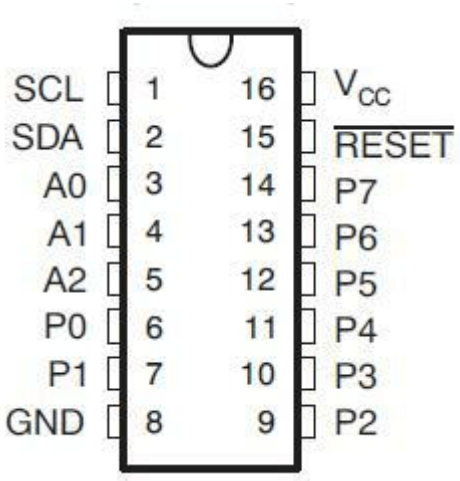
拡張ポート(I2C対応:I/O EXPANDER)

概要

ポートの数がもう少しほしい場合には、止む無く、ピン数の多いPICを使わざるを得ないのですが、検索してみま すと、TI社から、REMOTE 8-BIT I2C I/O EXPANDERという製品が提供されており、これを使用すると、ポート 数を簡単に増やすことができます。

そこで、これと同等の機能を、PICを使って実現することにしました。

- I2Cで制御可能とする。
- ポートの数は、8ビットとする。
- 入出力モードを設定可能とする。
- 接続台数は、4台まで可能とする。



TI社/PCA9557

動作原理

I2Cのスレーブ機能としての、基本的な構造は、前回製作したLCDモニタ(I2C対応)と同じです。 ポートの入出力モードや入出力データを設定するために、2バイトのメモリを使用します。0x00入出力モード設定用(TRISAやTRISBに相当) 0x01入出力データ設定用(PORTAやPORTBに相当)

<ポートの入出力モード設定> 通常のポートの入出力モード設定と同様にしました。 例えば、

```
TRISB = 0x00001111;
```

とすることでPORTBを、上位4ビットを出力モード、下位4ビットを入力モードにすることができます。 これをマスター側から設定するには、アドレス(0x00)のメモリに、0x00001111を書き込みます。

```
Soft_I2C_Start();
Soft_I2C_Write(0xC0);
Soft_I2C_Write(0x00);
Soft_I2C_Write(0b00001111);
Soft_I2C_Stop();
```

