

簡易小型電圧計(7セグLCD)

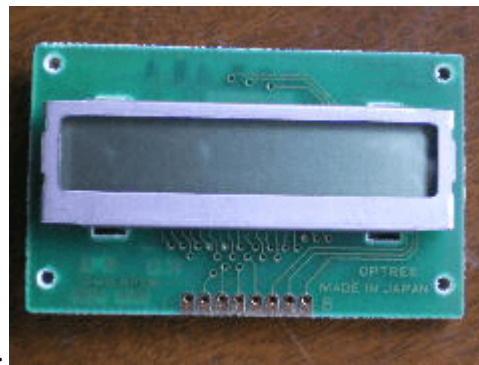
概要

久しぶりに大阪日本橋を散策したところデジットで格安(200円)の7セグLCDを見つけました。

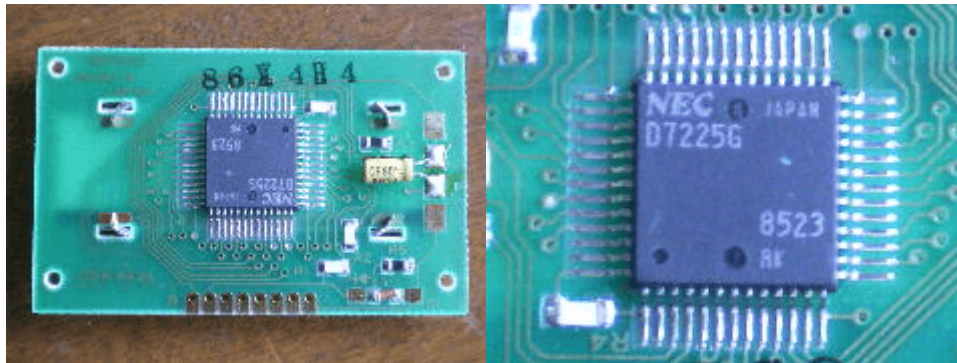
一般的によく見かけるLCDは、一文字が5×8のドットで構成されているので、英数字、カタカナ、特殊文字等の豊富な種類の文字を表示することができます。

今回購入した7セグLCDは、一文字が7セグ形式(a,b,c,d,e,f,g,DP)になっているので、一般的なLCDに比べると表示できる文字の種類は少なくなりますが、数字に特化した用途には十分使えそうです。

また、制御信号線も一般的なLCDでは少なくとも6本必要ですが、7セグLCDは5本で制御可能なので、ピン数の少ないマイコン(PIC12F683等)にも接続することができます。



<OPTREX PWB842B(7SEG LCD)の概観>





<付属の簡易説明書>

動作原理

A/D変換でアナログデータを1000回測定し、その平均値を7セグLCDに3時分割モードで表示させます。

動作原理(ハードウェア)

◎A/D変換

- PIC内臓のADC(10ビット逐次変換方式)でアナログデータを1000回測定し、平均値を求めます。
- 基準電圧を5V(Vdd)に設定し、分解能を約5mVとします□(5000mV÷1024)

7セグLCD

- コントローラにμPD7225G(NEC社製)を使用したOPTREX社製の7セグLCD(PWB842B)を接続します。
- コマンドモードのみで制御するため、5ピン(Command/Data)はHi(Vdd)に固定接続します。
- これにより全体を5本の線で制御することが可能となります。

<PWB842Bのピン配置>

SI	3ピン	入力	□Serial Input□ シリアル・データ(コマンド/データ)の入力端子で表示のためのデータおよびμPD7225の動作を制御する19種類のコマンドを入力します。
----	-----	----	---------------------------------------------------------------------------------------

