ST7301/ ST7302/ ST7303

轉圖程序範例

Ver. 1.3

2018/11/06

ST7301 SERIES .NET C# 轉圖程式範例

目錄

1. 64 色 （64-COLOR，TYPE 2） 寫圖

- **格式**

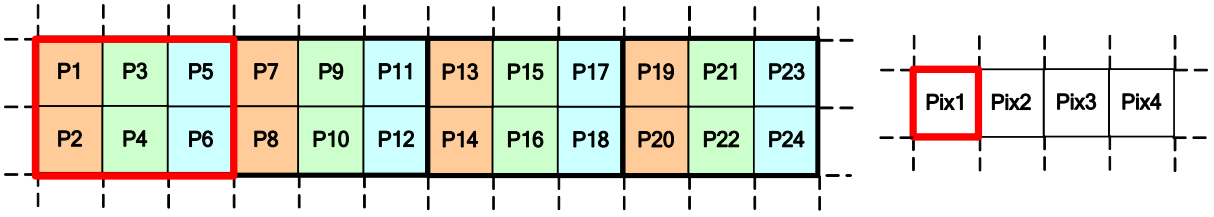
  TYPE2: 三筆 DATA 寫入 IC，才會有效的顯示於螢幕上 (P1~P24 為一單位)

TYPE2: There are 3 write operations for 24-bit data. (Set by "BPS=1" of command 0x3Ah)

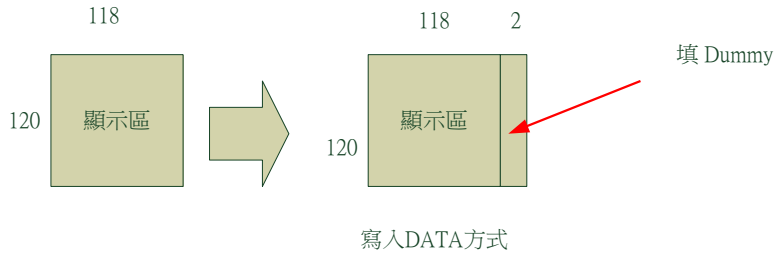
Command	A0	D7	D6	D5	D4	D3	D2	D1	D0
DDRAM write	0	0	0	1	0	1	1	0	0
1st write	1	P1	P2	P3	P4	P5	P6	P7	P8
2nd write	1	P9	P10	P11	P12	P13	P14	P15	P16
3rd write	1	P17	P18	P19	P20	P21	P22	P23	P24

64Color:

64 Color (Set by "MC8=0" of command 0x3Ah)

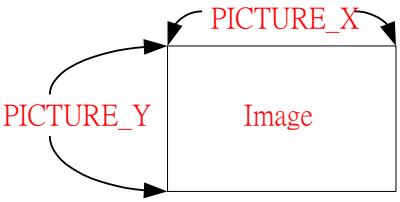


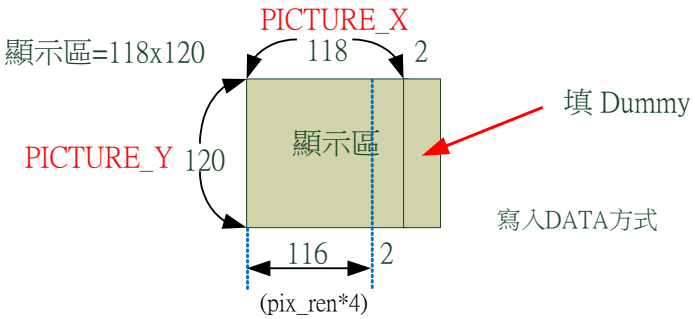
- X方向，不可整除處理



118/4 不可整除，轉換圖檔時，必須118+2=120，才可被4整除，2 列(119,120)這空間必須填Dummy，在以下程式範例P13~P24 填0。若未做此Dummy，就會有畫面異常現象。

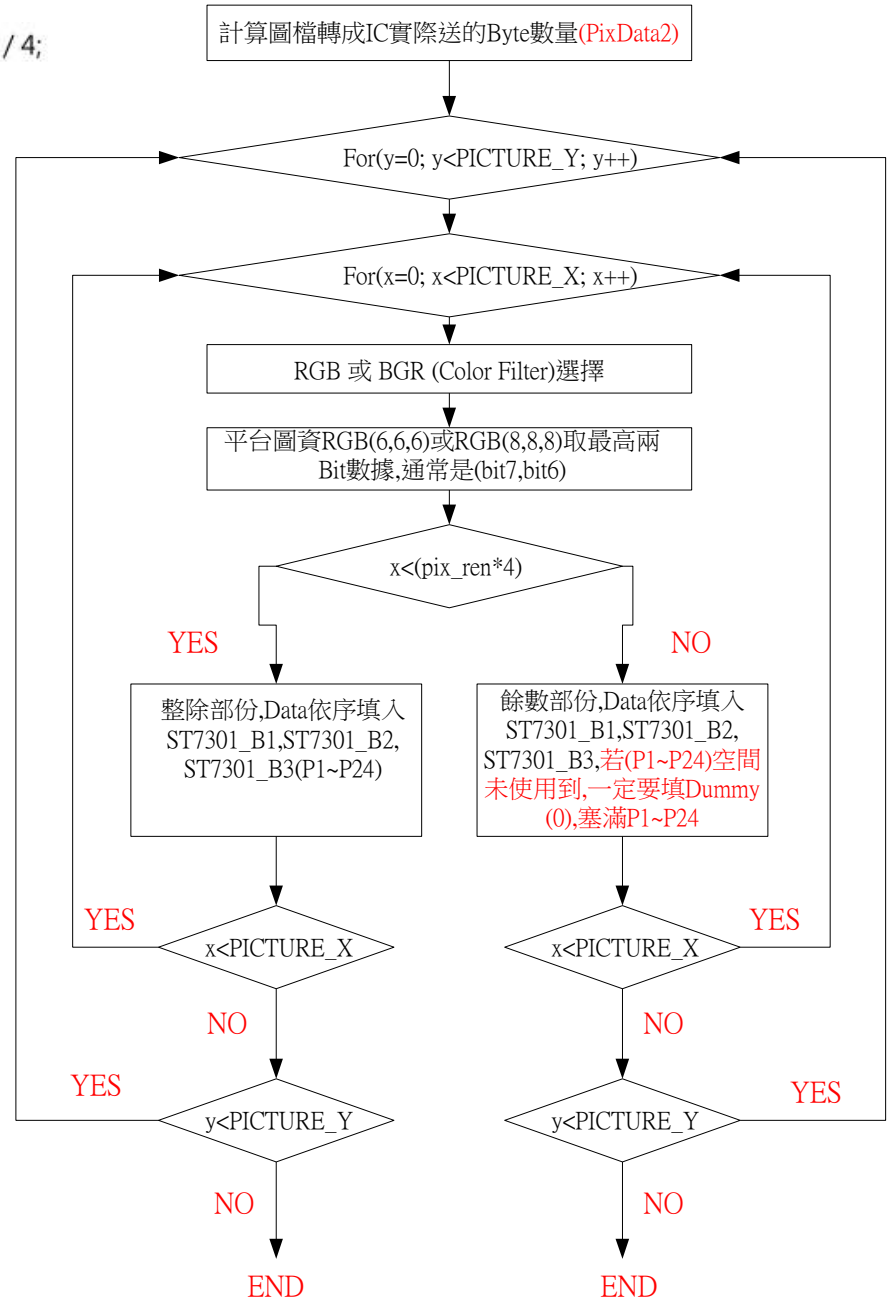
- **程式流程**






```
pix_ren = PICTURE_X / 4;

if (x < (pix_ren * 4))
{
    整除部份,Data依序填入
    ST7301_B1,ST7301_B2,
    ST7301_B3(P1~P24)
}
else
{
    餘數部份,Data依序填入
    ST7301_B1,ST7301_B2,
    ST7301_B3,若(P1~P24)空間
    未使用到,一定要填Dummy
    (0),塞滿P1~P24
}
```