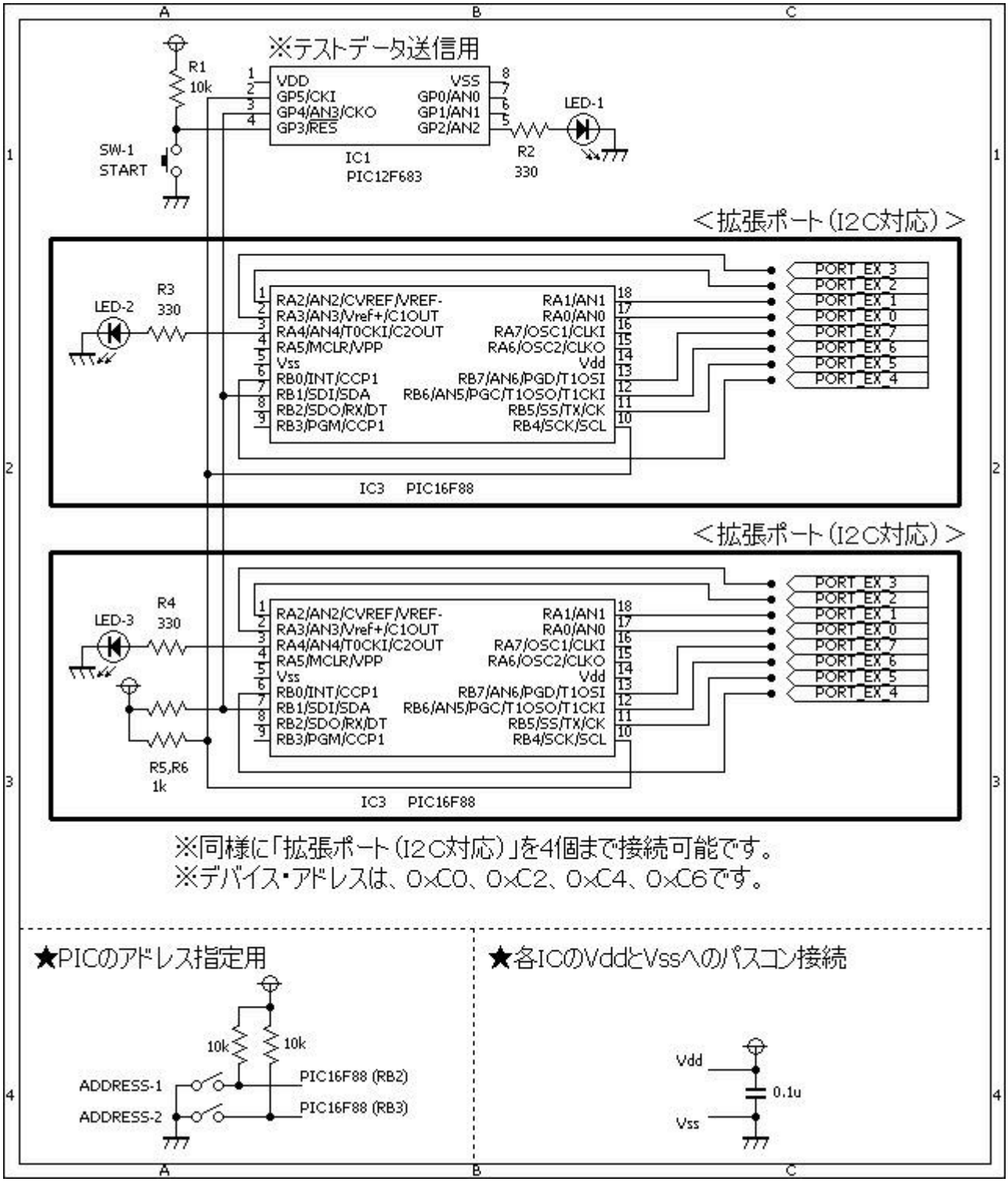
<ポートへのデータ出力> マスター側から、アドレス(0x01)のメモリに、出力データを書き込みます。

```
Soft_I2C_Start();
Soft_I2C_Write(0xC0);
Soft_I2C_Write(0x01);
Soft_I2C_Write(0b10100000);
Soft_I2C_Stop();
```

<ポートからのデータ入力> マスター側から、アドレス(0x01)のメモリを、読み込みます。

```
Soft_I2C_Start();
Soft_I2C_Write(0xC0);
Soft_I2C_Write(0x01);
Soft_I2C_Start();
Soft_I2C_Write(0xC1);
dat = Soft_I2C_Read(NO_ACK);
Soft_I2C_Stop();
```

