MS51 RGB LED Control by UART

Example Code Introduction for 8-bit NuMicro® Family

Document Information

Application	This sample code uses MS51 PWM to control RGB LED by UART.
BSP Version	MS51_Series_BSP_Keil_V1.00.004
Hardware	NuTiny-MS51FB V1.1

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 MS51 PWM to control RGB LED through UART input corresponding to PWM duty. Users can input the PWM duty corresponding to RGB through the terminal tools to achieve the adjustment of brightness and color control.

1.1 Principle

1.1.1 PWM and LED Status

RGB LED lights are divided into common anode and common cathode. Please refer to Figure 1-1 and Figure 1-3 for the hardware circuit. Please refer to Figure 1-2 and Figure 1-4 for the correspondence between RGB LED and PWM duty status. The user can define the hardware action state through '#define Com_Pole' in the program code.

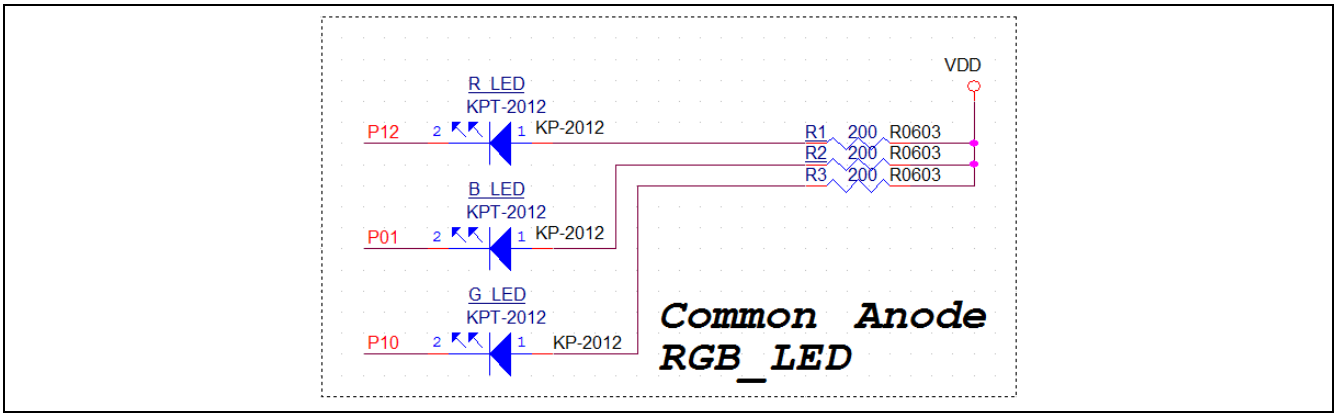

Figure 1-1 Common Anode RGB Diagram

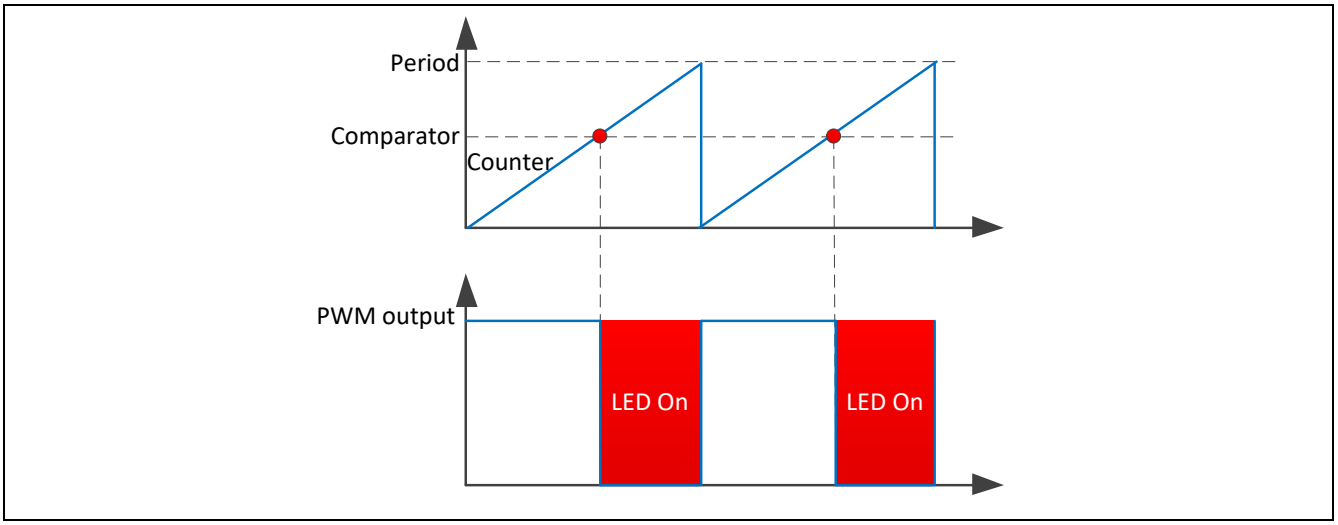

Figure 1-2 PWM and Light Status at Common Anode RGB

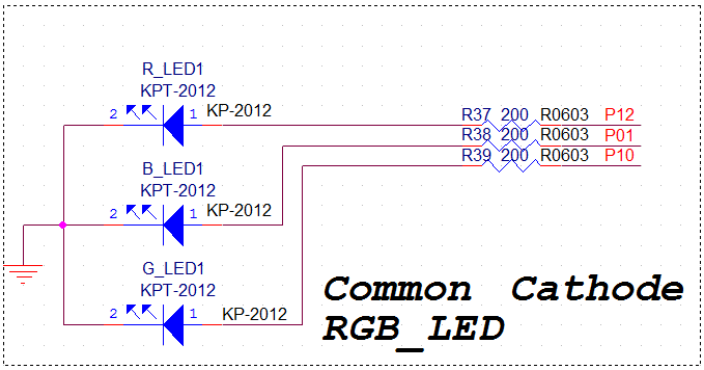

Figure 1-3 Common Cathode RGB Diagram

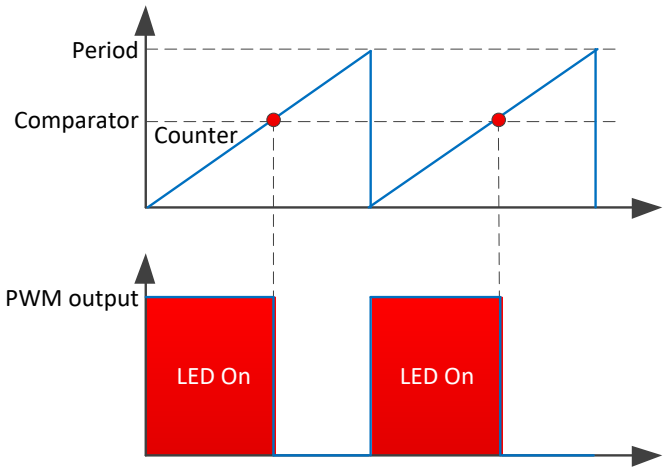

Figure 1-4 PWM and Light Status at Common Cathode RGB

1.1.2 UART Protocol Format

