TEL

3色12点VU/リニアLEDレベルメーター(品番JVU-0301)説明書

概要

本キットは、マイクロチップ社、8ビットマイクロコントローラ、ミッドレンジ、エンハンスシリーズPIC16F1826を使用した3色12点LEDバーグラフ(棒状)表示VUレベルメーター(品番JVU-0301完成品)です。

通常のVUレベルメーターとして、オーディオ信号などの入力信号を対数(VU)スケールで棒状表示するほか、リニア(直線)表示切替が可能で、DC直流電圧をリニアスケールで3色12点LEDバーグラフ(棒状)表示ができます。

特長および機能

1. 3色(緑、黄、赤)12点LED表示
2. VU/リニア(直線)表示切換可
3. 高速/低速サンプリング速度切換可
4. 各色LEDを直列接続し、定電流ドライブすることにより消費電流の低減化(約60mA 全LED点灯時)
5. PIC内蔵10ビットA/D変換モジュール使用
6. A/D変換値の移動平均によるソフトウェアーローパスフィルター採用
7. 利得4倍(12dB)の全波整流入力アンプ付
8. 推奨電源電圧は12V

電気的特性

1. 動作電源電圧	10V～15V(12V推奨)	
2. 動作電流	約60mA @12V、全LED点灯時	
	約20mA @12V、全LED消灯時	
3. 入力アンプゲイン	12dB(4倍)	
4. 入力電圧範囲		
AC入力	0～0.256Vrms(0～0.717Vp-p)*注1	
DC入力	0～0.256V*注1	
5. サンプリング速度	1mS/4mS(高速/低速)	
6. 表示速度	16mS/64mS(高速/低速)	
7. ADC基準電圧	1.024V	
8. 分解能	1mV/bit	
9. LED定電流源	15mA/LEDセグメント	

準備するもの

1. 12V直流電源、100mA以上のもの。弊社12V三端子電源キットJPS－0161＋12Vなど
2. 3ピンのピンソケット(メス)2個

使い方

1. VU表示レベルメーターとして使用する場合
SW1スイッチをAC IN 側にスライドする。入力にカップリング電解コンデンサー(C5)が入り、交流カップリングとなる。
コネクターCN2の3番ピンに入力信号、2番ピンにグランド(GND)を接続。信号源にはオーディオ信号などの交流信号を使う。
＜調整の仕方＞基準となる信号を入力し、半固定抵抗(TP1)を回し、所定のレベルのLEDを点灯させる。入力には利得4倍(12dB)の全波整流入力アンプが入っているので、本機の最大入力電圧(*注1)は0.256Vrms(0.717Vp-p)となる。このレベルを超える入力は、半固定抵抗(TP1)を左回し、最大入力レベルを超えないようにすること。CPU入力電圧がVdd(5V)を超えると故障の原因となる。
12点LED、デシベル(dB)及び10ビットA/D換算値の関係を表1に示す。

2. リニア(直線)表示レベルメーターとして使用する場合
SW1スイッチをDC IN 側にスライドする。入力バイポーラ電解コンデンサー(C5)がショートされ、直流カップリングとなる。
コネクターCN2の3番ピンに入力信号、2番ピンにグランド(GND)を接続。信号源は、DC直流電圧で、入力電圧をリニアスケールでLEDバーグラフ表示できる。
＜調整の仕方＞基準となる電圧用意し、半固定抵抗(TP1)を回し、所定のレベルのLEDを点灯させる。入力には利得4倍(12dB)の全波整流入力アンプが入っているので、本機の最大入力電圧(*注1)は256mV(0.256V)となる。このレベルを超える入力は、半固定抵抗(TP1)を左回し、最大入力 レベルを超えないようにすること。
CPU入力電圧がVdd(5V)を超えると故障の原因となる。
12点LEDと10ビットA/D換算値の関係を表2に示す。

