LCD制御ライブラリ(mikroCコンパチブル)

概要

mikroCには、標準でLCD制御ライブラリが含まれているので、LCDの制御命令、信号のタイミング、制御手順 などを意識することなく、容易にLCDを利用することが出来ます。

しかし、今回はLCD制御の勉強も兼ねて、LCD制御ライブラリ」を自作してみました。 ライブラリの仕様は、mikroCコンパチブルとしました。従って、関数名さえ変更すれば、どちらのライブラリでも 同じように動作させることが出来ます。

mikroC版LCD制御ライブラリ	自作LCD制御ライブラリ
Lcd_Init	Lcd2_Init
Lcd_Out	Lcd2_Out
Lcd_Out_Cp	Lcd2_Out_Cp
Lcd_Chr	Lcd2_Chr
Lcd_Chr_Cp	Lcd2_Chr_Cp
Lcd_Cmd	Lcd2_Cmd

動作原理

PICにLCDを接続する方法には、次の4種類があります。

- 接続線6本(4ビットモード+ビジーフラグ(BF)未使用)
- 接続線7本(4ビットモード+ビジーフラグ(BF)使用)
- 接続線10本(8ビットモード+ビジーフラグ(BF)未使用)
- 接続線11本(8ビットモード+ビジーフラグ(BF)使用)

今回は、接続線の最も少ない(4ビットモード+ビジーフラグ(BF)未使用)を採用しました。

<LCD制御ライブラリで提供する関数>

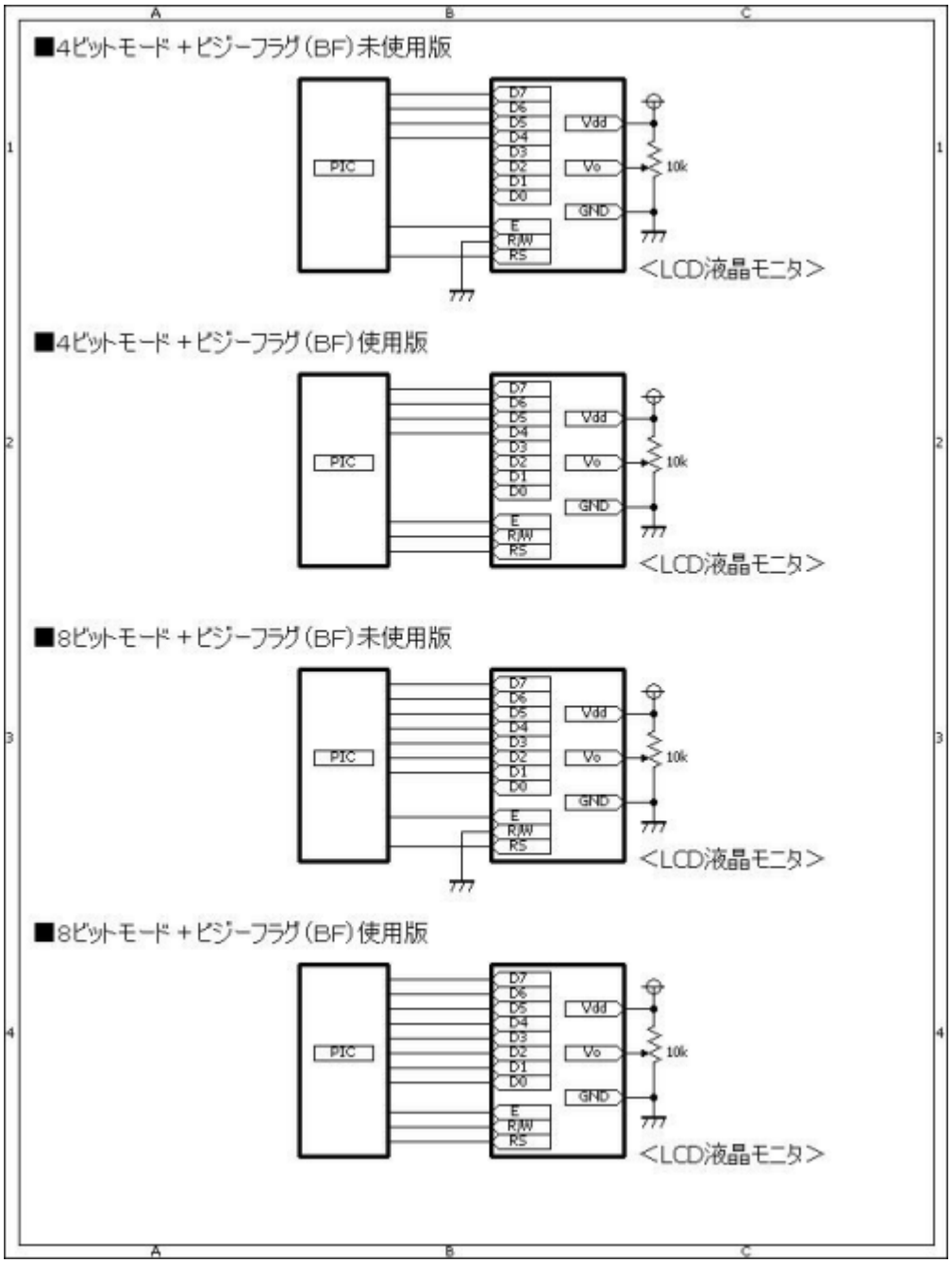
- Lcd2_Init();
  LCDを初期化します。
  データビット(7,6,5,4)、レジスタ選択信号(RS)、イネーブル信号(E)を指定します。
- Lcd2_Out(char row, char column, char *text);
  行(row)と列(column)を指定して、LCDに文字列(text)を表示します。
- Lcd2_Out_Cp(char *text);
  カレントカーソルに、LCDに文字列(text)を表示します。
- Lcd2_Chr(char row, char column, char out_char);
  行(row)と列(column)を指定して、LCDに文字(out_char)を表示します。
- Lcd2_Chr_Cp(char out_char);
  カレントカーソルに、LCDに文字(out_char)を表示します。
- Lcd2_Cmd(char out_char);
  LCDにコマンド(out_char)を送信します。

