Using M032 to Drive WS2812B LED Strip

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code is an implementation of controlling the SPI transfer protocol of M032 from PC side via USB HID interface to drive WS2812B LED strip.
BSP Version	M031 BSP CMSIS V3.03.000
Hardware	NuMaker-M032KI Ver1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Overview

This sample code uses M032 series microcontrollers to drive WS2812B through MOSI of SPI0 or USPI0 to light RGB-LED strip. A user can control LED strip through USB interface using the PC AP Tool "*LEDDongle_v1_0.exe*".

1.1 Principle

The WS2812B is an intelligent control LED light source that integrates the control circuit and RGB chip into a 5050 component package. It includes an intelligent digital port data latch and a signal reshaping amplification drive circuit. The data transfer protocol uses single NZR communication mode. The DIN port receives data from the controller after the pixel power-on reset. The first pixel collects initial 24-bit data and then sends it to the internal data latch, and the other data reshaped by the internal signal reshaping amplification circuit is sent to the next cascade pixel through the DO port. The signal can be decreased up-to 24-bit for each pixel after transmission. The pixels follow auto reshaping transmission technology so that the pixel cascade number is not limited to the transmission of the signal. The number only depends on the speed of transmission of the signal.

T0H	0 code ,high voltage time	0.4us	±150ns
T1H	1 code ,high voltage time	0.85us	±150ns
T0L	0 code ,low voltage time	0.85us	±150ns
T1L	1 code ,low voltage time	0.4us	±150ns
RES	low voltage time	Above 50μs	

Table 1-1 Data Transfer Time (TH + TL = 1.25μs ± 600ns)