- **程式流程說明**

Step1: 先算出轉成IC實際Data數量，包含無法整除的Data數量，轉換後全部Data存在PixData2暫存器

Step2: 轉圖y地址設定，第一層迴圈

Step3: 轉圖x地址設定，第二層迴圈，所有的轉圖都在此迴圈內完成

Step4: LCM Color filter 選擇 (RGB or BGR)

平臺提供RGB (6,6,6)或RGB(8,8,8)圖資，取最高兩bit(b7,b6)資料

Step5: 可整除部分移入ST7301_B1，ST7301_B2，ST7301_B3 暫存器(依序移入P1~P24)

Step6: 不可整除部分移入ST7301_B1，ST7301_B2，ST7301_B3 暫存器，有Dummy部分寫入0x00(依序移入P1~P24)

Step7: 重複Step2~Step6，直到每點轉換完成

- **程式範例**

```
int pix_ren = PICTURE_X / 4;
 total_byte = PICTURE_X * PICTURE_Y * 0.75f;     //Step 1

 if ((PICTURE_X % 4) == 0)
 {
      remainder = 0;
 }
 else
 {
      double SEG_remainder = PICTURE_Y * (4 - (PICTURE_X % 4)) * 0.75;

      remainder = (int)SEG_remainder + 1;   //無法整除部分
 }

byte[] PixData2 = new byte[(int)total_byte + remainder];   //總共要送給IC Byte數量
int count = 0;
int temp = 0;
int temp1 = 0;
int temp_R, temp_G, temp_B;
int TYPE_Count = 0;

for (int y = 0; y < PICTURE_Y; y++)   // Step2: 轉圖y地址設定
 {
      for (int x = 0; x < PICTURE_X; x++) // Step3: 轉圖x地址設定
      {

           if (checkBox36.Checked == true)   // Step4: Color Filter RGB 或 BGR
           {
                temp = pic.GetPixel(x, y).R;
                temp = temp & 0xC0;
                temp_R = temp;

                temp1 = pic.GetPixel(x, y).G;
                temp1 = temp1 & 0xC0;
                temp_G = temp1;
```

```
                temp1 = temp1 >> 2;
                temp = temp | temp1;

                temp1 = pic.GetPixel(x, y).B;
                temp1 = temp1 & 0xC0;
                temp_B = temp1;
                temp1 = temp1 >> 4;
                temp = temp | temp1;
            }

            else
            {

                temp = pic.GetPixel(x, y).B;
                temp = temp & 0xC0;
                temp_R = temp;

                temp1 = pic.GetPixel(x, y).G;
                temp1 = temp1 & 0xC0;
                temp_G = temp1;
                temp1 = temp1 >> 2;
                temp = temp | temp1;

                temp1 = pic.GetPixel(x, y).R;
                temp1 = temp1 & 0xC0;
                temp_B = temp1;
                temp1 = temp1 >> 4;
                temp = temp | temp1;
            }
pic.SetPixel(x, y, Color.FromArgb(temp_R, temp_G, temp_B)); //實際轉出的圖，顯示在螢幕上
             if (TYPE_Count == 4)
                TYPE_Count = 0;

              if (x < (pix_ren * 4))    // Step5可整除部分
            {

                if (TYPE_Count == 0)
```

```
                {
                    ST7301_B1 = temp;
                    ST7301_B1 = ST7301_B1 & 0xFC;

                }

                if (TYPE_Count == 1)
                {
                    ST7301_B1 = ((temp >> 6) & 0x03) | ST7301_B1;
                    ST7301_B2 = temp;
                    ST7301_B2 = (ST7301_B2 << 2) & 0xF0;
                    PixData2[count] = (byte)ST7301_B1;
                    count++;

                }

                if (TYPE_Count == 2)
                {
                    ST7301_B2 = ((temp >> 4) & 0x0F) | ST7301_B2;
                    ST7301_B3 = (temp << 4) & 0xC0;
                    PixData2[count] = (byte)ST7301_B2;
                    count++;

                }

                if (TYPE_Count == 3)
                {

                    ST7301_B3 = ((temp >> 2) & 0x3F) | ST7301_B3;
                    PixData2[count] = (byte)ST7301_B3;
                    count++;

                }
            }
else // Step6:不可整除部分，無此Source輸出， (若P1~P24未使用到，一定要填0 (Dummy) 塞滿)
            {
                if (((x + 1) % 4) == 1)
```

```
{
    ST7301_B1 = temp;
    ST7301_B1 = ST7301_B1 & 0xFC;

    if ((x + 1) == PICTURE_X) //是否是X最後一筆
    {
        PixData2[count] = (byte)ST7301_B1;
        count++;

        PixData2[count] = 0x00;
        count++;

        PixData2[count] = 0x00;
        count++;
    }
}

if (((x + 1) % 4) == 2)
{
    ST7301_B1 = ((temp >> 6) & 0x03) | ST7301_B1;
    ST7301_B2 = temp;
    ST7301_B2 = (ST7301_B2 << 2) & 0xF0;

    PixData2[count] = (byte)ST7301_B1;
    count++;

    if ((x + 1) == PICTURE_X)
    {

        PixData2[count] = (byte)ST7301_B2;
        count++;

        PixData2[count] = 0x00;
        count++;
    }
}
```

```
                    if (((x + 1) % 4) == 3)
                    {
                        ST7301_B2 = ((temp >> 4) & 0x0F) | ST7301_B2;
                        ST7301_B3 = (temp << 4) & 0xC0;

                        PixData2[count] = (byte)ST7301_B2;
                        count++;


                        if ((x + 1) == PICTURE_X)
                        {

                            PixData2[count] = (byte)ST7301_B3;
                            count++;
                        }
                    }


                        TYPE_Count = 3;

                }//if (x < (pix_ren * 4)) else

                    TYPE_Count++;
            }//for (int x = 0; x < PICTURE_X; x++)
        }
    USB.wrdata_bulk(PixData2); //轉好的DATA，寫入ST7301
```

