MS51 驅動 RGB 7 段 LED

NuMicro® 1T 8051 系列微控制器範例代碼介紹

文件資訊

應用簡述	使用 MS51 32K 系列驅動 RGB 彩色 7 段 LED 顯示
BSP 版本	MS51_Series_BSP_Keil_V1.00.006
開發平臺	NuMaker-MS51PC V1.1 & PWM_ARGB_Control_Board V1.0

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. 概述

1.1 原理

傳統 7 段顯示為單一顏色，使用者只需把正極接電，不同的負極接地就能控制七段顯示不同的數字。

RGB 是色光的色彩模式。R 代表紅色，G 代表綠色，B 代表藍色，三種色彩疊加形成了其它的色彩。因為三種顏色都有 256 個亮度水準級，所以三種色彩疊加就形成 1670 萬種顏色。

所以當 RGB 應用於 7 段 LED 顯示時，不僅需要點亮，還需要控制 RGB 三色不同的色階，達成混合顏色的效果。

本範例使用 ADH-Tech 的 ACD8143 RGB LED 模塊來實現，LED 腳位定義與內部驅動線路如下：共點亮 2 個 RGB LED 分別定義為 DIG. 1、DIG.2 。

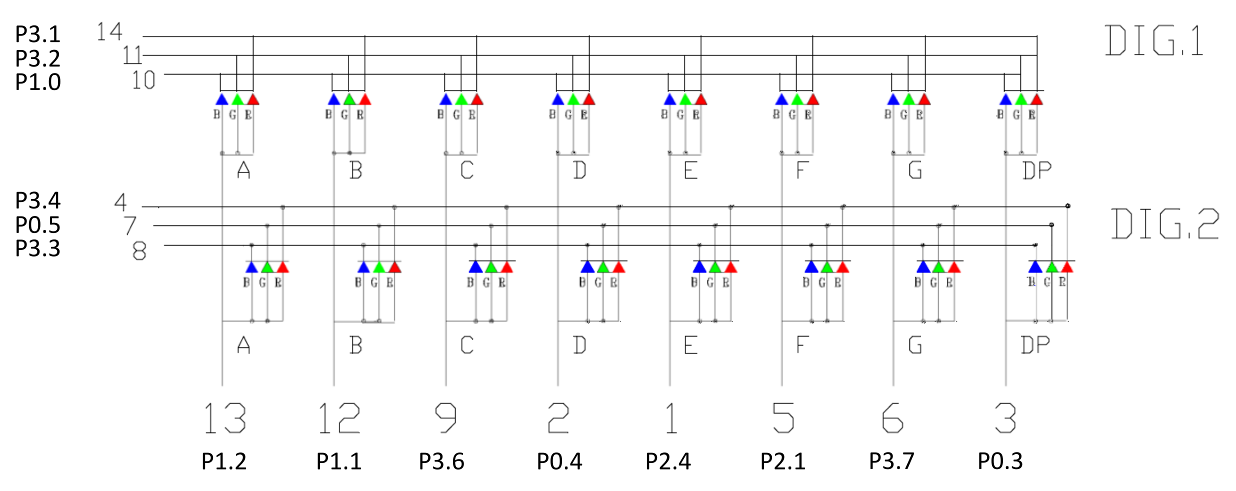


圖 1-1 ACD8143 腳位定義與內部驅動線路

腳位 10\11\14 分別代表需要點亮 DIG.1 的 PWM 色階 B\G\R 控制， 腳位 4\7\8 用於點亮 DIG.2，PWM 輸出控制 255 級色階。

其餘腳位 13\12\9\2\1\5\63 直接以 GPIO 控制點亮，程式內定義名稱為 SEG, 分別對應 SEG_A ~ SEG_G 以及 SEG_DP。

例如當需要點亮 DIG1 的 A 段綠色時，設定 LED 11 腳 PWM 輸出 Duty 定義色階，13 腳 GPIO 定為 1 來進行 Enable SEG 配置。