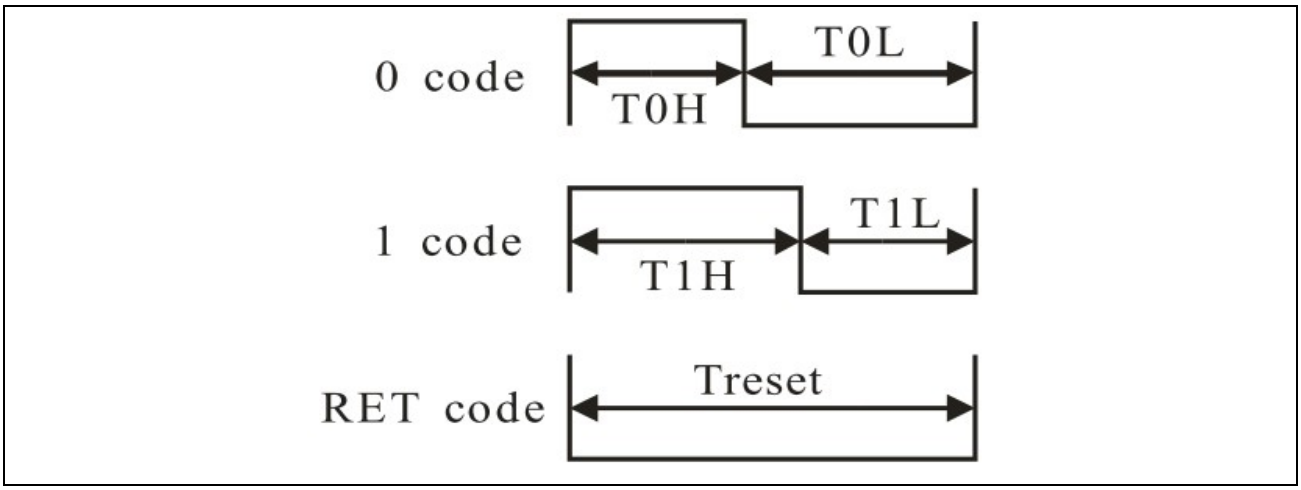


Figure 1-1 Sequence Chart

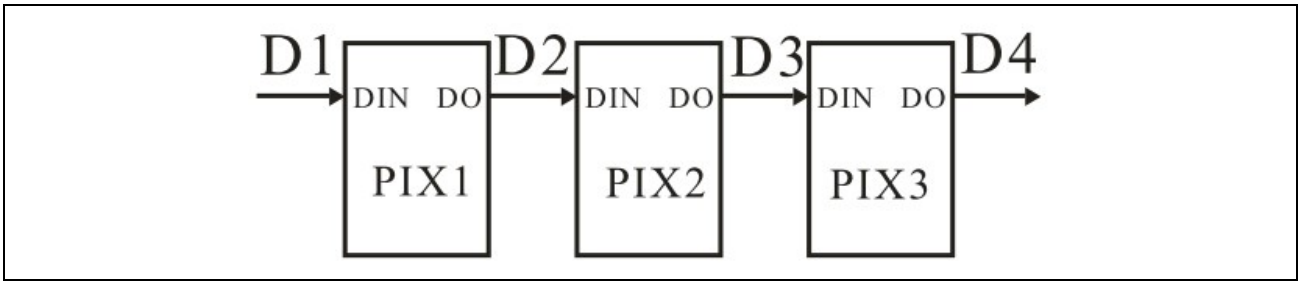


Figure 1-2 Cascade Method

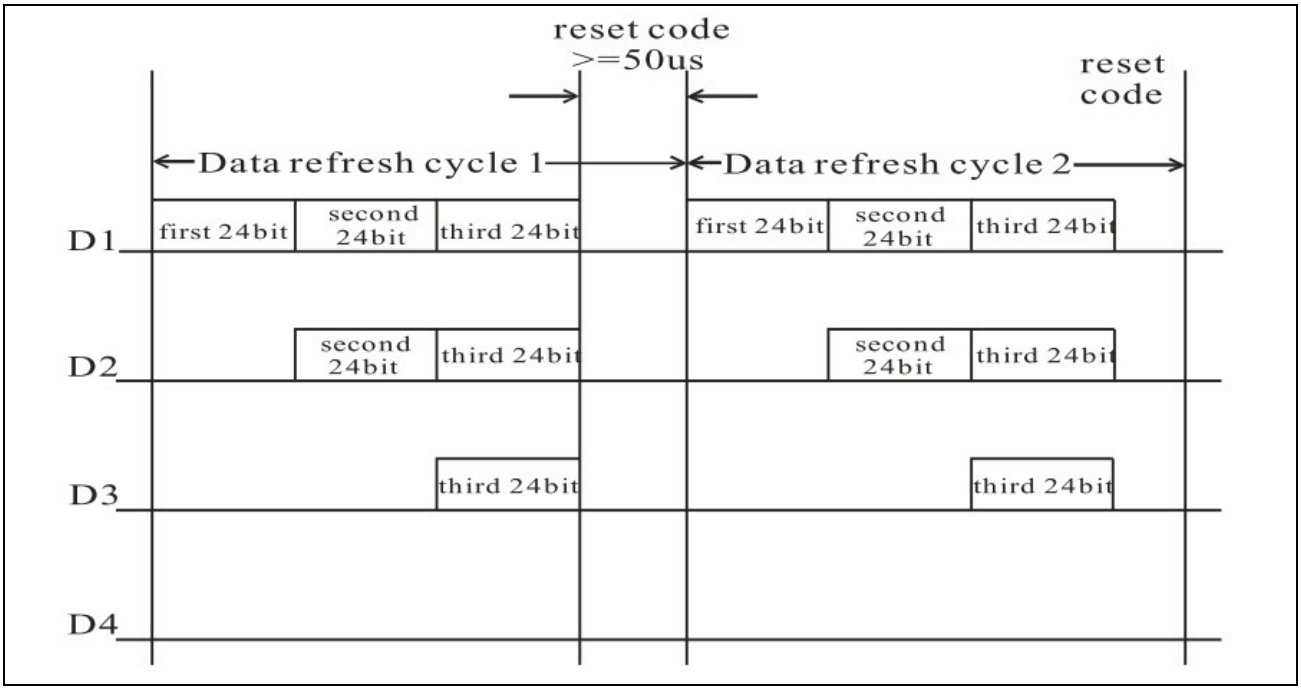


Figure 1-3 Data Transmission Method

1.2 Demo Result

Connect the M032 USB to the computer, open the [Device Manager] console, and you can see the HID-compliant device in the Human Interface Devices area, as shown in Figure 1-4. Then open the PC tool "*LEDDongle_v1_0.exe*" and click the **Connect** icon to connect LED dongle, as shown in Figure 1-5. The PC tool uses USB protocol to control LED dongle. For more information about the USB protocol, please refer to the LED_Dongle_USB_Protocol_V1_0.pdf in Tool folder. Set LED number to 100, select **RAINBOW** mode and then click the **Set** icon to demonstrate rainbow effect, and the result is shown in Figure 1-6. Note that the sample code does not support firmware update feature of PC tool.

