ADCモジュール(SPI&PIC)

概要

市販されている各種センサーには、SPI(Serial Peripheral Interface)方式に対応したものが多くあります。

- A/Dコンバータ【MCP3208【12ビット、8チャネル】
- D/Aコンバータ【MCP4922【12ビット、2チャネル】
- 気圧センサ【SCP1000】
- 温度センサ【MAX6627】
- 加速度センサ【MMA7455L】

PICに、これらのセンサーを接続して使用する場合には、PIC側をSPIのマスタ【(master)】センサー側をスレーブ(slave)として制御することになります。

このように、PICにセンサーを接続する限りにおいては、PIC側にスレーブ機能を搭載する必要性は殆どありません。 しかし、PIC同士をSPIで接続して通信を行う場合には、どちらかのPICにスレーブ機能を搭載する必要があります。

そこで今回は、PIC間をSPI方式で接続し、マスター&スレーブ通信を行わせてみました。 只単に、通信を行わせるだけでは実用性がありませんので、次のような仕様とし、実用性を高めました。尚、マスター機能とスレーブ機能を1つのPICに搭載し、起動時のスイッチによって、どちらかに切り替えるようにしました。

<マスター側の仕様>

- スレーブ側にADCの開始を指示する。(4チャネル分)
- 各チャネルのデータを受信し、LCDに表示する。

<スレーブ側の仕様>

- マスター側からの指示で、ADCを開始する。(4チャネル分)
- マスター側からの指示で、各チャネルのデータを送信する。

動作原理

PIC16F88には、SSP(Synchronous Serial Port)モジュールが搭載されています。 SSPは、SPIやI2CIの通信をハードウェアで実現するためのインターフェース・モジュールです。 SSPをSPIモードで使用することにより、マスター機能およびスレーブ機能を実現することが出来ます。

<mikroCが提供するSPIライブラリ>

SPIx_Init	SPIモジュールを初期化します。(マスターモードに特化)
SPIx_Init_Advanced	SPIモジュールの初期化します。(詳細な設定が可能)
SPIx_Read	SPI通信で、1バイトを受信します。
SPIx_Write	SPI通信で、1バイトを送信します。
SPI_Set_Active	SPIモジュールが2個搭載されている場合に、どちらをアクティブにするかを設定します。

<SPIの初期化>

- マスターモードではmikroCで提供されているSPI初期化関数(SPI1_Init)をそのまま使用します。
- スレーブモードではSPI1_Init関数の後で、スレーブ用に再設定します。