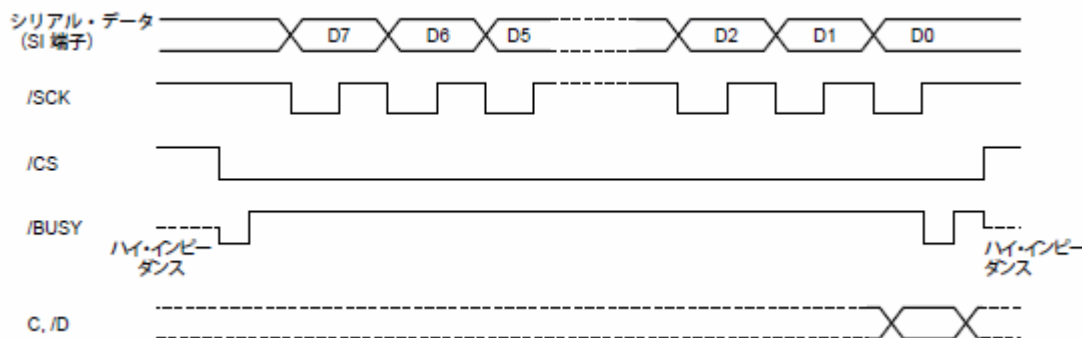
/SCK	2ピン	入力	□Serial Clock□ シリアル・データ(SI入力)のシフト・クロックで、立ち上がりエッジでSI入力の内容がシリアル・レジスタに1ビットずつ読み込まれます。 /SCK入力は/CS = 0のとき/BUSY = 1であれば有効となり□/BUSY = 0であれば無視されます。また/CS = 1のときは/BUSYに関係なく無視されます。
C,/D	5ピン	入力	□Command/Data□ SI端子より入力したシリアル・データがコマンドかまたはデータであるかを示す入力端子です□Lowでデータ□Hiでコマンドを表します。
/BUSY	4ピン	出力	アクティブLowの出力端子で、シリアル・データの入力禁止 / 許可を指示します□Lowで禁止、ハイで許可を表します。
/CS	6ピン	入力	□Chip Select□ CSをHiからLowにすることでμPD7225のSCKカウンタはクリアされ、シリアル・データの入力が可能となります。また、同時にデータ・ポインタを0番地にイニシャライズします。シリアル・データを入力後□/CSをHiにするとデータ・メモリの内容がディスプレイ・データ・ラッチに転送されLCDに表示されます。
/RESET	1ピン	入力	アクティブLowのリセット入力端子です。
Vdd	7ピン	-	+5V
Vss	8ピン	-	GND

動作原理(ソフトウェア)

メイン関数(main)

- ADCを初期化します。
- 7セグLCDを初期化します。
- ADCでアナログデータを1000回測定し、その平均値を求め、電圧値に換算します。
- 結果を7セグLCDに表示します。

