LCDモニタ□(I2C対応)

概要

LCD(Liquid Crystal Display:液晶ディスプレイ)を制御するには、少なくとも、7本のポートが必要となります。 そのため□PIC12F683(8ピン)等のようにポート数が少ないPICでは□LCDを制御することができません。

そこで、以前に、RS232Cのインターフェイスを持った□LCDモニターを製作しました。 必要なポート数は、1本のみと、とても便利なのですが、この方法では、1台のLCDへの表示しかできません。

そこで、今回は、1台のPICから、複数のLCDへメッセージを表示できる方法を実現してみました。

動作原理

<PIC16F88が内蔵するSSPモジュール> Synchronous Serial Port(SSP)モジュールは、他の周辺機器やマイクロコントローラ装置とコミュニケートする のに役立つ、シリアルインタフェースです。これらの周辺機としては□EEPROM□シフトレジスタ、ディスプレイド ライバ□A/Dコンバータ□D/Aコンバータ等が考えられます。

SSPモジュールは、次の2つの動作モードを持っています。

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I2C)

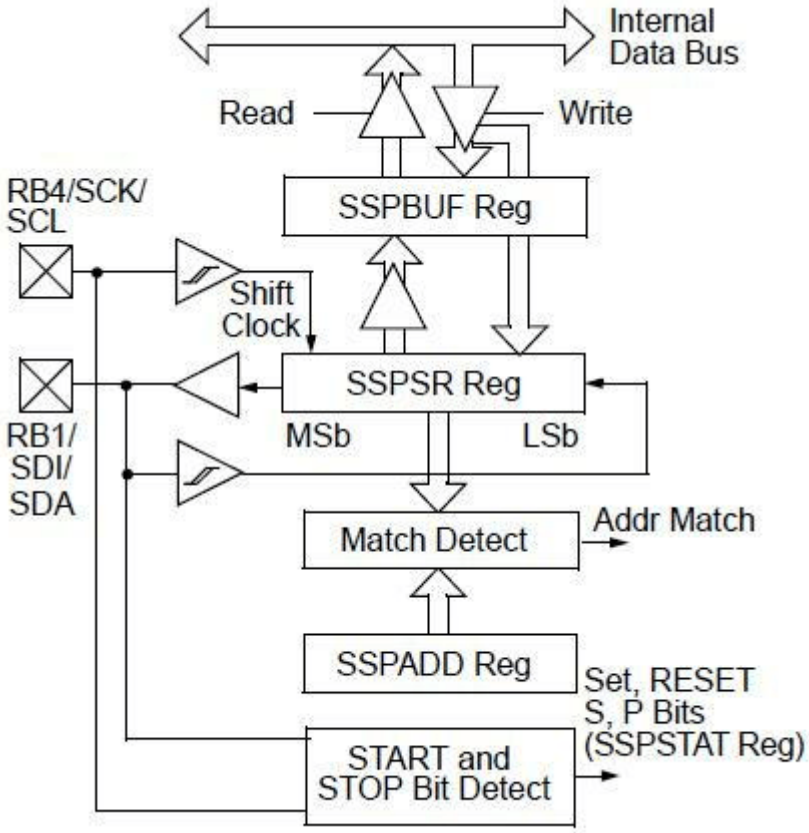
I2C(Inter-Integrated-Circuit)インターフェースは、同期式のシリアル通信の規格で、非同期式シリアル通信 (RS-232C等)に比べて、