2. 黑白（MONO 2-COLOR，TYPE 2） 寫圖

- **格式**

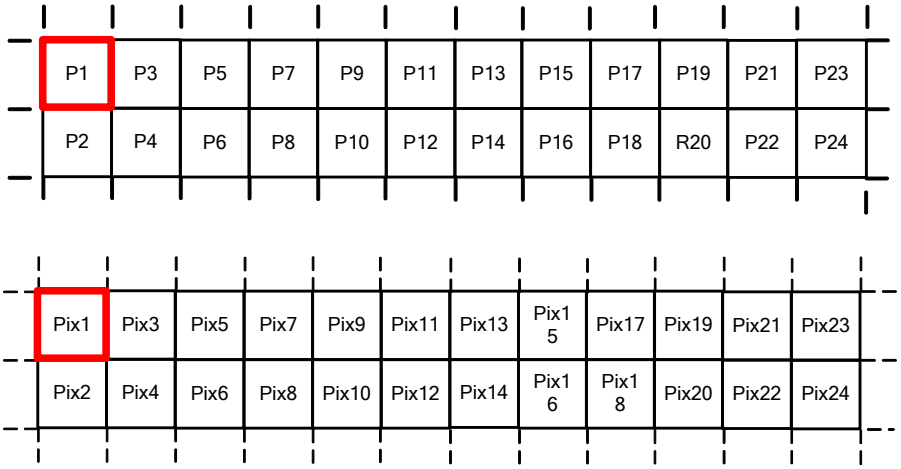
  Type2: 三筆 DATA 寫入 IC，才會有效的顯示於螢幕上 (P1~P24 為一單位)

TYPE2: There are 3 write operations for 24-bit data. (Set by “BPS=1” of command 0x3Ah)

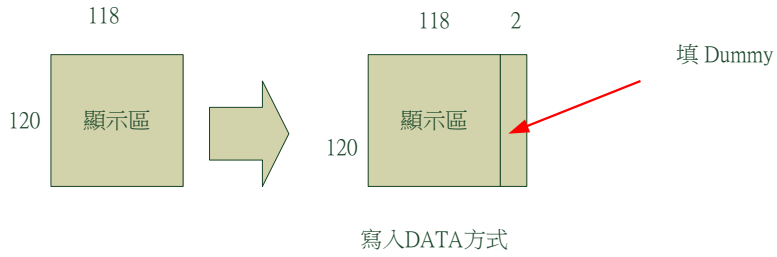
Command	A0	D7	D6	D5	D4	D3	D2	D1	D0
DDRAM write	0	0	0	1	0	1	1	0	0
1st write	1	P1	P2	P3	P4	P5	P6	P7	P8
2nd write	1	P9	P10	P11	P12	P13	P14	P15	P16
3rd write	1	P17	P18	P19	P20	P21	P22	P23	P24

Mono 2-Color:

Mono (Set by “MC8=0” of command 0x3Ah)

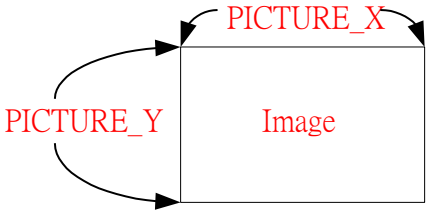


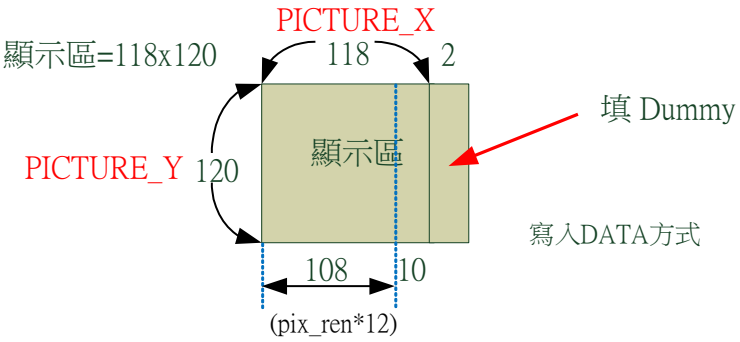
- X方向，不可整除處理



118/12 不可整除，轉換圖檔時，必須118+2=120，才可被12整除，2(119,120)這空間必須填Dummy，在以下程式範例P21~P24 填0。若未做此Dummy，就會有畫面異常現象。