回路図



ソースコード

port_ex_i2c.c

```
//*****************************************************************************
*
/*
  ＜拡張ポート□□□□対応）＞
*/
```

```
//*****************************************************************************
*

#define       LED             PORTA.F4
#define       SW_ADDR1    PORTB.F2
#define       SW_ADDR2    PORTB.F3
#define       ADDR_BASE    0xC0

#define       PORT_EX_0    PORTA.F0
#define       PORT_EX_1    PORTA.F1
#define       PORT_EX_2    PORTA.F2
#define       PORT_EX_3    PORTA.F3
#define       PORT_EX_4    PORTB.F0
#define       PORT_EX_5    PORTB.F5
#define       PORT_EX_6    PORTB.F6
#define       PORT_EX_7    PORTB.F7

#define       TRIS_EX_0    TRISA.F0
#define       TRIS_EX_1    TRISA.F1
#define       TRIS_EX_2    TRISA.F2
#define       TRIS_EX_3    TRISA.F3
#define       TRIS_EX_4    TRISB.F0
#define       TRIS_EX_5    TRISB.F5
#define       TRIS_EX_6    TRISB.F6
#define       TRIS_EX_7    TRISB.F7

#define       ON            1
#define       OFF             0

#define       DATA_SIZE    2

//*****************************************************************************
*

static unsigned    short    data[DATA_SIZE], pnt, flg, tmp, stat;

//*****************************************************************************
*

void   i2c_Write(unsigned short dat)
{
    while (SSPSTAT.BF == 1)
        ;
    while (1) {
        SSPCON.WCOL = 0;
        SSPBUF = dat;
        if (SSPCON.WCOL == 1)
            continue;
        //
        SSPCON.CKP = 1;
```

```
        return;
    }
}

//*********************************************************************
*

void   i2c_Handler()
{
    stat = SSPSTAT;
    tmp = SSPSTAT & 0b00101101;
    //
    if (tmp == 0b00001001) {    //書き込みモード、デバイスアドレス
        tmp = SSPBUF;
        flg = 0;
        pnt = 0;
        return;
    }
    if (tmp == 0b00101001) {    //書き込みモード、データ
        if (flg == 0) {
            pnt = SSPBUF;
            flg = 1;
            return;
        }
        if (flg == 1) {
            data[pnt] = SSPBUF;
            pnt++;
            return;
        }
    }
    if (tmp == 0b00001100) {    //読み込みモード、デバイスアドレス
        i2c_Write(data[pnt]);
        pnt++;
        return;
    }
    if (tmp == 0b00101100) {    //読み込みモード、データ□□□□□
        i2c_Write(data[pnt]);
        pnt++;
        return;
    }
    if (tmp == 0b00101000) {    //読み込みモード、データ□□□□□□□□
        tmp = SSPBUF;
        SSPCON = 0x36;
        return;
    }
    LED = ON;
    Delay_ms(100);
    LED = OFF;
}

//*********************************************************************
```

```
*

void   interrupt()
{
    if (PIR1.SSPIF == 1) {
        PIR1.SSPIF = 0;
        //
        LED = ON;
        i2c_Handler();
        LED = OFF;
    }
}

//*****************************************************************************
*

void   ByteToBit(unsigned short number, char *output)
{
    short    cnt;
    //
    for (cnt = 0; cnt < 8; cnt++) {
        if ((number & 0x80) != 0)
            output[cnt] = '1';
        else
            output[cnt] = '0';
        number <<= 1;
    }
    output[8] = 0x00;
}

//*****************************************************************************
*

void   main()
{
    unsigned    short    cnt, TRIS_EX;