- 高速通信
- 複数スレーブ(Slave)接続

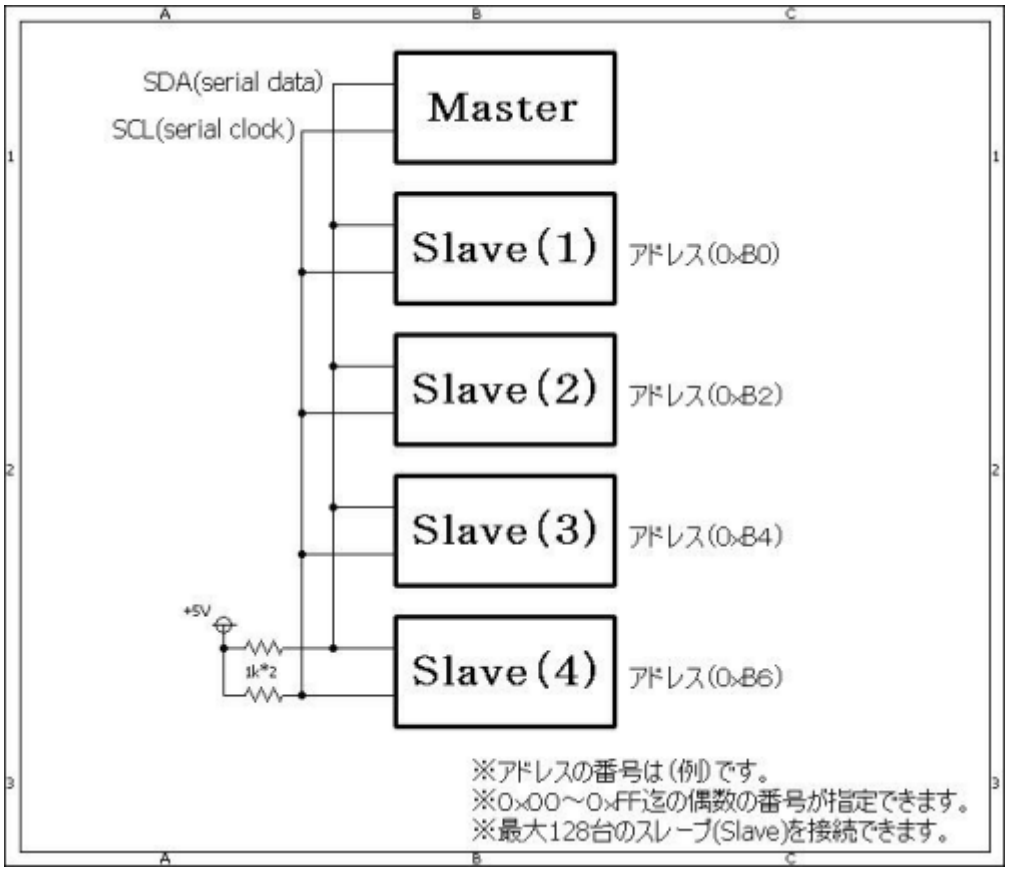
等の特長があり、今回の仕様には打って付けです。

接続できるスレーブの数は、0x00~0xFF迄の偶数の番号のアドレスが指定できるので、最大128台となります。但し、予約されているアドレス(下表参照)がありますので、この分が使えないことになります。

<I2Cモード時のSSPブロックダイアグラム>


Internal
Data Bus
Read
Write
RB4/SCK/
SCL
SSPBUF Reg
Shift
Clock
SSPSR Reg
RB1/
SDI/
SDA
MSb
LSb
Match Detect
Addr Match
SSPADD Reg
Set, RESET
S, P Bits
(SSPSTAT Reg)
START and
STOP Bit Detect

