電圧&電流計アダプタV4(汎用型)

概要

電圧計&電流計を各種製作してきましたが、電圧および電流の測定範囲については、どれも固定されたものになっています。

- 電圧計&電流計アダプタ→電圧=0V~25V、電流=0A~2A
- 電圧計&電流計アダプタV2(バー表示)→電圧=0V~55V、電流=0A~4.5A
- 電圧&電流計V3(7セグ表示)→電圧=0V~25V、電流=0A~2A

しかし、用途によっては測定範囲(上限値)を上げたい(または下げたい)場合が、多々発生します。その都度プログラムを修正するのは煩わしいものです。

そこで今回は、プログラムを修正せずに測定範囲を設定可能な「電圧&電流計アダプタ」を製作してみました。

<仕様>

- 電圧の測定表示は、分解能=100mVとします。<50点バー表示機能搭載>
- 電流の測定表示は、分解能=100mAとします。<50点バー表示機能搭載>
- 電圧の測定範囲を設定可能とします。
  - 最小5V=5V×1倍
  - 最大5000V=5V×1000倍→表示上は99.9Vが最大となります。
- 電流検出抵抗の値を設定可能とします。
  - 0.01Ω~10Ω
- 電流の測定範囲を設定可能とします。
  - 電流検出抵抗の値と電圧増福率の値の組み合わせで範囲を決定します。
  - 最小500μA=5V÷(10Ω×1000倍)→表示上は100mAが最小となります。
  - 最大500A=5V÷(0.01Ω×1倍)→表示上は99.9Aが最大となります。

動作原理

設定変更可能な、電流検出抵抗(R)、電圧増幅率(G)、電圧分圧率(D)の値と、ADCで取り込んだ電圧(V1,V2)を用いて、電圧および電流を算出し表示します。

```
電圧=(V1×D)-(V2÷G)
電流=(V2÷G)÷R
```

動作原理(ハードウェア)

◎電圧&電流検出

- 回路は特定しませんが、以下の値が含まれている事とします。
  - 電流検出抵抗(R)、電圧増幅率(G)、電圧分圧率(D)、電圧増幅後の電圧(V2)、電圧分圧後の電圧(V1)
- 下図の回路図中の電圧&電流検出回路は参考です。

動作原理(ソフトウェア)

◎メイン関数(main)

- ADCの初期化を行います。
- LCDの初期化を行います。
- 定数(R､G､D)をEEPROMより読み込みます｡(read_literal)
  R=i_resistor､G=i_scale､D=v_scale
- スイッチ(SW_MODE)がオフ状態であれば測定表示関数を呼び出します｡(measrement)
- スイッチ(SW_MODE)がオン状態であれば定数設定関数を呼び出します｡(configuration)

