specifications*

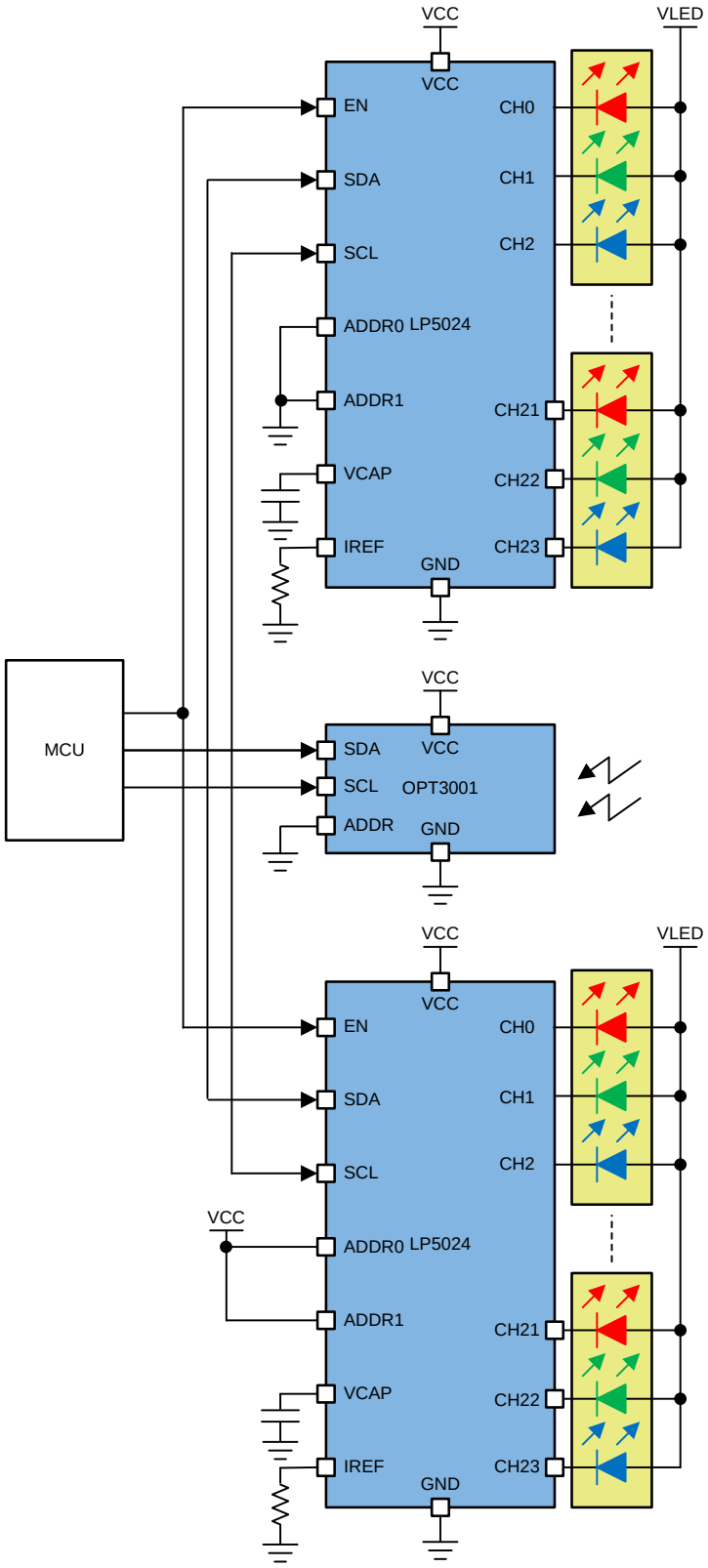
**表 1. Key System Specifications**

| PARAMETER | SPECIFICATIONS |
|---|---|
| Input voltage range | 3 V to 5.5 V |
| Output current | 8 mA/channel maximum |
| LED number | 16 RGB |
| LED type | 19-337/R6GHBHC-A01/2T |
| Lighting pattern style | 7 |
| Analog dimming control range | 1% to 100% |

# 2 System Overview

## 2.1 Block Diagram

図 1. TIDA-050011 Block Diagram



## 2.2 Design Considerations

In this reference design, two 24-channel, 12-bit, PWM ultralow-quiescent-current, I2C RGB LED drivers (LP5024) are used to drive a 16-RGB-LED module with constant current control and smooth dimming effect. The LP5024 devices improve the user experience in color mixing and brightness control for both live effects and coding efforts. The optimized performance for RGB LEDs makes the LP5024 an excellent fit for human-machine interaction applications. OPT3001 senses the ambient light dynamically. The MSP432P401R LaunchPad sends the control signal to adjust LP5024 analog current and generate the various lighting patterns.

Several considerations are taken into account for this particular design:

- LED map (ring) for meeting the requirement of popular human-machine interaction style.
- LED size, numbers and the diffuse design for meeting lighting pattern uniformity.
- Analog dimming in the difference ambient light lux without losing dimming resolution in lighting pattern.

These considerations apply to most human-machine interaction end equipment with day and night vision designs in some way, but the designer must decide the particular considerations to take into account for a specific design.

## 2.3 Highlighted Products

The following highlighted products are used in this reference design. The key features for selecting the devices for this reference design are outlined in the following subsections. For the complete details of the highlighted devices, refer to their respective product data sheets.

### 2.3.1 LP5024 24-Channel, 12-Bit, PWM Ultralow-Quiescent-Current, I2C RGB LED Driver

The LP5024 device is an 24-channel constant current-sink LED driver. The LP5024 device improves the user experience in color mixing and brightness control, from both live effects and coding efforts. The optimized performance for RGB LEDs makes it a potential choice for human-machine-interaction applications.

The LP5024 device controls each LED output with a 12-bit PWM resolution at 29-kHz switching frequency, which helps achieve a smooth dimming effect and eliminates audible noise. The independent color mixing and brightness control registers make the software coding straightforward. When targeting fade-in, fade-out type breathing effect, the global R, G, B bank control reduces the microcontroller loading significantly. The LP5024 device also implements a PWM phase-shifting function to help reduce the input power budget when LEDs turn on simultaneously.

### 2.3.2 SimpleLink™ MSP432P401R LaunchPad™ Development Kit

The SimpleLink™ MSP-EXP432P401R LaunchPad™ development kit enables one to develop high-performance applications that benefit from low-power operation. This kit features the MSP432P401R LaunchPad, which includes a 48-MHz Arm® Cortex®-M4F, 80-µA/MHz active power and 660-nA RTC operation, 14-bit 1- MSPS differential SAR ADC, and AES256 accelerator. All pins of the MSP-EXP432P401R device are fanned out for easy access. These pins make it easy to plug in 20-pin and 40-pin BoosterPack™ modules that add additional functionality including *Bluetooth*® low energy, Wi-Fi® wireless connectivity, and more.

### 2.3.3 OPT3001 Digital Ambient Light Sensor (ALS) With High-Precision Human-Eye Response

The OPT3001 is a sensor that measures the intensity of visible light. The spectral response of the sensor tightly matches the photopic response of the human eye and includes significant infrared rejection.

The OPT3001 is a single-chip lux meter, measuring the intensity of light as visible by the human eye. The precision spectral response and strong IR rejection of the device enables the OPT3001 to accurately meter the intensity of light as seen by the human eye regardless of light source. The strong IR rejection also aids in maintaining high accuracy when industrial design calls for mounting the sensor under dark glass for aesthetics. The OPT3001 is designed for systems that create light-based experiences for humans, and an ideal preferred replacement for photodiodes, photoresistors, or other ambient light sensors with less human eye matching and IR rejection.