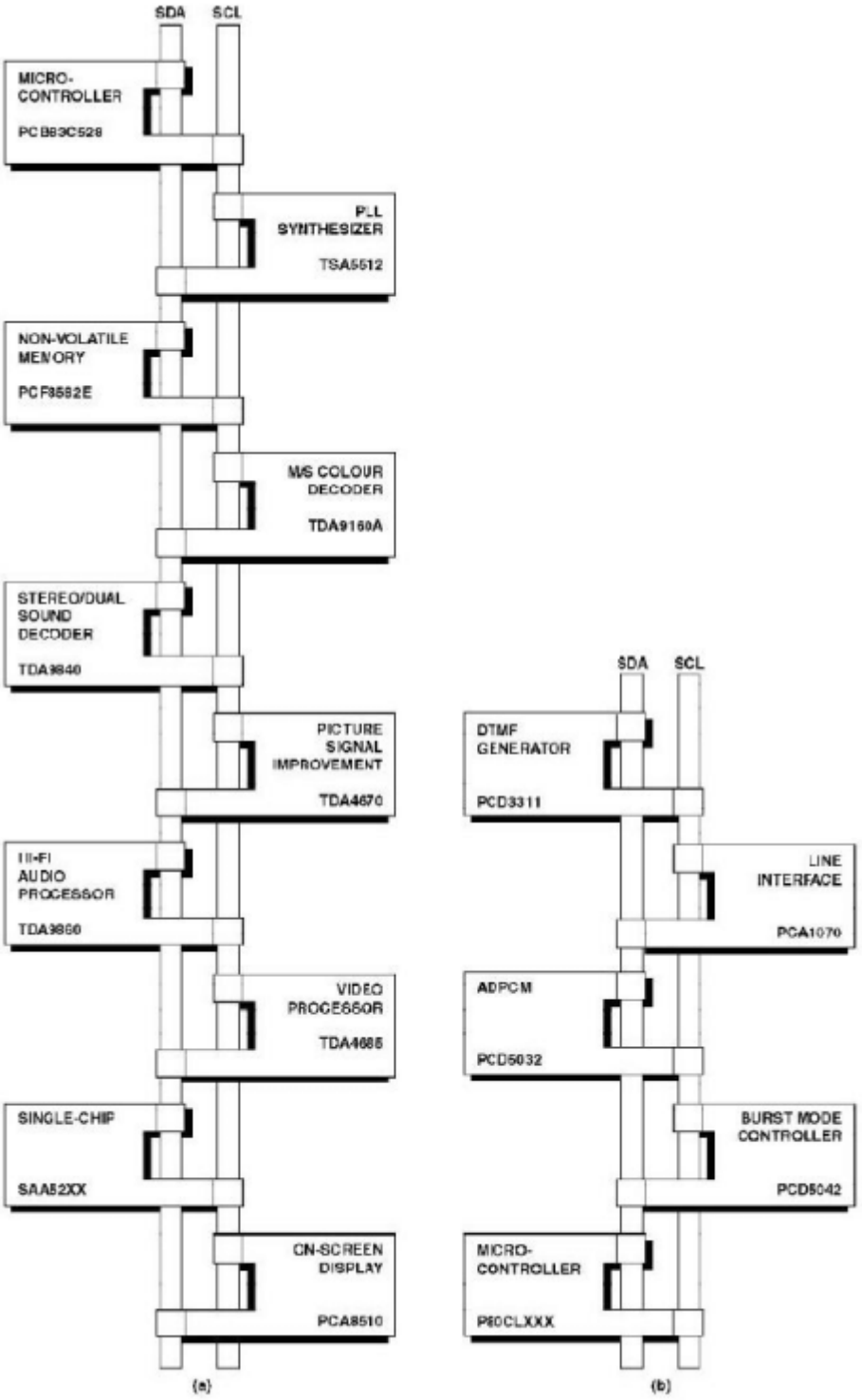
<接続例>


SDA(serial data)
SCL(serial clock)
Master
Slave(1)
アドレス(0x80)
Slave(2)
アドレス(0x82)
Slave(3)
アドレス(0x84)
Slave(4)
アドレス(0x86)
+5V
1k*2
※アドレスの番号は(例)です。
※0x00～0xFF迄の偶数の番号が指定できます。
※最大128台のスレーブ(Slave)を接続できます。