The input command format is shown as Table 1-1. Byte 0, Byte 4, Byte 8 and Byte 12 are fixed characters and need to be exactly the same as the table and the format needs to match 13 Byte. User must strictly follow the format and enter the command correctly to successfully set to the PWM register. Otherwise, an error message will pop up and it will keep the last correct parameter.

For various input command format analysis and display messages, please refer to Table 1-2.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
‘ (’	‘ 0 ’	‘ 0 ’	‘ 0 ’	‘ , ’	‘ 0 ’	‘ 0 ’	‘ 0 ’
Start Byte	LED_R duty(000-100)			Separate Byte	LED_B duty(000-100)		
Byte 8	Byte 9	Byte 10	Byte 11	Byte 12			
‘ , ’	‘ 0 ’	‘ 0 ’	‘ 0 ’	‘) ’			
Separate Byte	LED_G duty(000-100)			End Byte			

Table 1-1 Input Command Format Description

Item	Input	State	Description
1.	(050,100,002)	O	Message: Current Parameter Setting: R_Duty=50, B_Duty=100, G_Duty=2.
2.	(050,100,02)	X	Message: Please enter a command consisting of 13 characters according to the format requirements. Format:(R[000-100],B[000-100],G[000-100]),EX:(050,100,020). Current Parameter Setting: R_Duty=50, B_Duty=100, G_Duty=2.
3.	(050,100,020	X	Message: Please enter a command consisting of 13 characters according to the format requirements. Format:(R[000-100],B[000-100],G[000-100]),EX:(050,100,020). Current Parameter Setting: R_Duty=50, B_Duty=100, G_Duty=2.
4.	(050,100,120)	X	Message: The input parameter range is (0-100). Format:(R[000-100],B[000-100],G[000-100]),EX:(050,100,020). Current Parameter Setting: R_Duty=50, B_Duty=100, G_Duty=2.