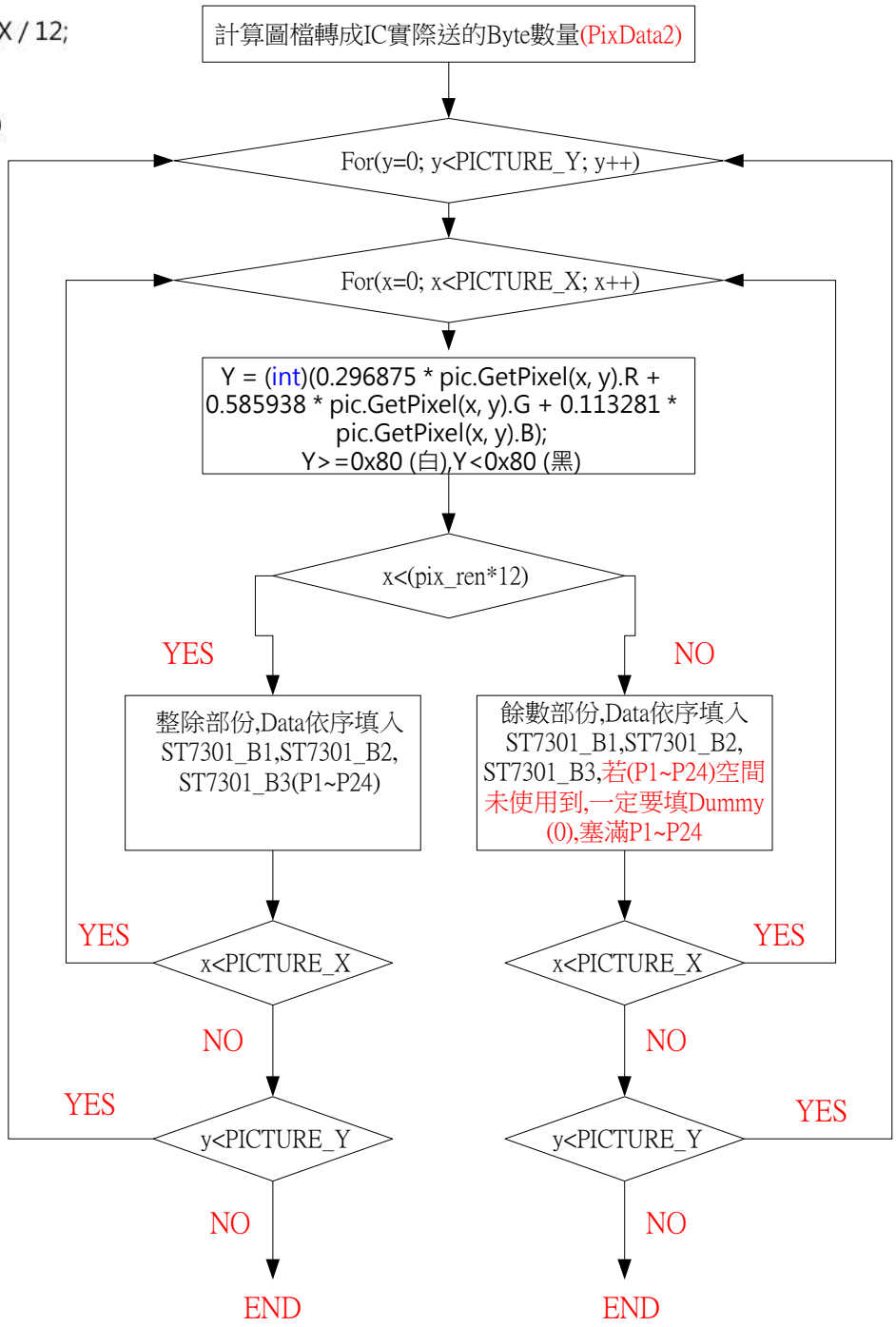
- 程式流程






```
pix_ren = PICTURE_X / 12;

if (x < (pix_ren * 12))
{
    整除部份,Data依序填入
    ST7301_B1,ST7301_B2,
    ST7301_B3(P1~P24)
}
else
{
    餘數部份,Data依序填入
    ST7301_B1,ST7301_B2,
    ST7301_B3,若(P1~P24)空間
    未使用到,一定要填Dummy
    (0),塞滿P1~P24
}
```



- **程式流程說明**

Step1: 先算出轉成IC實際Data數量，包含無法整除的Data數量，轉換後全部Data存在PixData2暫存器

Step2: 轉圖y地址設定，第一層迴圈

Step3: 轉圖x地址設定，第二層迴圈

所有的轉圖都在此迴圈內完成

Step4: 運用公式，將圖點轉成黑白

Step5: 可整除部分移入ST7301_B1，ST7301_B2，ST7301_B3 暫存器(依序移入P1~P24)

Step6: 不可整除部分移入ST7301_B1，ST7301_B2，ST7301_B3 暫存器，有Dummy部分寫入0x00(依序移入P1~P24)