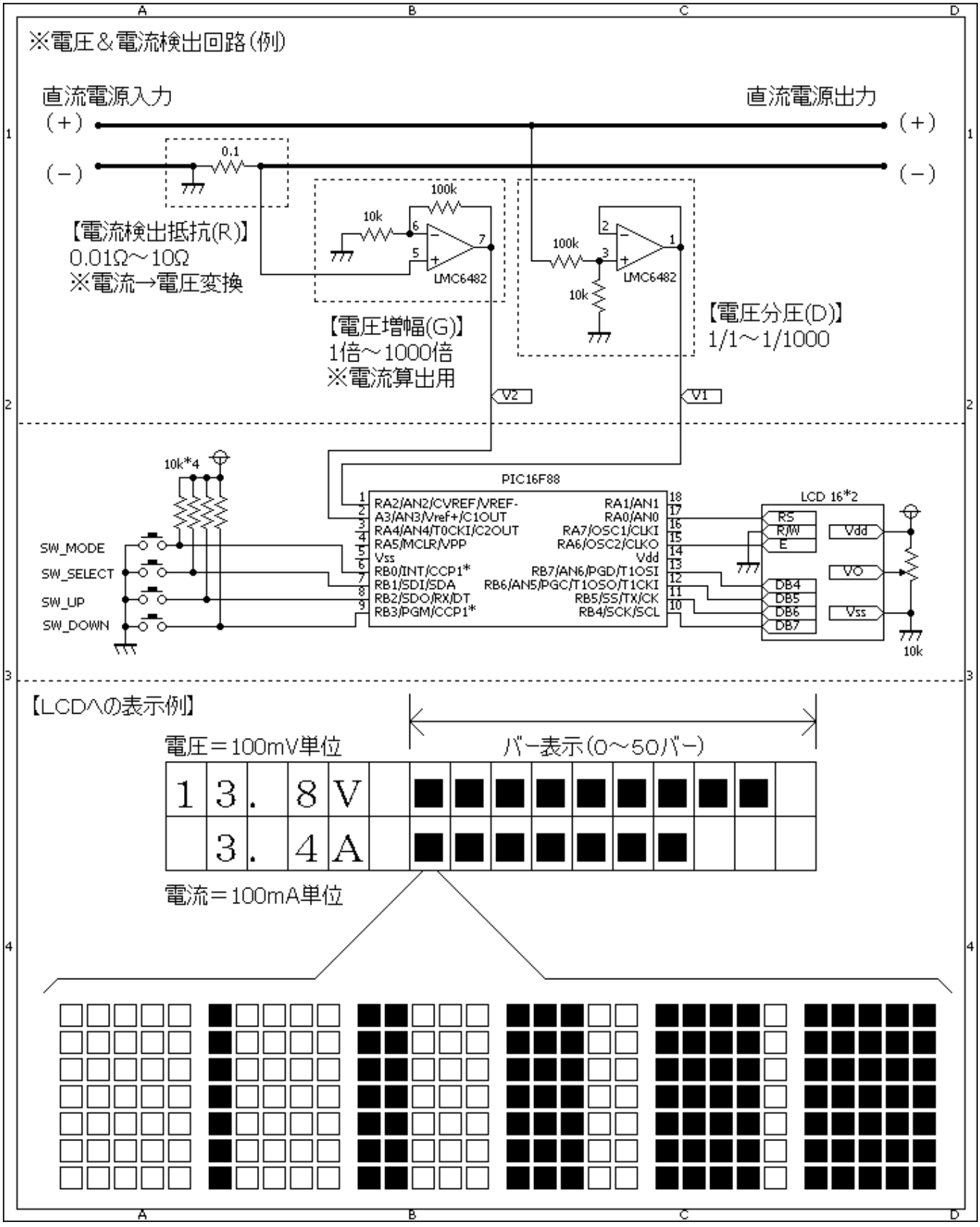
◎測定表示関数(measrement)

- V1､V2の値をA/D変換で取り込みます。
- V1､V2の値をバー表示します｡(BarDisp)
- V1､V2､R､G､Dより出力電圧および出力電流を求めます。
- 求めた値をLCDに表示します。

◎定数設定関数(configuration)

- 定数(R､G､D)の値をLCDに表示します。
- スイッチ(SW_SELECT)で変更する定数(R､G､D)を選択します。
- 選択した定数を変更(+1、-1)します。
- 変更した値をEEPROMに書き込みます｡(write_literal)