Table 1-2 Input Format States Description

1.2 Demo Result

1.2.1 PuTTY Settings

This sample code is matched with “PuTTY” terminal application. Input the correct RGB LED duty format on the PC side to achieve RGB dimming. PuTTY is set as follows:

1. Open PuTTY application and click '**Terminal**'.

2. Click '**Force on**' under '**Local echo**' to enable input value display, as Figure 1-5.

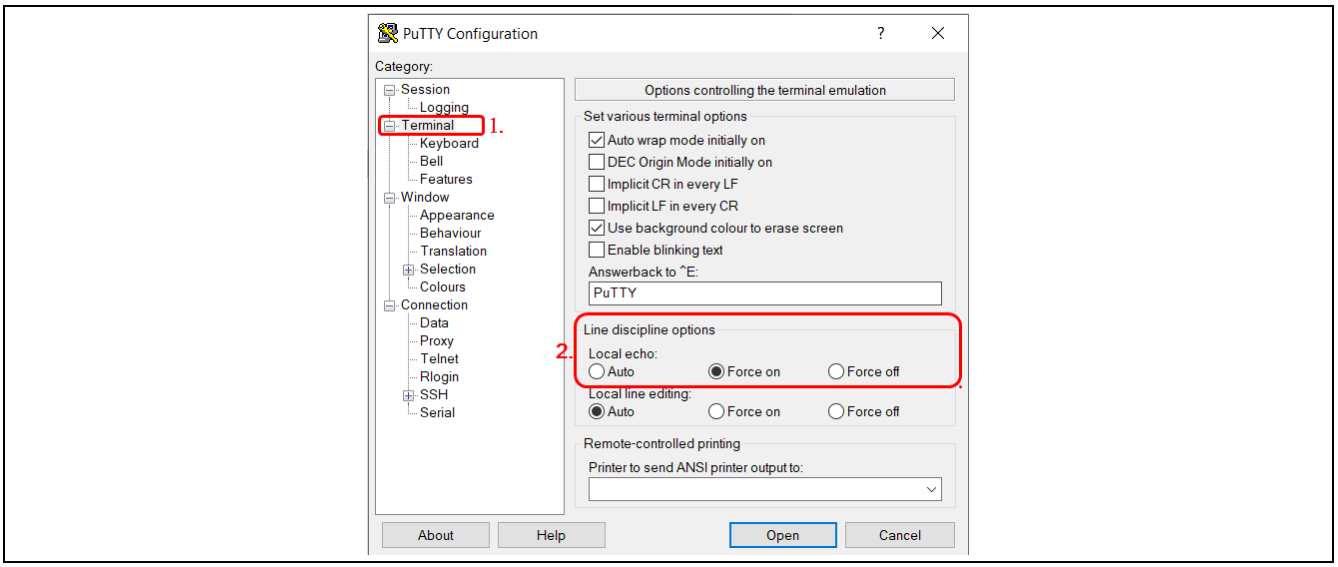

Figure 1-5 Terminal Settings for PuTTY

3. Click '**Session**'.

4. Click '**Serial' under 'Serial line**' then select port number and Baud rate, as Figure 1-6.