<7セグLCDへの1バイト送信シーケンス>



7セグLCD制御関数 7セグLCDを制御するための各種関数群を提供します。

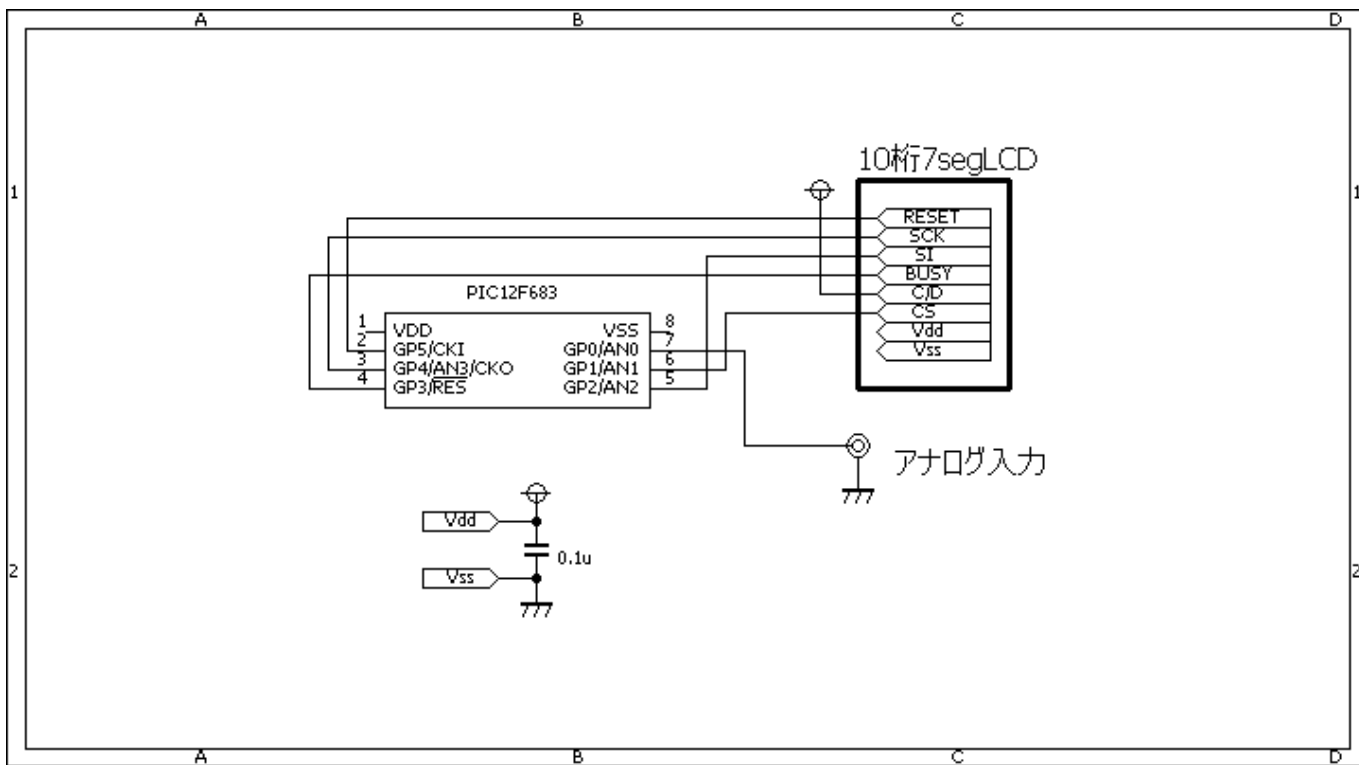
<7セグLCD制御関数リスト>

lcd7seg_init	7セグLCDを初期化します。
lcd7seg_spi	7セグLCDに、シリアル・データ(コマンド/データ)を1バイト(MSB□LSB)分送信します。
lcd7seg_cmd	7セグLCDに、コマンド列を送信します。
lcd7seg_chr	7セグLCDに、表示位置を指定して文字を表示します。
lcd7seg_out	7セグLCDに、表示位置を指定して文字列を表示します。