回路図



ソースコード

vi_meter_v4.c

```
//*********************************************************************
*
/*
    『電圧＆電流計□□□汎用型）』
*/
//*********************************************************************
*
//■■■マクロ定義■■■
#define BYTE            unsigned  char
#define WORD            unsigned  int
#define DWORD           unsigned  long
//
#define SW_MODE         PORTB.B0
#define SW_SELECT       PORTB.B1
#define SW_UP           PORTB.B2
#define SW_DOWN         PORTB.B3
//*********************************************************************
*
//■■■関数宣言■■■
extern  void    main();
extern  int     ADC_Get_Sample_average(unsigned short channel);
extern  void    lcd_custom_init();
extern  void    RegistCustomChar();
extern  void    BarDisp(char row, char column, short dat, short bar);
extern  void    measrement();
extern  void    configuration();
extern  void    read_literal();
extern  void    write_literal();
//*********************************************************************
*
union {
        long        _long;
        short       _short[4];
}       v_scale, i_scale, i_resistor;
short   configuration_mode = 0;
char    buf[10];
//*********************************************************************
*
void    main()
{
        OSCCON = 0b01110000;
        CMCON  = 0b00000111;
        ANSEL  = 0b00001100;
        TRISA  = 0b10111110;
        TRISB  = 0b00001111;
        //
        ADC_Init();
        lcd_custom_init();
        read_literal();
        //
        if (SW_MODE == 1) {
```

```
                while (1) {
                        measrement();
                }
        } else {
                while (1) {
                        configuration();
                }
        }
}
//*****************************************************************************
*
void    measrement()
{
        double  v1, v2, i;
        long    tmp;
        //
        v1 = ADC_Get_Sample_average(2);
        v2 = ADC_Get_Sample_average(3);
        //
        BarDisp(1, 7, v1 / 21, 50);
        BarDisp(2, 7, v2 / 21, 50);
        //
        v1 = (v1 * v_scale._long) * 4.8828125;
        v2 = (v2 / i_scale._long) * 4.8828125;
        i = (v2 / i_resistor._long) * 100.0;
        //
        tmp = (long)v1 - (long)v2;
        if ((tmp % 100) > 50) {
                tmp = (tmp / 100) + 1;
        } else {
                tmp = tmp / 100;
        }
        WordToStr(tmp, buf);
        buf[0] = buf[2];
        buf[1] = (buf[3] == ' ') ? '0' : buf[3];
        buf[2] = '.';
        buf[3] = buf[4];
        buf[4] = 'V';
        buf[5] = 0x00;
        Lcd_Out(1, 1, buf);
        //
        tmp = (long)i;
        if ((tmp % 100) > 50) {
                tmp = (tmp / 100) + 1;
        } else {
                tmp = tmp / 100;
        }
        WordToStr(tmp, buf);
        buf[0] = buf[2];
        buf[1] = (buf[3] == ' ') ? '0' : buf[3];
        buf[2] = '.';
```

```
        buf[3] = buf[4];
        buf[4] = 'A';
        buf[5] = 0x00;
        Lcd_Out(2, 1, buf);
}
//*********************************************************************
*
void    configuration()
{
        long    tmp;
        //
        WordToStr(v_scale._long, buf);
        buf[5] = ' ';
        buf[6] = 0x00;
        Lcd_Out(1, 1, buf);
        WordToStr(i_scale._long, buf);
        buf[5] = ' ';
        buf[6] = 0x00;
        Lcd_Out(2, 1, buf);
        WordToStr(i_resistor._long, buf);
        buf[5] = '0';
        buf[6] = 'm';
        buf[7] = 0xF4;
        buf[8] = ' ';
        buf[9] = 0x00;
        Lcd_Out(2, 8, buf);
        switch (configuration_mode) {
        case 0:
                Lcd_Chr(1, 1, '[');
                Lcd_Chr(1, 6, ']');
                break;
        case 1:
                Lcd_Chr(2, 1, '[');
                Lcd_Chr(2, 6, ']');
                break;
        case 2:
                Lcd_Chr(2, 8, '[');
                Lcd_Chr(2, 16, ']');
                break;
        }
        //
        if (SW_SELECT == 0) {
                configuration_mode++;
                if (configuration_mode == 3) {
                        configuration_mode = 0;
                }
        }
        if (SW_UP == 0) {
                switch (configuration_mode) {
                case 0:
                        tmp = v_scale._long;
```

```
                        if (tmp < 1000) {
                                tmp++;
                        }
                        v_scale._long = tmp;
                        break;
                case 1:
                        tmp = i_scale._long;
                        if (tmp < 1000) {
                                tmp++;
                        }
                        i_scale._long = tmp;
                        break;
                case 2:
                        tmp = i_resistor._long;
                        if (tmp < 1000) {
                                tmp++;
                        }
                        i_resistor._long = tmp;
                        break;
                }
                write_literal();
        }
        if (SW_DOWN == 0) {
                switch (configuration_mode) {
                case 0:
                        tmp = v_scale._long;
                        if (tmp > 1) {
                                tmp--;
                        }
                        v_scale._long = tmp;
                        break;
                case 1:
                        tmp = i_scale._long;
                        if (tmp > 1) {
                                tmp--;
                        }
                        i_scale._long = tmp;
                        break;
                case 2:
                        tmp = i_resistor._long;
                        if (tmp > 1) {
                                tmp--;
                        }
                        i_resistor._long = tmp;
                        break;
                }
                write_literal();
        }
        Delay_ms(100);
}
//******************************************************************************
```