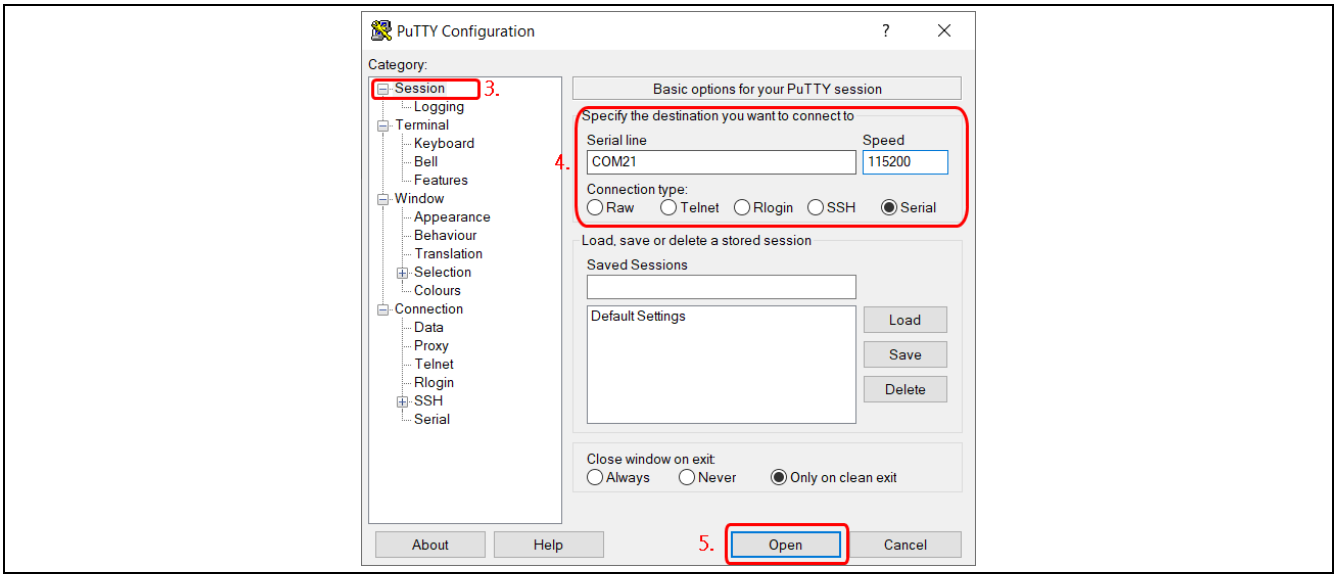

Figure 1-6 Session Setting for PuTTY

1.2.2 Action Window

The demo result is shown in Figure 1-7. The PWM duty can be modified correctly only when the correct format is input. In addition, when the user inputs the first character, the input command needs to be completed within 10 seconds; otherwise it will be judged as a format error.

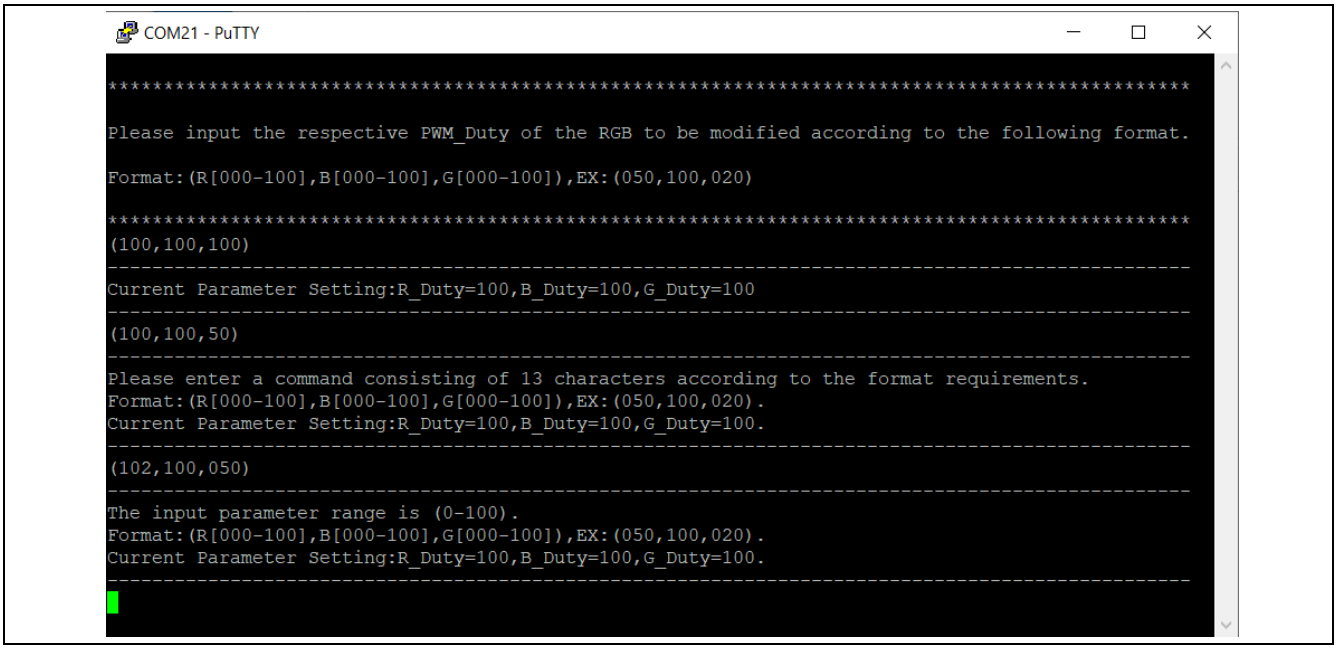

Figure 1-7 Demo Result

2. Code Description

The code instructions are listed below:

1. Initialize the MS51 in main.c, including HIRC, UART, and Timer 0.

```
void main(void)
{
    /************************************************************************/
    /* Clock Setting                                                        */
    /************************************************************************/
    MODIFY_HIRC(HIRC_24);

    /************************************************************************/
    /* Uart Setting                                                         */
    /************************************************************************/
    P06_QUASI_MODE;
    P07_QUASI_MODE;
    UART_Open(24000000, UART0_Timer3, 115200);
    ENABLE_UART0_INTERRUPT;                    /* Enable UART0 interrupt */
    ENABLE_UART0_PRINTF;

    ENABLE_GLOBAL_INTERRUPT;                   /* Global interrupt enable */

    ALL_GPIO_QUASI_MODE;

    /************************************************************************/
    /* PWM Setting                                                          */
    /************************************************************************/
    PWM0_P12_OUTPUT_ENABLE;
    PWM2_P10_OUTPUT_ENABLE;
    PWM4_P01_OUTPUT_ENABLE;


    PWM_IMDEPENDENT_MODE;
    PWM_CLOCK_DIV_8;
    PWMPH = 0x0B;
    PWMPL = 0xB7;
    /************************************************************************
      PWM frequency = Fpwm/((PWMPH,PWMPL) + 1) <Fpwm = Fsys/PWM_CLOCK_DIV>
                    = (24MHz/8)/(0xBB7 + 1)
                    = 1KHz (1ms)
    ************************************************************************/

    /************************************************************************/
    /* Timer0 Setting                                                       */
    /************************************************************************/
    ENABLE_TIMER0_MODE1;
    TIMER0_FSYS_DIV12;
    TH0 = TH0_INIT;
    TL0 = TL0_INIT;
```

2. When the final UART receives a command character, the characters received by UART Serial_ISR() function will be triggered.

```
void Serial_ISR(void) interrupt 4
{
    if (RI)                                //Uart Receive Data
    {
        if (SBUF == 0x0d || SBUF == ')') //hit enter or ')' input is completed
        {
            u32cnt = 0;
```

```
            bTIMER0FLAG = 1;
            clr_TCON_TR0;
        }

        if (u8RVCT < 12)
        {
            if (SBUF == 0x7F || SBUF   == 0X08)          //Delete or Backspace
            {
                u8RVCT--;
            }
            else
            {
                au8UR0RVBUFFER[u8RVCT] = SBUF;
                u8RVCT++;
            }

            if (u8RVCT == 1)
            {
                ENABLE_TIMER0_INTERRUPT;                 /*enable Timer0 interrupt*/
                set_TCON_TR0;
            }

        }
        else if (u8RVCT == 12)
        {
            if (SBUF == 0x7F || SBUF   == 0X08)         //Delete or Backspace
            {
                u8RVCT--;
            }
            else
            {
                au8UR0RVBUFFER[u8RVCT] = SBUF;
                bUART0RVFLAG = 1;
                u8RVCT = 0;
                clr_TCON_TR0;
            }

        }

        clr_SCON_RI;                                    //Clear RI (Receive Interrupt).
    }
}
```

3. Timer0 is responsible for counting the input time. If the input time is too long, it will be judged as a format error.

```
void Timer0_ISR(void) interrupt 1      /*interrupt address is 0x000B */
{
    _push_(SFRS);

    TH0 = TH0_INIT;
    TL0 = TL0_INIT;
    TF0 = 0 ;
    P12 = ~P12;                        /* GPIO toggle when interrupt  */
    u32cnt++;

    if (u32cnt == 1000)                //10ms * 1000 = 10000 ms, /* Input command time
    {
        u32cnt = 0;
        bTIMER0FLAG = 1;
        clr_TCON_TR0;
    }

    bUART0RVFLAG = 0;
    _pop_(SFRS);
```

```
}
```

