周波数カウンタV5(集大成版)

概要

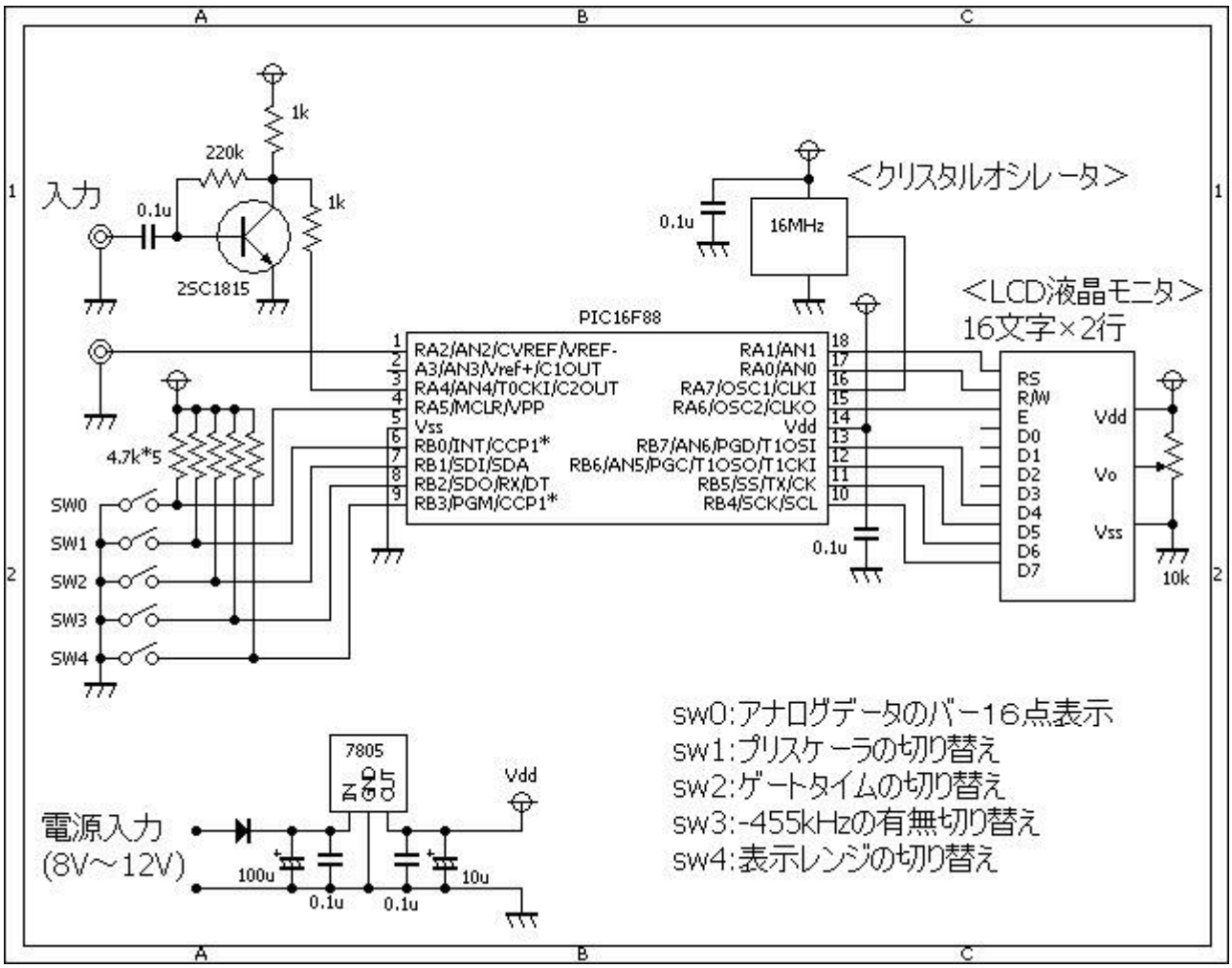
今迄に多数の周波数カウンタを製作してきましたが、機能や構造の見直しを含めここに集大成してみました。

動作原理

特に拘った箇所は、基準時間(1秒、0.1秒)を発生させる仕組みです。 独自に発案した方式で、PIC内臓の「TIMER1モジュールのみを使用します。これにより、プログラムの構造をシンプルにすることが出来ます。 <仕様>

- アナログデータ(0~5V)のバー16点表示
  - sw0=1無し
  - sw0=0有り
- プリスケーラの切り替え
  - sw1=11/1
  - sw1=01/8
- ゲートタイムの切り替え
  - sw2=11秒
  - sw2=00.1秒
- -455kHzの有無切り替え
  - sw3=1-0kHz
  - sw3=0-455kHz
- 表示レンジの切り替え
  - sw4=1Hz表示
  - sw4=0kHz表示