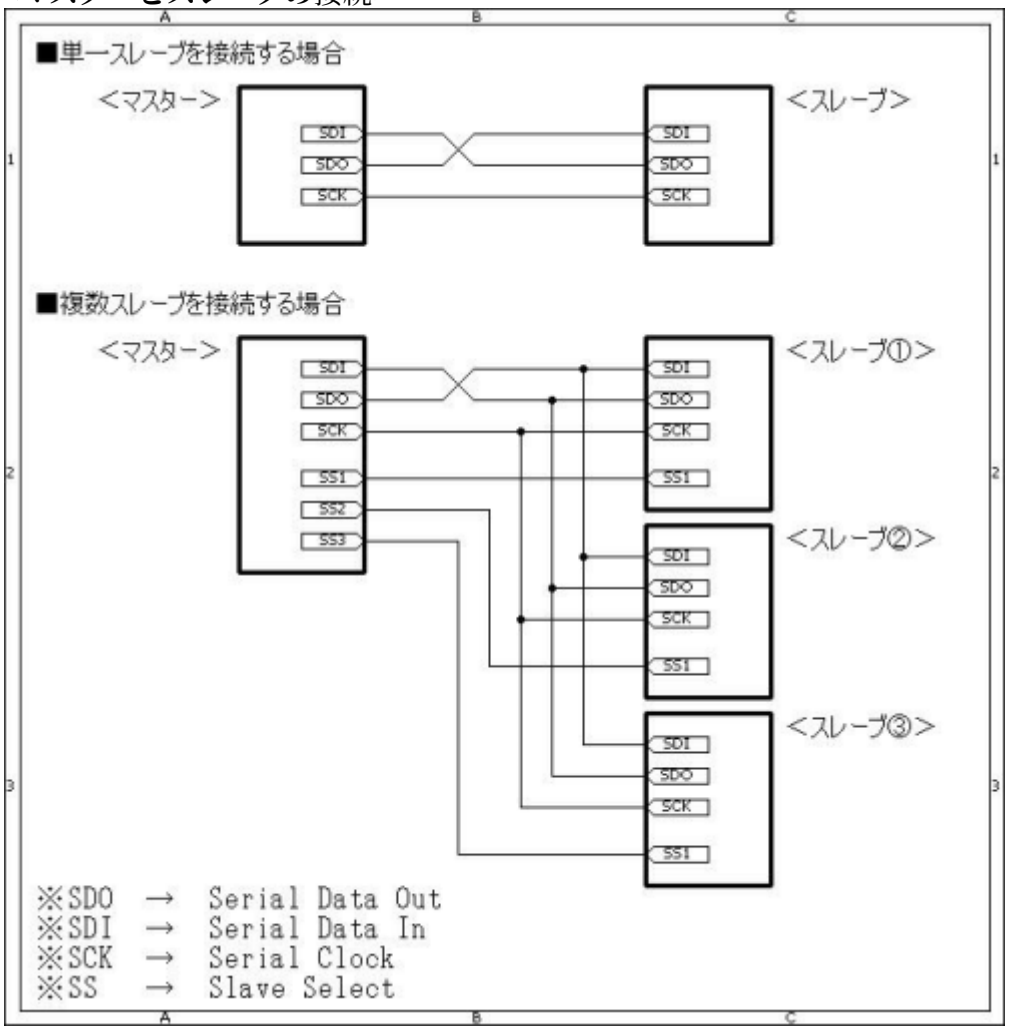
<マスター側の処理>

- スレーブ側にADC開始コマンドを送信します(ADC_START)
- チャネルを指定して、データ受信コマンドを送信します(ADC_CH_1~ADC_CH_4)
- スレーブ側からAD変換値を受信します(ad1~ad4)
- 受信したAD変換値を電圧値に換算し、LCDに表示します。

