```
/*  ////////////////////////////////////////////////
 *   PIC16F88にSC1602互換LCD表示パネルを接続してi2cのスレー
 *   ブとして表示を行う実験用ルーチンです。sc2002とは4ビット
 *   モード接続されます。
 *
 *   V0.1 2011.09.30  V1.0 2011.10.03
 *              MPLAB v8.73&HI-TECH C V9.82
 *              #define _LEGACY_HEADERS
 *
 *      by nobcha (c)2011
 *
 *      i2c経由データは必ず２バイト単位でくるとします。(2,4,6)
 *      １バイト目はRSビット、２バイト目がLCDに書き込むデータです
 *      ストロベリーリナックスの拡張コマンドはサポートしてません
 *
 *      RA0-3:SC1602は4ビットモ－ドとし、RA0－3で接続します。
 *      RA6:LCD EN bit (enable)はRA6につなぎます。
 *      RA7:LCD RS bit (RS)はRA7につなぎます。
 *      RB1:SDA CCP
 *      RB4:SCL CCP
 *      RB6:動作確認用LED
 *      RB5:1:Debug 0:normal
 *
 *      SC1602 pin connection via 4bit mode
 *              #1      Vdd=5V
 *              #2      Vss=GND
 *              #3      LCD contrust center of 2k VOL
 *              #4      RS      RB0
 *              #5      R/W     GND
 *              #6      E       RB2
 *
 *              #11-14  DATA    RA0-3
 *
 *
 *
 * ////////////////////////////////////////////////// */
#define _LEGACY_HEADERS
#include        <htc.h>
#include        "lcd.h"
#include        "delay.h"

#define _XTAL_FREQ = 8000000
#define PIC_CLOCK = 8000000
#define I2C_ADR 0x7C

__CONFIG(INTCLK & WDTDIS & PWRTDIS & BORDIS & MCLREN & LVPDIS &
         DEBUGEN & UNPROTECT & FCMDIS & IESODIS & DEBUGEN);
/* ////////////////////////////////////////////  */

#define DEBUG PORTB&0x20

unsigned char   buffer[20];                     // 受信バッファ
unsigned char   stat[20];
char     i , j, rcv_count=0 ,cnt=0;

void    init(void){
        for(char i=0;i<55;i++){                 // waiting for clock settled
                _delay(100);
        }
        OSCCON = 0b01110110;                    // 内部クロック8MHz IOFS INTCLK
        for(char i=0;i<55;i++){                 // waiting for clock settled
                _delay(100);
        }
        ANSEL  = 0x00;                          // 全ポートデジタル
        OPTION = 0x00;                          // オプション設定なし
        INTCON = 0x00;                          // 割込み使用せず

        TRISA = 0b00100000;                     // RA5は入力
        TRISB = 0b00110010;                     // RB1とRB4,5はi2c用で入力設定
                                                // port direction: 1:input
        PORTB = 0b00000000;                     //
        PORTA = 0b00000000;                     //

        }
```

```
                                                main_slave
void    ssp_init(void){
        SSPCON = 0x16;                          // SSPEN,SSP_slave_7bit
        SSPADD = I2C_ADR;                       // SSP ADDRESS SET
        PIE1 = 0x80;                            // SSPIE enable
        }

unsigned char lcd_rs;
interrupt i2c_slave(){                                                  // i2c slave receive int func
        GIE=0;
        if(SSPIF==1){
                SSPIE=0;
                SSPIF=0;
                buffer[rcv_count]=SSPBUF;                               // dataをバッファに入れる
                stat[rcv_count]=SSPSTAT;                                // statusをステイタスバッファに入
れる

                if ((SSPSTAT&0x20)==0x20){                              // DATA?
                        if ((rcv_count==1 )|(rcv_count==3) |(rcv_count==5)){
                                lcd_rs =((buffer[rcv_count]&0x40)<<1);
                        }
                        else {
                                lcd_write_rs(buffer[rcv_count], lcd_rs);
                    }
                }
                rcv_count++;                                            // 受信バイト数
                        if(rcv_count>10){ lcd_putch(0x3F);}
                SSPIE=1;
        }

        GIE=1;
}


char    MSG1[] = "PIC i2c slave v0.3 ";
char    PORTB_DATA;

void main(void){
        init();                                                         // ポート初期化
        lcd_init();                                                     // LCDの初期化
        cnt=0;

        if ((PORTB&0x20)==0x20){                                        //      debug mode

                lcd_goto(40);                                           // 2行目にLCD動作表示
                lcd_puts(MSG1);                                         // LCDに初期表示
                }

        ssp_init();                                                     // SSPの初期化
        SSPIF = 0;                                                      // SSPIF clear
        PORTB_DATA = 0b00010010;                                        // PB6は動作モニター用LED消灯 RB1
とRB4はi2c

        rcv_count=0;

        while(1){                                                       // 繰り返し
                SSPIE=1;
                PEIE=1;
                GIE=1;
                SSPEN=1;
                while((SSPSTAT&0x10) ==0x00) { }                        // 受信のストップ状態チェック

                GIE=0;                                                  // int diable

                if ((PORTB&0x20)==0x20){                                //      debug mode
                        for(i=0;i<(rcv_count);i++){
                                lcd_goto(i*5 | ((cnt&0x01))<<6);
                                lcd_putch(0x23);                        // #
                                lcd_putch(((buffer[i]>>4)&0x0F)|0x30);
                                lcd_putch(((buffer[i])&0x0F)|0x30);
                                lcd_putch(((stat[i]>>4)&0x0F)|0x30);
                                lcd_putch(((stat[i])&0x0F)|0x30);
                        }
                        lcd_goto(0x53);                                 // 2行目にLCD動作表示
                        lcd_putch(cnt++ | 0x30);
                                if(cnt>0x09){cnt=0;}
```

```
                lcd_putch(0x3C);

                lcd_putch(((rcv_count) & 0x0F ) | 0x30);        // 下位4ビットを取り出しバイト
                lcd_putch(0x3E);
                lcd_putch(0x20);

                for(j=0;j<100;j++){                             // 表示時間を待たせる
                        _delay(100);
                      }
            }
            rcv_count=0;
            RB6 = RB6 ^= 1;
        }
}
```