Measurements can be made from 0.01 lux up to 83 k lux without manually selecting full-scale ranges by using the built-in, full-scale setting feature. This capability allows light measurement over a 23-bit effective dynamic range.

## *2.4 System Design Theory*

Two 24-channel, 12-bit, PWM ultralow-quiescent-current, I2C RGB LED drivers (LP5024) are used to drive a 16-RGB-LED module with constant current control and a smooth dimming effect. OPT3001 senses the ambient light dynamically. The MSP432P401R LaunchPad sends the PWM control signal to adjust LP5024 analog current and refresh the LP5024 device registers to generate the various lighting patterns.

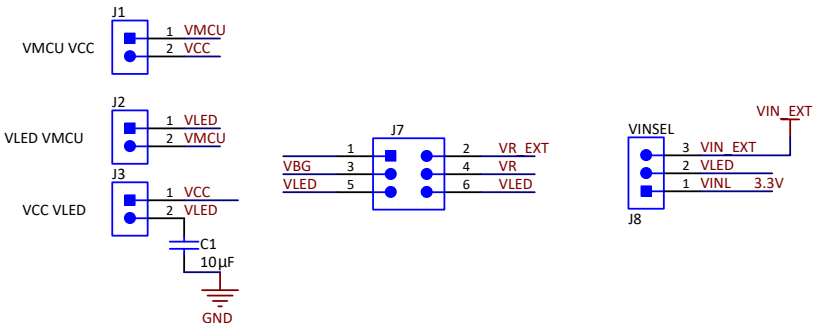
### 2.4.1 Power Supply Design Theory

The LEDs (VLED) are powered by the external DC power directly or 3.3 V from the MSP432P401R LaunchPad through J8 (VINSEL).

The 3.3-V blue and green LEDs, which forward voltage can use separate power from the red LED because the red LED forward voltage is 1.8 V. The lower input voltage for the red LED reduces the power dissipation from the LED driver. J7 is the configuration jumper.

LP5024 supports 1.8-V, 3.3-V, and 5-V $I_2C$ communication logic leves, which means the difference power rail can be used in VCC, $I_2C$ interface and VLED. The device also can be configured through J1, J2, and J3.

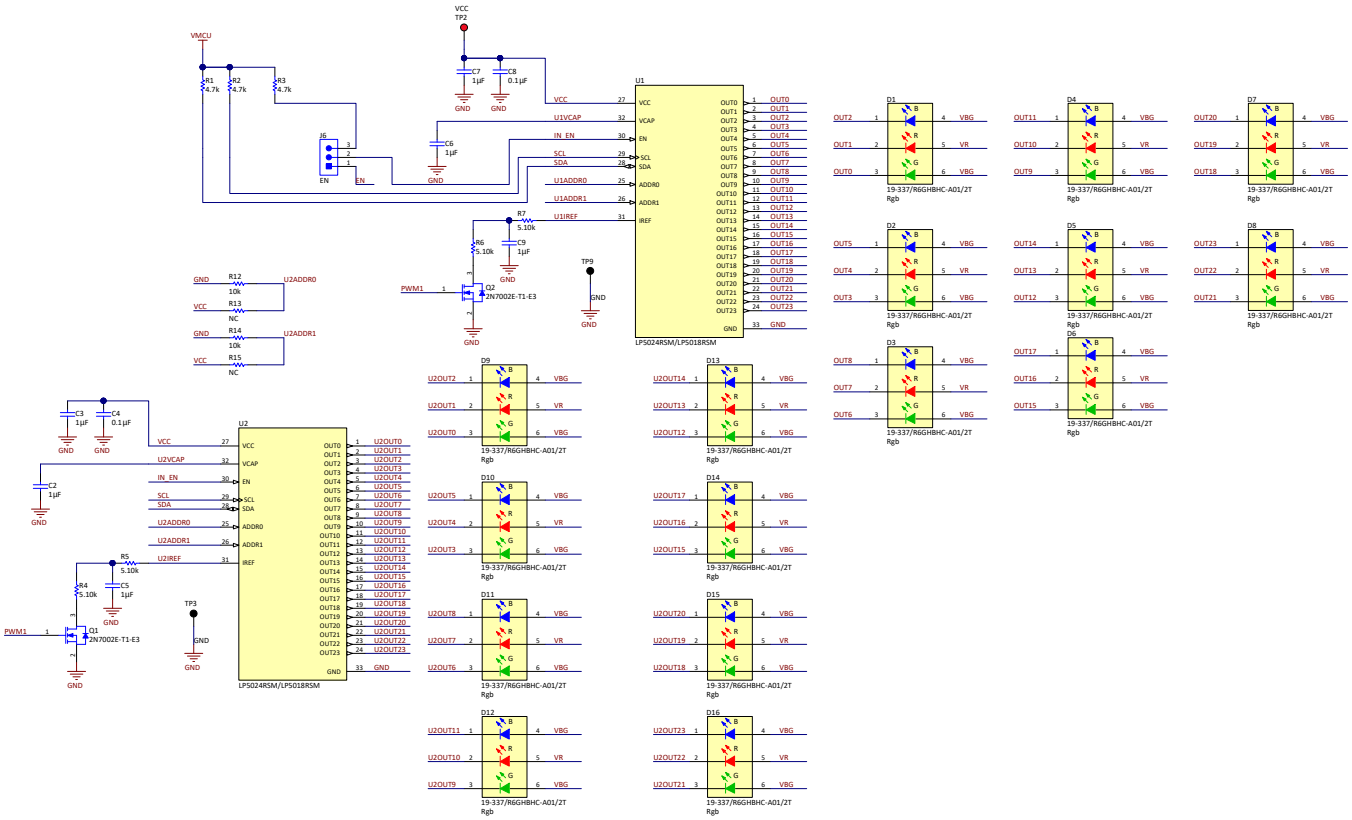
**図 2. Power Design**



### 2.4.2 LED Design

16 RGB LED modules were placed as the LED rings for the popular human-machine interaction style. This design uses two LP5024 devices.

**図 3. LED Design**




#### 2.4.2.1 LED Current

The maximum current of each LED is 8 mA.

The constant-current value ($I_{SET}$) of all 24 channels is set by a single external resistor, $R_{IREF}$. The value of $R_{IREF}$ can be calculated by 式 1.

$$R_{IREF} = K_{IREF} \times \frac{V_{IREF}}{I_{SET}}$$

where:

- $K_{IREF}$ = 105
- $V_{IREF}$ = 0.7 V (1)

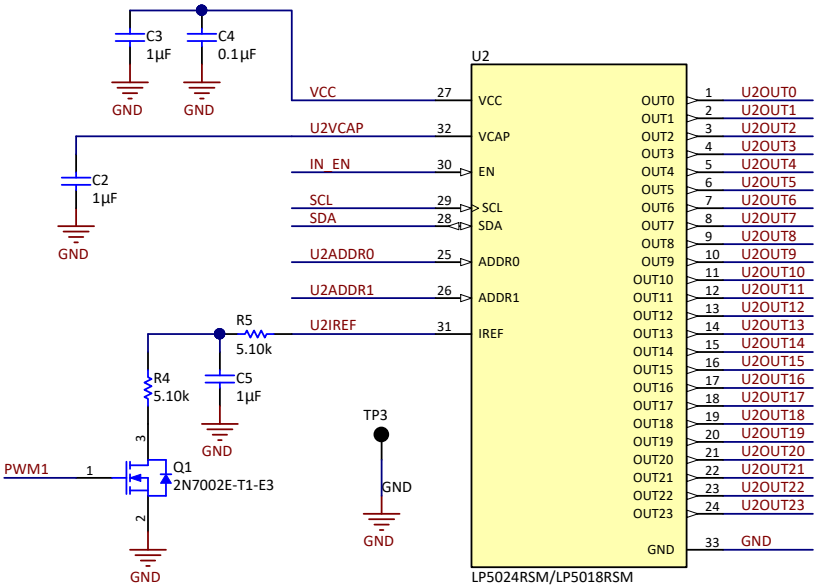
The maximum current of each LED is 8 mA.