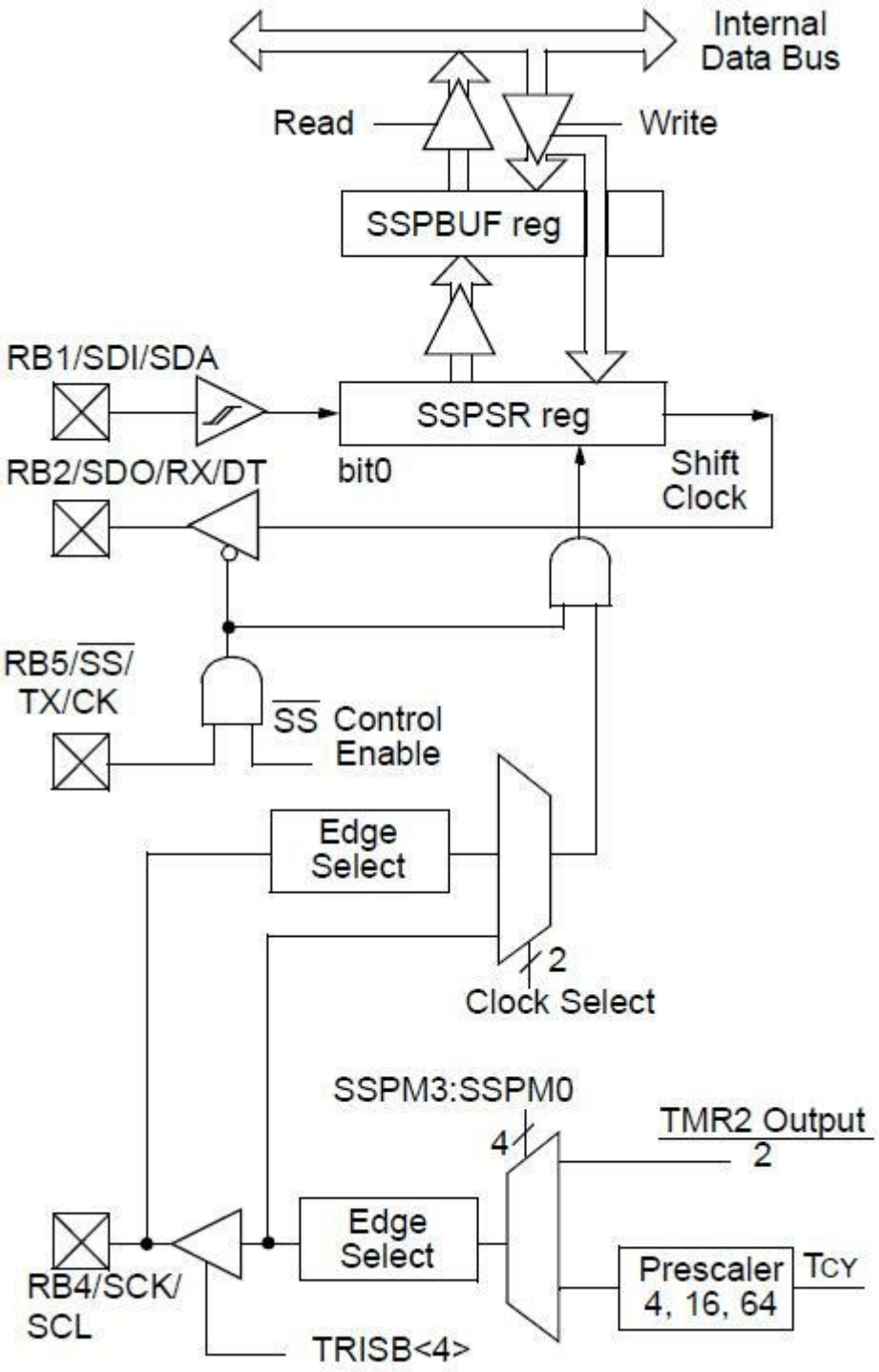
<スレーブ側の処理>

- マスター側からADC開始コマンドを受信すると、4チャネル分を全てAD変換し、結果を保存します。
- マスター側から、データ受信コマンドを受信すると、指定されたチャネルのAD変換値を送信します。