    //
    CMCON  = 0b00000111;        //コンパレータは使用しない。
    ANSEL  = 0b00000000;        //A/Dコンバータは使用しない。
    OSCCON = 0b01110000;        //クロックは内臓8MHzを使用する。
    TRISA  = 0b11101111;        //PORTAを設定する。
    TRISB  = 0b11111111;        //PORTBを設定する。
    OPTION_REG.NOT_RBPU = 0;    //PORTBをプルアップする。
    //□□□を設定する。
    SSPSTAT.SMP = 1;
    SSPSTAT.CKE = 1;
    SSPCON.WCOL = 0;
    SSPCON.SSPOV = 0;
    SSPCON.SSPEN = 1;
    SSPCON.CKP = 1;
```

```
    SSPCON.SSPM0 = 0;
    SSPCON.SSPM1 = 1;
    SSPCON.SSPM2 = 1;
    SSPCON.SSPM3 = 0;
    SSPADD = ADDR_BASE + ((SW_ADDR2 == 1) ? 4 : 0) + ((SW_ADDR1 == 1) ?
2 : 0);
    PIE1.SSPIE = 1;
    PIR1.SSPIF = 0;
    //
    LED = OFF;
    pnt = 0;
    flg = 0;
    data[0] = 0b11111111;
    data[1] = 0b11111111;
    TRIS_EX = data[0];
    for (cnt = 0; cnt < 10; cnt++) {
        LED = ON;
        Delay_ms(50);
        LED = OFF;
        Delay_ms(50);
    }
    //割り込み(全体)の設定
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    while (1) {
        //
        if (data[0].F0 != TRIS_EX.F0) {
            TRIS_EX.F0 = data[0].F0;
            TRIS_EX_0 = TRIS_EX.F0;
        }
        if (TRIS_EX.F0 == 1) {
            data[1].F0 = PORT_EX_0;
        } else {
            PORT_EX_0 = data[1].F0;
        }
        //
        if (data[0].F1 != TRIS_EX.F1) {
            TRIS_EX.F1 = data[0].F1;
            TRIS_EX_1 = TRIS_EX.F1;
        }
        if (TRIS_EX.F1 == 1) {
            data[1].F1 = PORT_EX_1;
        } else {
            PORT_EX_1 = data[1].F1;
        }
        //
        if (data[0].F2 != TRIS_EX.F2) {
            TRIS_EX.F2 = data[0].F2;
            TRIS_EX_2 = TRIS_EX.F2;
        }
```

```
        if (TRIS_EX.F2 == 1) {
            data[1].F2 = PORT_EX_2;
        } else {
            PORT_EX_2 = data[1].F2;
        }
        //
        if (data[0].F3 != TRIS_EX.F3) {
            TRIS_EX.F3 = data[0].F3;
            TRIS_EX_3 = TRIS_EX.F3;
        }
        if (TRIS_EX.F3 == 1) {
            data[1].F3 = PORT_EX_3;
        } else {
            PORT_EX_3 = data[1].F3;
        }
        //
        if (data[0].F4 != TRIS_EX.F4) {
            TRIS_EX.F4 = data[0].F4;
            TRIS_EX_4 = TRIS_EX.F4;
        }
        if (TRIS_EX.F4 == 1) {
            data[1].F4 = PORT_EX_4;
        } else {
            PORT_EX_4 = data[1].F4;
        }
        //
        if (data[0].F5 != TRIS_EX.F5) {
            TRIS_EX.F5 = data[0].F5;
            TRIS_EX_5 = TRIS_EX.F5;
        }
        if (TRIS_EX.F5 == 1) {
            data[1].F5 = PORT_EX_5;
        } else {
            PORT_EX_5 = data[1].F5;
        }
        //
        if (data[0].F6 != TRIS_EX.F6) {
            TRIS_EX.F6 = data[0].F6;
            TRIS_EX_6 = TRIS_EX.F6;
        }
        if (TRIS_EX.F6 == 1) {
            data[1].F6 = PORT_EX_6;
        } else {
            PORT_EX_6 = data[1].F6;
        }
        //
        if (data[0].F7 != TRIS_EX.F7) {
            TRIS_EX.F7 = data[0].F7;
            TRIS_EX_7 = TRIS_EX.F7;
        }
```

```
        if (TRIS_EX.F7 == 1) {
            data[1].F7 = PORT_EX_7;
        } else {
            PORT_EX_7 = data[1].F7;
        }
    }
}

//*********************************************************************
*
```