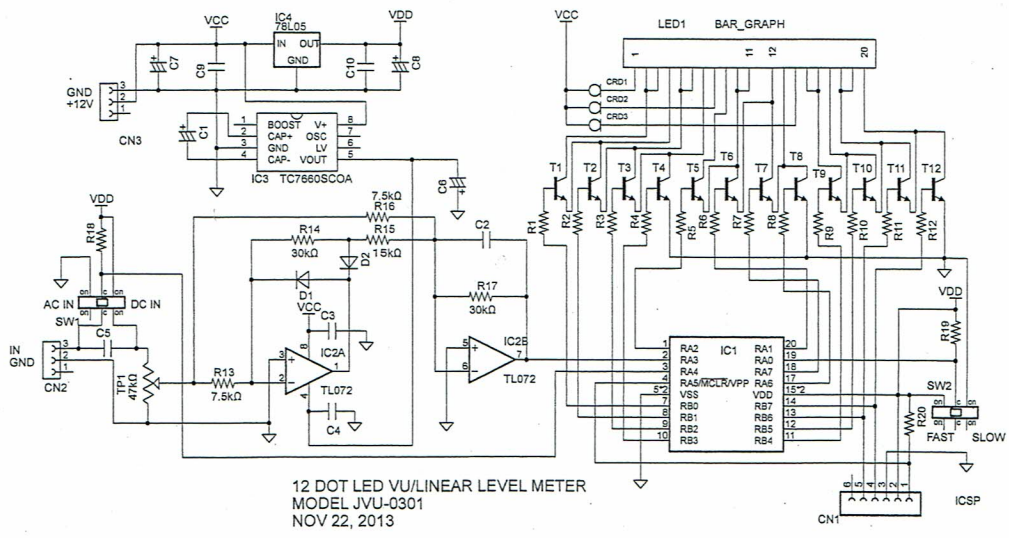
3. 高速/低速サンプリング速度切換
音声、音楽のように振幅変化の早い入力信号に対しては、速度切換を高速(FAST)にすると表示速度が速くなり、より正確な表示が可能となる。

プログラムについて

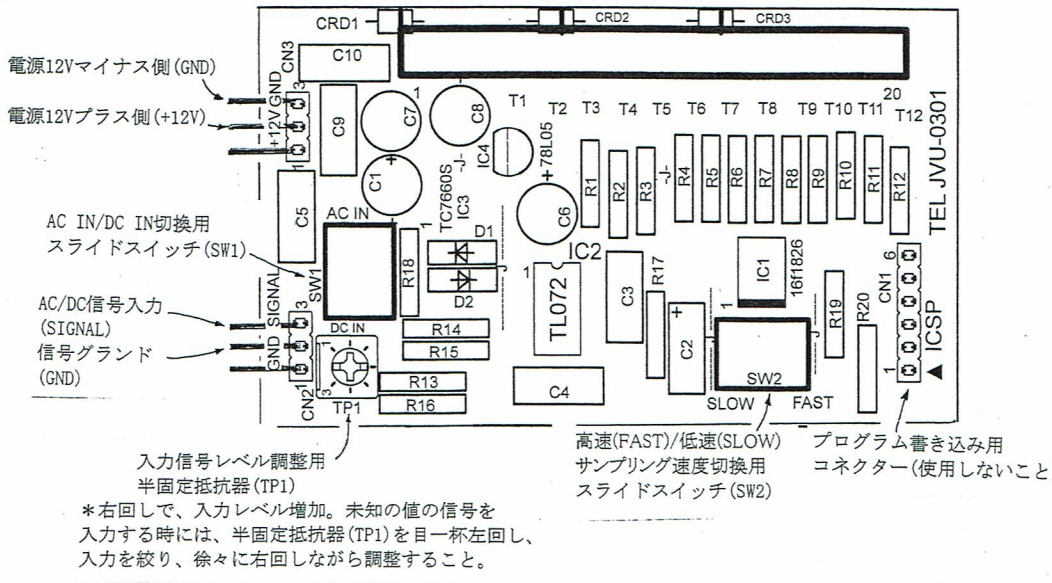
MPLAB XC8 C コンパイラー ライトモードV1.21を使用。
ROM(プログラムメモリ) 2048ワード中533使用 約26%
RAM(データーメモリ) 884バイト中47バイト使用 約5%
参考資料としてCソースファイルのリスト(Listing)を添付する。
＜プログラムの特記事項＞
サンプリング周期にウオッチドックタイマー(WDT)を使う。
A/Dサンプル、変換時以外はCPU及びその周辺モジュールがスリープ状態となっている。 また、16個のA/D変換値の移動平均を使い、プログラムによるローパスフィルターを採用した。
＜注意＞CONFIGワードのコードプロテクションビット はOFFになっており、ICプログラムメモリの上書き、読込みが自由にできますが、ICSPを使用しプログラムを変更した場合には、保証できなくなりますから注意してください。また、プログラムに関するご質問はお受けできませんのでご了承ください。