Step7: 重複Step2~Step6，直到每點轉換完成

- 程式範例

```
if (PICTURE_Y % 2 != 0)
 {
      MessageBox.Show("Gate must be even");
      return;
 }
int total_byte;
      int pix_ren = PICTURE_X / 12;

      if ((PICTURE_X % 12) == 0)
      {
           total_byte = (PICTURE_X * PICTURE_Y) / 8;
      }
      else
      {
           int to = PICTURE_X / 12;
           to = to * 12;
           total_byte = ((to + 12) * PICTURE_Y) / 8;
      }

      float TYPE2_Size = (PICTURE_X * PICTURE_Y) / 8;
      byte[] PixData2 = new byte[(int)total_byte];   //Step1  總共byte數量
      int count = 0;
      int Y0, Y1;
      int pixelvalue = 0;
      int TYPE_Count = 0;
      for (int y = 0; y < PICTURE_Y; y = y + 2)   //Step2  轉圖y地址設定
      {
           for (int x = 0; x < PICTURE_X; x++)   //Step3  轉圖x地址設定
      {
Y0 = (int)(0.296875 * pic.GetPixel(x, y).R + 0.585938 * pic.GetPixel(x, y).G + 0.113281 * pic.GetPixel(x, y).B);
//Step4:運用公式，將圖點轉成黑白
Y1 = (int)(0.296875 * pic.GetPixel(x, y + 1).R + 0.585938 * pic.GetPixel(x, y + 1).G + 0.113281 * pic.GetPixel(x, y + 1).B);
pic.SetPixel(x, y, Color.FromArgb(Y0 & 0x80, Y0 & 0x80, Y0 & 0x80)); //顯示轉換後圖至PC
pic.SetPixel(x, y + 1, Color.FromArgb(Y1 & 0x80, Y1 & 0x80, Y1 & 0x80));
```

```
Y0 = Y0 & 0x80;
Y1 = Y1 & 0x80;
if (TYPE_Count == 12)
    TYPE_Count = 0;
if (x < (pix_ren * 12))   //Step5 可整除部分
{
    if (TYPE_Count == 0)
    {
        pixelvalue = Y0;
        pixelvalue = (Y1 >> 1) | pixelvalue;
    }
    if (TYPE_Count == 1)
    {
        pixelvalue = (Y0 >> 2) | pixelvalue;
        pixelvalue = (Y1 >> 3) | pixelvalue;
    }
    if (TYPE_Count == 2)
    {
        pixelvalue = (Y0 >> 4) | pixelvalue;
        pixelvalue = (Y1 >> 5) | pixelvalue;
    }
    if (TYPE_Count == 3)
    {
        pixelvalue = (Y0 >> 6) | pixelvalue;
        pixelvalue = (Y1 >> 7) | pixelvalue;
        PixData2[count] = (byte)pixelvalue;
        count++;
    }
    if (TYPE_Count == 4)
    {
        pixelvalue = Y0;
        pixelvalue = (Y1 >> 1) | pixelvalue;
    }
    if (TYPE_Count == 5)
    {
        pixelvalue = (Y0 >> 2) | pixelvalue;
        pixelvalue = (Y1 >> 3) | pixelvalue;
```

```
            }
            if (TYPE_Count == 6)
            {
                pixelvalue = (Y0 >> 4) | pixelvalue;
                pixelvalue = (Y1 >> 5) | pixelvalue;
            }
            if (TYPE_Count == 7)
            {
                pixelvalue = (Y0 >> 6) | pixelvalue;
                pixelvalue = (Y1 >> 7) | pixelvalue;
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
            if (TYPE_Count == 8)
            {
                pixelvalue = Y0;
                pixelvalue = (Y1 >> 1) | pixelvalue;
            }
            if (TYPE_Count == 9)
            {
                pixelvalue = (Y0 >> 2) | pixelvalue;
                pixelvalue = (Y1 >> 3) | pixelvalue;
            }
            if (TYPE_Count == 10)
            {
                pixelvalue = (Y0 >> 4) | pixelvalue;
                pixelvalue = (Y1 >> 5) | pixelvalue;
            }
            if (TYPE_Count == 11)
            {
                pixelvalue = (Y0 >> 6) | pixelvalue;
                pixelvalue = (Y1 >> 7) | pixelvalue;
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
        }
        else
```

```
{   //Step 6不可整除部分
    if (((x + 1) % 12) == 1)
    {
        pixelvalue = Y0;
        pixelvalue = (Y1 >> 1) | pixelvalue;
        if ((x + 1) == PICTURE_X) //是否是X最後一筆
        {
            PixData2[count] = (byte)pixelvalue;
            count++;
            PixData2[count] = 0x00;
            count++;
            PixData2[count] = 0x00;
            count++;
        }
    }
    if (((x + 1) % 12) == 2)
    {
        pixelvalue = (Y0 >> 2) | pixelvalue;
        pixelvalue = (Y1 >> 3) | pixelvalue;
        if ((x + 1) == PICTURE_X)
        {
            PixData2[count] = (byte)pixelvalue;
            count++;
            PixData2[count] = 0x00;
            count++;
            PixData2[count] = 0x00;
            count++;
        }
    }
    if (((x + 1) % 12) == 3)
    {
        pixelvalue = (Y0 >> 4) | pixelvalue;
        pixelvalue = (Y1 >> 5) | pixelvalue;
        if ((x + 1) == PICTURE_X)
        {
            PixData2[count] = (byte)pixelvalue;
            count++;
```

```
                PixData2[count] = 0x00;
                count++;
                PixData2[count] = 0x00;
                count++;
            }
        }
        if (((x + 1) % 12) == 4)
        {
            pixelvalue = (Y0 >> 6) | pixelvalue;
            pixelvalue = (Y1 >> 7) | pixelvalue;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = (byte)pixelvalue;
                count++;
                PixData2[count] = 0x00;
                count++;
                PixData2[count] = 0x00;
                count++;
            }
            else
            {
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
        }
        if (((x + 1) % 12) == 5)
        {
            pixelvalue = Y0;
            pixelvalue = (Y1 >> 1) | pixelvalue;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = (byte)pixelvalue;
                count++;
                PixData2[count] = 0x00;
                count++;
            }
        }
```

```
if (((x + 1) % 12) == 6)
{
    pixelvalue = (Y0 >> 2) | pixelvalue;
    pixelvalue = (Y1 >> 3) | pixelvalue;
    if ((x + 1) == PICTURE_X)
    {
        PixData2[count] = (byte)pixelvalue;
        count++;
        PixData2[count] = 0x00;
        count++;
    }
}
if (((x + 1) % 12) == 7)
{
    pixelvalue = (Y0 >> 4) | pixelvalue;
    pixelvalue = (Y1 >> 5) | pixelvalue;
    if ((x + 1) == PICTURE_X)
    {
        PixData2[count] = (byte)pixelvalue;
        count++;
        PixData2[count] = 0x00;
        count++;
    }
}
if (((x + 1) % 12) == 8)
{
    pixelvalue = (Y0 >> 6) | pixelvalue;
    pixelvalue = (Y1 >> 7) | pixelvalue;
    if ((x + 1) == PICTURE_X)
    {
        PixData2[count] = (byte)pixelvalue;
        count++;
        PixData2[count] = 0x00;
        count++;
   }
    else
    {
```

```
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
        }
        if (((x + 1) % 12) == 9)
        {
            pixelvalue = Y0;
            pixelvalue = (Y1 >> 1) | pixelvalue;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
        }
        if (((x + 1) % 12) == 10)
        {
            pixelvalue = (Y0 >> 2) | pixelvalue;
            pixelvalue = (Y1 >> 3) | pixelvalue;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
        }
        if (((x + 1) % 12) == 11)
        {
            pixelvalue = (Y0 >> 4) | pixelvalue;
            pixelvalue = (Y1 >> 5) | pixelvalue;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
        }
        TYPE_Count = 11;
    }// if (x < (pix_ren * 12)) else
TYPE_Count++;
```

```
            }//for (int x = 0; x < PICTURE_X; x++)
        }
    USB.wrdata_bulk(PixData2); //轉好完成Data送到ST7301 RAM
```