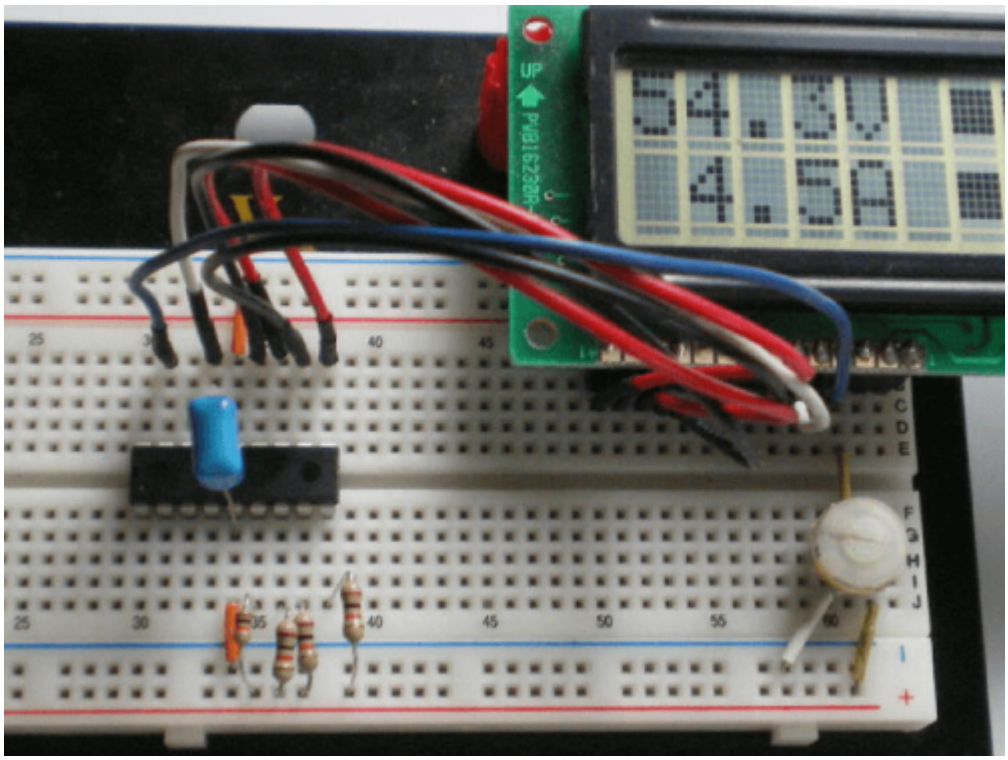
```
*
int     ADC_Get_Sample_average(unsigned short channel)
{
        int     cnt;
        long    ad;
        //
        ad = 0;
        for (cnt = 0; cnt < 1000; cnt++) {
                ad += ADC_Get_Sample(channel);
        }
        return (ad /1000);
}
//*****************************************************************************
*
//■■■定数読み込み関数■■■
void    read_literal()
{
        v_scale._short[0] = Eeprom_Read(0);
        v_scale._short[1] = Eeprom_Read(1);
        v_scale._short[2] = Eeprom_Read(2);
        v_scale._short[3] = Eeprom_Read(3);
        v_scale._long = ((v_scale._long < 1) || (v_scale._long > 1000))
? 1 : v_scale._long;
        //
        i_scale._short[0] = Eeprom_Read(4);
        i_scale._short[1] = Eeprom_Read(5);
        i_scale._short[2] = Eeprom_Read(6);
        i_scale._short[3] = Eeprom_Read(7);
        i_scale._long = ((i_scale._long < 1) || (i_scale._long > 1000))
? 1 : i_scale._long;
        //
        i_resistor._short[0] = Eeprom_Read(8);
        i_resistor._short[1] = Eeprom_Read(9);
        i_resistor._short[2] = Eeprom_Read(10);
        i_resistor._short[3] = Eeprom_Read(11);
        i_resistor._long = ((i_resistor._long < 1) || (i_resistor._long
> 1000)) ? 100 : i_resistor._long;
}
//*****************************************************************************
*
//■■■定数書き込み関数■■■
void    write_literal()
{
        Eeprom_Write(0, v_scale._short[0]);
        Eeprom_Write(1, v_scale._short[1]);
        Eeprom_Write(2, v_scale._short[2]);
        Eeprom_Write(3, v_scale._short[3]);
        //
        Eeprom_Write(4, i_scale._short[0]);
        Eeprom_Write(5, i_scale._short[1]);
        Eeprom_Write(6, i_scale._short[2]);
```

```
        Eeprom_Write(7, i_scale._short[3]);
        //
        Eeprom_Write(8, i_resistor._short[0]);
        Eeprom_Write(9, i_resistor._short[1]);
        Eeprom_Write(10, i_resistor._short[2]);
        Eeprom_Write(11, i_resistor._short[3]);
}
//*****************************************************************************
*
//■■■□□□初期化関数■■■
sbit LCD_RS at RA0_bit;
sbit LCD_EN at RA6_bit;
sbit LCD_D7 at RB4_bit;
sbit LCD_D6 at RB5_bit;
sbit LCD_D5 at RB6_bit;
sbit LCD_D4 at RB7_bit;
sbit LCD_RS_Direction at TRISA0_bit;
sbit LCD_EN_Direction at TRISA6_bit;
sbit LCD_D7_Direction at TRISB4_bit;
sbit LCD_D6_Direction at TRISB5_bit;
sbit LCD_D5_Direction at TRISB6_bit;
sbit LCD_D4_Direction at TRISB7_bit;
//
void    lcd_custom_init()
{
        short   cnt;
        //
        Delay_ms(100);
        Lcd_Init();
        RegistCustomChar();
        Lcd_Cmd(_LCD_CURSOR_OFF);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "V&I meter v4");
        for (cnt = 0; cnt <= 80; cnt++) {
                BarDisp(2, 1, cnt, 80);
                Delay_ms(10);
        }
        Lcd_Cmd(_LCD_CLEAR);
}
//*****************************************************************************
*
//■■■キャラクタ登録関数■■■
const char character0[] = { 0, 0, 0, 0, 0, 0, 0, 0};
const char character1[] = { 0,16,16,16,16,16, 0, 0};
const char character2[] = { 0,24,24,24,24,24, 0, 0};
const char character3[] = { 0,28,28,28,28,28, 0, 0};
const char character4[] = { 0,30,30,30,30,30, 0, 0};
const char character5[] = { 0,31,31,31,31,31, 0, 0};
//
void    RegistCustomChar()
{
```