※mikroCコンパチブルにしましたので、詳細については、其方(HELP)を参照してください。 ※LCD制御ライブラリの動作確認にはPIC16F88を使用しましたが、他のPICでも同様に動作します。

<動作確認用のプログラムの処理>

- LCDを初期化します。
- オープニングデモを行います。
  - Lcd2_Chr関数を使用して、文字を表示します。
  - Lcd2_Chr_Cp関数を使用して、文字を表示します。
  - Lcd2_Out関数を使用して、文字を表示します。
  - Lcd2_Out_Cp関数を使用して、文字を表示します。
  - Lcd2_Cmd関数を使用して、画面をクリアします。
  - Lcd2_Chr関数を使用して、画面の各行各列に ■ を、順次表示します。
  - Lcd2_Cmd関数を使用して、画面のオン/オフを繰り返します。
  - Lcd2_Cmd関数を使用して、カーソルをホームに戻します。
  - Lcd2_Chr関数を使用して、画面の各行各列にスペースを、順次表示します。
- アナログデータ(CH1~CH4)を取り込み、値を表示します。

<LCDの接続方法>

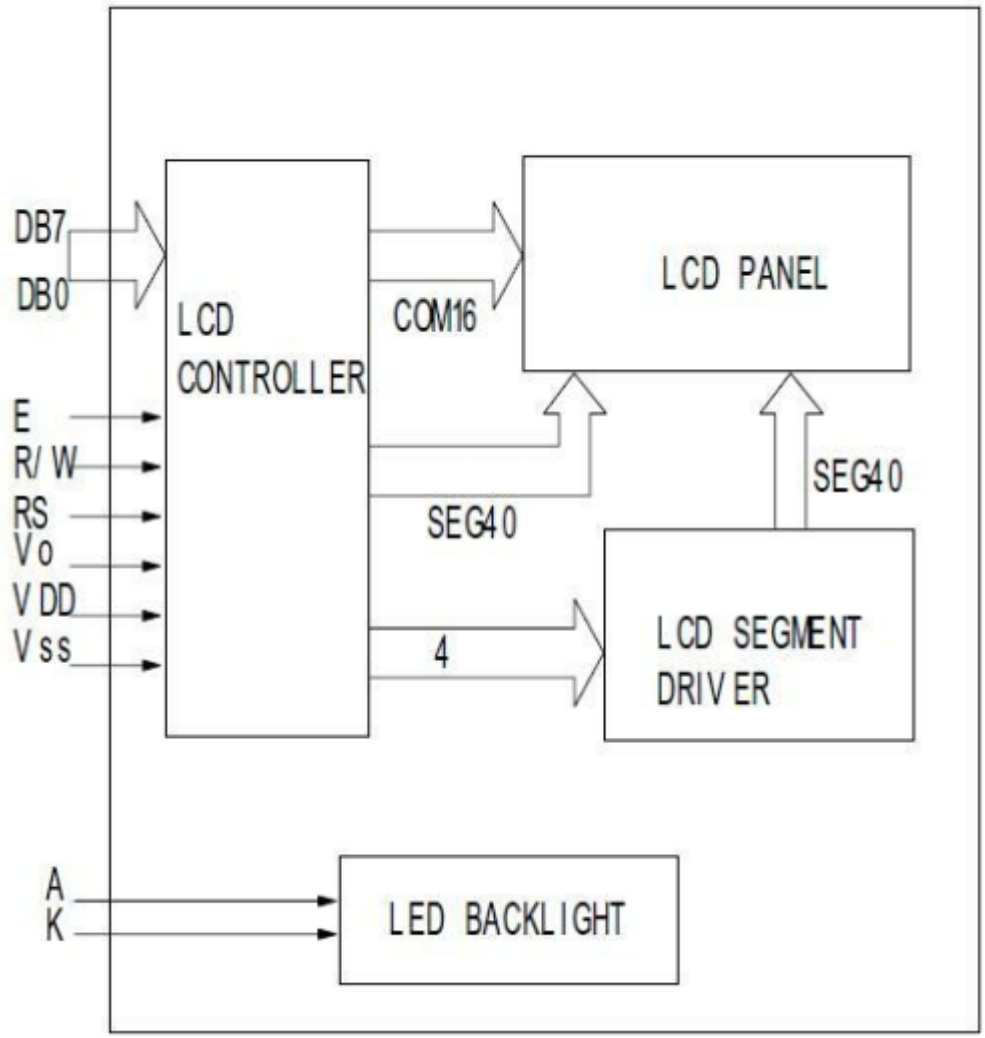


<LCDの制御命令>

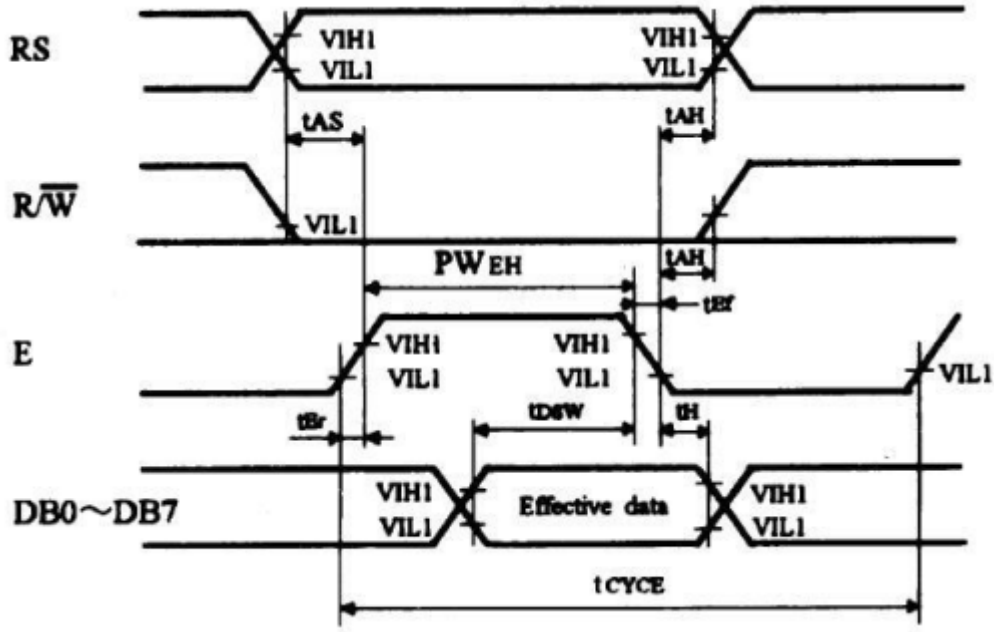
制御命令	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
表示クリア	0	0	0	0	0	0	0	0	0	1
カーソルホーム	0	0	0	0	0	0	0	0	1	*
エントリーモードセット	0	0	0	0	0	0	0	1	I/D	S
表示オン／オフコントロール	0	0	0	0	0	0	1	D	C	B
カーソル／表示シフト	0	0	0	0	0	1	S/C	R/L	*	*
ファンクションセット	0	0	0	0	1	DL	N	F	*	*
CGRAMアドレスセット	0	0	0	1	CGRAMアドレス					
DDRAMアドレスセット	0	0	1	DDRAMアドレス						
BF／アドレス読み出し	0	1	BF	アドレスカウンタ						
CGRAM/DDRAMへの書き込み	1	0	書き込みデータ							
CGRAM/DDRAMから読み出し	1	1	読み出しデータ							