<予約されているアドレス>

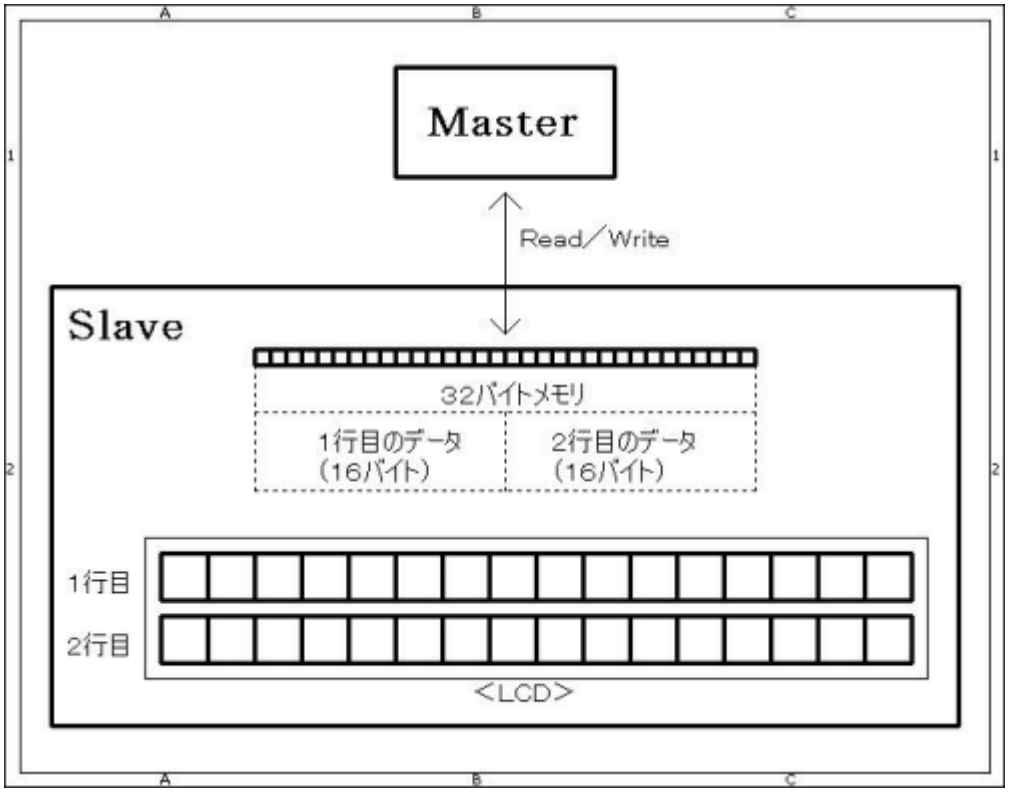
スレーブアドレス	R/$\overline{W}$ビット	説　明
0000　000	0	ゼネラル・コール・アドレス
0000　000	1	スタート・バイト(1)
0000　001	X	CBUSアドレス(2)
0000　010	X	異なるバス・フォーマット用に予約されているアドレス(3)
0000　011	X	将来の利用のために予約
0000　1XX	X	Hsモード・マスター・コード
1111　1XX	X	将来の利用のために予約
1111　0XX	X	10ビット・スレーブ・アドレス指定