<参考> ※テスト用(PIC12F683)のプログラムです。

port_ex_i2c_master.c

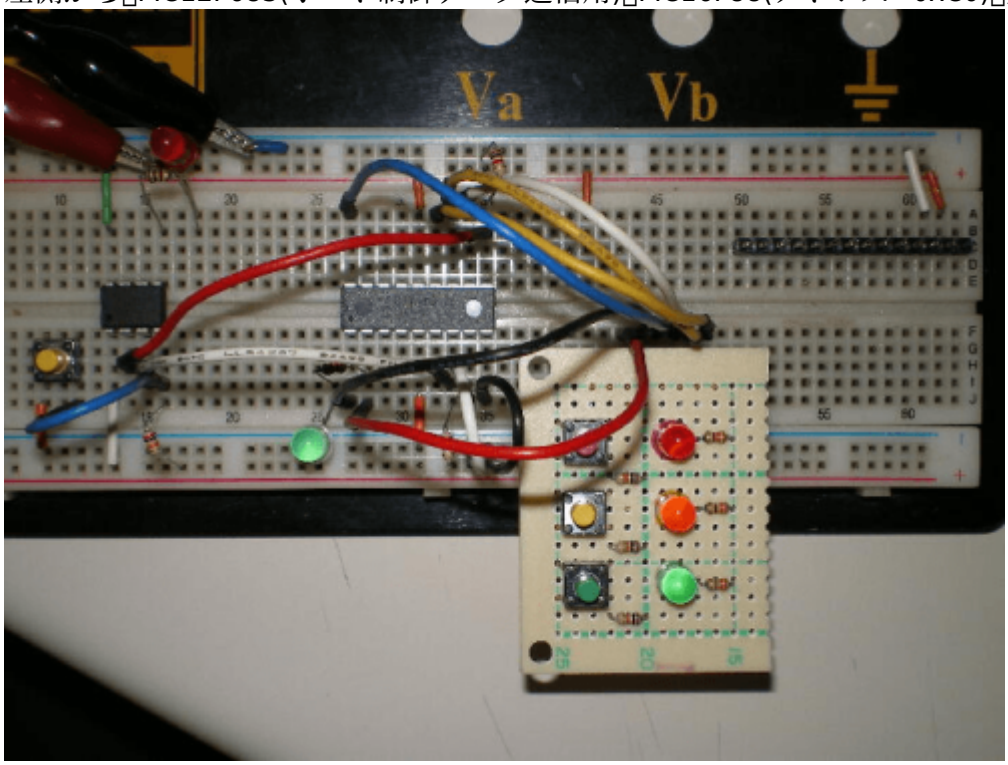
```
//*********************************************************************
*
/*
  『拡張ポートのテストデータ送信用（マスター）』
*/
//*********************************************************************
*

#define       SW            GPIO.F3
#define       LED            GPIO.F2

#define       ON            1
#define       OFF            0

#define       ACK            1
#define       NO_ACK        0

//*********************************************************************
*

void   SwitchONcheck()
{
    while (Button(&GPIO, 3, 1, 0) == 0)
        ;
    while (Button(&GPIO, 3, 1, 1) == 0)
        ;
}

//*********************************************************************
*

void main()
{
    unsigned    short    cnt, dat;
    //
```

```
    CMCON0 = 0b00000111;
    ANSEL.ANS0 = 0;
    ANSEL.ANS1 = 0;
    ANSEL.ANS2 = 0;
    ANSEL.ANS3 = 0;
    ADCON0.VCFG = 0;
    TRISIO = 0b00001011;
    OSCCON = 0b01110000;
    //
    for (cnt = 0; cnt < 10; cnt++) {
        LED = ON;
        Delay_ms(50);
        LED = OFF;
        Delay_ms(50);
    }
    //
    Soft_I2C_Config(&GPIO, 4, 5);    // SDA, SCL
    //
    while (1) {
        SwitchONcheck();
        //TRIS_EX = 0b00001111
        Soft_I2C_Start();
        Soft_I2C_Write(0xC0);
        Soft_I2C_Write(0x00);
        Soft_I2C_Write(0b00001111);
        Soft_I2C_Stop();
        //
        SwitchONcheck();
        //PORT_EX = 0x10100000
        Soft_I2C_Start();
        Soft_I2C_Write(0xC0);
        Soft_I2C_Write(0x01);
        Soft_I2C_Write(0b10100000);
        Soft_I2C_Stop();
        //
        SwitchONcheck();
        //PORT_EX = 0x01010000
        Soft_I2C_Start();
        Soft_I2C_Write(0xC0);
        Soft_I2C_Write(0x01);
        Soft_I2C_Write(0b01010000);
        Soft_I2C_Stop();
        //
        SwitchONcheck();
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0xC0);
        Soft_I2C_Write(0x01);
        Soft_I2C_Start();
        Soft_I2C_Write(0xC1);
```