回路図



ソースコード

FreqCounterV5.c

```
//*****************************************************************************
*
/*
  <周波数カウンター>

□□sw0:アナログデータのバー１６点表示
□□□□□sw0=1□無し
□□□□□sw0=0□有り

□□sw1:プリスケーラの切り替え
□□□□□sw1=1□1/1
□□□□□sw1=0□1/8

□□sw2:ゲートタイムの切り替え
□□□□□sw2=1□1秒
□□□□□sw2=0□0.1秒

□□sw3:-455kHzの有無切り替え
```

```
□□□□・sw3=1□-0kHz
□□□□・sw3=0□-455kHz

□□sw4:表示レンジの切り替え
□□□□・sw4=1□Hz表示
□□□□・sw4=0□kHz表示
*/
//*****************************************************************************
*
/*
<PICのピンアサイン>
Pin-01□アナログデータ入力
Pin-02□未使用
Pin-03□信号入力
Pin-04□アナログデータの表示切替SW
Pin-05□Vss(GND)
PIN-06□プリスケーラの切替SW
Pin-07□ゲートタイムの切替SW
Pin-08□-455kHzの切替SW
Pin-09□表示レンジの切替SW
Pin-10□LCD:D7
Pin-11□LCD:D6
Pin-12□LCD:D5
Pin-13□LCD:D4
Pin-14□Vdd(+5V)
Pin-15□LCD:EN
Pin-16□クロック入力(16MHz入力)
Pin-17□LCD:WR
Pin-18□LCD:RS
*/
//*****************************************************************************
*

#define       sw0        PORTA.F5
#define       sw1        PORTB.F0
#define       sw2        PORTB.F1
#define       sw3        PORTB.F2
#define       sw4        PORTB.F3

#define       GATETIME_100MSEC    10
#define       GATETIME_1SEC            1

//*****************************************************************************
*

static short    MeasurementCnt;

void   interrupt()
{
    if (PIR1.TMR1IF == 1) {
        PIR1.TMR1IF = 0;
```

```
        //
        MeasurementCnt--;
        if (MeasurementCnt == 0) {
            TRISA.F4 = 0;         //ゲートを閉める。
            PORTA.F4 = 0;
            T1CON.TMR1ON = 0;     // TIMER1を停止する。
        }
    }
}

//*****************************************************************************
*

unsigned   long    FreqMeasurement(unsigned short gateTime)
{
    unsigned    long    freq;
    //
    TRISA.F4 = 0;         //ゲートを閉める。
    PORTA.F4 = 0;
    // TIMER0の設定
    INTCON.T0IF = 0;
    TMR0 = 0;
    // TIMER1の設定
    PIR1.TMR1IF = 0;
    switch (gateTime) {
    case GATETIME_1SEC:
        MeasurementCnt = 8;
        TMR1L = 0xE0;         // 500000=(1/16000000) * 4 * 8
        TMR1H = 0x5E;         // 0x5EE0=65536-(500000-(65536*7))
        break;
    case GATETIME_100MSEC:
        MeasurementCnt = 1;
        TMR1L = 0xB0;         // 50000=(0.11/16000000) * 4 * 8
        TMR1H = 0x3C;         // 0x3CB0=65536-50000
        break;
    }
    //
    freq = 0;
    //割り込みを許可する。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //開始
    T1CON.TMR1ON = 1;
    //
    Delay_Cyc(2);
    asm    nop
    asm    nop
    asm    nop
    asm    nop
    asm    nop
```