<I2Cバスを応用した2つの製品例>※□I2Cバス仕 様 書バージョン 2.1」より (a) 高性能高集積テレビ (b) DECTコードレス電話基地局

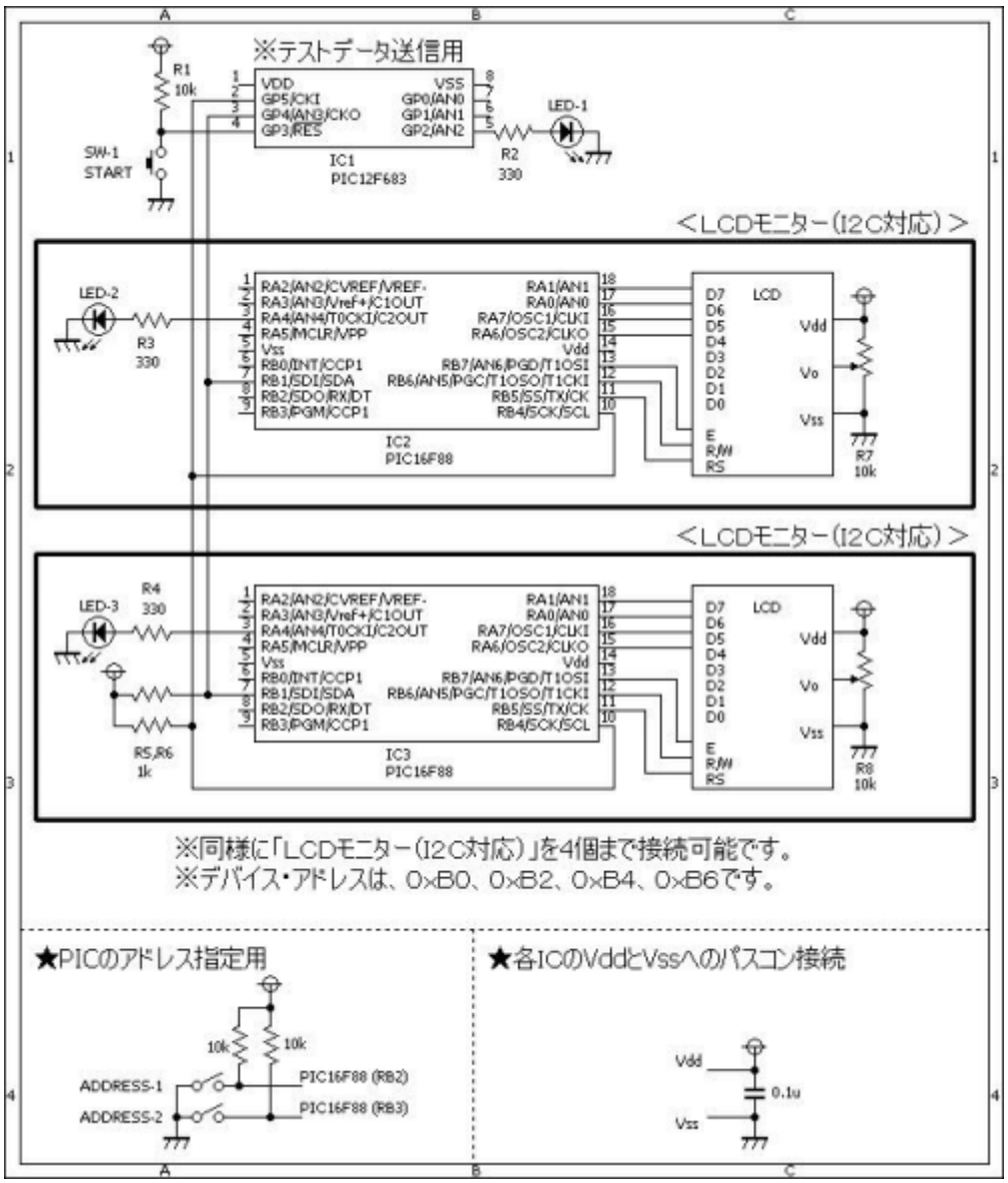


<今回製作した□LCDモニターの仕組み> マスター側から見ると、スレーブのメモリ(32バイト)にデータをセットするだけです。 スレーブ側では、メモリにセットされたデータを、LCDに表示させます。

- 32バイトの前半の16バイトが1行目のデータになります。
- 32バイトの後半の16バイトが2行目のデータになります。


Master
Read／Write
Slave
32バイトメモリ
1行目のデータ
(16バイト)
2行目のデータ
(16バイト)
1行目
2行目
<LCD>

回路図



ソースコード

LcdMonitor_I2C.c

```
//*****************************************************************************
*
/*
    □□□□モニタ□□□□□対応）＞
*/
//*****************************************************************************
*

#define       LED             PORTA.F4
#define       SW_ADDR1    PORTB.F2
#define       SW_ADDR2    PORTB.F3
#define       ADDR_BASE    0xB0
```

```
#define       ON            1
#define       OFF            0

#define       DATA_SIZE    32

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

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, buf[10], mode;
    //
    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  = 0b00101100;        //PORTAを設定する。
    TRISB  = 0b00011111;        //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;
    for (cnt = 0; cnt < 32; cnt++) {
        data[cnt] = ' ';
    }
    //LCDを初期化する。
    Lcd_Custom_Config(&PORTA,1,0,7,6,&PORTB,5,6,7);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(1, cnt + 1, 0xFF);
        Delay_ms(50);
    }
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(2, cnt + 1, 0xFF);
        Delay_ms(50);
```

```
    }
    //割り込み(全体)の設定
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    while (1) {
        for (cnt = 0; cnt < 16; cnt++) {
            Lcd_Custom_Chr(1, cnt + 1, data[cnt]);
        }
        for (cnt = 0; cnt < 16; cnt++) {
            Lcd_Custom_Chr(2, cnt + 1, data[cnt + 16]);
        }
    }
}

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

<参考> ※テスト用(PIC12F683)のプログラムです。

LcdMonitor_I2C_master.c