lcd7seg_dot 7セグLCDに、表示位置を指定してドットのオン/オフを制御します。

※AVRで7セグLCDを制御されている「[Recny](#)」さんの【[電子工作::ジャンクLCD](#)】を参考にさせて頂きました。

回路図



ソースコード

[volt_meter_7seg_lcd_v1_00.c](#)

```
//*****  
*  
/*  
    <簡易小型電圧計（7セグ）  
  
    社製<μPD7225>プログラマブルLCDコントローラ/ドライバ使用  
    *μPD7225は、ソフトウェアでプログラム可能なLCD[Liquid Crystal Display]液晶表示）  
    コントローラ/ドライバです。  
    マイクロコンピュータ応用システムにおいてCPUとシリアルにインタフェースしスタティッ  
    ク、2,3,4  
    時分割のLCDをダイレクトに制御駆動します。また特定のセグメント・パターンを発生する  
    セグメント・デコーダ  
    を内蔵しています。その他プリンキング（点滅）動作を制御することができます。  
*/  
//*****  
*
```

```

//LCD
sbit LCD_RESET at GP5_bit;
sbit LCD_SCK   at GP4_bit;
sbit LCD_SI    at GP2_bit;
sbit LCD_CS    at GP1_bit;
sbit LCD_BUSY  at GP3_bit;
//
#define BYTE    unsigned short
#define WORD    unsigned int
#define DWORD   unsigned long
//*****
*
extern void     main();
extern void     lcd7seg_init();
extern void     lcd7seg_spi(BYTE dt);
extern void     lcd7seg_cmd(BYTE *dt, BYTE len);
extern void     lcd7seg_chr(BYTE addr, BYTE dt);
extern void     lcd7seg_out(BYTE addr, BYTE *dt);
extern void     lcd7seg_dot(BYTE addr, BYTE on);
extern BYTE     INIT_CMD[];
extern BYTE     LCD_HEX[];
extern BYTE     DOT_ON[];
extern BYTE     DOT_OFF[];
//*****
*
BYTE     INIT_CMD[] = {
    0x48,          //MODE SET 1/3
    0x31,          //Synchronized transfer
    0x20,          //Clear Data Memory
    0x11,          //Display ON
    0x18,          //Blinking OFF
    0x14};        //WITHOUT SEGMENT decoder

//
BYTE     LCD_HEX[] = {
    0xD3,0xD5,0xD3,          //0
    0xD3,0xD0,0xD0,          //1
    0xD1,0xD7,0xD2,          //2
    0xD3,0xD7,0xD0,          //3
    0xD3,0xD2,0xD1,          //4
    0xD2,0xD7,0xD1,          //5
    0xD2,0xD7,0xD3,          //6
    0xD3,0xD1,0xD0,          //7
    0xD3,0xD7,0xD3,          //8
    0xD3,0xD7,0xD1,          //9
    0xD3,0xD3,0xD3,          //A
    0xD2,0xD6,0xD3,          //B
    0xD0,0xD5,0xD3,          //C
    0xD3,0xD6,0xD2,          //D
    0xD0,0xD7,0xD3,          //E
    0xD0,0xD3,0xD3,          //F
    0xD7,0xD7,0xD3,          //all(0xFF)

```

```
    0xD0, 0xD0, 0xD0,      //space
    0xD0, 0xD2, 0xD0,      //-
    0xD0, 0xD5, 0xD0};    //=  
  
//  
BYTE   DOT_ON[] = { 0xB4, 0xB0, 0xB0};  
BYTE   DOT_OFF[] = { 0x93, 0x97, 0x93};  
//*****  
*  
//      フセグ初期化関数  
void   lcd7seg_init()  
{  
    LCD_CS = 1;  
    LCD_SCK = 1;  
    LCD_SI = 1;  
    LCD_RESET = 0;  
    Delay_us(1000);  
    LCD_RESET = 1;  
    Delay_us(1000);  
    //  
    lcd7seg_cmd(INIT_CMD, 6);  
}  
//*****  
*  
//      8ビット送信関数  
void   lcd7seg_spi(BYTE dt)  
{  
    short   cnt;  
    //  
    while (LCD_BUSY == 0) {  
    }  
    for (cnt = 0; cnt < 8; cnt++) {  
        LCD_SI = ((dt << cnt) & 0b10000000) == 0 ? 0 : 1;  
        LCD_SCK = 0;  
        LCD_SCK = 1;  
    }  
    while (LCD_BUSY == 0) {  
    }  
}  
//*****  
*  
//      コマンド設定関数  
void   lcd7seg_cmd(BYTE *dt, BYTE len)  
{  
    short   cnt;  
    //  
    LCD_CS = 0;  
    for (cnt = 0; cnt < len; cnt++) {  
        lcd7seg_spi(*(dt + cnt));  
    }  
    LCD_CS = 1;  
}  
}
```