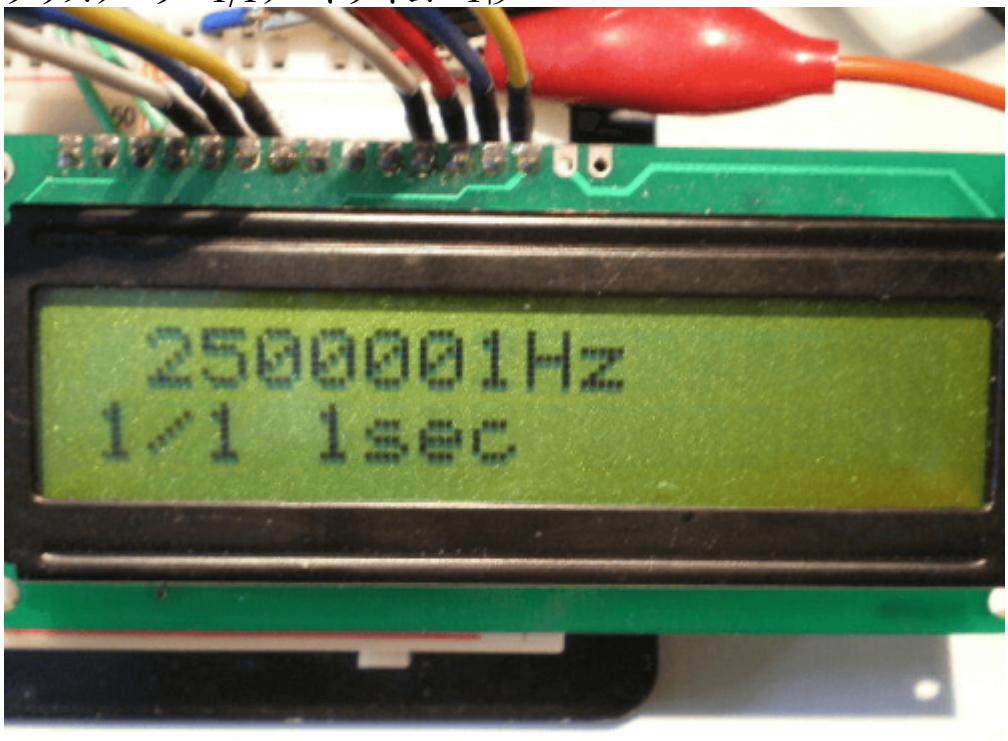
```
    asm    nop
    asm    nop
    asm    nop
    //
    TRISA.F4 = 1;         //ゲートを開ける。
    //測定
    while (T1CON.TMR1ON != 0) {
        if (INTCON.T0IF == 1) {
            INTCON.T0IF = 0;
            freq++;
        }
    }
    if (INTCON.T0IF == 1) {
        INTCON.T0IF = 0;
        freq++;
    }
    freq = (freq * 256) + TMR0;
    return (freq);
}

//*****************************************************************************
*

void main()
{
    static    char*                         msg;
    static    unsigned    long    freq, temp;    // 0...4294967295
    static    unsigned    char    buf[20], cnt, prescaler, gateTime;
    static    unsigned    int        ad;
    //アナログの設定
    ANSEL  = 0b00000100;    // ad2を使用する。
    //ポートの設定
    TRISA  = 0b10111100;
    TRISB  = 0b00001111;
    // TIMER0の設定
    INTCON = 0b00000000;
    OPTION_REG = 0b01101000;
    // TIMER1の設定
    PIE1.TMR1IE = 1;
    PIR1.TMR1IF = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1CKPS1 = 1;
    T1CON.TMR1ON = 0;
    //
    prescaler = 1;
    gateTime = GATETIME_1SEC;
    //
    Lcd_Custom_Config(&PORTB,4,5,6,7,&PORTA,1,0,6);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
```

```
    while (1) {
        freq = FreqMeasurement(gateTime);
        //補正
        freq *= prescaler * gateTime;
        //アナログデータのバー表示
        if (sw0 == 0) {
            ad = Adc_Read(2) / 60;
            for (cnt = 0; cnt < ad; cnt++)
                buf[cnt] = 0xFF;
            for (; cnt < 16; cnt++)
                buf[cnt] = ' ';
            buf[cnt] = 0x00;
            Lcd_Custom_Out(2, 1, buf);
        }
        //プリスケーラの切り替え
        if (sw1 == 1) {
            OPTION_REG.PSA  = 1;
            OPTION_REG.PS1  = 0;
            prescaler = 1;
            msg = "1/1 ";
        } else {
            OPTION_REG.PSA  = 0;
            OPTION_REG.PS1  = 1;
            prescaler = 8;
            msg = "1/8 ";
        }
        if (sw0 == 1)
            Lcd_Custom_Out(2, 1, msg);
        //ゲートタイムの切り替え
        if (sw2 == 1) {
            gateTime = GATETIME_1SEC;
            msg = "1sec   ";
        } else {
            gateTime = GATETIME_100MSEC;
            msg = "0.1sec ";
        }
        if (sw0 == 1)
            Lcd_Custom_Out(2, 5, msg);
        // □□□□□Hzの有無
        if (sw3 == 0) {
            freq -= 455000;
            msg = "-455k";
        } else {
            msg = "     ";
        }
        if (sw0 == 1)
            Lcd_Custom_Out(2, 12, msg);
        //表示レンジの切り替え
        if (sw4 == 1) {
            LongToStr(freq, buf);
```