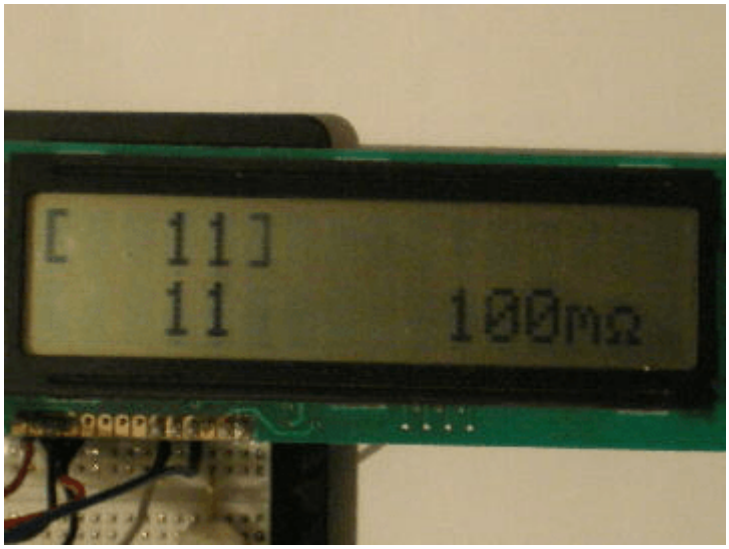
```
        char    i;
        //
        LCD_Cmd(64);
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character0[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character1[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character2[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character3[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character4[i]);
        }
        for (i = 0; i<=7; i++) {
                LCD_Chr_Cp(character5[i]);
        }
        LCD_Cmd(_LCD_RETURN_HOME);
}
//*********************************************************************
*
//■■■バー表示関数■■■
void    BarDisp(char row, char column, short dat, short bar)
{
        short   j, k, cnt;
        //
        j = dat / 5;
        k = dat - (j * 5);
        //
        if (row == 1)
                Lcd_Cmd(_LCD_FIRST_ROW);
        else
                Lcd_Cmd(_LCD_SECOND_ROW);
        //
        for (cnt = 1; cnt < column; cnt++) {
                Lcd_Cmd(_LCD_MOVE_CURSOR_RIGHT);
        }
        //
        for (cnt = 0; cnt < j; cnt++) {
                Lcd_Chr_Cp(5);
        }
        Lcd_Chr_Cp(k);
        for (cnt++; cnt < (bar / 5); cnt++) {
                Lcd_Chr_Cp(' ');
        }
}
//*********************************************************************
```