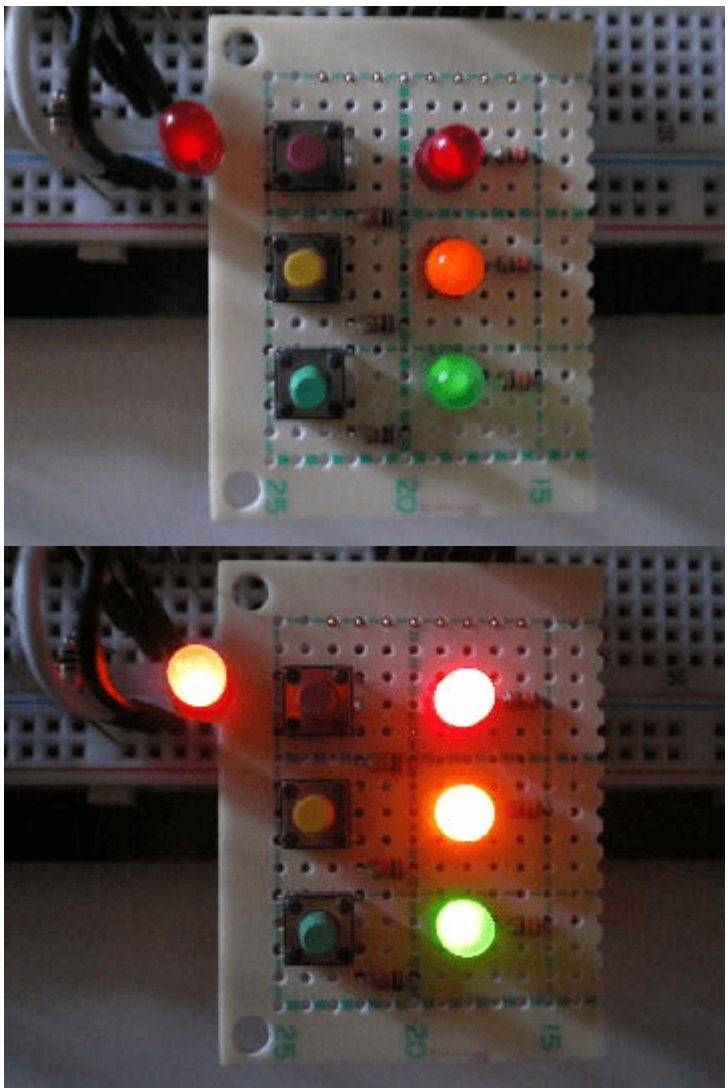
```
        dat = Soft_I2C_Read(NO_ACK);
        Soft_I2C_Stop();
        //
        SwitchONcheck();
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0xC0);
        Soft_I2C_Write(0x01);
        Soft_I2C_Write(dat << 4);
        Soft_I2C_Stop();
        //
        for (cnt = 0; cnt < 10; cnt++) {
            LED = ON;
            Delay_ms(50);
            LED = OFF;
            Delay_ms(50);
        }
    }
}

//*****************************************************************************
*
```

動作確認

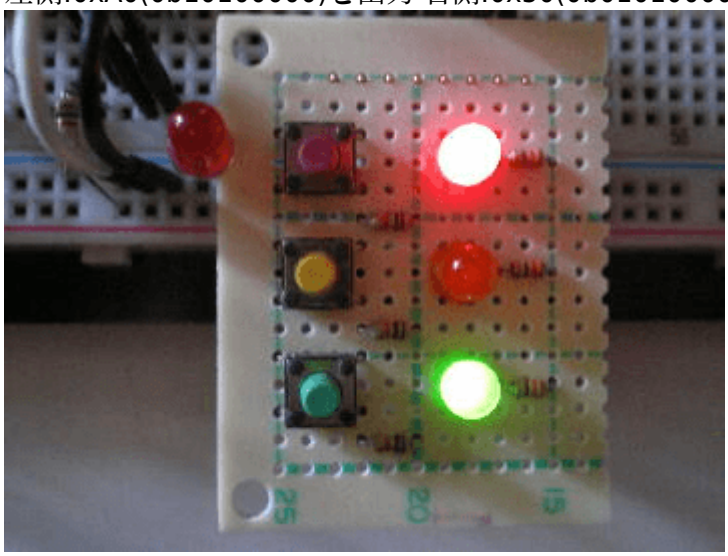
左側から□PIC12F683(ポート制御データ送信用)□PIC16F88(アドレス=0xC0)□入出力基板です。