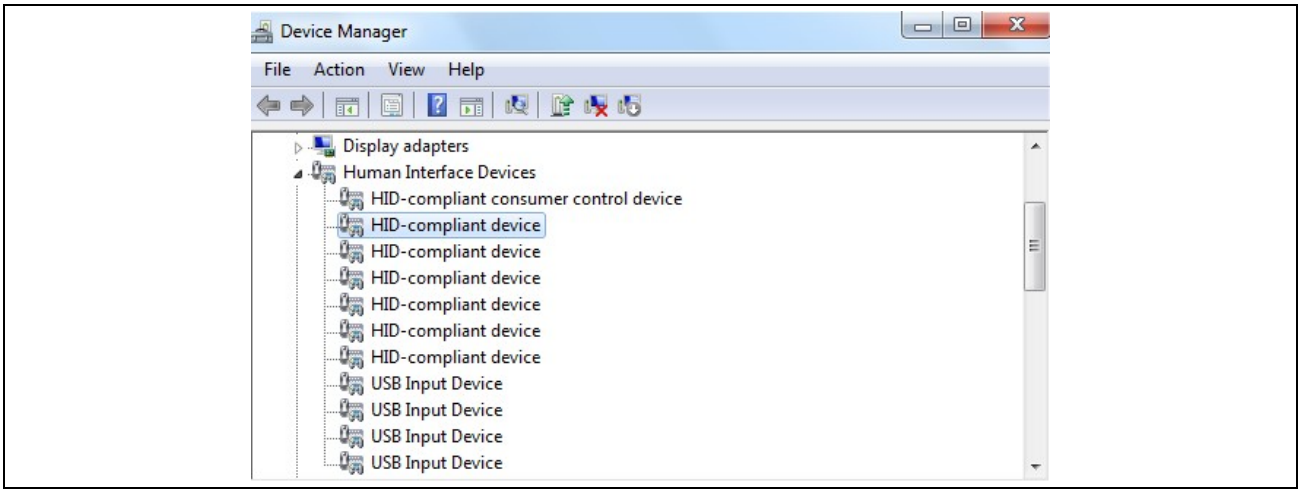


Figure 1-4 Device Manager

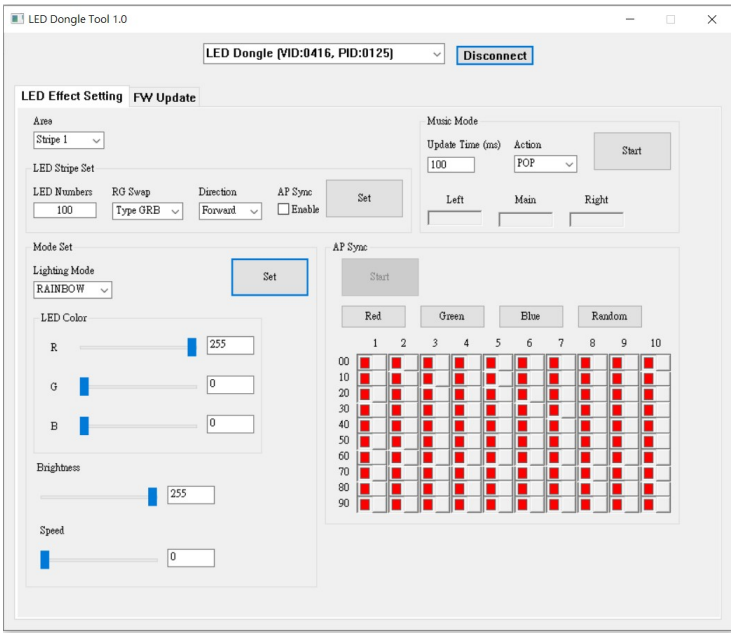


Figure 1-5 LED Dongle Tool

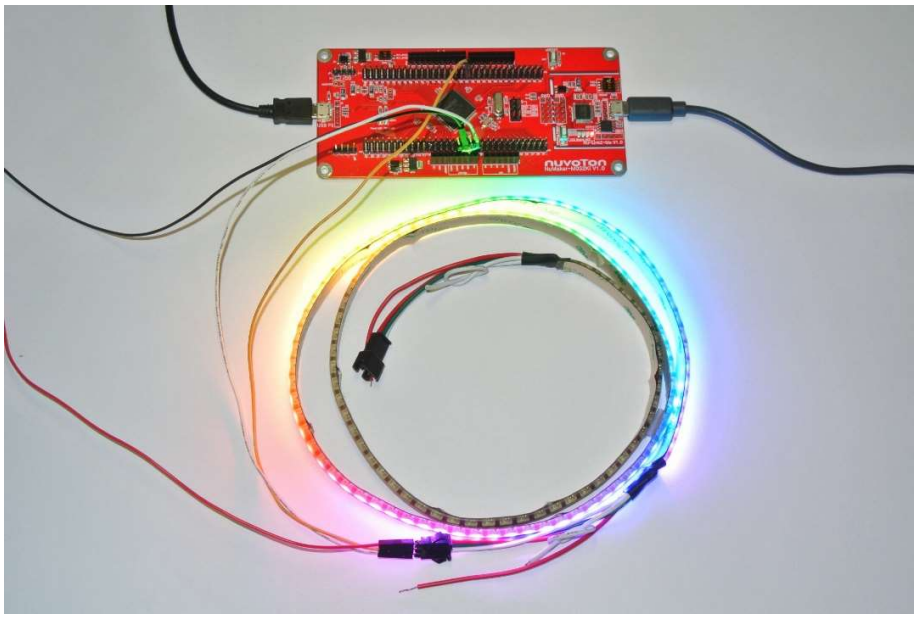


Figure 1-6 Demo Result

2 Code Description

2.1 main.c

First, initialize TIMER, USB, SPI and PDMA for SPI TX operations. Then, read LED strip setting from Data Flash and SPI write 0 code to reset LED strip. This sample code is always polling PDMA transfer status, and thus SPI will write data to WS2812B LED strip with PDMA again whenever PDMA transfer is done.

```
int32_t main(void)
{
    uint32_t i;

    /* Unlock write-protected registers */
    SYS_UnlockReg();

    /* Init system and multi-funcition I/O */
    SYS_Init();

    /* Open Timer0 in periodic mode, enable interrupt and 1000 interrupt tick per second
      to add counter*/
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 1000);
    TIMER_EnableInt(TIMER0);
    NVIC_EnableIRQ(TMR0_IRQn);

    /* Set Led configuration */
    PDMA_Open(PDMA, (1 << USPI_TX_PDMA_CH) | (1 << SPI_TX_PDMA_CH));
    Bar1_Initial();
    Bar2_Initial();

    /* Open USB controller */
    USBD_Open(&gsInfo, HID_ClassRequest, NULL);
    /*Init Endpoint configuration for HID */
    HID_Init();
    /* Start USB device */
    USBD_Start();
    /* Enable USB device interrupt */
    NVIC_EnableIRQ(USBD_IRQn);

    /* Read setting from Data Flash */
    ReadStoredSetting();

    /* Reset Lighting Bar */
    Reset_Bar1_LED();
    while(!Bar1_Setting.fPDMA_Done);
    Reset_Bar2_LED();
    while(!Bar2_Setting.fPDMA_Done);

    /* Start Timer */
    TIMER_Start(TIMER0);

    /* Use TIMER0 as seed of random function */
    srand(TIMER0->CNT);

    while (1)
    {
        /* Serial Transmission Done */
        if(Bar1_Setting.fPDMA_Done)
        {
```

```
            if(Bar1_Setting.AP_Sync == 1)
            {
                /* Check Area */
                if(RcvBuffer[1] & 0x01)
                {