- **エントリーモードセット**
  I/D:文字**コードをDDRAM/CGRAMに書き込み、読み出し時のアドレス**増減方向(1→+、0→-)
  S:S=1**のとき、**書き込み時の表示**シフトあり**(I/D=1:左**シフト**I/D=0:右**シフト**)**
- 表示**オン/オフコントロール**
  D:1**で**表示**オン、0で**表示**オフ**DDRAM**の内容は**保持。
  C:1**でカーソル**表示**オン、0でオフ。**
  B:1**でカーソル**位置**ブリンクオン、0でオフ。ブリンクは**全**ドット黒と**文字**の切り換え。**
- **カーソル**/表示**シフト**
  DDRAM**の内容を変えずにカーソル**の移動と表示**シフトを**行う。
  S/C:1**で**表示全体、0**でカーソル**位置**S/C=1ではカーソル**も一緒に移動。
  R/L:1**で右**シフト、0で**左**シフト。**
- **ファンクションセット**
  ファンクションセットは他の制御命令**より**先に実行**しなければならない。**
  DL:**インタフェイスデータ**長設定。1**で8ビット**(DB7-0使用)、0**で4ビット**(DB7-4使用)。
  N:**デューティ**の設定。1**で1/16、0で1/8または1/11。**
  F:文字**フォントの**設定。通常F=0**で**使用。