依據上述定義，配合 NuMaker-MS51PC，設計控制板線路如下：

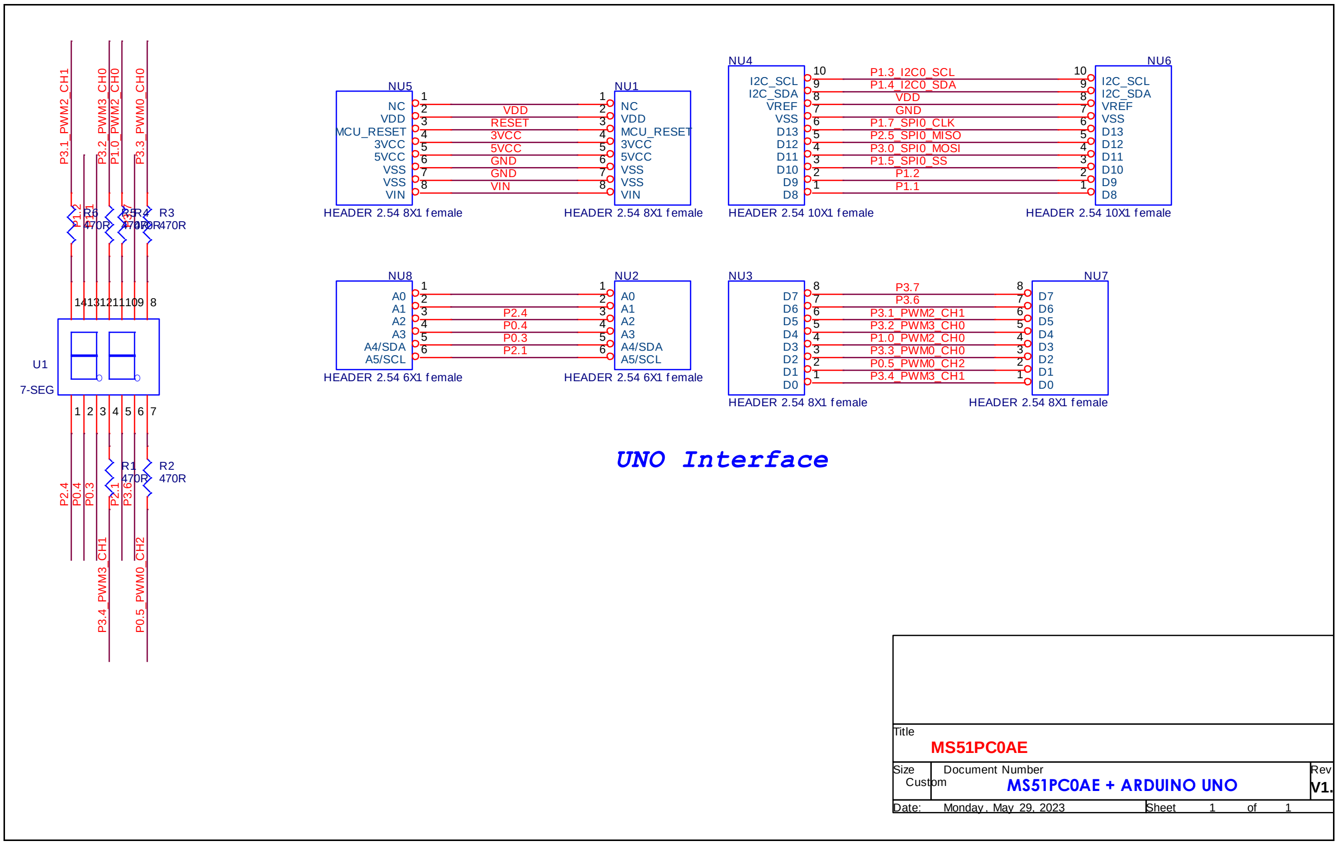


圖 1-1 ACD8143 控制板線路圖

1.2 執行結果

通過 Access Port UART 傳輸固定格式 RGB LED 顯示對應顯示的數字與顏色

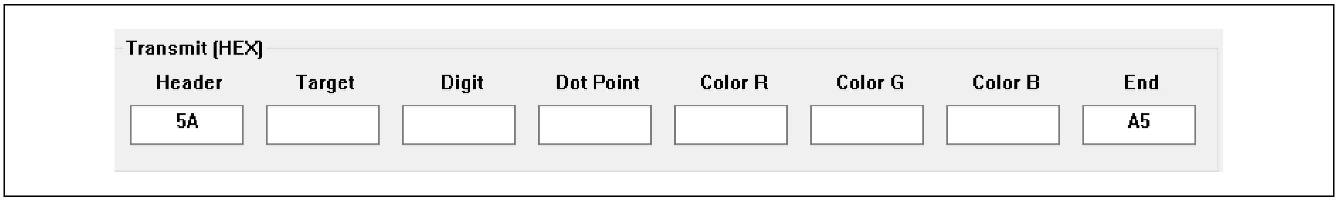
UART 傳輸格式如下，每次傳輸定義一個 7seg

Target: 0 = DIG.1, 1 = DIG.2;