3. 黑白紅（B/W/R 3-COLOR，TYPE 2） 寫圖

- **格式**

Type2: 三筆 DATA 寫入 IC，才會有效的顯示於螢幕上 (P1~P24 為一單位)

TYPE2: There are 3 write operations for 24-bit data. (Set by "BPS=1" of command 0x3Ah)

Command	A0	D7	D6	D5	D4	D3	D2	D1	D0
DDRAM write	0	0	0	1	0	1	1	0	0
1st write	1	P1	P2	P3	P4	P5	P6	P7	P8
2nd write	1	P9	P10	P11	P12	P13	P14	P15	P16
3rd write	1	P17	P18	P19	P20	P21	P22	P23	P24

B/W/R 3-Color:

黑白紅 (Set by "MC8=0" of command 0x3Ah)

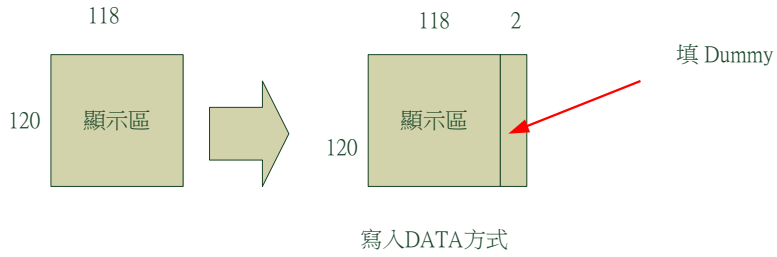
P1 黑/白	P3 紅/黑	P5 黑/白	P7 紅/黑	P9 黑/白	P11 紅/黑	P13 黑/白	P15 紅/黑	P17 黑/白	P19 紅/黑	P21 黑/白	P23 紅/黑
P2 黑/白	P4 紅/黑	P6 黑/白	P8 紅/黑	P10 黑/白	P12 紅/黑	P14 黑/白	P16 紅/黑	P18 黑/白	P20 紅/黑	P22 黑/白	P24 紅/黑

程式範例: if (checkBox36.Checked == true) **//判斷 黑/白 與 紅/黑 位置**

(Color Filter)

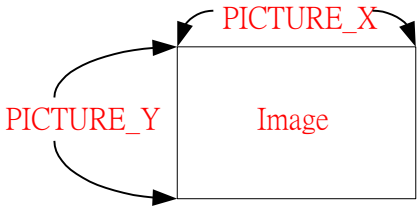
Pix1	Pix3	Pix5	Pix7	Pix9	Pix11
Pix2	Pix4	Pix6	Pix8	Pix10	Pix12

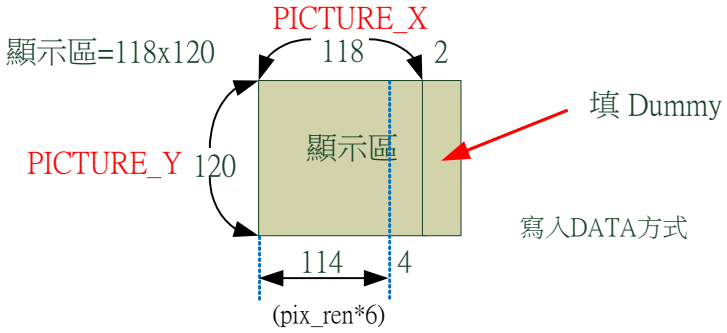
- X方向，不可整除處理



118/6 不可整除，轉換圖檔時，必須118+2=120，才可被6整除，2 (119,120)這空間必須填Dummy，在以下程式範例P17~P24 填0。若未做此Dummy，就會有畫面異常現象。