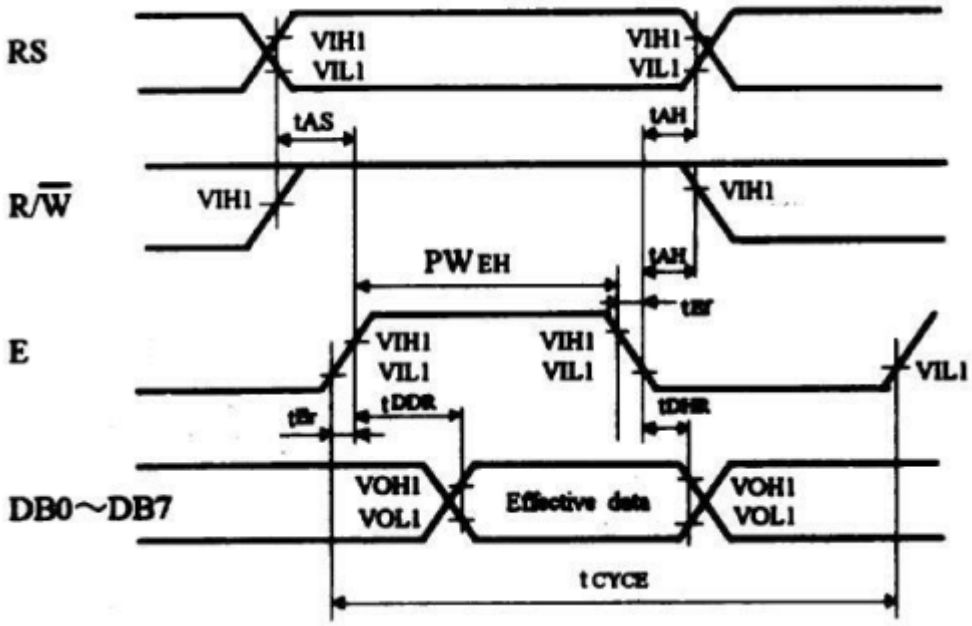
<LCD**のブロックダイアグラム**>



<LCDの書き込みタイミングチャート>



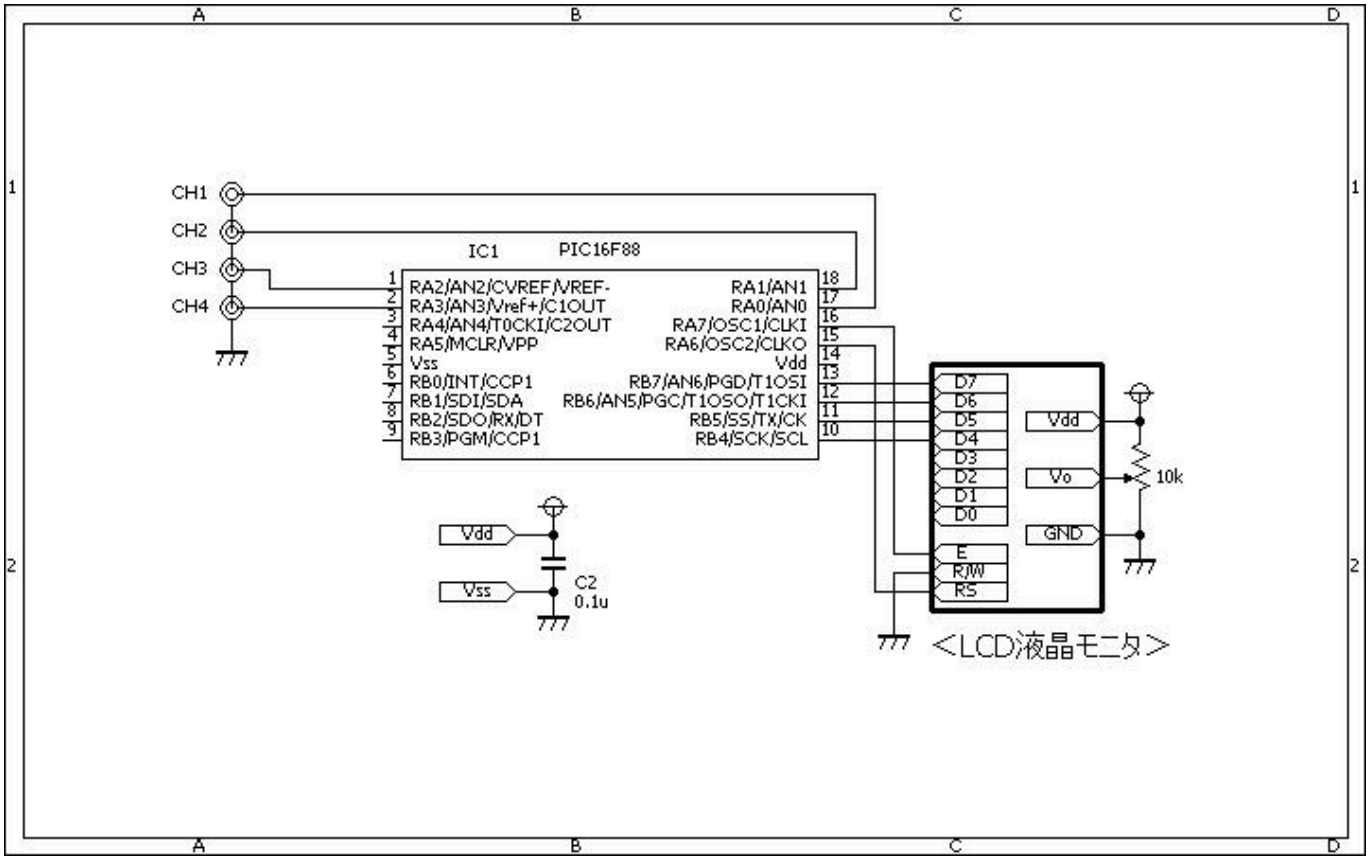
<LCDの読み込みタイミングチャート>※今回は使用しません。



<LCDのタイミング時間>

Item		Symbol	Min	Max	Unit
Enable Cycle Time		T_{CYCE}	500	--	ns
Enable Pulse Width	High Level	PW_{EH}	230	--	ns
Enable Rise/Fall Time		t_{Er}, t_{Ef}	--	20	ns
Address Set-up Time	RS, R/W to E	t_{AS}	40	--	ns
Address Hold Time		t_{AH}	10	--	ns
Data Set-up Time		t_{DSW}	80	--	ns
Data Delay Time		t_{DDR}	--	160	ns
Data Hold Time (Writing)		t_H	10	--	ns
Data Hold Time (Reading)		t_{DHR}	5	--	ns
Clock Oscillation Frequency		f_{OSC}	270 (TYP)		KHz