Digit = 0~9, A~F;

Dot Point: 0 = Disable, 1 = Enable;

Color R/G/B = 0x00 ~ 0xFF



Transmit (HEX)

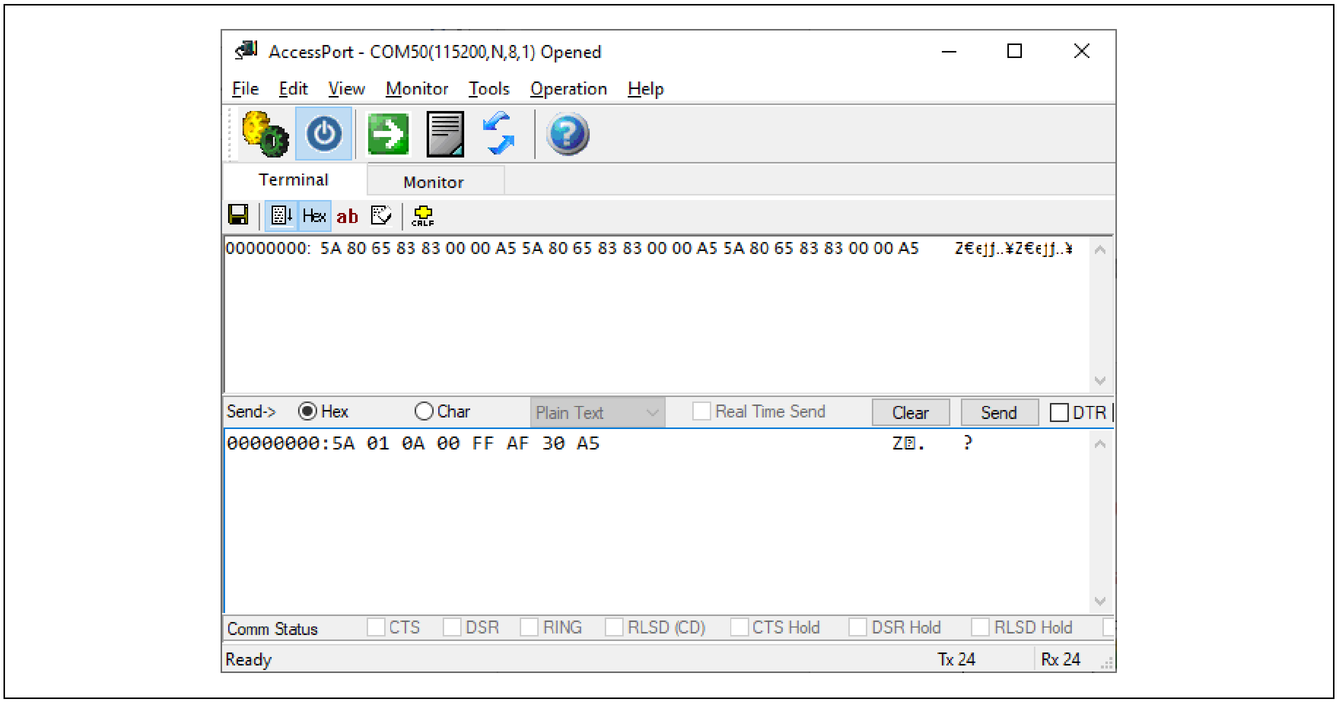
Header	Target	Digit	Dot Point	Color R	Color G	Color B	End
5A							A5

圖 1-2 UART 傳輸格式定義

輸入範例，

第一組輸入 5A 00 03 00 69 AA 00 A5 控制 DIG.1 顯示數字 [3] ，顏色為嫩綠色

第二組輸入 5A 01 0A 00 FF AF 30 A5 控制 DIG.1 顯示字元 [A] ，顏色為嫩黃色



AccessPort - COM50(115200,N,8,1) Opened

```
00000000:  5A 80 65 83 83 00 00 A5 5A 80 65 83 83 00 00 A5 5A 80 65 83 83 00 00 A5      Z€eƒƒ..¥Z€eƒƒ..¥
```