Select R4, R5, R6, R7 with a value of 5.1 kΩ.

### 2.4.2.2 LP5024 Analog Dimming Circuit Design

The source current of IREF pin determines the LP5024 constant-current value ($I_{SET}$). The input PWM signal filters it as DC voltage with a large capacitor. The IREF is calculated from equations based on the schematic.

**図 4. Analog Dimming Schematic**




$$I_{IREF} = \frac{V_{IREF} \times D}{R4 + R5 \times D}$$

$$I_{outx} = K_{IREF} \times I_{IREF} = \frac{K_{IREF} \times V_{IREF} \times D}{R4 + R5 \times D}$$

where

- K = R5 / (R4 + R5)
- D is duty cycle of the PWM

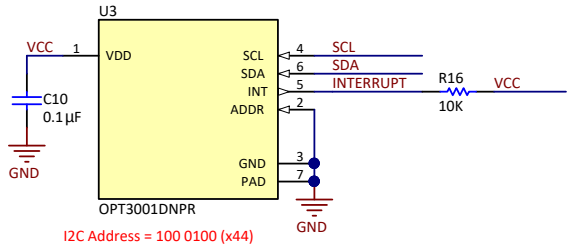
Make R4 = R5, then K = 1/2.

$$I_{outx} = \frac{K_{IREF} \times V_{IREF} \times D}{R4 \times (1 + D)}$$

### 2.4.3 OPT3001 Design

Use the OPT3001 typical application to measure ambient light lux. Make sure that the C1 capacitor is placed close to the VDD pin.

**図 5. OPT3001 Design**



### 2.4.4 Relation Between Pattern Brightness and Ambient Light Lux

When lux is 0, the minium brightness is 1%. When lux is higher than 2000, the maximum brightness is 100%. When the lux is between 0 and 2000, the brightness has a linear relation with the lux.

### 2.4.5 Pattern Brightness Algorithm

Read the lux value from the OPT3001 device register and covert it a decimal format. Next, adjust the pattern brightness based on the lux value through PWM.

After the reading the lux, use the following sample code to transfer it to decimal format.

```
unsigned long int OPT3001_getLux()
{
    /* Specify slave address for OPT3001 */
    I2C_setslave(OPT3001_SLAVE_ADDRESS);

    uint16_t exponent = 0;
    uint32_t result = 0;
    int16_t raw;
    raw = I2C_read16(RESULT_REG);
    /*Convert to LUX*/
    //extract result & exponent data from raw readings
    result = raw&0x0FFF;
    exponent = (raw>>12)&0x000F;
    //convert raw readings to LUX
    switch(exponent){
    case 0: //*0.015625
        result = result>>6;
        break;
    case 1: //*0.03125
        result = result>>5;
        break;
    case 2: //*0.0625
        result = result>>4;
        break;
    case 3: //*0.125
        result = result>>3;
        break;
    case 4: //*0.25
        result = result>>2;
        break;
    case 5: //*0.5
        result = result>>1;
        break;
    case 6:
        result = result;
        break;
    case 7: //*2
        result = result<<1;
        break;
    case 8: //*4
        result = result<<2;
        break;
    case 9: //*8
        result = result<<3;
        break;
    case 10: //*16
        result = result<<4;
        break;
    case 11: //*32
        result = result<<5;
        break;
    }
```

```
        return result;
    }
```

Adjust the brightness using the following function.

```
void  adjustb(void)
{
    float lux;
    lux = OPT3001_getLux();
            /* Adjust Peak Current */
                    if (lux < 2000)
                        pwmConfig.dutyCycle = ((2000*0.01) + (lux*0.98))/2000 * 128;

                    else
                        pwmConfig.dutyCycle = 128;
                    MAP_Timer_A_generatePWM(TIMER_A0_BASE, &pwmConfig);
}
```

### 2.4.6 Lighting Pattern Design

Define the LP5024 register operation function with the following code:

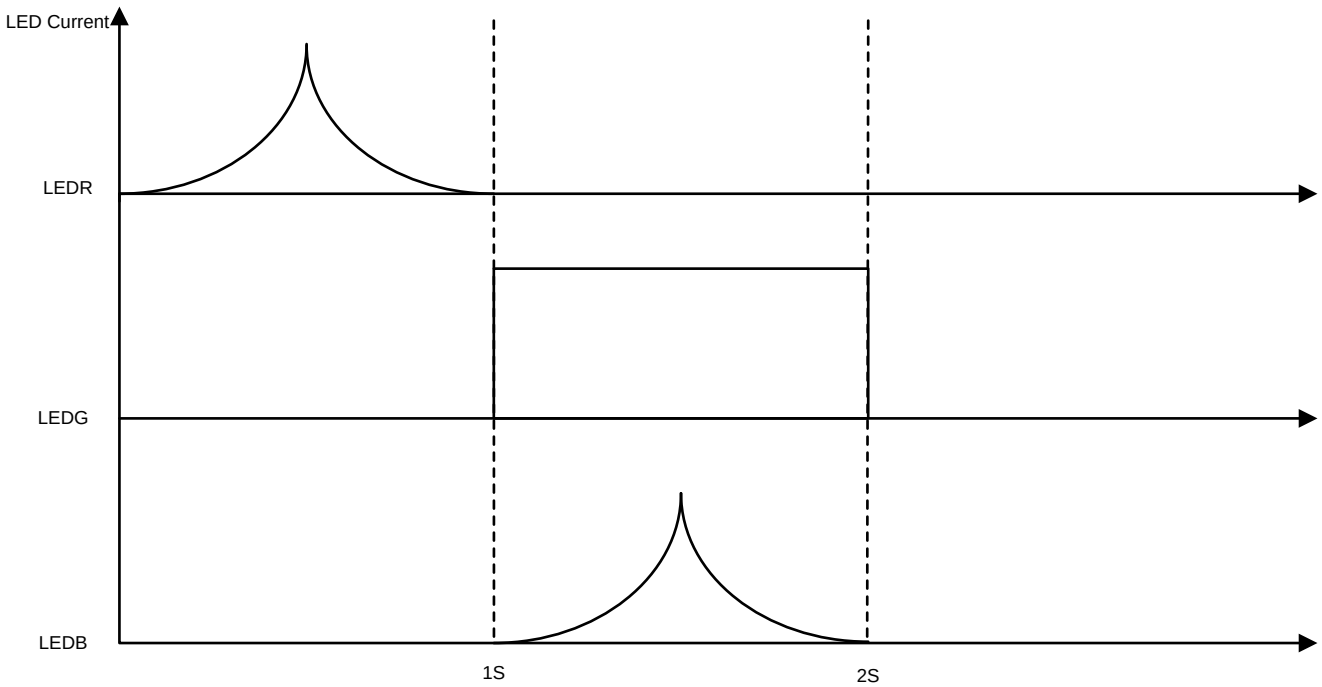
```
void Shutdown()
{
    GPIO_setOutputLowOnPin(GPIO_PORT_P5, GPIO_PIN6);//Set Enable to Low to enter SHUTDOWN mode
}

void Initialization()
{
    GPIO_setOutputHighOnPin(GPIO_PORT_P5, GPIO_PIN6);//Set Enable to High to enter INITIALIZATION
mode
}

void Mode_Select_Standby()//Set Chip_EN=0 to enter STANDBY mode
{
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS);
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x00);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x00); // send register data
}

void Mode_Select_Normal()//Set Chip_EN=1 to enter NORMAL mode
{
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS);
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x00);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0x40); // send register data
}

/*Device Configure*/
void Device_Config1(bool Log_Scale_EN, bool Power_Save_EN, bool Auto_Incr_EN, bool
PWM_Dithering_EN, bool Max_Current_Option, bool LED_Global_Off)
{
    char
config1=Log_Scale_EN*32+Power_Save_EN*16+Auto_Incr_EN*8+PWM_Dithering_EN*4+Max_Current_Option*2+LE
D_Global_Off;
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS);
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x01);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,config1); // send register data
}

/*Bank Select*/
void LED_CONFIG0(bool LED7_Bank_EN, bool LED6_Bank_EN, bool LED5_Bank_EN, bool LED4_Bank_EN, bool
LED3_Bank_EN, bool LED2_Bank_EN, bool LED1_Bank_EN, bool LED0_Bank_EN)
{
    char
config0=LED7_Bank_EN*128+LED6_Bank_EN*64+LED5_Bank_EN*32+LED4_Bank_EN*16+LED3_Bank_EN*8+LED2_Bank_
EN*4+LED1_Bank_EN*2+LED0_Bank_EN;
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS);
```

```
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x02);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,config0); // send register data
}

void Bank_Brightness_Set(char BANK_BRIGHTNESS)
{
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS);
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x03);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,BANK_BRIGHTNESS); // send register data
}
void Bank_Brightness_Set_U1(char BANK_BRIGHTNESS)
{
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS1);
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x03);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,BANK_BRIGHTNESS); // send register data
}
void Bank_Brightness_Set_U2(char BANK_BRIGHTNESS)
{
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS2);
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x03);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,BANK_BRIGHTNESS); // send register data
}

void Bank_Color_Set(char BANK_A_COLOR, char BANK_B_COLOR, char BANK_C_COLOR)
{
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS);
    /*Set Bank Red Color Gray*/
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x04);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,BANK_A_COLOR); // send register data

    /*Set Bank Green Color Gray*/
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x05);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,BANK_B_COLOR); // send register data

    /*Set Bank Blue Color Gray*/
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x06);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,BANK_C_COLOR); // send register data
}

void LED_Brightness_Set(char LED_Number, char LED_Brightness)
{
    if(LED_Number<8)
    {
        MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS1);
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, LED_Number+0x07);  //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,LED_Brightness); // send register data
    }
    else
    {
        MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS2);
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, LED_Number-
8+0x07);  //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,LED_Brightness); // send register data
    }
}

void LED_Color_Set(char LED_Number, char GS_Red, char GS_Green, char GS_Blue)
{
    if(LED_Number<8)
    {
        MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS1);
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, LED_Number*3+0x0F);  //send register
address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,GS_Red); // send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, LED_Number*3+0x10);  //send register
address
```

```
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,GS_Green); // send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, LED_Number*3+0x11);  //send register
address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,GS_Blue); // send register data
    }
    else
    {
        MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS2);
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, (LED_Number-
8)*3+0x0F);  //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,GS_Red); // send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, (LED_Number-
8)*3+0x10);  //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,GS_Green); // send register data
        MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, (LED_Number-
8)*3+0x11);  //send register address
        MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,GS_Blue); // send register data
    }
}

void Reset()
{
    MAP_I2C_setSlaveAddress(EUSCI_B1_BASE, SLAVE_ADDRESS);
    MAP_I2C_masterSendMultiByteStart(EUSCI_B1_BASE, 0x27);  //send register address
    MAP_I2C_masterSendMultiByteFinish(EUSCI_B1_BASE,0xFF); // send register data
}
```

#### 2.4.6.1 Breathing

**図 6. LED Current Waveform of Breathing**




The sample code for breathing is as follows:

```
void breath_fresh_Red(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(0);
```