                    /* Mapping Color to LED Format */
                    if (Bar1_Setting.LED_Type == Type_GRB)
                    {
                        Set_GRB_Array(Bar1_Setting.LED_Data, cBar1_LED,
                           (uint8_t*)(RcvBuffer + 2), 0xFF);
                    }
                    else if (Bar1_Setting.LED_Type == Type_RGB)
                    {
                        Set_RGB_Array(Bar1_Setting.LED_Data, cBar1_LED,
                           (uint8_t*)(RcvBuffer + 2), 0xFF);
                    }

                    /* Set Serial Data */
                    Set_Bar1_LED(cBar1_LED);
                }
            }
            else
            {
                for(i = 0; i < cBar1_LED * 24; i++)
                {
                    *(Bar1_Setting.LED_Data + i) = WSData0;
                }
                /* Light Bar 1 LED data */
                Bar1_Setting.Mode_FUNC(&Bar1_Setting);

                /* Set Serial Data */
                Set_Bar1_LED(cBar1_LED);
            }
        }

        ...

    }
}
```

2.2 Serial_LED.c

Mode_Function array includes 11 lighting effects for LED strip function. The SPI clock is set at 6.4 MHz (156.25 ns) and a data bit is represented by an 8-bit SPI waveform. Convert logic 0 into 0 code and the binary is 0xC0, which means 312.5ns high and 937.5ns low. Convert logic 1 into 1 code and the binary is 0xFC, which means 937.5ns high and 312.5ns low. According to the mode set by the PC tool, fill data in Bar1LEDData array and then trigger PDMA to transfer serial data to WS2812B LED strip.

```
void * const Mode_Function[15] = {FUNC_Off, FUNC_Static, FUNC_Breathing, FUNC_Strobe,
                                  FUNC_Cycling, FUNC_Random, FUNC_Music, FUNC_Wave,
                                  FUNC_Spring, FUNC_Water, FUNC_Rainbow
                                 };

/* Initial Bar1 Setting */
uint8_t Bar1LEDData[cBar1_LED*24];
```

```