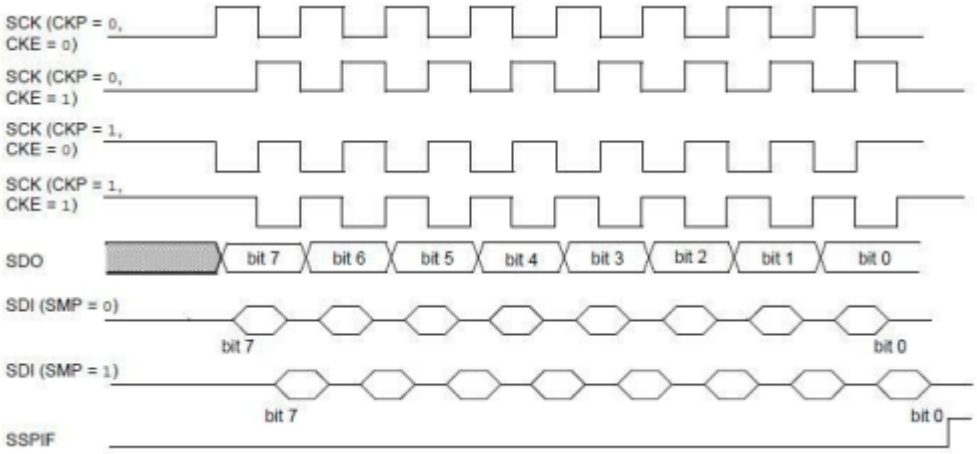
<マスターとスレーブの接続>



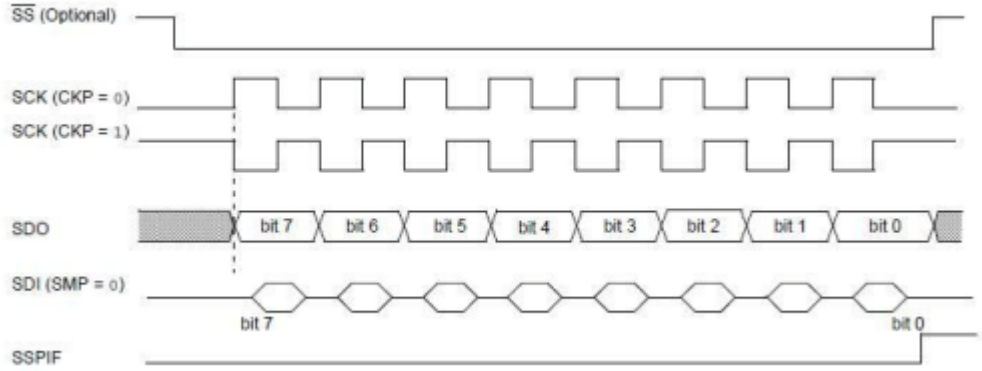
<SSPモジュールのブロックダイアグラム>



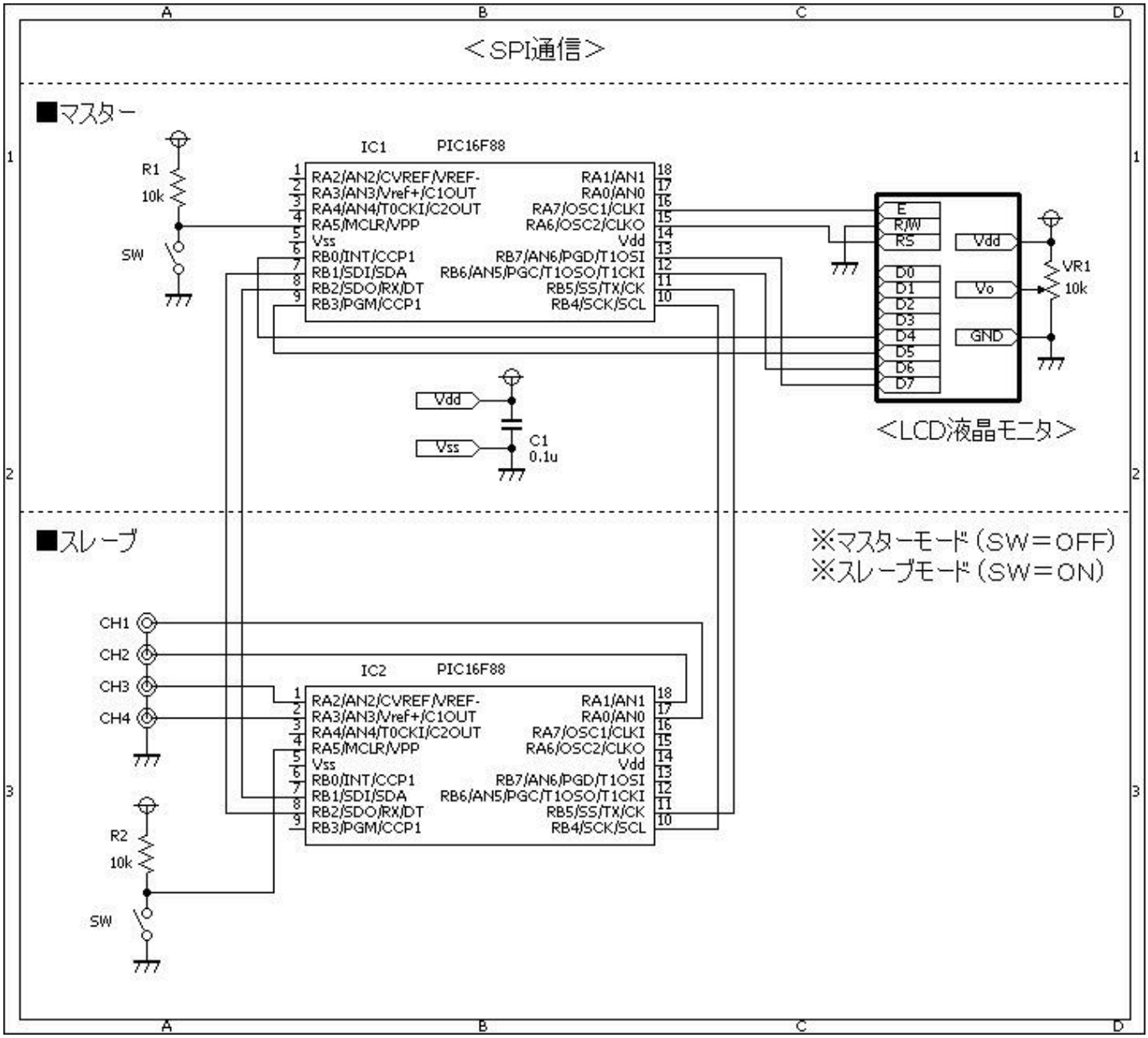
<マスター側のタイミングチャート>



<スレーブ側のタイミングチャート>



回路図



ソースコード

spi_test.c

```
//**********************************************************************
*
/*
□□□□□□テスト（マスターモード&スレーブモード）＞
  ■マスターの処理
□□□□□□□開始コマンドを送信する。
□□□□□□□□□□□□□のデータを受信する。（データ受信コマンド□□□□□□
□□□□□□□□□□□□□のデータを受信する。（データ受信コマンド□□□□□□
□□□□□□□□□□□□□のデータを受信する。（データ受信コマンド□□□□□□
□□□□□□□□□□□□□のデータを受信する。（データ受信コマンド□□□□□□
□□□□□□□□□□□□□のデータを□□□に表示する。
  ■スレーブの処理
      ・マスターからのコマンドを受信する。
```

```
□□□□□□□開始コマンドであれば□□□□□□□□を□□変換する。
      ・データ受信コマンドであれば、指定されたチャネルのデータを送信する。
*/
//*****************************************************************
*
//■■■関数宣言■■■
extern  void    main();
extern  void    SPI1_Write_Ss(unsigned short data_);
extern  unsigned  short  SPI1_Read_Ss(unsigned short buffer);
//*****************************************************************
*
//■■■マクロ定義■■■
//LCD
sbit  LCD_RS  at  RA6_bit;