回路図



ソースコード

◎動作確認用のプログラム

lcd_display.c

```
//*****************************************************************************
*
/*
□□□□□□制御ライブラリ□□□□□□□コンパチブル）＞
*/
//*****************************************************************************
*
//■■■関数宣言■■■
extern  void    main();
extern  void    opening_demonstration();
//*****************************************************************************
*
//■■■インクルード■■■
#include "lcd_lib_4bit.h"
//*****************************************************************************
*
//■■■マクロ定義■■■
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 RB5_bit;
sbit LCD_D4 at RB4_bit;
// Pin direction
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 TRISB5_bit;
sbit LCD_D4_Direction at TRISB4_bit;
//********************************************************************
*
//■■■メイン関数■■■
void    main()
{
        char    buf[16];
        double  ad;
        //
        OSCCON = 0b01110000;
        ANSEL  = 0b00001111;
        TRISA  = 0b00101111;
        TRISB  = 0b00000000;
        //
        Lcd2_Init();
        Lcd2_Cmd(_LCD_CLEAR);
        Lcd2_Cmd(_LCD_CURSOR_OFF);
        //
        opening_demonstration();
        //
        Lcd2_Out(1, 6, "mV");
        Lcd2_Out(1, 14, "mV");
        Lcd2_Out(2, 6, "mV");
        Lcd2_Out(2, 14, "mV");
        while (1) {
                ad = Adc_Read(0);
                ad *= 4.8828125;
                WordToStr(ad, buf);
                Lcd2_Out(1, 1, buf);
                //
                ad = Adc_Read(1);
                ad *= 4.8828125;
                WordToStr(ad, buf);
                Lcd2_Out(1, 9, buf);
                //
                ad = Adc_Read(2);
                ad *= 4.8828125;
                WordToStr(ad, buf);
                Lcd2_Out(2, 1, buf);
                //
                ad = Adc_Read(3);
                ad *= 4.8828125;
```

```
                WordToStr(ad, buf);
                Lcd2_Out(2, 9, buf);
                //
                Delay_ms(500);
        }
}
//*****************************************************************************
*
//■■■オープニングデモ関数■■■
void    opening_demonstration()
{
        short   cnt;
        //
        Lcd2_Chr(1, 1, 'j');
        Lcd2_Chr(1, 2, 'f');
        Lcd2_Chr(1, 3, '3');
        Lcd2_Chr(1, 4, 's');
        Lcd2_Chr(1, 5, 'f');
        Lcd2_Chr(1, 6, 'b');
        Delay_ms(1000);
        //
        Lcd2_Chr_Cp(' ');
        Lcd2_Chr_Cp('J');
        Lcd2_Chr_Cp('F');
        Lcd2_Chr_Cp('3');
        Lcd2_Chr_Cp('S');
        Lcd2_Chr_Cp('F');
        Lcd2_Chr_Cp('B');
        Delay_ms(1000);
        //
        Lcd2_Out(2, 1, "JF3SFB");
        Delay_ms(1000);
        //
        Lcd2_Out_Cp(" jf3sfb");
        Delay_ms(1000);
        //
        Lcd2_Cmd(_LCD_CLEAR);
        for (cnt = 0; cnt < 16; cnt++) {
                Lcd2_Chr(1, cnt + 1, 0xFF);
                Delay_ms(100);
        }
        for (cnt = 0; cnt < 16; cnt++) {
                Lcd2_Chr(2, cnt + 1, 0xFF);
                Delay_ms(100);
        }
        Delay_ms(1000);
        //
        for (cnt = 0; cnt < 10; cnt++) {
                Lcd2_Cmd(_LCD_TURN_OFF);
                Delay_ms(100);
                Lcd2_Cmd(_LCD_TURN_ON);
```

```
                Delay_ms(100);
        }
        //
        Lcd2_Cmd(_LCD_RETURN_HOME);
        for (cnt = 0; cnt < 16; cnt++) {
                Lcd2_Chr(1, cnt + 1, ' ');
                Delay_ms(100);
        }
        for (cnt = 0; cnt < 16; cnt++) {
                Lcd2_Chr(2, cnt + 1, ' ');
                Delay_ms(100);
        }
        Delay_ms(1000);
}
//*****************************************************************************
*
```