volatile LED_Setting_T Bar1_Setting = {0, 0, 100, 1, 255, 0, 0, 0xFF, 0, Dir_Forward,
                                       Type_GRB, 1, 1, 0, FUNC_Static, Bar1LEDData, 0, 0,
                                       0, Music_POP
                                       };

void Set_RGB_Single(uint8_t *LED_DATA, uint32_t TotalLED, uint8_t Data_R, uint8_t Data_G,
uint8_t Data_B)
{
    uint32_t i;
    int8_t k;
    uint32_t Temp;

    for(i = 0; i < TotalLED; i++)
    {
        /* R */
        Temp = i*24 + 0*8 + 7;
        for(k = 7; k >= 0; k--)
        {
            if(Data_R & (1 << k))
            {
                *(LED_DATA + (Temp-k)) = WSData1;
            }
            else
            {
                *(LED_DATA + (Temp-k)) = WSData0;
            }
        }
        /* G */
        Temp = i*24 + 1*8+7;
        for(k = 7; k >= 0; k--)
        {
            if(Data_G & (1 << k))
            {
                *(LED_DATA + (Temp-k)) = WSData1;
            }
            else
            {
                *(LED_DATA + (Temp-k)) = WSData0;
            }
        }
        /* B */
        Temp = i*24 + 2*8+7;
        for(k = 7; k >= 0; k--)
        {
            if(Data_B & (1 << k))
            {
                *(LED_DATA + (Temp-k)) = WSData1;
            }
            else
            {
                *(LED_DATA + (Temp-k)) = WSData0;
            }
        }
    }
}

void Set_Bar1_LED(uint32_t LEDNum)
{
    /* Set SPI0_MOSI MFP function */
    Bar1_Open();

    /* Set transfer width (8 bits) and transfer count */
    PDMA->DSCT[SPI_TX_PDMA_CH].CTL |= (((LEDNum*24)) << PDMA_DSCT_CTL_TXCNT_Pos) |
      PDMA_OP_BASIC;
    /* Set source/destination address and attributes */
```

```
    PDMA_SetTransferAddr(PDMA, SPI_TX_PDMA_CH, (uint32_t)Bar1LEDData, PDMA_SAR_INC,
      (uint32_t)&SPI0->TX, PDMA_DAR_FIX);

    Bar1_Setting.fPDMA_Done = 0;
    /* Enable SPI master DMA function */
    SPI_TRIGGER_TX_PDMA(SPI0);
}
```