4. As to format error message, when the input format is wrong, another order will be discarded and a format error message will pop up.

```
void Format_Err(void)
{
    for (u8i = 0; u8i <= 12; u8i++)
    {
        au8UR0RVBUFFER[u8i] = 0;
    }

    u8RVCT = 0;
    bTIMER0FLAG = 0;
    clr_TCON_TR0;
    printf("\n-------------------------------------------------------------------------------
           --------------------\n");
    printf("Please enter a command consisting of 13 characters according to the format
           requirements.");
    printf("\nFormat:(R[000-100],B[000-100],G[000-100]),EX:(050,100,020).");
    printf("\nCurrent Parameter Setting:R_Duty=%bd,B_Duty=%bd,G_Duty=%bd.", u8SysColor_R,
           u8SysColor_B, u8SysColor_G);
    printf("\n-------------------------------------------------------------------------------
           --------------------\n");
}
```

5. Analyze the correctness of the format. If it is correct, change the PWM Duty. Otherwise, keep the original setting without modification.

```
    printf("\n*******************************************************************************
           ********************\n");
    printf("\nPlease input the respective PWM_Duty of the RGB to be modified according to
           the following format.\n");
    printf("\nFormat:(R[000-100],B[000-100],G[000-100]),EX:(050,100,020)\n");

    printf("\n*******************************************************************************
           ********************\n");

    while (1)
    {
        if (bTIMER0FLAG == 1)           //Enter less than 12 characters.
        {
            bTIMER0FLAG = 0;
            Format_Err();
        }

        P14 ^= 1;

        if (bUART0RVFLAG)               //Uart receives completion and starts parsing.
        {
            if (au8UR0RVBUFFER[0] == '(') //Input format parsing.
            {
                if (au8UR0RVBUFFER[4] == ',' && au8UR0RVBUFFER[8] == ',')
                {
                    if (au8UR0RVBUFFER[12] == ')')
                    {
                        u8Color_R = (au8UR0RVBUFFER[1] - 0x30) * 100;
                        u8Color_R += (au8UR0RVBUFFER[2] - 0x30) * 10;
                        u8Color_R += (au8UR0RVBUFFER[3] - 0x30);
                        u8Color_B = (au8UR0RVBUFFER[5] - 0x30) * 100;
                        u8Color_B += (au8UR0RVBUFFER[6] - 0x30) * 10;
                        u8Color_B += (au8UR0RVBUFFER[7] - 0x30);
                        u8Color_G = (au8UR0RVBUFFER[9] - 0x30) * 100;
                        u8Color_G += (au8UR0RVBUFFER[10] - 0x30) * 10;
```

```
                        u8Color_G += (au8UR0RVBUFFER[11] - 0x30);

                        if (u8Color_R <= 100 && u8Color_B <= 100 && u8Color_G <= 100)
                        //Input Parameter range.
                        {
                            bTIMER0FLAG = 0;
                            u8SysColor_R = u8Color_R;
                            u8SysColor_B = u8Color_B;
                            u8SysColor_G = u8Color_G;

                            printf("\n-------------------------------------------------
                                    ------------------------------------------\n");
                            printf("Current Parameter Setting:
                                    R_Duty=%bd,B_Duty=%bd,G_Duty=%bd", u8SysColor_R,
                                    u8SysColor_B, u8SysColor_G);
                            printf("\n-------------------------------------------------
                                    ------------------------------------------\n");

                            PWM0H = HIBYTE((MAKEWORD(PWMPH, PWMPL) / 99) * u8SysColor_R);
                            PWM0L = LOBYTE((MAKEWORD(PWMPH, PWMPL) / 99) * u8SysColor_R);