sbit  LCD_EN  at  RA7_bit;
sbit  LCD_D7  at  RB7_bit;
sbit  LCD_D6  at  RB6_bit;
sbit  LCD_D5  at  RB3_bit;
sbit  LCD_D4  at  RB0_bit;
sbit  LCD_RS_Direction  at  TRISA6_bit;
sbit  LCD_EN_Direction  at  TRISA7_bit;
sbit  LCD_D7_Direction  at  TRISB7_bit;
sbit  LCD_D6_Direction  at  TRISB6_bit;
sbit  LCD_D5_Direction  at  TRISB3_bit;
sbit  LCD_D4_Direction  at  TRISB0_bit;
//SPI
sbit  Slave_Select  at  RB5_bit;
sbit  Slave_Select_Direction  at  TRISB5_bit;
sbit  SCK_Direction  at  TRISB4_bit;
//switch
sbit  SW_MODE  at  RA5_bit;
#define  MASTER_MODE  1
#define  SLAVE_MODE   0
//ADC
#define  ADC_START    0
#define  ADC_CH_1     1
#define  ADC_CH_2     2
#define  ADC_CH_3     3
#define  ADC_CH_4     4
//other
#define  INPUT_MODE   1
#define  OUTPUT_MODE  0
#define  WORD         unsigned  int
#define  BYTE         unsigned  short
//*****************************************************************
*
//■■■□□□書き込み関数（スレーブセレクト型）■■■
void    SPI1_Write_Ss(unsigned short data_)
{
        Slave_Select = 0;
        SPI1_Write(data_);
```