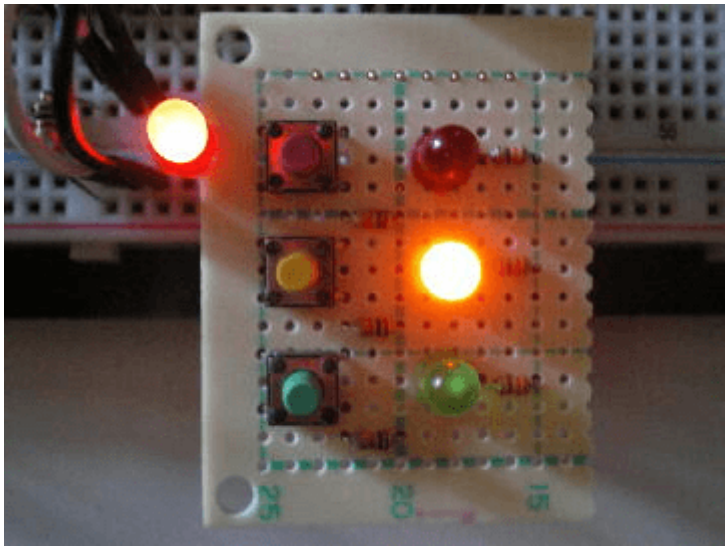


左側:起動直後 右側:ポートの入出力モード設定(上位4ビットを出力モード、下位4ビットを入力モード)