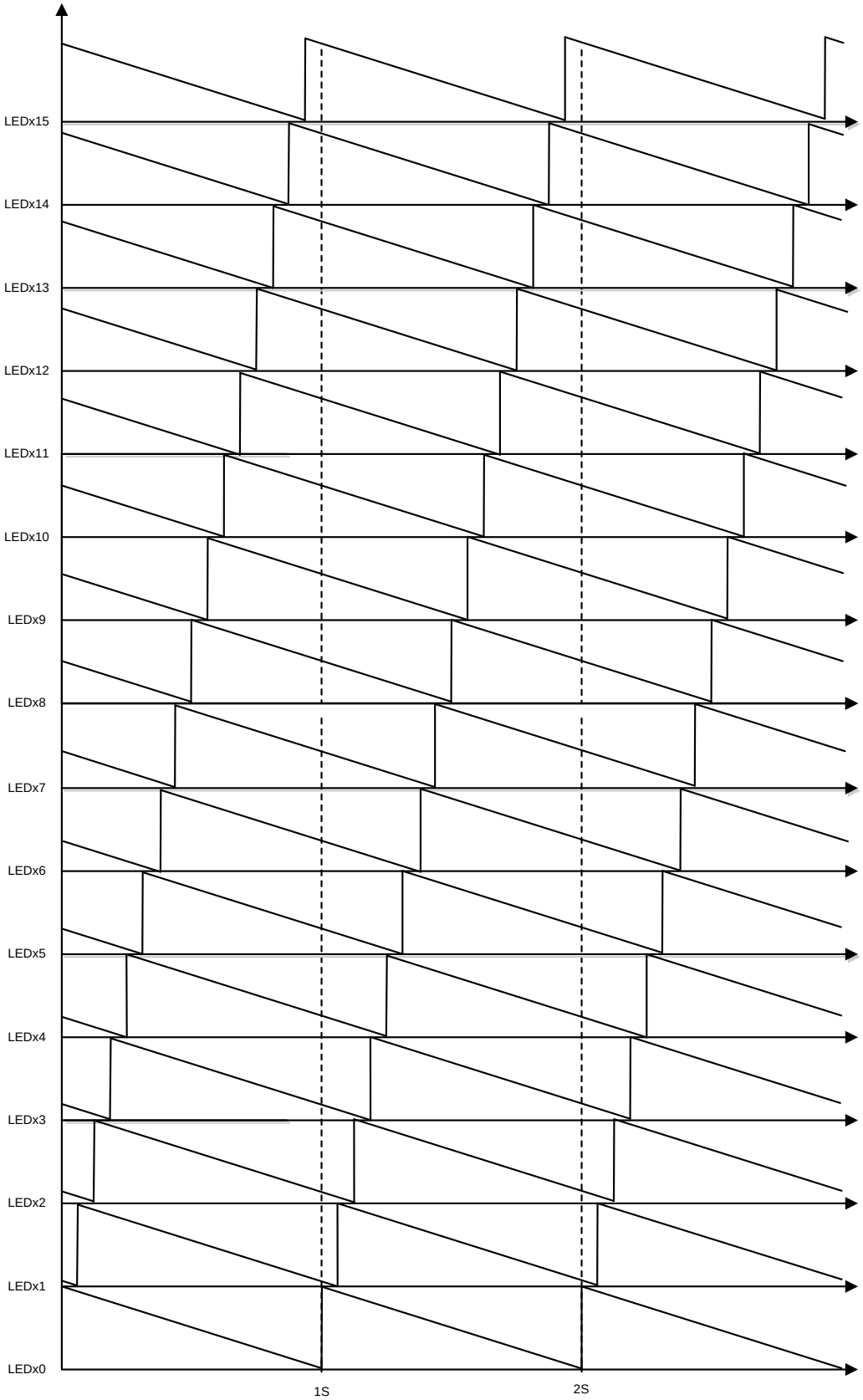
```
    Bank_Color_Set(255,0,0);//Bank color R:255 G:0 B:0
    int i;
    for(i=0;i<256;i++)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
    for(i=255;i>=0;i--)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
}

void breath_fresh_Green(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(0);
    Bank_Color_Set(0,255,0);//Bank color R:0 G:255 B:0
    int i;
    for(i=0;i<256;i++)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
    for(i=255;i>=0;i--)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
}

void breath_fresh_Blue(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(0);
    Bank_Color_Set(0,0,255);//Bank color R:200 G:0 B:0
    int i;
    for(i=0;i<256;i++)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
    for(i=255;i>=0;i--)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
}

void breath_fresh_Yellow(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(0);
```

```
    Bank_Color_Set(255,255,0);//Bank color R:255 G:255 B:0
    int i;
    for(i=0;i<256;i++)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
    for(i=255;i>=0;i--)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
}

void breath_fresh_Pink(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(0);
    Bank_Color_Set(255,0,255);//Bank color R:200 G:0 B:0
    int i;
    for(i=0;i<256;i++)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
    for(i=255;i>=0;i--)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
}

void breath_fresh_Teal(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(0);
    Bank_Color_Set(0,255,255);//Bank color R:200 G:0 B:0
    int i;
    for(i=0;i<256;i++)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
    for(i=255;i>=0;i--)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
}

void breath_fresh_White(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(0);
```

```
    Bank_Color_Set(255,255,255);//Bank color R:255 G:255 B:255
    int i;
    for(i=0;i<256;i++)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
    for(i=255;i>=0;i--)
    {
        Bank_Brightness_Set_U1(i);
        Bank_Brightness_Set_U2(i);
        delay_ms(5);
    }
}

void breath_fresh_BreathingWithCorlorChanging(void)
{
    Mode_Select_Normal();
    Device_Config1(1,1,1,1,0,0);
    LED_CONFIG0(1,1,1,1,1,1,1,1);//Bank control set, every LED in BANK
    Bank_Brightness_Set(255);
    //Bank_Color_Set(200,0,0);//Bank color R:200 G:0 B:0
    int i;
    for(i=0;i<201;i++)
    {
        Bank_Color_Set(0,0,i);
        delay_ms(5);
    }
    for(i=0;i<256;i++)
    {
        Bank_Color_Set(0,i,201);
        delay_ms(5);
    }
    for(i=255;i>0;i--)
    {
        Bank_Color_Set(0,i,201);
        delay_ms(5);
    }

    for(i=0;i<256;i++)
    {
        Bank_Color_Set(i,0,201);
        delay_ms(5);
    }
    for(i=255;i>0;i--)
    {
        Bank_Color_Set(i,0,201);
        delay_ms(5);
    }
    for(i=200;i>0;i--)
    {
        Bank_Color_Set(0,0,i);
        delay_ms(5);
    }
}
```