*

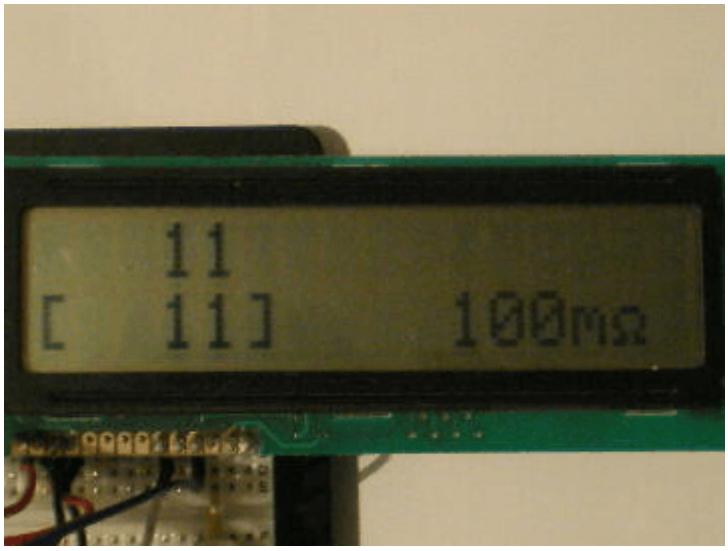
動作確認



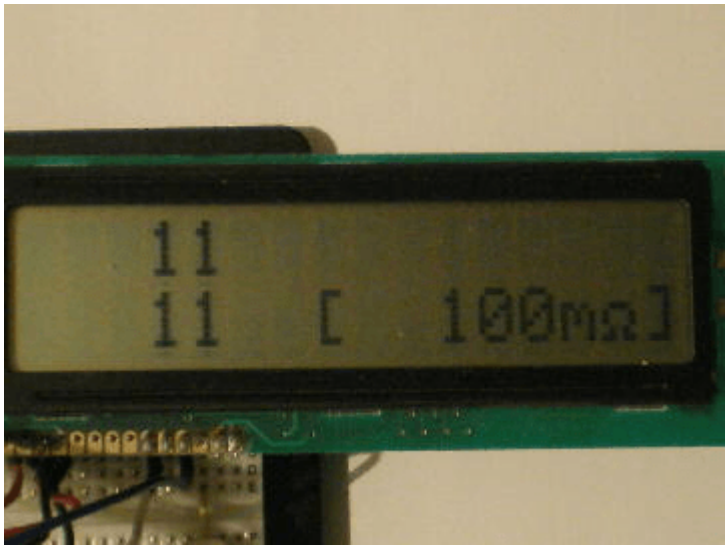
スイッチ(SW_MODE)をオン状態で起動すると設定モードになります。 スイッチ(SW_SELECT)で変更する定数を選択します。【】マークが移動します。 左側:電圧測定用の分圧抵抗の比率(D)を設定します。



右側:電流測定用の増幅率(G)を設定します。



電流測定用の電流検出抵抗(R)の値をmΩ単位で設定します。



スイッチ(SW_MODE)をオフ状態で起動すると測定モードになります。 実際の測定時の表示画面です。