```
//*********************************************************************
*
/*
□□□□□モニター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)
        ;
}
```

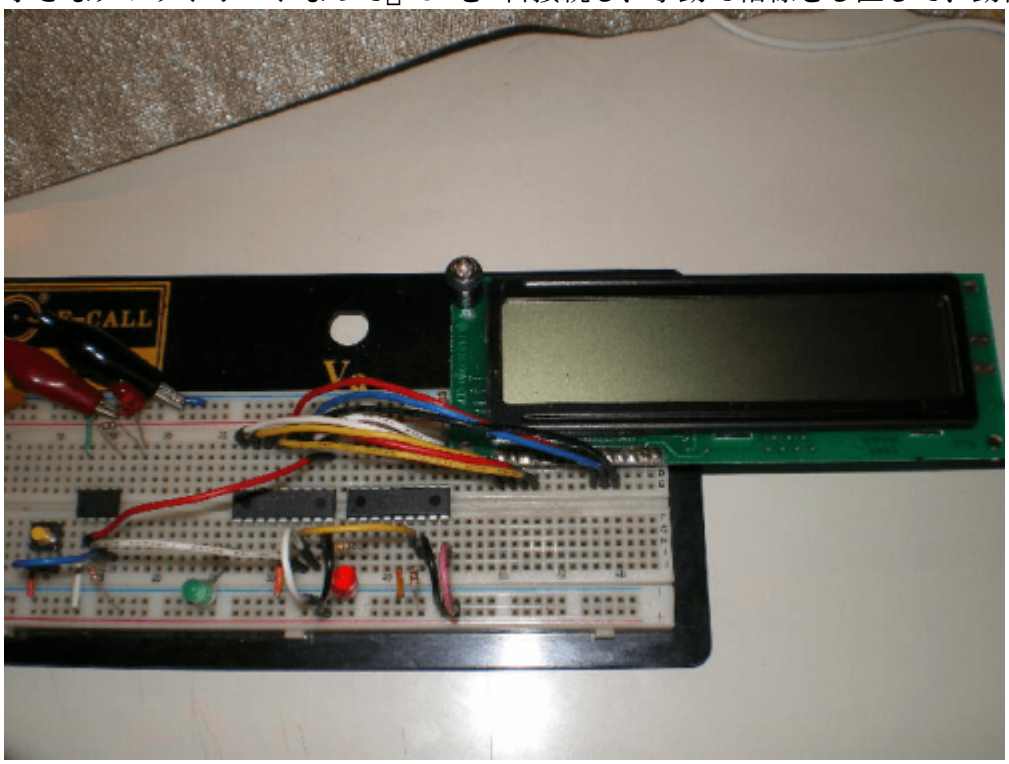
```
//*********************************************************************
*

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();
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0xB4);
        Soft_I2C_Write(0x00);
        for (cnt = 0; cnt < 32; cnt++) {
            Soft_I2C_Write('0' + cnt);
        }
        Soft_I2C_Stop();
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0xB4);
        Soft_I2C_Write(0x00);
        Soft_I2C_Start();
        Soft_I2C_Write(0xB5);
        for (cnt = 0; cnt < 32; cnt++) {
            if (cnt < 31)
                dat = Soft_I2C_Read(ACK);
            else
                dat = Soft_I2C_Read(NO_ACK);
            //
            if (dat != ('0' + cnt)) {
                LED = ON;
                Delay_ms(1000);
```