注1:全LEDを点灯させるために必要な最低入力電圧範囲のこと。
入力電圧範囲の可変は、半固定抵抗(TP1)で調整する。
本機の表示は、入力電圧に対し相対的に点灯するので、入力電圧範囲は半固定抵抗(TP1)を使い自由に決めることができる。例えば、基準電圧として6Vを入力し、半固定抵抗(TP1)を左回し、LED12(全LED)を点灯させた時、各LEDあたりの電圧は0.5Vとなる。

3色12点VU/リニアLEDレベルメーター
(品番JVU-0301)回路図



3色12点VU/リニアLEDレベルメーター
(品番JVU-0301)組立図



＊右回しで、入力レベル増加。未知の値の信号を入力する時には、半固定抵抗器(TP1)を目一杯左回し、入力を絞り、徐々に右回しながら調整すること。

表1;12点LED、デシベル(dB)及び10ビットA/D換算値の関係(VU表示)

10ビットA/D換算値＝$408x10^{Gv/20}$

点灯LED番号	0	1	2	3	4	5	6	7	8	9	10	11	12
電圧ゲインGv(dB)	-20	-15	-10	-7	-5	-3	-1	0	+1	+3	+5	+8	
10ビットA/D換算値	＜20	40	72	129	182	229	288	363	408	457	576	725	1024

表2:12点LEDと10ビットA/D換算値の関係(リニアー表示)

点灯LED番号	0	1	2	3	4	5	6	7	8	9	10	11	12
10ビットA/D換算値	＜76	155	234	313	392	471	550	629	708	787	866	945	1024