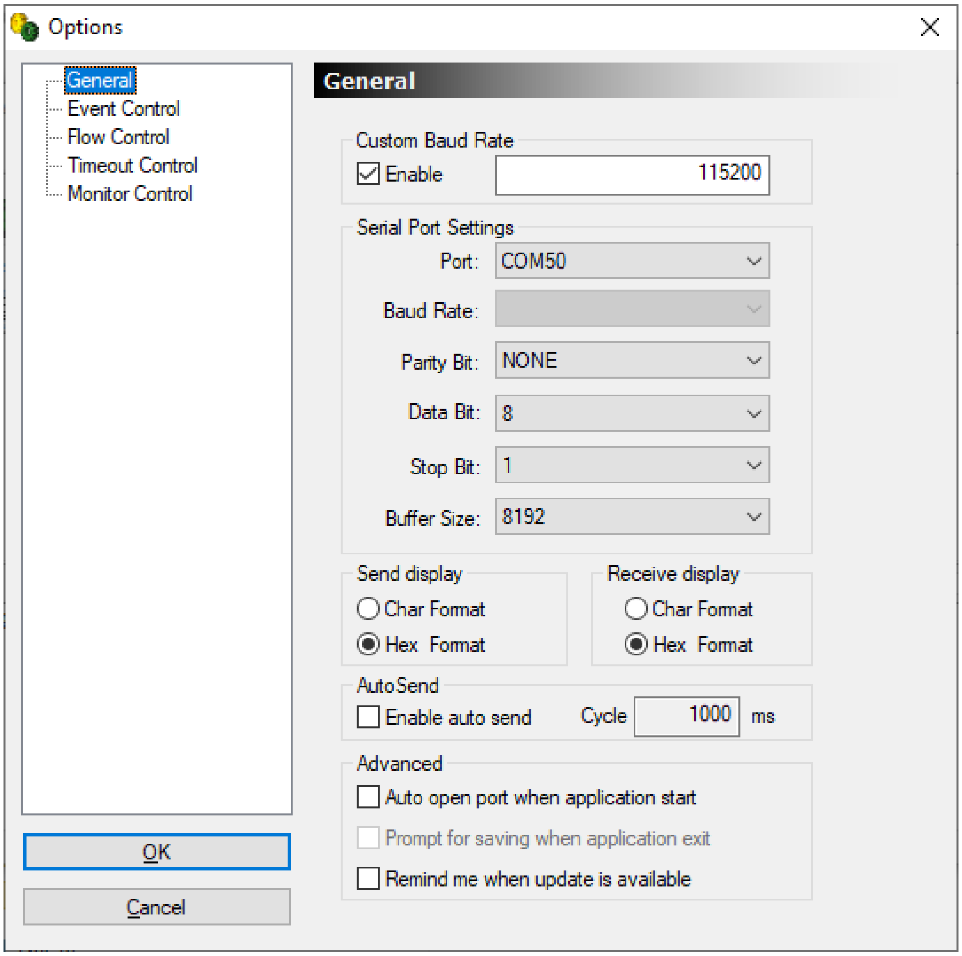
Send-> ● Hex ○ Char

```
00000000:5A 01 0A 00 FF AF 30 A5                    Z□.    ?
```

Comm Status ☐ CTS ☐ DSR ☐ RING ☐ RLSD (CD) ☐ CTS Hold ☐ DSR Hold ☐ RLSD Hold

Ready Tx 24 Rx 24

圖 1-3 UART 輸入與返回值



Options

General

Custom Baud Rate

☑ Enable 115200

Serial Port Settings

Port: COM50

Baud Rate:

Parity Bit: NONE

Data Bit: 8

Stop Bit: 1

Buffer Size: 8192

Send display

○ Char Format

● Hex Format

Receive display

○ Char Format

● Hex Format

AutoSend

☐ Enable auto send Cycle 1000 ms

Advanced

☐ Auto open port when application start

☐ Prompt for saving when application exit

☐ Remind me when update is available

圖 1-4 Access Port 配置示意圖

2. 代碼介紹

計算原代碼位元組數的函式，代碼位於 *main.c*。

定義 7 段顯示器不同顯示字元的 SEG。SEG_DP 在每個 PWM 更新的週期判斷是否需要點亮。

```
uint8_t Show_Char[16][8] =
{
    /* SEG_A, SEG_B, SEG_C, SEG_D, SEG_E, SEG_F, SEG_G */
    {      1,      1,      1,      1,      1,      1,      0},    // 0
    {      0,      1,      1,      0,      0,      0,      0},    // 1
    {      1,      1,      0,      1,      1,      0,      1},    // 2
    {      1,      1,      1,      1,      0,      0,      1},    // 3
    {      0,      1,      1,      0,      0,      1,      1},    // 4
    {      1,      0,      1,      1,      0,      1,      1},    // 5
    {      1,      0,      1,      1,      1,      1,      1},    // 6
    {      1,      1,      1,      0,      0,      0,      0},    // 7
    {      1,      1,      1,      1,      1,      1,      1},    // 8
    {      1,      1,      1,      1,      0,      1,      1},    // 9
    {      1,      1,      1,      0,      1,      1,      1},    // A
    {      0,      0,      1,      1,      1,      1,      1},    // B
    {      1,      0,      0,      1,      1,      1,      0},    // C
    {      0,      1,      1,      1,      1,      0,      1},    // D
    {      1,      0,      0,      1,      1,      1,      1},    // E
    {      1,      0,      0,      0,      1,      1,      1},    // F
};
```

SEG 所用到的 GPIO 初始配置。

```
    ……
void Segment_Initial(void)
{
    /* Set all segment pins to output low */
    SEG_A = SEG_B = SEG_C = SEG_D = \
    SEG_E = SEG_F = SEG_G = SEG_DP = 0;
    /* Set all segment pins to output mode */
    P12_PUSHPULL_MODE;    // SEG_A
    P11_PUSHPULL_MODE;    // SEG_B
    P36_PUSHPULL_MODE;    // SEG_C
    P04_PUSHPULL_MODE;    // SEG_D
    P24_PUSHPULL_MODE;    // SEG_E
    P21_PUSHPULL_MODE;    // SEG_F
    P37_PUSHPULL_MODE;    // SEG_G
    P03_PUSHPULL_MODE;    // SEG_DP

    /* Initial value */
    *(uint8_t *)&Scan_Pos = 0x01;
}
```

色階所用到的PWM初始配置。

```
void PWM_Initial(void)
{
    /* Set all PWM pins to output mode */
    P31_PUSHPULL_MODE;    // Dig1_R
    P32_PUSHPULL_MODE;    // Dig1_G
    P10_PUSHPULL_MODE;    // Dig1_B
    P34_PUSHPULL_MODE;    // Dig2_R
    P05_PUSHPULL_MODE;    // Dig2_G
    P33_PUSHPULL_MODE;    // Dig2_B

    /* Set PWM clock source */
    PWM0_ClockSource(PWM_FSYS, 1);
    PWM123_ClockSource(PWM2, 1);
    PWM123_ClockSource(PWM3, 1);

    /* Set PWM period value */
    /* PWM0 for Dig2_G, Dig2_B */
    SFRS = 0;
    PWM0PH = PWM_FEQ_H;
    PWM0PL = PWM_FEQ_L;
    /* PWM2 for Dig1_R, Dig1_B */
    SFRS = 2;
    PWM2PH = PWM_FEQ_H;
    PWM2PL = PWM_FEQ_L;
    /* PWM3 for Dig1_G, Dig1_G */
    PWM3PH = PWM_FEQ_H;
    PWM3PL = PWM_FEQ_L;

    /* Reset SFRS */
    SFRS = 0;

    /* Enable PWM output */
    ENABLE_PWM2_CH1_P31_OUTPUT;    // Dig1_R
    ENABLE_PWM3_CH0_P32_OUTPUT;    // Dig1_G
    ENABLE_PWM2_CH0_P10_OUTPUT;    // Dig1_B
    ENABLE_PWM3_CH1_P34_OUTPUT;    // Dig2_R
    ENABLE_PWM0_CH2_P05_OUTPUT;    // Dig2_G
    ENABLE_PWM0_CH0_P33_OUTPUT;    // Dig2_B
};
```