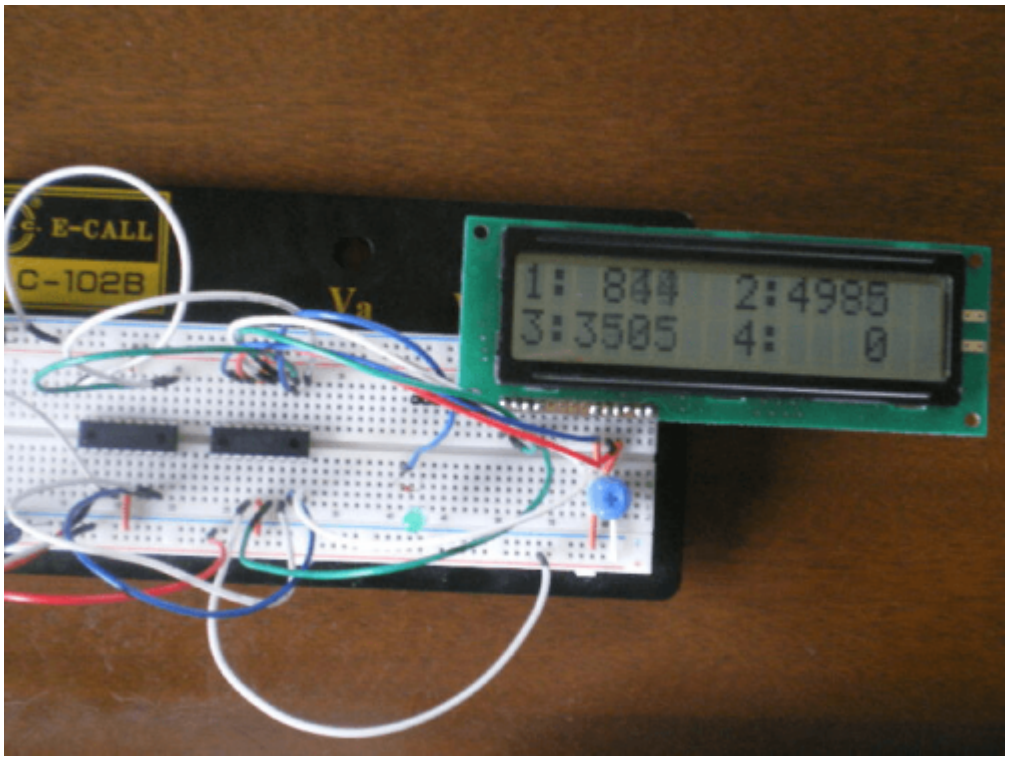
```
        Slave_Select = 1;
}
//*********************************************************************
*
//■■■□□□読み込み関数（スレーブセレクト型）■■■
BYTE    SPI1_Read_Ss(unsigned short buffer)
{
    BYTE    rd, dummy;
    //
        Slave_Select = 0;
        rd = SPI1_Read(dummy);
        Slave_Select = 1;
        return (rd);
}
//*********************************************************************
*
//■■■メイン関数■■■
void    main()
{
    int      ad1, ad2, ad3, ad4;
        short    dummy, mode;
        char     buf[8];
        double   ad;
        //
        OSCCON = 0b01110000;
        ANSEL  = 0b00001111;
        TRISA  = 0b11101111;
        //□□□初期化
        Lcd_Init();
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Cmd(_LCD_CURSOR_OFF);
        Lcd_Out(1, 1, "SPI test");
        if (SW_MODE == 1) {  //マスターモード
            Lcd_Out(2, 1, " -> Master mode");
        } else {  //スレーブモード
            Lcd_Out(2, 1, " -> Slave mode");
    }
        Delay_ms(1000);
        Lcd_Cmd(_LCD_CLEAR);
        //□□□初期化
        ADC_Init();
        //□□□初期化
        if (SW_MODE == 1) {  //マスターモード
                mode = MASTER_MODE;
                //
                Slave_Select = 1;
                Slave_Select_Direction = OUTPUT_MODE;
                SPI1_Init();
            SSPSTAT.CKE = 0;
                //
                Lcd_Out(1, 1, "1:");
```

```
                Lcd_Out(1, 9, "2:");
                Lcd_Out(2, 1, "3:");
                Lcd_Out(2, 9, "4:");
                Delay_us(500);
        } else {  //スレーブモード
                mode = SLAVE_MODE;
                //
                Slave_Select_Direction = INPUT_MODE;
                SPI1_Init();
            SSPSTAT.CKE = 0;
                //
                SCK_Direction = INPUT_MODE;  //SCKピンを入力モードにする。
                SSPCON.SSPM3 = 0;
                SSPCON.SSPM2 = 1;
                SSPCON.SSPM1 = 0;
                SSPCON.SSPM0 = 0;
        }
        //
        while (1) {
                switch (mode) {
                case MASTER_MODE:  //マスターモード
                    //□□□開始コマンド
                        SPI1_Write_Ss(ADC_START);
                        Delay_us(500);
                        //データ受信コマンド□□□□□
                        SPI1_Write_Ss(ADC_CH_1);
                        Delay_us(100);
                        ad1 = SPI1_Read_Ss(dummy) << 8;
                        ad1 |= SPI1_Read_Ss(dummy);
                        Delay_us(100);
                        //データ受信コマンド□□□□□
                        SPI1_Write_Ss(ADC_CH_2);
                        Delay_us(100);
                        ad2 = SPI1_Read_Ss(dummy) << 8;
                        ad2 |= SPI1_Read_Ss(dummy);
                        Delay_us(100);
                        //データ受信コマンド□□□□□
                        SPI1_Write_Ss(ADC_CH_3);
                        Delay_us(100);