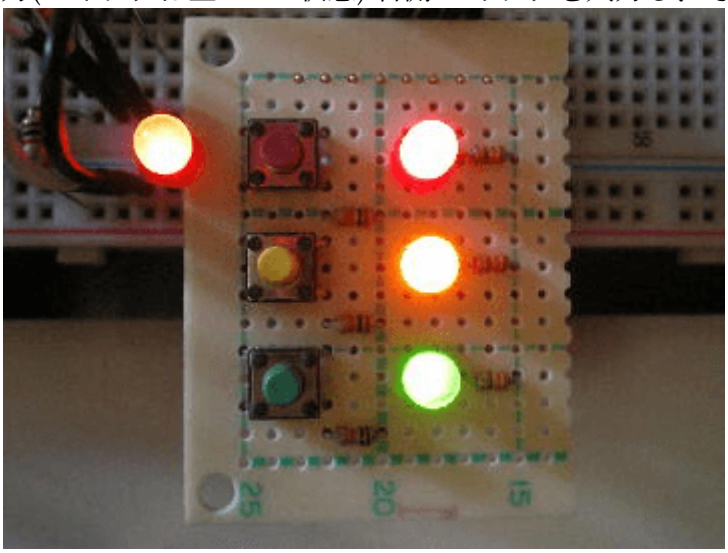


左側:0xA0(0b10100000)を出力 右側:0x50(0b01010000)を出力

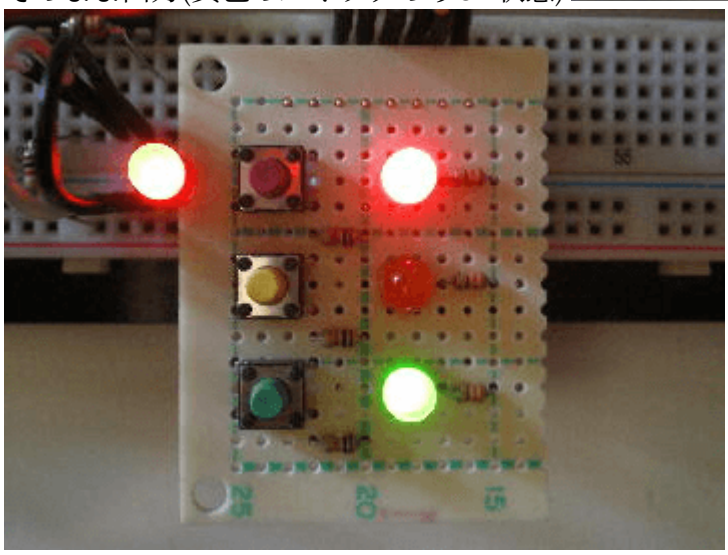




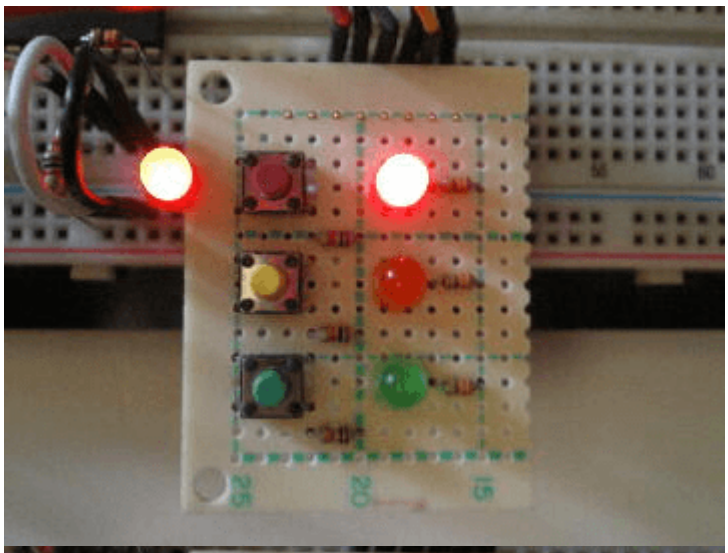
左側:スイッチを入力し、それをそのまま出力(スイッチは全てOFF状態) 右側:スイッチを入力し、それを

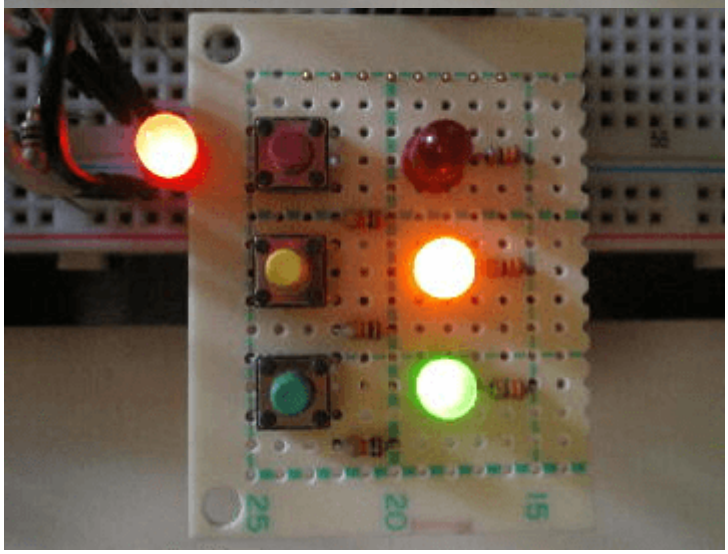


そのまま出力(黄色のスイッチのみON状態)



左側:スイッチを入力し、それをそのまま出力(黄色のスイッチと緑色のスイッチがON状態) 右側:スイッチを入力し、それをそのまま出力(赤色のスイッチのみON状態)





如何ですか? これでピン数の少ないPICに、簡単にポート数を増やすことができますね!{ }!

今回は、最大4台の拡張ポートを接続できるようにしましたが□PIC16F88のポートには、まだ3ピン未使用の ポートがありますので、プログラムを少し修正するだけで、最大32台(256ポート)まで増やすことができます。