每次更新色階，需要重新定義PWM的duty。

```
void Set_PWM_Duty(void)
{
    /* Digi_1 */
    if(((g_u8Scan_Pos == 0x07) && g_Dig_Setting[0].DP) || \
       ((g_u8Scan_Pos < 0x07) && (*(Show_Char[g_Dig_Setting[0].Char] + g_u8Scan_Pos))))
    {
        /* Enable segment */
        SFRS = 2;
        /* Dig1_R */
        PWM2C1H = ((255 - g_Dig_Setting[0].R) >> 4);
        PWM2C1L = ((255 - g_Dig_Setting[0].R) << 4);
        /* Dig1_G */
        PWM3C0H = ((255 - g_Dig_Setting[0].G) >> 4);
        PWM3C0L = ((255 - g_Dig_Setting[0].G) << 4);
        /* Dig1_B */
        PWM2C0H = ((255 - g_Dig_Setting[0].B) >> 4);
        PWM2C0L = ((255 - g_Dig_Setting[0].B) << 4);
    }
    else
    {
        /* Disable segment */
        SFRS = 2;
        /* Dig1_R */
        PWM2C1H = PWM_FEQ_H;
        PWM2C1L = PWM_FEQ_L;
        /* Dig1_G */
        PWM3C0H = PWM_FEQ_H;
        PWM3C0L = PWM_FEQ_L;
        /* Dig1_B */
        PWM2C0H = PWM_FEQ_H;
        PWM2C0L = PWM_FEQ_L;
    }

    /* Digi_2 */
   if(((g_u8Scan_Pos == 0x07) && g_Dig_Setting[1].DP) || \
       ((g_u8Scan_Pos < 0x07) && (*(Show_Char[g_Dig_Setting[1].Char] + g_u8Scan_Pos))))
    {
        /* Enable segment */
        SFRS = 2;
        /* Dig2_R */
        PWM3C1H = ((255 - g_Dig_Setting[1].R) >> 4);
        PWM3C1L = ((255 - g_Dig_Setting[1].R) << 4);
        SFRS = 0;
        /* Dig2_G */
        PWM0C2H = ((255 - g_Dig_Setting[1].G) >> 4);
        PWM0C2L = ((255 - g_Dig_Setting[1].G) << 4);
        /* Dig2_B */
        PWM0C0H = ((255 - g_Dig_Setting[1].B) >> 4);
        PWM0C0L = ((255 - g_Dig_Setting[1].B) << 4);
    }
    else
    {
        /* Disable segment */
        SFRS = 2;
        /* Dig2_R */
        PWM3C1H = PWM_FEQ_H;
        PWM3C1L = PWM_FEQ_L;
        SFRS = 0;
        /* Dig2_B */
        PWM0C0H = PWM_FEQ_H;
        PWM0C0L = PWM_FEQ_L;
        /* Dig2_G */
        PWM0C2H = PWM_FEQ_H;
        PWM0C2L = PWM_FEQ_L;
    }

    /* Reload new setting */
    set_PWM0CON0_LOAD;
    set_PWM2CON0_LOAD;
    set_PWM3CON0_LOAD;

    /* Wait new setting reload */
    while(PWM3CON0 & BIT6);

    /* Reset SFR */
    SFRS = 0;
}
```

主程式設定每 1ms 更新一次所有 GPIO 的狀態，達成循環點亮所有 LED。

完成 LED 輪詢點亮後，重置 Timer 0 定時並等候下一次定時中斷。

```
    while(1)
    {
        if(TF0 == 1)
        {
            /* Stop Timer0 */
            clr_TCON_TR0;

            /* Set all segment pins to output low */
            SEG_A = SEG_B = SEG_C = SEG_D = \
            SEG_E = SEG_F = SEG_G = SEG_DP = 0;

            /* Set PWM duty */
            Set_PWM_Duty();

            /* Scan segment pins */
            SEG_A  = Scan_Pos.A;
            SEG_B  = Scan_Pos.B;
            SEG_C  = Scan_Pos.C;
            SEG_D  = Scan_Pos.D;
            SEG_E  = Scan_Pos.E;
            SEG_F  = Scan_Pos.F;
            SEG_G  = Scan_Pos.G;
            SEG_DP = Scan_Pos.DP;
            (Scan_Pos.DP == 0x01)?(*(uint8_t *)&Scan_Pos = 0x01):(*(uint8_t *)&Scan_Pos
             <<= 1);
            (g_u8Scan_Pos == 0x07)?(g_u8Scan_Pos = 0):(g_u8Scan_Pos++);

            /* Set next time-out */
            TH0 = TH0_INIT;
            TL0 = TL0_INIT;

            /* Clear Timer0 overflow flag */
            TF0 = 0 ;

           /* Start Timer0 */
           set_TCON_TR0;
        }
```