                        ad3 = SPI1_Read_Ss(dummy) << 8;
                        ad3 |= SPI1_Read_Ss(dummy);
                        Delay_us(100);
                        //データ受信コマンド□□□□□
                        SPI1_Write_Ss(ADC_CH_4);
                        Delay_us(100);
                        ad4 = SPI1_Read_Ss(dummy) << 8;
                        ad4 |= SPI1_Read_Ss(dummy);
                        Delay_us(100);
                        //データ表示□□□□□
                        ad = ad1;
                        ad = ad * 4.8828125;
```

```
                        WordToStr(ad, buf);
                        Lcd_Out(1, 3, &buf[1]);
                        //データ表示□□□□□
                        ad = ad2;
                        ad = ad * 4.8828125;
                        WordToStr(ad, buf);
                        Lcd_Out(1, 11, &buf[1]);
                        //データ表示□□□□□
                        ad = ad3;
                        ad = ad * 4.8828125;
                        WordToStr(ad, buf);
                        Lcd_Out(2, 3, &buf[1]);
                        //データ表示□□□□□
                        ad = ad4;
                        ad = ad * 4.8828125;
                        WordToStr(ad, buf);
                        Lcd_Out(2, 11, &buf[1]);
                        //
                        Delay_ms(100);
                        break;
                case SLAVE_MODE:  //スレーブモード
                        switch (SPI1_Read(dummy)) {
                        case ADC_START:  //□□□開始コマンド
                                ad1 = ADC_Get_Sample(0);
                                ad2 = ADC_Get_Sample(1);
                                ad3 = ADC_Get_Sample(2);
                                ad4 = ADC_Get_Sample(3);
                                break;
                        case ADC_CH_1:  //データ受信コマンド□□□□□
                                SPI1_Write((ad1 >> 8) & 0xFF);
                                SPI1_Write(ad1 & 0xFF);
                                break;
                        case ADC_CH_2:  //データ受信コマンド□□□□□
                                SPI1_Write((ad2 >> 8) & 0xFF);
                                SPI1_Write(ad2 & 0xFF);
                                break;
                        case ADC_CH_3:  //データ受信コマンド□□□□□
                                SPI1_Write((ad3 >> 8) & 0xFF);
                                SPI1_Write(ad3 & 0xFF);
                                break;
                        case ADC_CH_4:  //データ受信コマンド□□□□□
                                SPI1_Write((ad4 >> 8) & 0xFF);
                                SPI1_Write(ad4 & 0xFF);
                                break;
                        }
                        break;
                }
        }
}
//*****************************************************************************
*
```

動作確認

左側のPICがスレーブ、右側のPICがマスターです。



マスターに接続しているLCDに、CH1~CH4の電圧値を表示します。