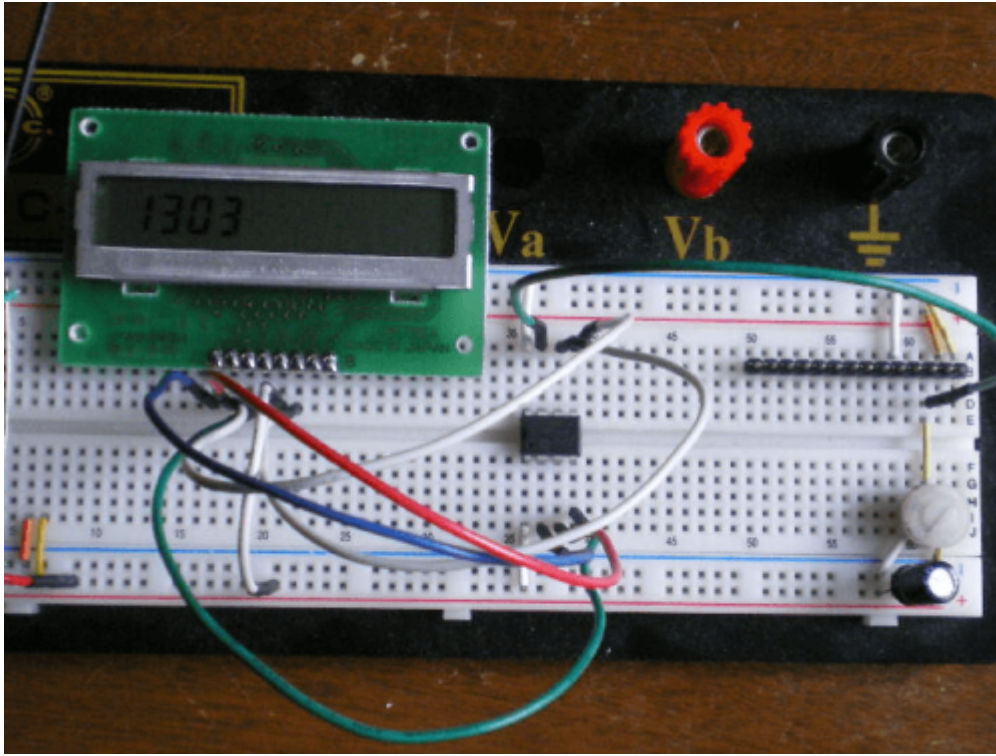
```
//*****  
*  
// 文字表示関数  
void lcd7seg_chr(BYTE addr, BYTE dt)  
{  
    LCD_CS = 0;  
    lcd7seg_spi(0xE0 | (27 - ((addr << 1) + (addr))));  
    switch (dt) {  
        case '0':  
            lcd7seg_spi(LCD_HEX[0]);  
            lcd7seg_spi(LCD_HEX[1]);  
            lcd7seg_spi(LCD_HEX[2]);  
            break;  
        case '1':  
            lcd7seg_spi(LCD_HEX[3]);  
            lcd7seg_spi(LCD_HEX[4]);  
            lcd7seg_spi(LCD_HEX[5]);  
            break;  
        case '2':  
            lcd7seg_spi(LCD_HEX[6]);  
            lcd7seg_spi(LCD_HEX[7]);  
            lcd7seg_spi(LCD_HEX[8]);  
            break;  
        case '3':  
            lcd7seg_spi(LCD_HEX[9]);  
            lcd7seg_spi(LCD_HEX[10]);  
            lcd7seg_spi(LCD_HEX[11]);  
            break;  
        case '4':  
            lcd7seg_spi(LCD_HEX[12]);  
            lcd7seg_spi(LCD_HEX[13]);  
            lcd7seg_spi(LCD_HEX[14]);  
            break;  
        case '5':  
            lcd7seg_spi(LCD_HEX[15]);  
            lcd7seg_spi(LCD_HEX[16]);  
            lcd7seg_spi(LCD_HEX[17]);  
            break;  
        case '6':  
            lcd7seg_spi(LCD_HEX[18]);  
            lcd7seg_spi(LCD_HEX[19]);  
            lcd7seg_spi(LCD_HEX[20]);  
            break;  