### 2.4.6.2 Chasing 1

図 7. LED Current Waveform of Single Color Chasing

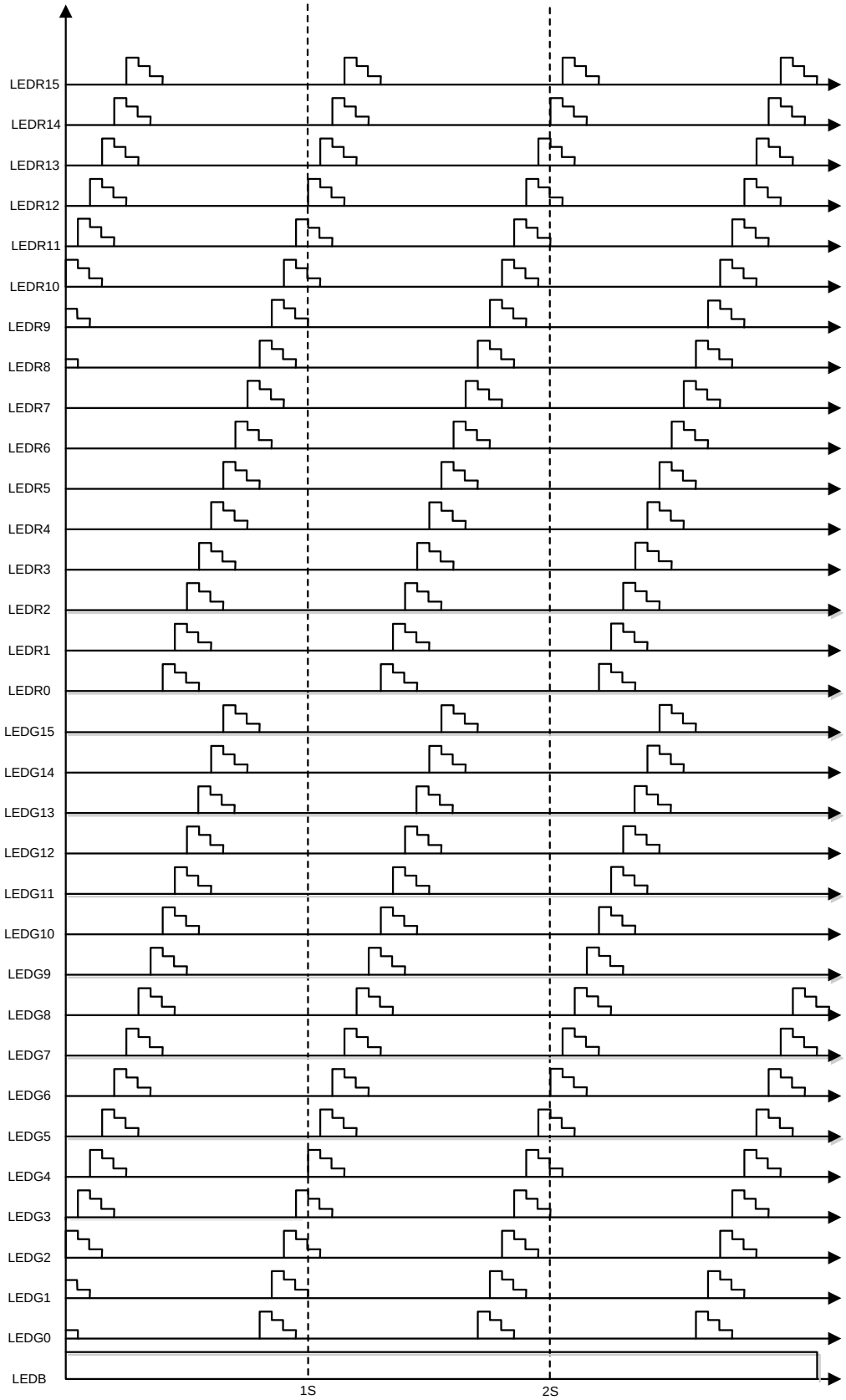


The sample code for the chasing effect is as follows:

```
void ChasingEffectFade_Red(void)
{
    //Mode_Select_Normal();
    //Device_Config1(1,1,1,1,0,0);
    int i;
    for(i=0;i<16;i++)
    {
        LED_Color_Set(i%16,6*(i%16),0,0);
        LED_Color_Set((i+1)%16,8*(i%16),0,0);
        LED_Color_Set((i+2)%16,10*(i%16),0,0);
        LED_Color_Set((i+3)%16,12*(i%16),0,0);
        LED_Color_Set((i+4)%16,14*(i%16),0,0);
        LED_Color_Set((i+5)%16,16*(i%16),0,0);
        LED_Color_Set((i+6)%16,0,0,0);
        LED_Color_Set((i+7)%16,0,0,0);
        LED_Color_Set((i+8)%16,6*(i%16),0,0);
        LED_Color_Set((i+9)%16,8*(i%16),0,0);
        LED_Color_Set((i+10)%16,10*(i%16),0,0);
        LED_Color_Set((i+11)%16,12*(i%16),0,0);
        LED_Color_Set((i+12)%16,14*(i%16),0,0);
        LED_Color_Set((i+13)%16,16*(i%16),0,0);
        LED_Color_Set((i+14)%16,0,0,0);
        LED_Color_Set((i+15)%16,0,0,0);
        delay_ms(32);
    }
    for(i=15;i>=0;i--)
    {
        LED_Color_Set(i%16,6*(i%16),0,0);
        LED_Color_Set((i+1)%16,8*(i%16),0,0);
        LED_Color_Set((i+2)%16,10*(i%16),0,0);
        LED_Color_Set((i+3)%16,12*(i%16),0,0);
        LED_Color_Set((i+4)%16,14*(i%16),0,0);
        LED_Color_Set((i+5)%16,16*(i%16),0,0);
        LED_Color_Set((i+6)%16,0,0,0);
        LED_Color_Set((i+7)%16,0,0,0);
        LED_Color_Set((i+8)%16,6*(i%16),0,0);
        LED_Color_Set((i+9)%16,8*(i%16),0,0);
        LED_Color_Set((i+10)%16,10*(i%16),0,0);
        LED_Color_Set((i+11)%16,12*(i%16),0,0);
        LED_Color_Set((i+12)%16,14*(i%16),0,0);
        LED_Color_Set((i+13)%16,16*(i%16),0,0);
        LED_Color_Set((i+14)%16,0,0,0);
        LED_Color_Set((i+15)%16,0,0,0);
        delay_ms(32);
    }
}
```