若有 UART 傳輸要求更改顯示內容，判定 UART 接受字串是否完整。接受完成後重新定義 PWM 色階並 UART 回傳更改結果。

```
            /* Check UART0 RX */
            if(UART0_RX_Flag)
            {
                /* Start UART RX */
                if(UART0_RX_TempCount == 0)
```

```
                    UART0_RX_TempCount = UART0_RX_Count;
                else
                {
                    /* UART stops */
                    if(UART0_RX_TempCount == UART0_RX_Count)
                    {
                        UART0_Timeout_Count++;
                        if(UART0_Timeout_Count == 5)
                        {
                            UART0_RX_Flag = 0;
                            UART0_TX_Flag = 1;
                            UART0_Timeout_Count = 0;
                        }
                    }
                    else
                    {
                        UART0_RX_TempCount = UART0_RX_Count;
                        UART0_Timeout_Count = 0;
                    }
                }
            }

        /* Set/Get command */
        if(UART0_TX_Flag)
        {
            /* Check valid data */
            if((UART0_RX_Data[0] == 0x5A) && (UART0_RX_Data[UART0_RX_Ptr-1] == 0xA5))
            {
                /* Set Command */
                if(UART0_RX_Data[1] < DIGI_COUNT)
                {
                    g_Dig_Setting[UART0_RX_Data[1]].Char = UART0_RX_Data[2];
                    g_Dig_Setting[UART0_RX_Data[1]].DP   = UART0_RX_Data[3];
                    g_Dig_Setting[UART0_RX_Data[1]].R    = UART0_RX_Data[4];