```
                LED = OFF;
            }
        }
        Soft_I2C_Stop();
        //
        for (cnt = 0; cnt < 10; cnt++) {
            LED = ON;
            Delay_ms(50);
            LED = OFF;
            Delay_ms(50);
        }
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0xB6);
        Soft_I2C_Write(0x00);
        for (cnt = 0; cnt < 32; cnt++) {
            Soft_I2C_Write('P' + cnt);
        }
        Soft_I2C_Stop();
        //
        Delay_ms(1000);
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0xB6);
        Soft_I2C_Write(0x00);
        Soft_I2C_Start();
        Soft_I2C_Write(0xB7);
        for (cnt = 0; cnt < 32; cnt++) {
            if (cnt < 31)
                dat = Soft_I2C_Read(ACK);
            else
                dat = Soft_I2C_Read(NO_ACK);
            //
            if (dat != ('P' + cnt)) {
                LED = ON;
                Delay_ms(1000);
                LED = OFF;
            }
        }
        Soft_I2C_Stop();
        //
        for (cnt = 0; cnt < 10; cnt++) {
            LED = ON;
            Delay_ms(50);
            LED = OFF;
            Delay_ms(50);
        }
    }
}
```

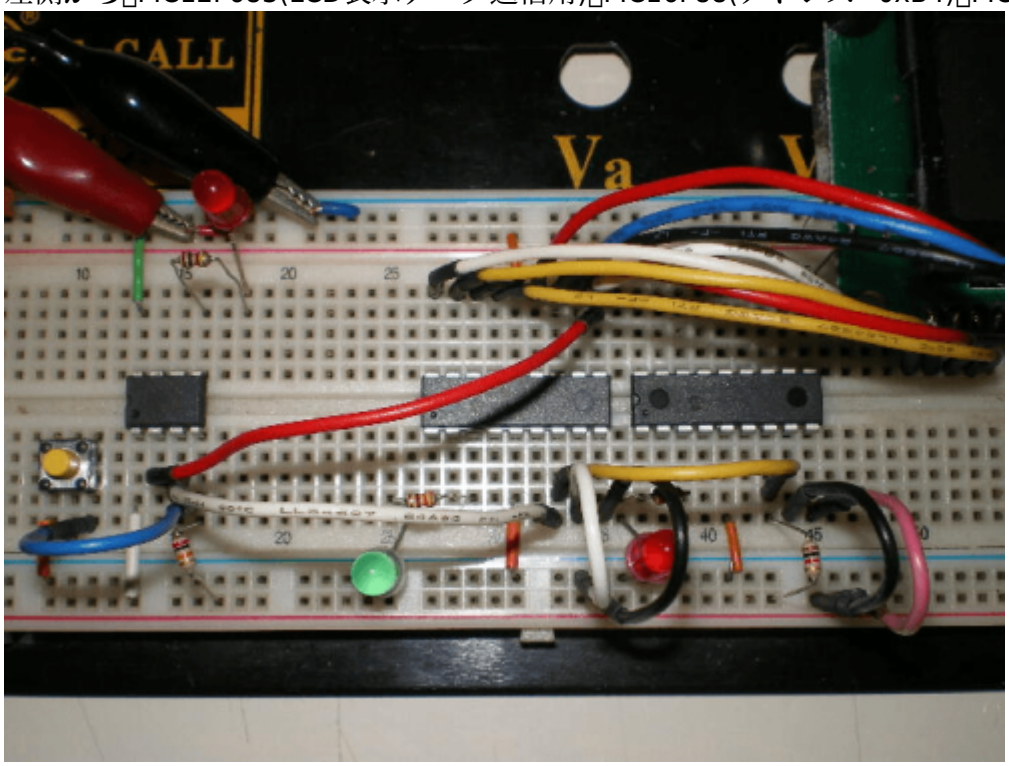
```
//*****************************************************************************
*
```

動作確認

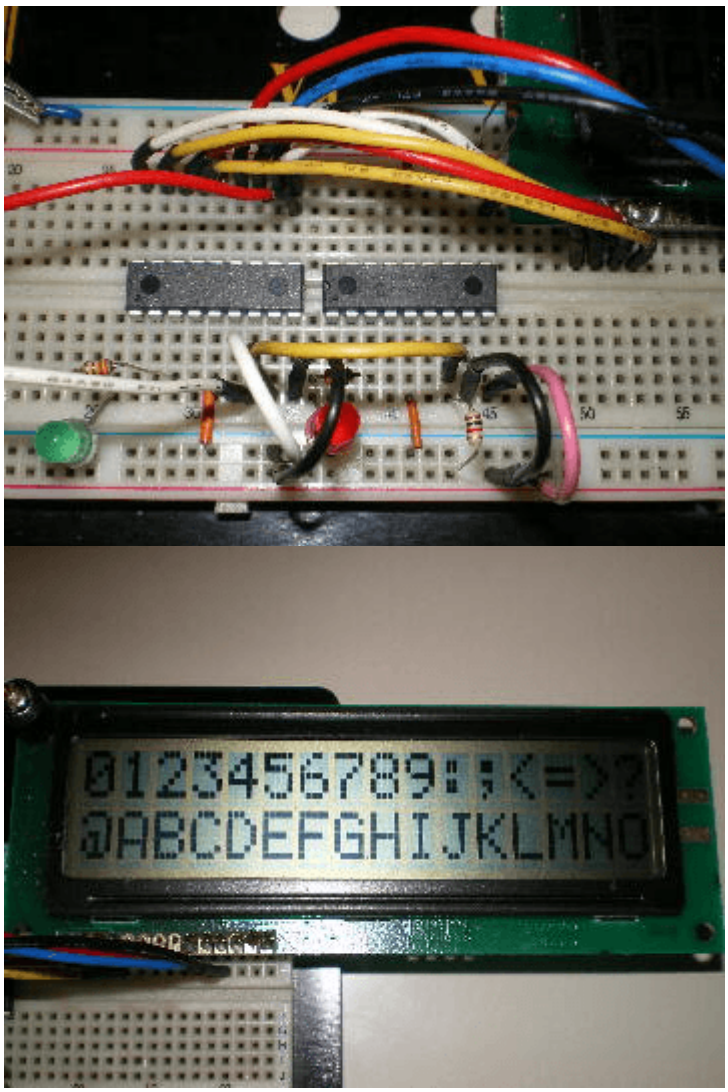
小さなブレッドボードなのでLCDを1台接続し、手動で結線をし直して、動作を確認しました。



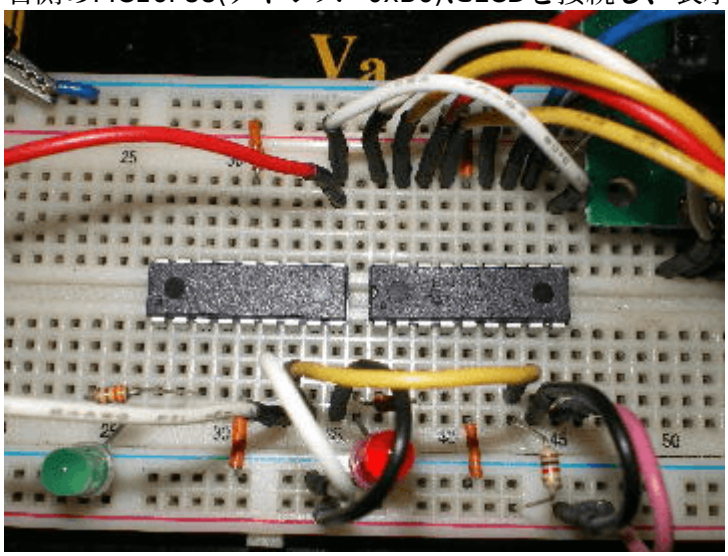
左側からPIC12F683(LCD表示データ送信用)PIC16F88(アドレス=0xB4)PIC16F88(アドレス=0xB6)



左側のPIC16F88(アドレス=0xB4)にLCDを接続し、表示データを送信したところです。



右側のPIC16F88(アドレス=0xB6)にLCDを接続し、表示データを送信したところです。



如何ですか? 1台のPICから、複数のLCDに対して、表示メッセージを送信することができるので、電子工作の範囲が広がり そうですね! 今回は、最大4台のLCDを接続できるようにしましたが□PIC16F88のポートには、まだ余裕がありますので、 プログラムを少し修正するだけで、8台、16台と増やすことができます。