### 2.4.6.3 Chasing 2

図 8. LED Current Waveform of Multi Color Chasing

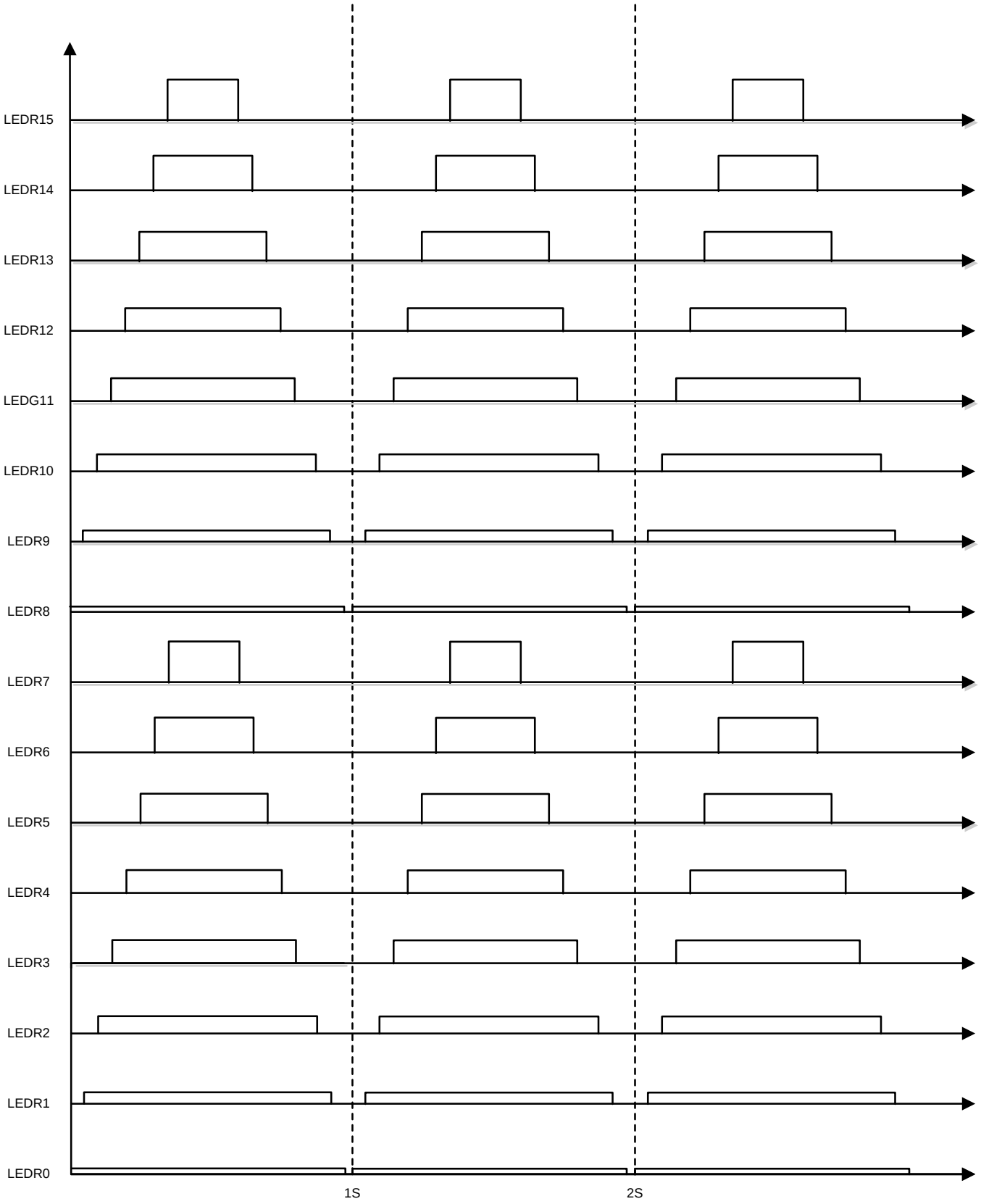


The sample code is as follows:

```
void ChasingEffectFade_Red(void)
{
    //Mode_Select_Normal();
    //Device_Config1(1,1,1,1,0,0);
    int i;
    for(i=0;i<16;i++)
    {
        LED_Color_Set(i%16,6*(i%16),0,0);
        LED_Color_Set((i+1)%16,8*(i%16),0,0);
        LED_Color_Set((i+2)%16,10*(i%16),0,0);
        LED_Color_Set((i+3)%16,12*(i%16),0,0);
        LED_Color_Set((i+4)%16,14*(i%16),0,0);
        LED_Color_Set((i+5)%16,16*(i%16),0,0);
        LED_Color_Set((i+6)%16,0,0,0);
        LED_Color_Set((i+7)%16,0,0,0);
        LED_Color_Set((i+8)%16,6*(i%16),0,0);
        LED_Color_Set((i+9)%16,8*(i%16),0,0);
        LED_Color_Set((i+10)%16,10*(i%16),0,0);
        LED_Color_Set((i+11)%16,12*(i%16),0,0);
        LED_Color_Set((i+12)%16,14*(i%16),0,0);
        LED_Color_Set((i+13)%16,16*(i%16),0,0);
        LED_Color_Set((i+14)%16,0,0,0);
        LED_Color_Set((i+15)%16,0,0,0);
        delay_ms(32);
    }
    for(i=15;i>=0;i--)
    {
        LED_Color_Set(i%16,6*(i%16),0,0);
        LED_Color_Set((i+1)%16,8*(i%16),0,0);
        LED_Color_Set((i+2)%16,10*(i%16),0,0);
        LED_Color_Set((i+3)%16,12*(i%16),0,0);
        LED_Color_Set((i+4)%16,14*(i%16),0,0);
        LED_Color_Set((i+5)%16,16*(i%16),0,0);
        LED_Color_Set((i+6)%16,0,0,0);
        LED_Color_Set((i+7)%16,0,0,0);
        LED_Color_Set((i+8)%16,6*(i%16),0,0);
        LED_Color_Set((i+9)%16,8*(i%16),0,0);
        LED_Color_Set((i+10)%16,10*(i%16),0,0);
        LED_Color_Set((i+11)%16,12*(i%16),0,0);
        LED_Color_Set((i+12)%16,14*(i%16),0,0);
        LED_Color_Set((i+13)%16,16*(i%16),0,0);
        LED_Color_Set((i+14)%16,0,0,0);
        LED_Color_Set((i+15)%16,0,0,0);
        delay_ms(32);
    }
}
```

### 2.4.6.4 Chasing 3

図 9. Current Wave of Custom Chasing

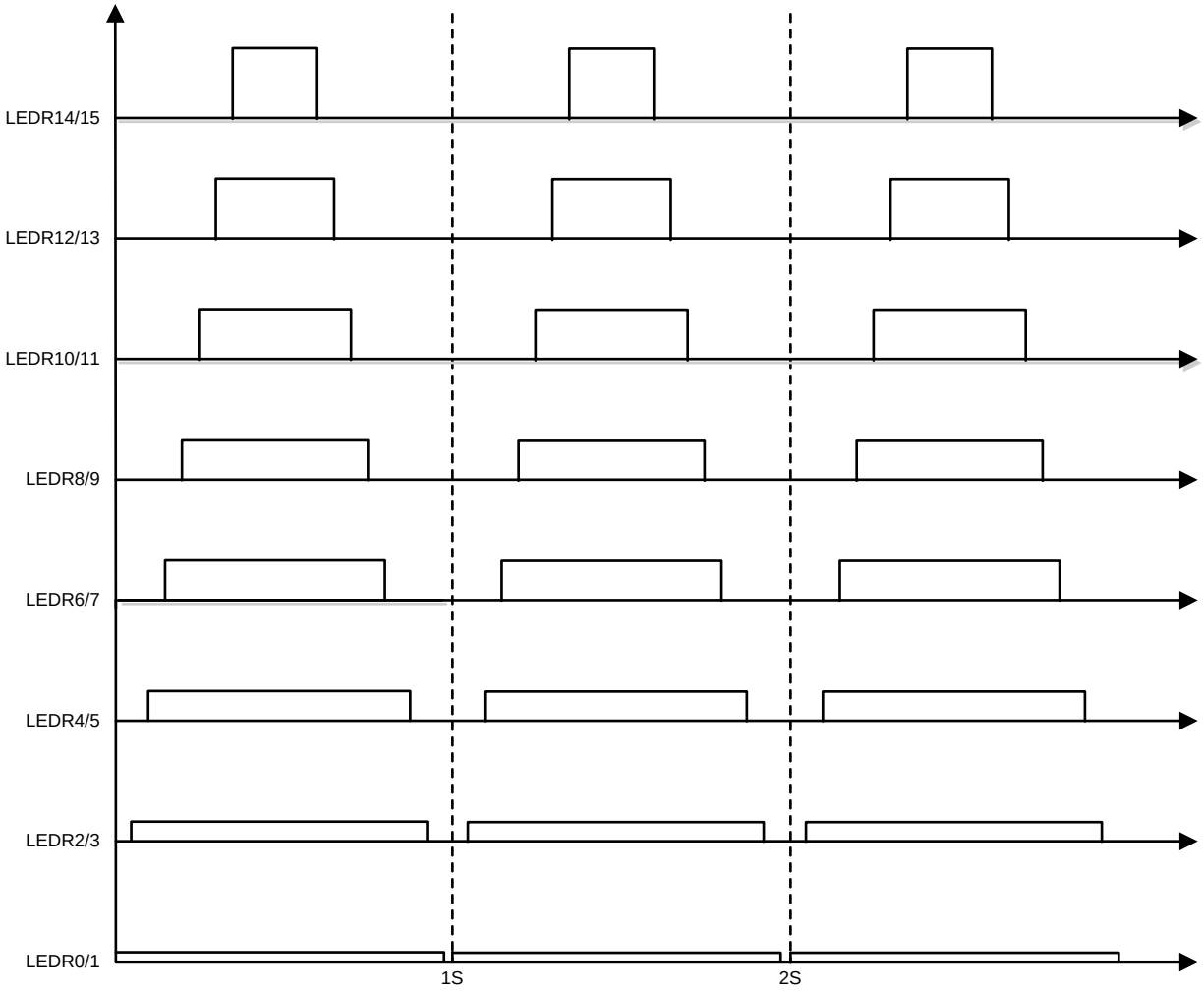


The sample code is as follows:

```
void ChasingEffect_Custom(void)
{
    //Mode_Select_Normal();
    //Device_Config1(1,1,1,1,0,0);
    int i;
    for(i=0;i<16;i++)
    {
        LED_Color_Set(i%16,103,103,0);
        LED_Color_Set((i+1)%16,193,193,0);
        LED_Color_Set((i+2)%16,233,233,0);
        LED_Color_Set((i+3)%16,0,0,200);
        LED_Color_Set((i+4)%16,0,0,200);
        LED_Color_Set((i+5)%16,0,0,200);
        LED_Color_Set((i+6)%16,0,0,200);
        LED_Color_Set((i+7)%16,0,0,200);
        LED_Color_Set((i+8)%16,0,70,12);
        LED_Color_Set((i+9)%16,0,121,25);
        LED_Color_Set((i+10)%16,0,252,50);
        LED_Color_Set((i+11)%16,0,0,200);
        LED_Color_Set((i+12)%16,0,0,200);
        LED_Color_Set((i+13)%16,0,0,200);
        LED_Color_Set((i+14)%16,0,0,200);
        LED_Color_Set((i+15)%16,0,0,200);
        delay_ms(30);
    }
}
```