TEL エレクトロニクス・キット （有）谷岡電子
〒164-0003 東京都中野区東中野１－５１－１３
大島ビル第一別館４０２ ☎ (03)3366-4552

参考資料

3色12点VU/リニアLEDレベルメーター

（品番JVU-0301）Cソースファイルリスト

```
/*****************************************************************
******* 12DOT 3COLOR LED VU/LINEAR LEVEL METER *******
* Project File: JVU_0301
* Source File : JVU_0301.c
* Features:
*  *PIC enhanced mid-range 16F1826
*  *12DOT 3COLOR LED BARGRAPH
*  *SINGLE POWER SUPPLY +12V
*  *HARDWARE SELECTABLE VU/LINEAR DISPLAY
*  *FAST/SLOW DISPLAY SPEED (DISPLAY TIME: 16mS/64mS)
*  *SOFTWARE LOW PASS FILTER REDUCES EFFECTIVELY FLICKERING LIGHT PROBLEM
* Pin Assignments:
*  *FAST/SLOW SWITCH INPUT <RA0>
*  *SIGNAL INPUT (ANALOG INPUT) <RA3>
*  *AC/DC SWITCH INPUT <RA4>
*  *BARGRAPH DRIVING OUTPUT <RA7,RA6> <RA2,RA1> <RB7:RB0>
* Compiler: xc8 v1.21
* Jan 4,2014: Coded by yuji Tanioka(TEL company)
*****************************************************************/
#include <xc.h>

#pragma config MCLRE = OFF, CP = OFF, BOREN = OFF, WDTE = SWDTEN,
#pragma config PWRTE = ON, CPD = OFF, CLKOUTEN = OFF, FCMEN = OFF
#pragma config WRT = OFF, STVREN = OFF, BORV = LO, LVP = OFF,
#pragma config FOSC = INTOSC, PLLEN = OFF

#define  _XTAL_FREQ     4000000  //4MHZ

typedef volatile bit Bool;                          //flags

#define         TRUE                    1
#define         FALSE                   0
#define         AC_DC_IN                RA4
#define         FAST_SLOW               RA0
#define         NSAMPLE                 16

//funcfion prototype declaration
void reg_init(void);                                //F13
void linear_dpy(unsigned int);                      //F22
void vu_dpy(unsigned int);                          //F23
void seg_table(unsigned char);                      //F24

//global variables
Bool   sleep_flag = FALSE;

    /*Main function*/
void main(void)
{
    static unsigned int sam_buf[NSAMPLE];
    static unsigned char pos = 0;
    static unsigned int adc_sum = 0;
    static unsigned int adc_ave;

    reg_init();                                     //F13

    for(;;)
    {

        if(FAST_SLOW)
            WDTCONbits.WDTPS = 0b00000;             //FAST INTERVAL 1mS
        else
```



```
            WDTCONbits.WDTPS = 0b00010;             //SLOW INTERVAL 4mS

        if(sleep_flag)
        {
            SWDTEN = 1;                 //WATCHDOG TIMER ENABLED IN SLEEP
            sleep_flag = FALSE;

            SLEEP();
        }

        SWDTEN = 0;

        ADON = 1;                                   //START A SAMPLING
        __delay(10);                                //acquisition time(10uS)
        GO_nDONE = 1;                               //START AN AD CONVERSION
        while(GO_nDONE);                            //WAIT UNTIL ADC COMPLETES

        adc_sum -= sam_buf[pos];                    //SUBTRACT OLD SAMPLE FROM TOTAL
        sam_buf[pos] = (unsigned int)ADRESH << 8  |  ADRESL;
                                                    //SAVE NEW SAMPLE
        adc_sum += sam_buf[pos];                    //ADD IT TO RUNNING TOTAL

        if(++pos == NSAMPLE)                        //DISPLAY INTERVAL 16mS/64mS
        {
            pos = 0;
            adc_ave = adc_sum >> 4;                 //DIVIDED BY 16
            if(AC_DC_IN)
                linear_dpy(adc_ave);                //LINEAR display, F22
            else
                vu_dpy(adc_ave);                    //VU display, F23
        }
        sleep_flag = TRUE;
    }
}

/*function definition F13, register initialization*/
void reg_init(void)
{
    OSCCON = 0b01101000;            //4Mhz,HF:IRCF=01101
    OSCTUNE = 0x00;
    TRISA = 0b00011001;             //<RA4:RA3:RA0>INPUT
    TRISB = 0b00000000;             //<RB7:RB0> OUTPUT
    ANSELA =0b00001000;             //<RA3> ANALOG
    ANSELB = 0x00;                  //DIGITAL I/O
    WPUA5 = 1;                      //WAKE PULL-UP RA5
    WPUB = 0x00;                    //WAKE PULL-UP DISABLED