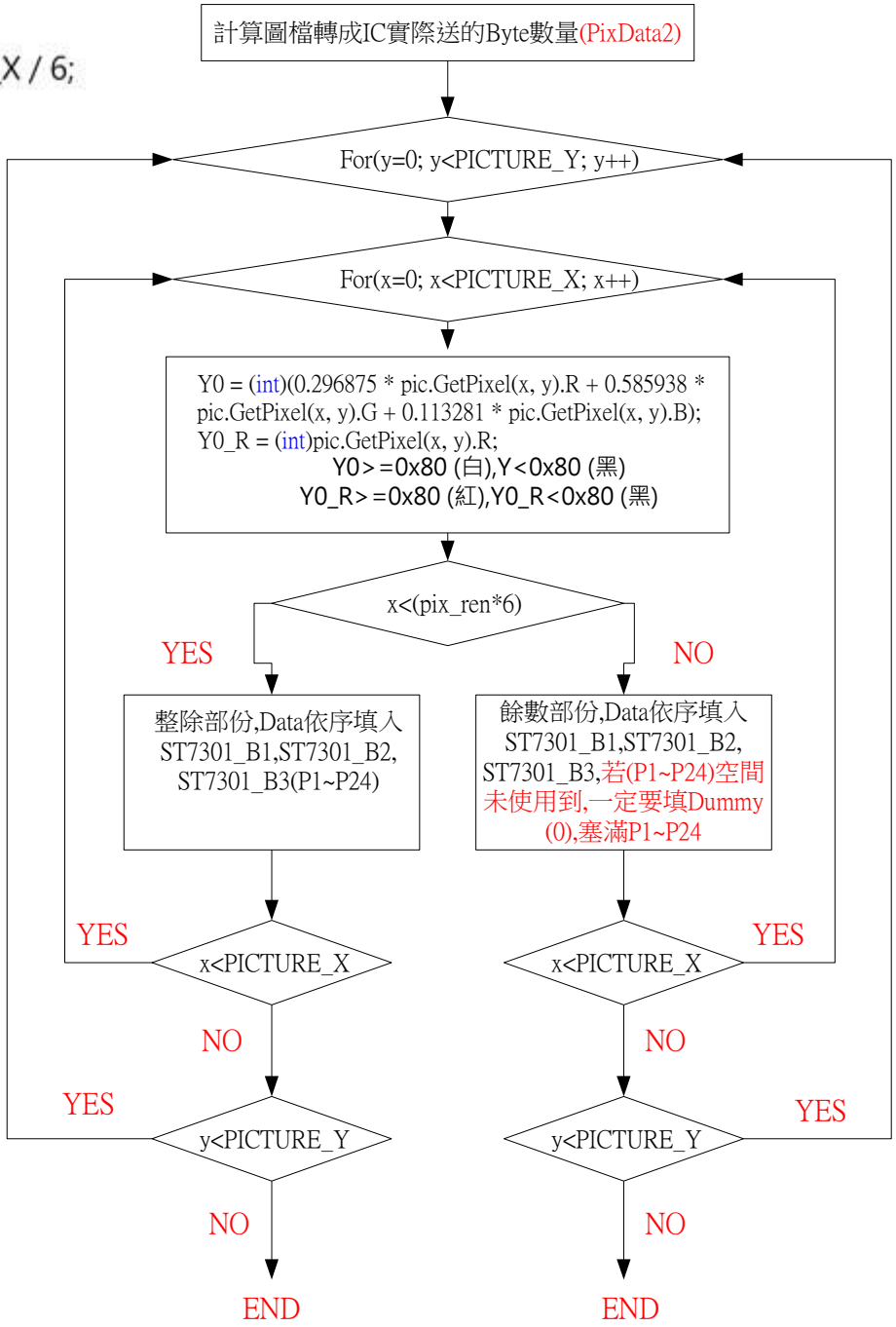
- 程式流程






```
pix_ren = PICTURE_X / 6;

if (x < (pix_ren * 6))
{
    整除部份,Data依序填入
    ST7301_B1,ST7301_B2,
    ST7301_B3(P1~P24)
}
else
{
    餘數部份,Data依序填入
    ST7301_B1,ST7301_B2,
    ST7301_B3,若(P1~P24)空間
    未使用到,一定要填Dummy
    (0),塞滿P1~P24
}
```



- **程式流程說明**

Step1: 先算出轉成IC實際Data數量，包含無法整除的Data數量，轉換後全部Data存在PixData2暫存器

Step2: 轉圖y地址設定，第一層迴圈

Step3: 轉圖x地址設定，第二層迴圈

所有的轉圖都在此迴圈內完成

Step4: 運用公式，將圖點轉成 黑/白 與 紅/黑

Step5: 可整除部分移入ST7301_B1，ST7301_B2，ST7301_B3 暫存器(依序移入P1~P24)

Step6: 不可整除部分移入ST7301_B1，ST7301_B2，ST7301_B3 暫存器，有Dummy部分寫入0x00(依序移入P1~P24)