                    g_Dig_Setting[UART0_RX_Data[1]].G    = UART0_RX_Data[5];
                    g_Dig_Setting[UART0_RX_Data[1]].B    = UART0_RX_Data[6];

                    /* Transmit reply */
                    DISABLE_UART0_INTERRUPT;
                    for(i = 0; i < 8; i++)
                        UART_Send_Data(UART0, Set_Reply[i]);
                    ENABLE_UART0_INTERRUPT;
                }
            }
            UART0_RX_Count = UART0_RX_Ptr = UART0_RX_TempCount = UART0_TX_Flag = 0;
        }
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - MS51_Series_BSP_Keil_V1.00.006
- IDE 版本
  - Keil uVision5 and PK51 Development Kit V9.52

3.2 硬體需求

- 電路元件
  - NuMaker-MS51PC V1.1
  - PWM_ARGB_Control_Board V1.0
- 線路示意圖
  - 通過 NuMaker-MS51PC Arduino 介面連接，以顯示範例代碼的執行結果。

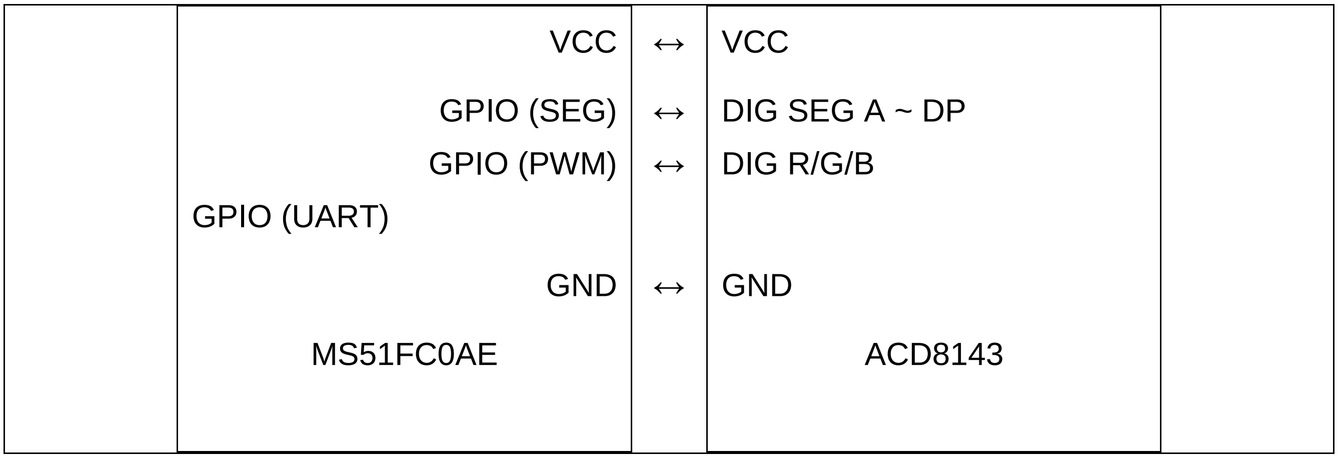


圖 3-1 線路示意圖

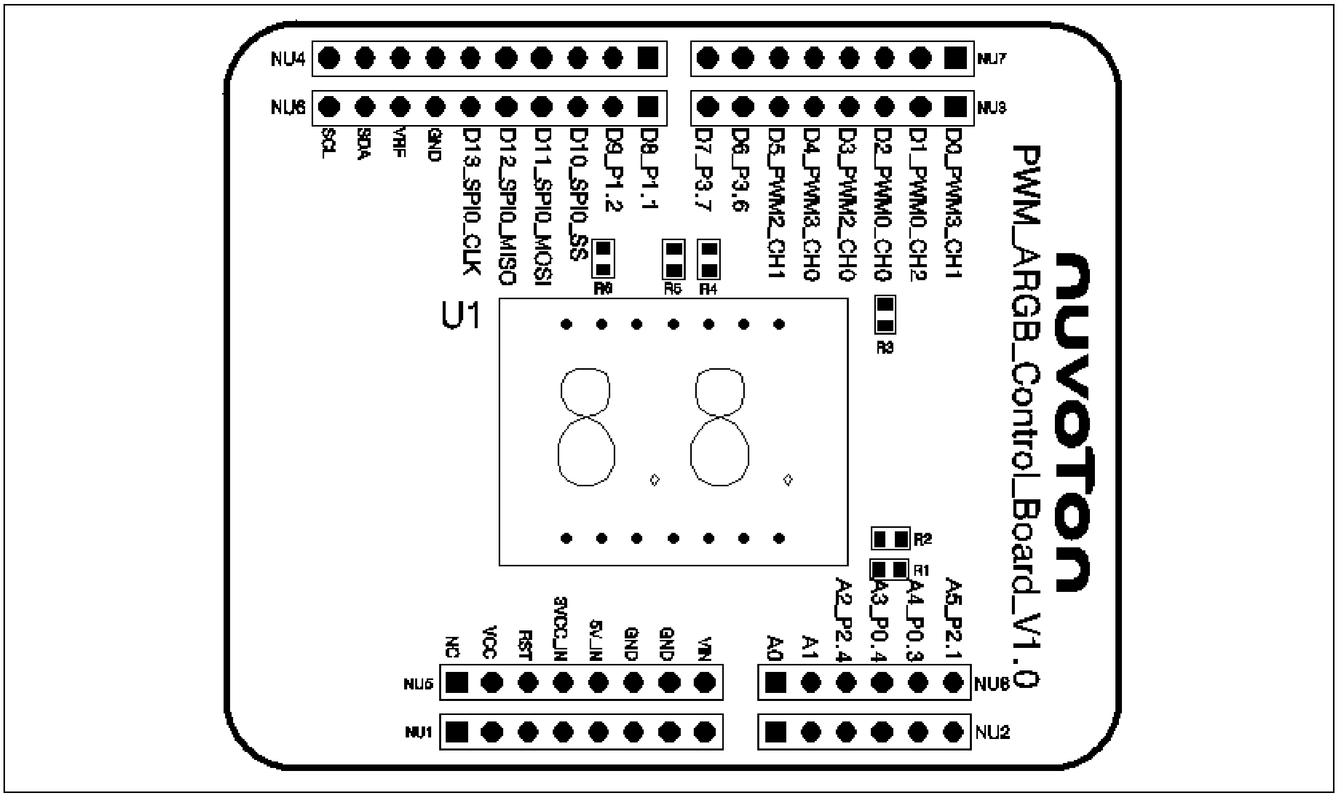


圖 3-2 PWM_ARGB_Control_Board PCB

4. 目錄資訊

🗁 EC_MS51_RGB_7Segment_UART_V1.00	
🗁 Library	Sample code header and source files
🗁 Device	MS51 compliant device header file
🗁 Startup	Startup file for MS51
🗁 StdDriver	All peripheral driver header and source files
🗁 SampleCode	
🗁 ExampleCode	Source file of example code

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *RGB_7Segment_UART.uvproj*。

2. 進入編譯模式介面

   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式

3. 進入除錯模式介面

   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2023.08.11	1.00	初始發布。

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.