    LATA7 = 1; LATA6 = 0;           //PORTA ALL LED OFF
    LATA2 = 1; LATA1 = 1;           //PORTA ALL LED OFF
    LATB = 0b01110111;              //PORTB ALL LED OFF

    //configure ADC
    ADCON0 = 0b00001100;
            //-00011--              Analog ch select CHS=00011 AN3
            //-------0              ADON=0 ADC disabled

    ADCON1 = 0b10010011;
            //1-------              ADFM=1, Right justified
            //-001----              Fosc/8, ADCS=001, Tad=2.0 us
            //-----0--              Vref- is connected to Vss
            //------11             ADPREF=11, Vref+ is connected to internal
            //                      FVR
```



```

    //configure FVR(fix voltage reference control register)
    FVRCON = 0b11000001;
            //1-------             FVREN=1, enabled
            //-1------             always '1' with LDO (16F1826)
            //----00--             off
            //------01             ADFVR=01, 1.024V
}

/*function definition F22, LINEAR SCALE DISPLAY*/
void linear_dpy(unsigned int val)                   //F22
{
    unsigned char led_seg;

    if(val < 76) led_seg = 0;
    else if(val < 155) led_seg = 1;
    else if(val < 234) led_seg = 2;
    else if(val < 313) led_seg = 3;
    else if(val < 392) led_seg = 4;
    else if(val < 471) led_seg = 5;
    else if(val < 550) led_seg = 6;
    else if(val < 629) led_seg = 7;
    else if(val < 708) led_seg = 8;
    else if(val < 787) led_seg = 9;
    else if(val < 866) led_seg = 10;
    else if(val < 945) led_seg = 11;
    else if(val <= 1024) led_seg = 12;

    seg_table(led_seg);                             //F24

}

/*function definition F23, LOG/UV SCALE DISPLAY*/
void vu_dpy(unsigned int val)                       //F23
{
    unsigned char led_seg;

    if(val < 20) led_seg = 0;
    else if(val < 40) led_seg = 1;
    else if(val < 72) led_seg = 2;
    else if(val < 129) led_seg = 3;
    else if(val < 182) led_seg = 4;
    else if(val < 229) led_seg = 5;
    else if(val < 288) led_seg = 6;
    else if(val < 363) led_seg = 7;
    else if(val < 408) led_seg = 8;
    else if(val < 457) led_seg = 9;
    else if(val < 576) led_seg = 10;
    else if(val < 725) led_seg = 11;
    else if(val <= 1024) led_seg =12;

    seg_table(led_seg);                             //F24
}

/*function definition F24, BARGRAPH SEGMENT TABLE*/
void seg_table(unsigned char seg_num)               //F24
{
    switch(seg_num)
    {
        case 0:
            LATA7 = 1; LATA6 = 0; LATA2 = 1; LATA1 = 1;
```



```
            LATB = 0b01110111; break;
        case 1:
            LATA7 = 1; LATA6 = 0; LATA2 = 1; LATA1 = 1;
            LATB = 0b01111111; break;
        case 2:
            LATA7 = 1; LATA6 = 0; LATA2 = 1; LATA1 = 1;
            LATB = 0b01111110; break;
        case 3:
            LATA7 = 1; LATA6 = 0; LATA2 = 1; LBTA1 = 1;
            LATB = 0b01111100; break;
        case 4:
            LATA7 = 1; LATA6 = 0; LATA2 = 1; LATA1 = 1;
            LATB = 0b01111000; break;
        case 5:
            LATA7 = 1; LATA6 = 1; LATA2 = 1; LATA1 = 1;
            LATB = 0b01111000; break;
        case 6:
            LATA7 = 1; LATA6 = 1; LATA2 = 0; LATA1 = 1;
            LATB = 0b01111000; break;
        case 7:
            LATA7 = 1; LATA6 = 1; LATA2 = 0; LATA1 = 0;

            LATB = 0b01111000; break;
        case 8:
            LATA7 = 0; LATA6 = 1; LATA2 = 0; LATA1 = 0;
            LATB = 0b01111000; break;
        case 9:
            LATA7 = 0; LATA6 = 1; LATA2 = 0; LATA1 = 0;
            LATB = 0b11111000; break;
        case 10:
            LATA7 = 0; LATA6 = 1; LATA2 = 0; LATA1 = 0;
            LATB = 0b11101000; break;
        case 11:
            LATA7 = 0; LATA6 = 1; LATA2 = 0; LATA1 = 0;
            LATB = 0b11001000; break;
        case 12:
            LATA7 = 0; LATA6 = 1; LATA2 = 0; LATA1 = 0;
            LATB = 0b10001000; break;
        default: break;
    }
}
```