Step7: 重複Step2~Step6，直到每點轉換完成

- **程式範例**

```
int total_byte;
int pix_ren = PICTURE_X / 6;
if ((PICTURE_X % 6) == 0)
{
    total_byte = (PICTURE_X * PICTURE_Y) / 4;
}
else
{
    int to = PICTURE_X / 6;
    to = to * 6;
    total_byte = ((to + 6) * PICTURE_Y) / 4;
}
float TYPE2_Size = (PICTURE_X * PICTURE_Y) / 4;
byte[] PixData2 = new byte[(int)total_byte]; ///Step1  總共byte數量
int count = 0;
int Y0, Y0_R, Y1, Y1_R;
int pixelvalue = 0;
int TYPE_Count = 0;
for (int y = 0; y < PICTURE_Y; y = y + 2) //Step2  轉圖y地址設定
{
    for (int x = 0; x < PICTURE_X; x++)   //Step3  轉圖x地址設定
    {
        Y0 = (int)(0.296875 * pic.GetPixel(x, y).R + 0.585938 * pic.GetPixel(x, y).G +
        0.113281 * pic.GetPixel(x, y).B);
        Y0_R = (int)pic.GetPixel(x, y).R;
       //Step4:運用公式，將圖點轉成  黑/白  與  紅/黑
        Y1 = (int)(0.296875 * pic.GetPixel(x, y + 1).R + 0.585938 * pic.GetPixel(x, y + 1).G
        + 0.113281 * pic.GetPixel(x, y + 1).B);
        Y1_R = (int)pic.GetPixel(x, y + 1).R;
        Y0 = Y0 & 0x80;
        Y0_R = Y0_R & 0x80;
        Y1 = Y1 & 0x80;
        Y1_R = Y1_R & 0x80;
        pic.SetPixel(x, y, Color.FromArgb(Y0_R, Y0, Y0));
        pic.SetPixel(x, y + 1, Color.FromArgb(Y1_R, Y1, Y1));
```

```
if (TYPE_Count == 6)
    TYPE_Count = 0;
if (x < (pix_ren * 6))   //Step5 可整除部分
{
    if (TYPE_Count == 0)
    {
        if (checkBox36.Checked == true) //判斷 黑/白 與 紅/黑 位置 (Color Filter)
        {
            Y0 = Y0 >> 2;
            Y1 = Y1 >> 3;
            Y1_R = Y1_R >> 1;
        }
        else
        {
            Y0_R = Y0_R >> 2;
            Y1 = Y1 >> 1;
            Y1_R = Y1_R >> 3;
        }
        pixelvalue = Y0 | Y0_R | Y1 | Y1_R;
        pixelvalue = pixelvalue & 0xF0;
    }
    if (TYPE_Count == 1)
    {
        if (checkBox36.Checked == true)
        {
            Y0 = Y0 >> 6;
            Y0_R = Y0_R >> 4;
            Y1 = Y1 >> 7;
            Y1_R = Y1_R >> 5;
        }
        else
        {
            Y0 = Y0 >> 4;
            Y0_R = Y0_R >> 6;
            Y1 = Y1 >> 5;
            Y1_R = Y1_R >> 7;
        }
```

```
            pixelvalue = pixelvalue | Y0 | Y0_R | Y1 | Y1_R;
            PixData2[count] = (byte)pixelvalue;
            count++;
        }
        if (TYPE_Count == 2)
        {
            if (checkBox36.Checked == true)
            {
                Y0 = Y0 >> 2;
                Y1 = Y1 >> 3;
                Y1_R = Y1_R >> 1;
            }
            else
            {
                Y0_R = Y0_R >> 2;
                Y1 = Y1 >> 1;
                Y1_R = Y1_R >> 3;
            }
            pixelvalue = Y0 | Y0_R | Y1 | Y1_R;
            pixelvalue = pixelvalue & 0xF0;
        }
        if (TYPE_Count == 3)
        {
            if (checkBox36.Checked == true)
            {
                Y0 = Y0 >> 6;
                Y0_R = Y0_R >> 4;
                Y1 = Y1 >> 7;
                Y1_R = Y1_R >> 5;
            }
            else
            {
                Y0 = Y0 >> 4;
                Y0_R = Y0_R >> 6;
                Y1 = Y1 >> 5;
                Y1_R = Y1_R >> 7;
            }
```

```
            pixelvalue = pixelvalue | Y0 | Y0_R | Y1 | Y1_R;
            PixData2[count] = (byte)pixelvalue;
            count++;
        }
        if (TYPE_Count == 4)
        {
            if (checkBox36.Checked == true)
            {
                Y0 = Y0 >> 2;
                Y1 = Y1 >> 3;
                Y1_R = Y1_R >> 1;
            }
            else
            {
                Y0_R = Y0_R >> 2;
                Y1 = Y1 >> 1;
                Y1_R = Y1_R >> 3;
            }
            pixelvalue = Y0 | Y0_R | Y1 | Y1_R;
            pixelvalue = pixelvalue & 0xF0;
        }
        if (TYPE_Count == 5)
        {
            if (checkBox36.Checked == true)
            {
                Y0 = Y0 >> 6;
                Y0_R = Y0_R >> 4;
                Y1 = Y1 >> 7;
                Y1_R = Y1_R >> 5;
            }
            else
            {
                Y0 = Y0 >> 4;
                Y0_R = Y0_R >> 6;
                Y1 = Y1 >> 5;
                Y1_R = Y1_R >> 7;
            }
```

```
                pixelvalue = pixelvalue | Y0 | Y0_R | Y1 | Y1_R;
                PixData2[count] = (byte)pixelvalue;
                count++;
            }
        }
        else
        {               //Step 6不可整除部分
            if (((x + 1) % 6) == 1)
            {
                if (checkBox36.Checked == true)
                {
                    Y0 = Y0 >> 2;
                    Y1 = Y1 >> 3;
                    Y1_R = Y1_R >> 1;
                }
                else
                {
                    Y0_R = Y0_R >> 2;
                    Y1 = Y1 >> 1;
                    Y1_R = Y1_R >> 3;
                }
                pixelvalue = Y0 | Y0_R | Y1 | Y1_R;
                pixelvalue = pixelvalue & 0xF0;
                if ((x + 1) == PICTURE_X) //是否是X最後一筆
                {
                    PixData2[count] = (byte)pixelvalue;
                    count++;
                    PixData2[count] = 0x00;
                    count++;
                    PixData2[count] = 0x00;
                    count++;
                }
            }
            if (((x + 1) % 6) == 2)
            {
                if (checkBox36.Checked == true)
                {
```

```
                Y0 = Y0 >> 6;
                Y0_R = Y0_R >> 4;
                Y1 = Y1 >> 7;
                Y1_R = Y1_R >> 5;
            }
            else
            {
                Y0 = Y0 >> 4;
                Y0_R = Y0_R >> 6;
                Y1 = Y1 >> 5;
                Y1_R = Y1_R >> 7;
            }
            pixelvalue = pixelvalue | Y0 | Y0_R | Y1 | Y1_R;
            PixData2[count] = (byte)pixelvalue;
            count++;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = 0x00;
                count++;
                PixData2[count] = 0x00;
                count++;
            }
        }
        if (((x + 1) % 6) == 3)
        {
            if (checkBox36.Checked == true)
            {
                Y0 = Y0 >> 2;
                Y1 = Y1 >> 3;
                Y1_R = Y1_R >> 1;
            }
            else
            {
                Y0_R = Y0_R >> 2;
                Y1 = Y1 >> 1;
                Y1_R = Y1_R >> 3;
            }
```

```
            pixelvalue = Y0 | Y0_R | Y1 | Y1_R;
            pixelvalue = pixelvalue & 0xF0;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = (byte)pixelvalue;
                count++;
                PixData2[count] = 0x00;
                count++;
            }
        }
        if (((x + 1) % 6) == 4)
        {
            if (checkBox36.Checked == true)
            {
                Y0 = Y0 >> 6;
                Y0_R = Y0_R >> 4;
                Y1 = Y1 >> 7;
                Y1_R = Y1_R >> 5;
            }
            else
            {
                Y0 = Y0 >> 4;
                Y0_R = Y0_R >> 6;
                Y1 = Y1 >> 5;
                Y1_R = Y1_R >> 7;
            }
            pixelvalue = pixelvalue | Y0 | Y0_R | Y1 | Y1_R;
            PixData2[count] = (byte)pixelvalue;
            count++;
            if ((x + 1) == PICTURE_X)
            {
                PixData2[count] = 0x00;
                count++;
            }
        }
        if (((x + 1) % 6) == 5)
        {
```

```
                    if (checkBox36.Checked == true)
                    {
                        Y0 = Y0 >> 2;
                        Y1 = Y1 >> 3;
                        Y1_R = Y1_R >> 1;
                    }
                    else
                    {
                        Y0_R = Y0_R >> 2;
                        Y1 = Y1 >> 1;
                        Y1_R = Y1_R >> 3;
                    }
                    pixelvalue = Y0 | Y0_R | Y1 | Y1_R;
                    pixelvalue = pixelvalue & 0xF0;
                    if ((x + 1) == PICTURE_X)
                    {
                        PixData2[count] = (byte)pixelvalue;
                        count++;
                    }
                }
                TYPE_Count = 5;
            }// if (x < (pix_ren * 6)) else
            TYPE_Count++;
        }//for (int x = 0; x < PICTURE_X; x++)
    }
    USB.wrdata_bulk(PixData2);
```