◎LCD制御ライブラリ

lcd_lib_4bit.c

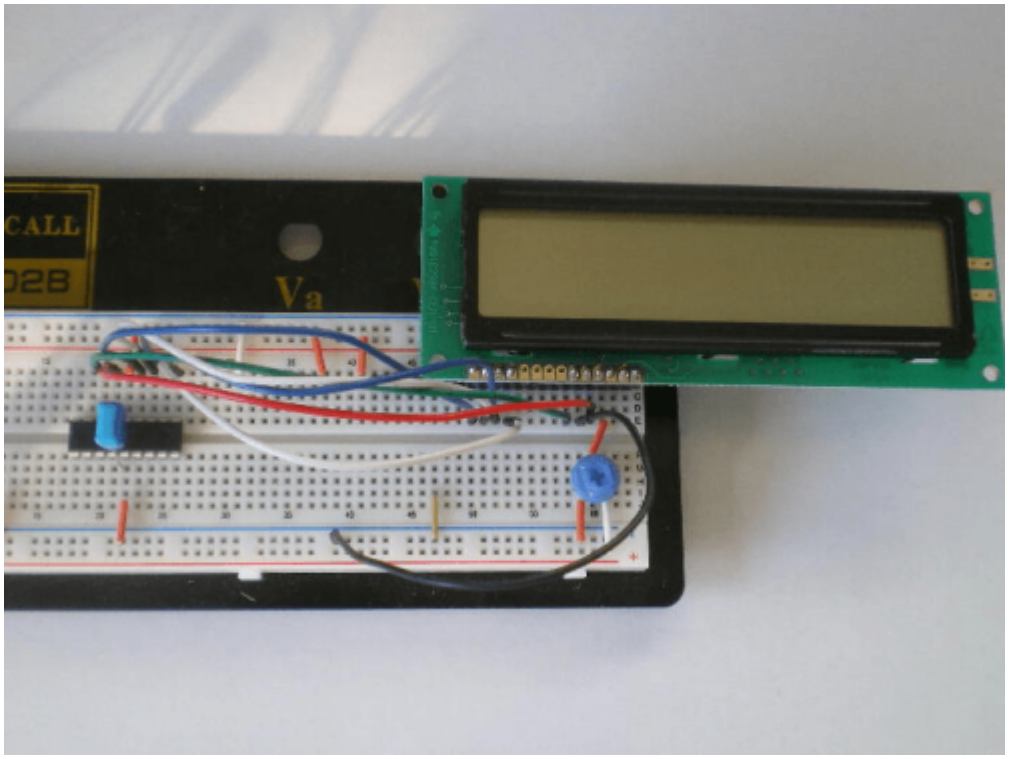
```
//*****************************************************************************
*
/*
□□□□□□制御ライブラリ□□□□□□□コンパチブル）＞
*/
//*****************************************************************************
*
#include "lcd_lib_4bit.h"
//*****************************************************************************
*
//■■■□□□文字出力（カレントカーソル位置）関数■■■
void    Lcd2_Chr_Cp(char out_char)
{
        LCD_D7 = out_char.B7;
        LCD_D6 = out_char.B6;
        LCD_D5 = out_char.B5;
        LCD_D4 = out_char.B4;
        LCD_RS = 1;
        LCD_EN = 1;
        asm         nop;
        LCD_EN = 0;
        //
        LCD_D7 = out_char.B3;
        LCD_D6 = out_char.B2;
        LCD_D5 = out_char.B1;
        LCD_D4 = out_char.B0;
        LCD_RS = 1;
        LCD_EN = 1;
        asm         nop;
```