                            PWM2H = HIBYTE((MAKEWORD(PWMPH, PWMPL) / 99) * u8SysColor_B);
                            PWM2L = LOBYTE((MAKEWORD(PWMPH, PWMPL) / 99) * u8SysColor_B);

                            set_SFRS_SFRPAGE;
                            //PWM4 and PWM5 duty setting is in SFP page 1
                            PWM4H = HIBYTE((MAKEWORD(PWMPH, PWMPL) / 99) * u8SysColor_G);
                            PWM4L = LOBYTE((MAKEWORD(PWMPH, PWMPL) / 99) * u8SysColor_G);
                            clr_SFRS_SFRPAGE;

                           /*************************************************************
                           ********
                             PWM high level duty = (PWM period/((PWMPH,PWMPL) + 1) * duty
                             percentage(%)

                           *************************************************************
                           ********/
#if Com_Pole    /* PWM output inversely enable */
                //Common anode: #define Com_Pole 1 ,Common cathode: #define Com_Pole 0
                            PWM0_OUTPUT_INVERSE;
                            PWM2_OUTPUT_INVERSE;
                            PWM4_OUTPUT_INVERSE;
#endif
                        /*-------- PWM start run--------------*/
                            set_PWMCON0_LOAD;
                            set_PWMCON0_PWMRUN;
                        }
                        else
                        {
                            for (u8i = 0; u8i <= 12; u8i++)
                            {
                                au8UR0RVBUFFER[u8i] = 0;
                            }

                            u8RVCT = 0;
                            bTIMER0FLAG = 0;
                            printf("\n-------------------------------------------------
                                    ------------------------------------------\n");
                            printf("The input parameter range is (0-100).");
                            printf("\nFormat:(R[000-100],B[000-100],G[000-
                                   100]),EX:(050,100,020).");
                            printf("\nCurrent Parameter Setting:
                                   R_Duty=%bd,B_Duty=%bd,G_Duty=%bd.", u8SysColor_R,
                                   u8SysColor_B, u8SysColor_G);
                            printf("\n-------------------------------------------------
                                    ------------------------------------------\n");
                        }
```

```
                    }
                    else
                        Format_Err();
                }
                else
                    Format_Err();
            }
            else
                Format_Err();

            bUART0RVFLAG = 0;
        }
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - MS51_Series_BSP_Keil_V1.00.004
- IDE version
  - Keil uVersion 5.28

3.2 Hardware Requirements

- Circuit components
  - NuTiny-MS51FB V1.1
  - RGB LED
- Pin Connect

V_{DD}	↔	V_{DD}
P12(PWM0_CH0)	↔	R Color
P01(PWM0_CH4)	↔	G Color
P10(PWM0_CH2)	↔	B Color
NuTiny-MS51FB		RGB LED

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

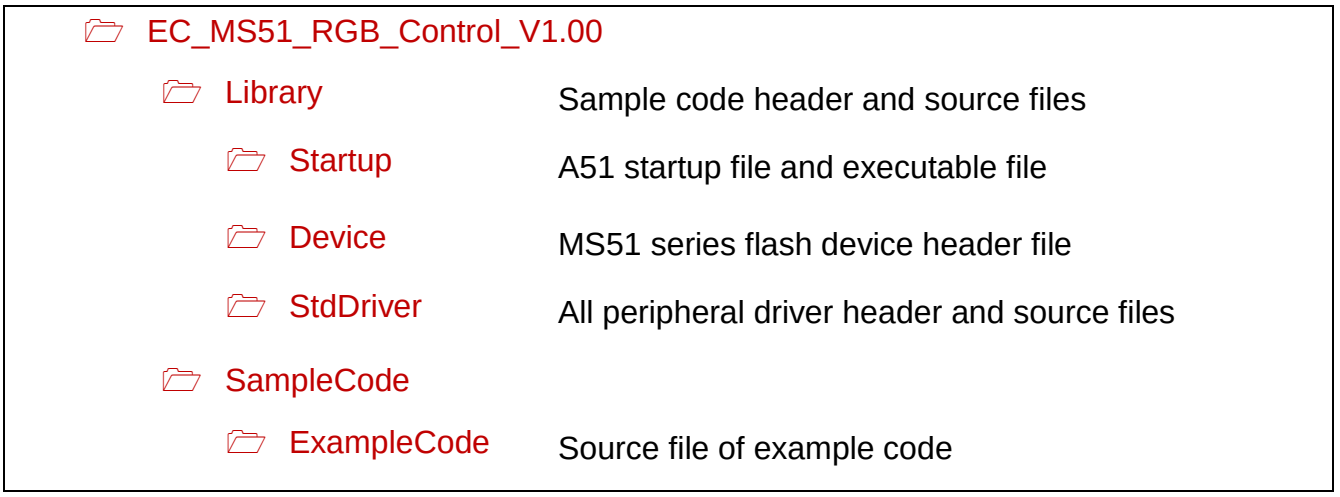

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *MS51_RGB_Control.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
3. Open PuTTY.
   - Press the Reset button.
   - Enter the PWM parameters corresponding to R, B, G according to the format (R[000-100], B[000-100],G[000-100]), e.g. (050,100,020).
   - If the input format is correct, modify PWM duty; otherwise, keep the original parameters without modification.

6. Revision History

Date	Revision	Description
2022.07.06	1.00	Initial version.

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.