2.3 hid_transfer.c

When a device receives OUT token that is issued by PC to send data, the function of EP3_Handler() will be executed in endpoint 3 interrupt handler. In the function of EP3_Handler(), the device will prepare data according to the general command received from PC for endpoint 2 buffer. After that, when the device receives IN token, the data stored in endpoint 2 buffer will be sent back to PC. The device will also set the related parameters of the LED strip according to the set strip command and store the setting in Data Flash.

```
void EP3_Handler(void)  /* Interrupt OUT handler */
{
    uint8_t *ptr;
    /* Interrupt OUT */
    ptr = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP3));

    /* Check Command Type */
    if(*ptr == General_ID)
    {
        HID_Get_General_CMD(ptr, USBD_GET_PAYLOAD_LEN(EP3));
    }
    else if(*ptr == SetStripe_ID)
    {
        HID_Get_SetStripe_CMD(ptr, USBD_GET_PAYLOAD_LEN(EP3));
    }
    else if(*ptr == SetMode_ID)
    {
        HID_Get_SetMode_CMD(ptr, USBD_GET_PAYLOAD_LEN(EP3));
    }

    USBD_SET_PAYLOAD_LEN(EP3, EP3_MAX_PKT_SIZE);
}
```

3 Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M031 Series BSP CMSIS v3.03.000
- IDE version
  - Keil uVersion 5.26
- PC tool
  - LEDDongle_v1_0.exe

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M032KI Ver1.0
  - WS2812B RGB-LED Strip
  - Micro USB cable

- Pin Connect

M032		WS2812B RGB-LED Strip1
		+5V
VSS	↔	VSS
SPI0_MOSI	↔	Din
		WS2812B RGB-LED Strip2
		+5V
VSS	↔	VSS
USPI0_MOSI	↔	Din
		PC (USB Host)
USB	↔	

Figure 3-1 Pin Connect

4 Directory Information

The directory structure is shown below.

EC_M032_SPI_WS2812B_LED_Strip_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code
Tool	PC tool LEDDongle_v1_0.exe and USB protocol document
NuLEDDongle	Source file of PC tool LEDDongle_v1_0.exe

Figure 4-1 Directory Structure

5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *SPI_WS2812B_LED_Strip.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6 Revision History

Date	Revision	Description
2021.09.27	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.