        case '7':  
            lcd7seg_spi(LCD_HEX[21]);  
            lcd7seg_spi(LCD_HEX[22]);  
            lcd7seg_spi(LCD_HEX[23]);  
            break;  
        case '8':  
            lcd7seg_spi(LCD_HEX[24]);  
            lcd7seg_spi(LCD_HEX[25]);  
    }  
}
```

```
        lcd7seg_spi(LCD_HEX[26]);
        break;
case '9':
    lcd7seg_spi(LCD_HEX[27]);
    lcd7seg_spi(LCD_HEX[28]);
    lcd7seg_spi(LCD_HEX[29]);
    break;
case 'A':
case 'a':
    lcd7seg_spi(LCD_HEX[30]);
    lcd7seg_spi(LCD_HEX[31]);
    lcd7seg_spi(LCD_HEX[32]);
    break;
case 'B':
case 'b':
    lcd7seg_spi(LCD_HEX[33]);
    lcd7seg_spi(LCD_HEX[34]);
    lcd7seg_spi(LCD_HEX[35]);
    break;
case 'C':
case 'c':
    lcd7seg_spi(LCD_HEX[36]);
    lcd7seg_spi(LCD_HEX[37]);
    lcd7seg_spi(LCD_HEX[38]);
    break;
case 'D':
case 'd':
    lcd7seg_spi(LCD_HEX[39]);
    lcd7seg_spi(LCD_HEX[40]);
    lcd7seg_spi(LCD_HEX[41]);
    break;
case 'E':
case 'e':
    lcd7seg_spi(LCD_HEX[42]);
    lcd7seg_spi(LCD_HEX[43]);
    lcd7seg_spi(LCD_HEX[44]);
    break;
case 'F':
case 'f':
    lcd7seg_spi(LCD_HEX[45]);
    lcd7seg_spi(LCD_HEX[46]);
    lcd7seg_spi(LCD_HEX[47]);
    break;
case 0xFF:
    lcd7seg_spi(LCD_HEX[48]);
    lcd7seg_spi(LCD_HEX[49]);
    lcd7seg_spi(LCD_HEX[50]);
    break;
case ' ':
    lcd7seg_spi(LCD_HEX[51]);
```