### 2.4.6.5 Water Drop

図 10. Current Wave of Water Drop

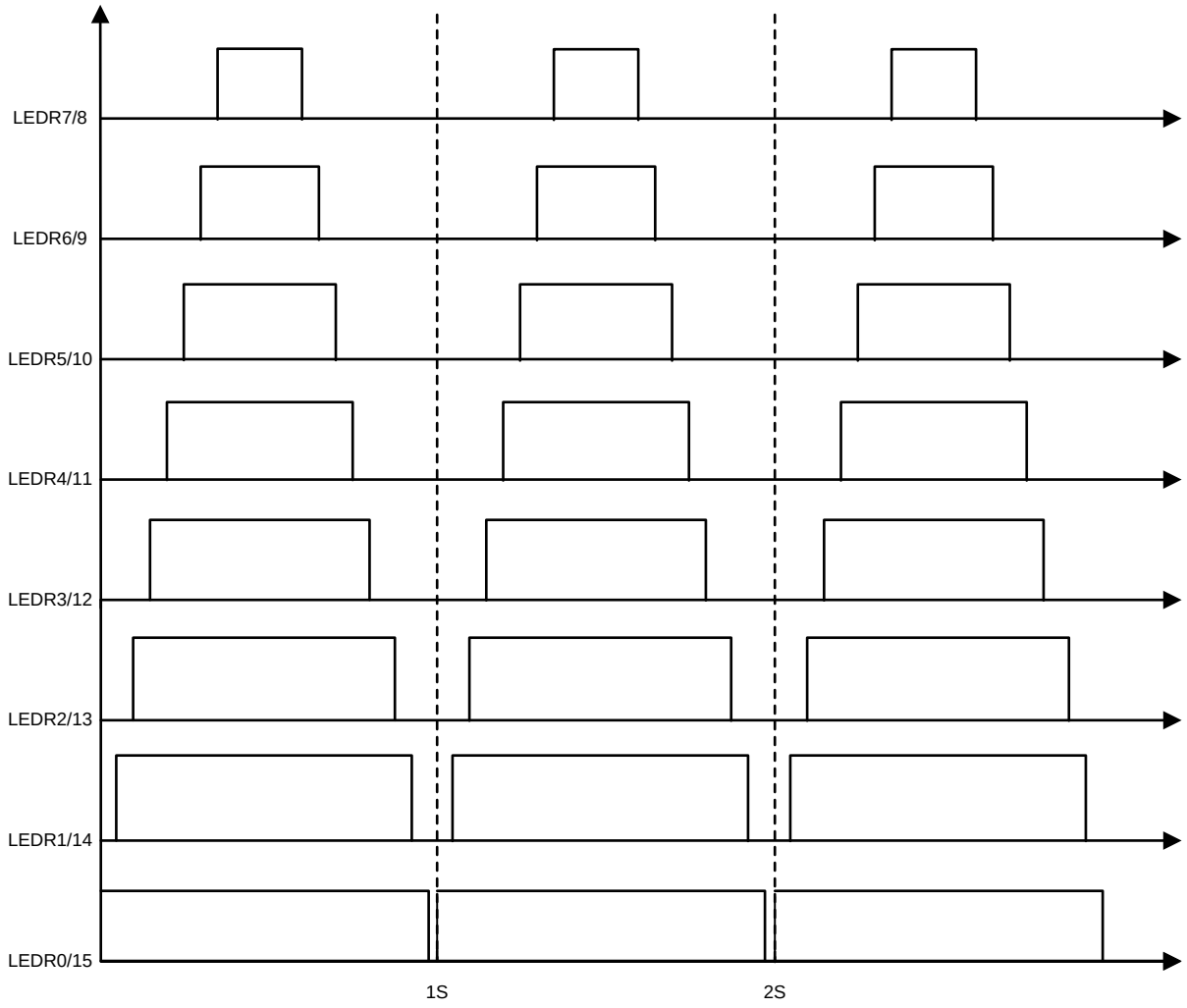



The sample code is as follows:

```
void Drop_effect_Red(void)
{
    //Mode_Select_Normal();
    //Device_Config1(1,1,1,1,0,0);
    int i;
    for (i=0;i<16;i++)
    {
        LED_Color_Set(i,(i+1)*16-1,0,0);
        delay_ms(32);
    }
    for(i=15;i>=0;i--)
    {
        LED_Color_Set(i,0,0,0);
        delay_ms(32);
    }
}
```

### 2.4.6.6 Others

図 11. Current Wave of Call




The sample code is as follows:

```
void Call_Effect(void)
{
    //Mode_Select_Normal();
    //Device_Config1(1,1,1,1,0,0);
    int i;
    for(i=0;i<8;i++)
    {
        LED_Color_Set(i,0,250,0);
        LED_Color_Set(15-i,0,250,0);
        delay_ms(32);
    }
    delay_ms(80);
    for(i=7;i>=0;i--)
    {
        LED_Color_Set(i,0,0,0);
        LED_Color_Set(15-i,0,0,0);
        delay_ms(32);
    }
    delay_ms(80);

}
```

# 3 Hardware, Software, Testing Requirements, and Test Results

## 3.1 Required Hardware and Software

### 3.1.1 Hardware

The following hardware is required for the testing:

- A personal computer with Windows® OS
- A 3.3-V dc power supply
- An MSP432 LaunchPad

### 3.1.2 Test Setup

After downloading the firmware from the TIDA-050011 product folder, connect a 3.3-V dc supply to the LED board input connector. Next, use the USB port to load the firmware to the MSP432 LaunchPad development kit through the Code Composer Studio integrated development environment (IDE). The LED driver and LED load boards can then run the pattern.

### 3.1.3 Test Results

All the effects are shown as expected.

# 4 Design Files

## 4.1 Schematics

To download the schematics, see the design files at TIDA-50011.

## 4.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDA-50011.

## 4.3 PCB Layout Recommendations

- Place the decoupling capacitor near the VCC and GND terminals.
- Make the GND traces as wide as possible for large GND currents. The maximum GND current is approximately 1.2 A.
- Route traces between the LED cathode and the device OUTXn pin as short and straight as possible to reduce wire inductance.
- Connect the thermal pad to the GND layer because the thermal pad is internally connected to GND to reduce device temperature.

### 4.3.1 Layout Prints

To download the layer plots, see the design files at TIDA-50011.

## 4.4 Altium Project

To download the Altium Designer® project files, see the design files at TIDA-50011.

## 4.5 Gerber Files

To download the Gerber files, see the design files at TIDA-50011.

## 4.6 Assembly Drawings

To download the assembly drawings, see the design files at TIDA-50011.

## 5 Software Files

To download the software files, see the design files at TIDA-50011.

## 6 Related Documentation

1. Texas Instruments, *LP5024 24-Channel I2C Constant-Current RGB LED Driver*
2. Texas Instruments, *MSP432P401R SimpleLink™ Microcontroller LaunchPad™ Development Kit (MSP-EXP432P401R) User's Guide*
3. Texas Instruments, *Digital Ambient Light Sensor (ALS) With High-Precision Human-Eye Response*

### *6.1 商標*



## 7 About the Author

**XING SU** is an application engineer at Texas Instruments for the LED Driver product group.

## 重要なお知らせと免責事項

TI は、技術データと信頼性データ（データシートを含みます）、設計リソース（リファレンス・デザインを含みます）、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションが適用される各種規格や、その他のあらゆる安全性、セキュリティ、またはその他の要件を満たしていることを確実にする責任を、お客様のみが単独で負うものとします。上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、TI の販売条件（www.tij.co.jp/ja-jp/legal/termsofsale.html）、または ti.com やかかる TI 製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。TI がこれらのリソースを提供することは、適用されるTI の保証または他の保証の放棄の拡大や変更を意味するものではありません。

## 重要なお知らせと免責事項

TI は、技術データと信頼性データ（データシートを含みます）、設計リソース（リファレンス・デザインを含みます）、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションが適用される各種規格や、その他のあらゆる安全性、セキュリティ、またはその他の要件を満たしていることを確実にする責任を、お客様のみが単独で負うものとします。上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、TI の販売条件（www.tij.co.jp/ja-jp/legal/termsofsale.html）、または ti.com やかかる TI 製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。TI がこれらのリソースを提供することは、適用されるTI の保証または他の保証の放棄の拡大や変更を意味するものではありません。