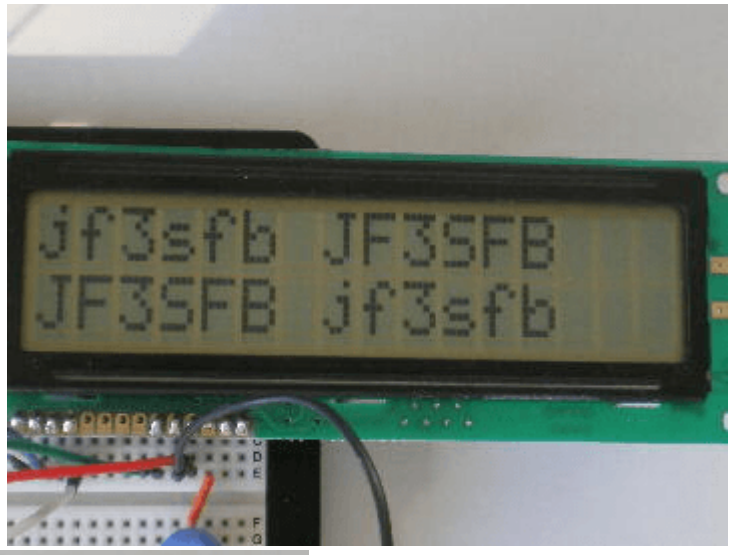
```
        LCD_EN = 0;
        //
        Delay_us(100);
}
//*********************************************************************
*
//■■■□□□文字出力（行列指定）関数■■■
void    Lcd2_Chr(char row, char column, char out_char)
{
        Lcd2_Cmd_8bit(0x80 + (column - 1) + ((row - 1) * 0x40));
        Lcd2_Chr_Cp(out_char);
}
//*********************************************************************
*
//■■■□□□文字列出力（カレントカーソル位置）関数■■■
void    Lcd2_Out_Cp(char *text)
{
        while (*text != 0x00) {
                Lcd2_Chr_Cp(*text);
                text++;
        }
}
//*********************************************************************
*
//■■■□□□文字列出力（行列指定）関数■■■
void    Lcd2_Out(char row, char column, char *text)
{
        Lcd2_Cmd_8bit(0x80 + (column - 1) + ((row - 1) * 0x40));
        Lcd2_Out_Cp(text);
}
//*********************************************************************
*
//■■■□□□コマンド出力関数■■■
void    Lcd2_Cmd(char cmd)
{
        Lcd2_Cmd_8bit(cmd);
        Delay_ms(10);
}
//*********************************************************************
*
//■■■□□□初期化関数■■■
void    Lcd2_Init()
{
        LCD_RS_Direction = 0;
        LCD_EN_Direction = 0;
        LCD_D7_Direction = 0;
        LCD_D6_Direction = 0;
        LCD_D5_Direction = 0;
        LCD_D4_Direction = 0;
        LCD_RS = 0;
```

```
        LCD_EN = 0;
        Delay_ms(20);
        Lcd2_Cmd_4bit(0b00110000);    //ファンクションセット(8bitモード)
        Delay_ms(10);
        Lcd2_Cmd_4bit(0b00110000);    //ファンクションセット(8bitモード)
        Delay_ms(10);
        Lcd2_Cmd_4bit(0b00110000);    //ファンクションセット(8bitモード)
        Delay_ms(10);
        Lcd2_Cmd_4bit(0b00100000);    //ファンクションセット(4bitモード)
        Delay_ms(10);
        Lcd2_Cmd_8bit(0b00101000);    //ファンクションセット(4bitモード,1/16
デューティ,5×7ドット)
        Delay_ms(10);
        Lcd2_Cmd_8bit(0b00001000);    //表示オフ
        Delay_ms(10);
        Lcd2_Cmd_8bit(0b00000001);    //表示クリア
        Delay_ms(10);
        Lcd2_Cmd_8bit(0b00000110);    //エントリーモードセット
        Delay_ms(10);
}
//*********************************************************************
*
//■■■□□□初期化用コマンド出力(4bit)関数■■■
void    Lcd2_Cmd_4bit(char cmd)
{
        LCD_D7 = cmd.B7;
        LCD_D6 = cmd.B6;
        LCD_D5 = cmd.B5;
        LCD_D4 = cmd.B4;
        LCD_RS = 0;
        LCD_EN = 1;
        asm        nop;
        LCD_EN = 0;
        //
        Delay_us(100);
}
//*********************************************************************
*
//■■■□□□初期化用コマンド出力(8bit)関数■■■
void    Lcd2_Cmd_8bit(char cmd)
{
        LCD_D7 = cmd.B7;
        LCD_D6 = cmd.B6;
        LCD_D5 = cmd.B5;
        LCD_D4 = cmd.B4;
        LCD_RS = 0;
        LCD_EN = 1;
        asm        nop;
        LCD_EN = 0;
        //
        LCD_D7 = cmd.B3;
```

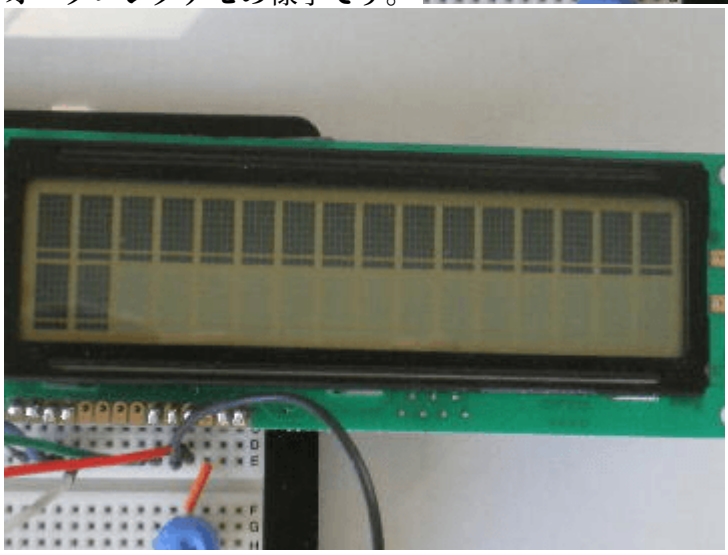
```
        LCD_D6 = cmd.B2;
        LCD_D5 = cmd.B1;
        LCD_D4 = cmd.B0;
        LCD_RS = 0;
        LCD_EN = 1;
        asm        nop;
        LCD_EN = 0;
        //
        Delay_us(100);
}
//*****************************************************************************
*
```

動作確認





オープニングデモの様子です。



4チャネル分のアナログデータを、左上:CH1□右上:CH2□左下:CH3□右下:CH4の順番に表示します。