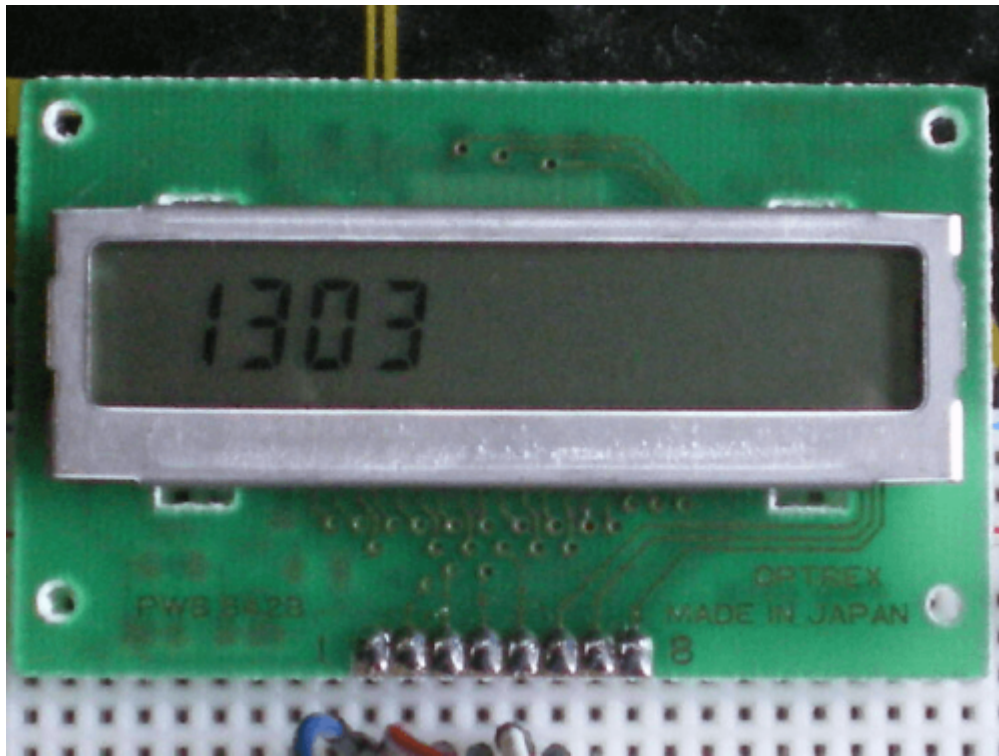
```
        lcd7seg_spi(LCD_HEX[52]);
        lcd7seg_spi(LCD_HEX[53]);
        break;
    case '-':
        lcd7seg_spi(LCD_HEX[54]);
        lcd7seg_spi(LCD_HEX[55]);
        lcd7seg_spi(LCD_HEX[56]);
        break;
    case '=':
        lcd7seg_spi(LCD_HEX[57]);
        lcd7seg_spi(LCD_HEX[58]);
        lcd7seg_spi(LCD_HEX[59]);
        break;
}
LCD_CS = 1;
}
//*****
*
// 文字列表示関数
void lcd7seg_out(BYTE addr, BYTE *dt)
{
    while (*dt != 0x00) {
        lcd7seg_chr(addr, *dt);
        dt++;
        addr++;
    }
}
//*****
*
// ドット表示関数
void lcd7seg_dot(BYTE addr, BYTE dot)
{
    LCD_CS = 0;
    lcd7seg_spi(0xE0 | (27 - ((addr << 1) + (addr))));
    if(dot == 1) {
        lcd7seg_spi(DOT_ON[0]);
        lcd7seg_spi(DOT_ON[1]);
        lcd7seg_spi(DOT_ON[2]);
    } else {
        lcd7seg_spi(DOT_OFF[0]);
        lcd7seg_spi(DOT_OFF[1]);
        lcd7seg_spi(DOT_OFF[2]);
    }
    LCD_CS = 1;
}
//*****
*
// メイン関数
void main()
{
    int cnt;
```

```
double ad;
char buf[10];
//
OSCCON = 0b11100000;
CMCON0 = 0b00000111;
ANSEL = 0b00000001;
TRISIO = 0b00001001;
//
ADC_Init();
//
lcd7seg_init();
//
for (cnt = 0; cnt < 10; cnt++) {
    lcd7seg_chr(cnt, 0xFF);
    Delay_ms(50);
}
for (cnt = 0; cnt < 10; cnt++) {
    lcd7seg_chr(cnt, '=');
    Delay_ms(50);
}
for (cnt = 0; cnt < 10; cnt++) {
    lcd7seg_chr(cnt, '-');
    Delay_ms(50);
}
for (cnt = 0; cnt < 10; cnt++) {
    lcd7seg_chr(cnt, ' ');
    Delay_ms(50);
}
//
while (1) {
    //□□□変換&換算
    ad = 0.0;
    for (cnt = 0; cnt < 1000; cnt++) {
        ad += ADC_Get_Sample(0);
    }
    ad = (ad * 4.8828125) / 1000.0;
    //結果表示
    WordToStr(ad, buf);
    lcd7seg_out(0, &buf[1]);
}
//*****
*
```

動作確認



電圧(1303mV)を測定して



みました。

From:
<http://www.deepsky.jp/wiki/> - うごくといいな

Permanent link:
<http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic12f683:40&rev=1588141079>

Last update: 2025/10/17 14:27