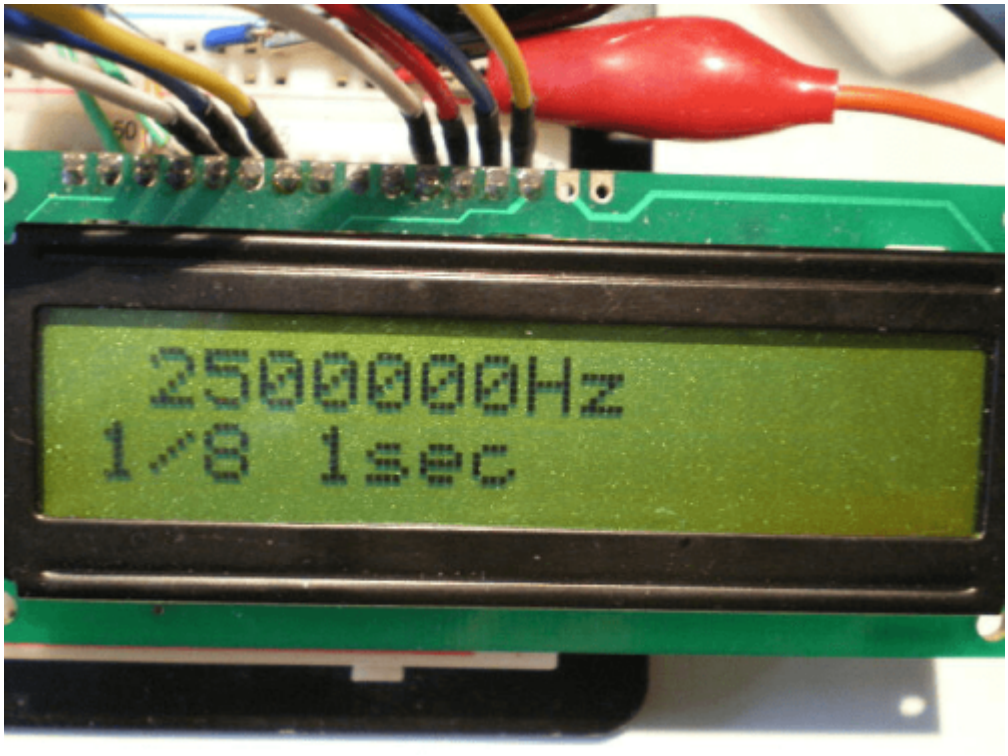
```
            msg = "Hz ";
        } else {
            temp = freq / 1000;
            if ((freq - (temp * 1000)) > 500) {
                temp++;
            }
            LongToStr(temp, buf);
            msg = "kHz";
        }
        Lcd_Custom_Out(1, 9, msg);
        //周波数の表示
        Lcd_Custom_Out(1, 1, &buf[3]);
    }
}

//*****************************************************************************
*
```

動作確認

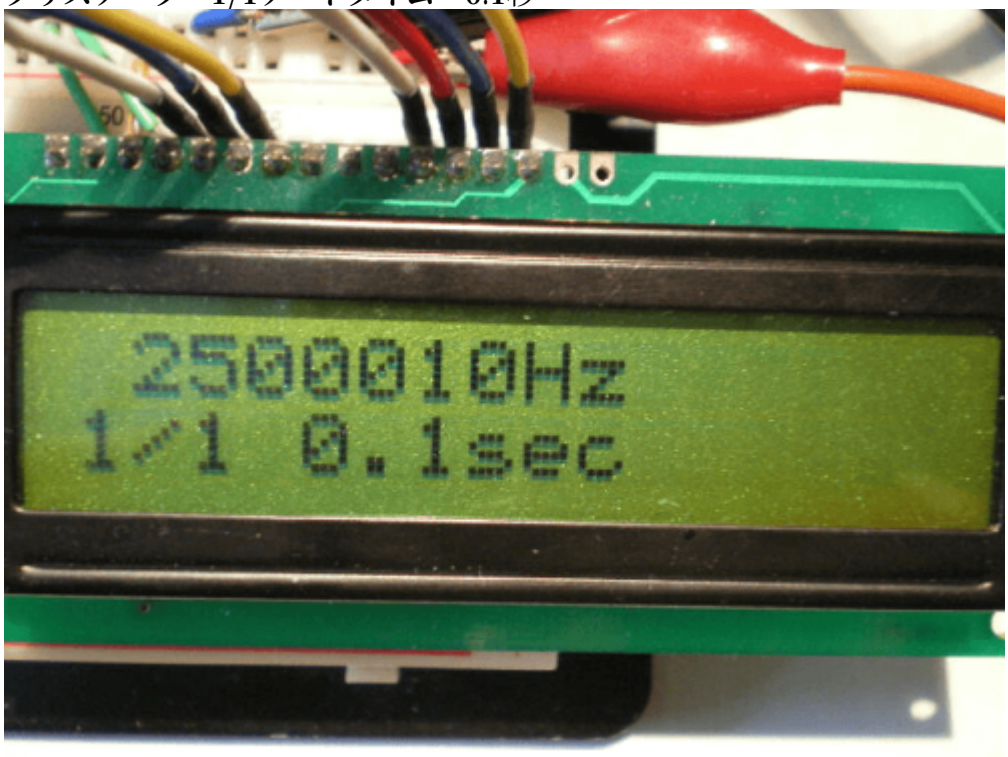
プリスケーラ→1/1ゲートタイム→1秒



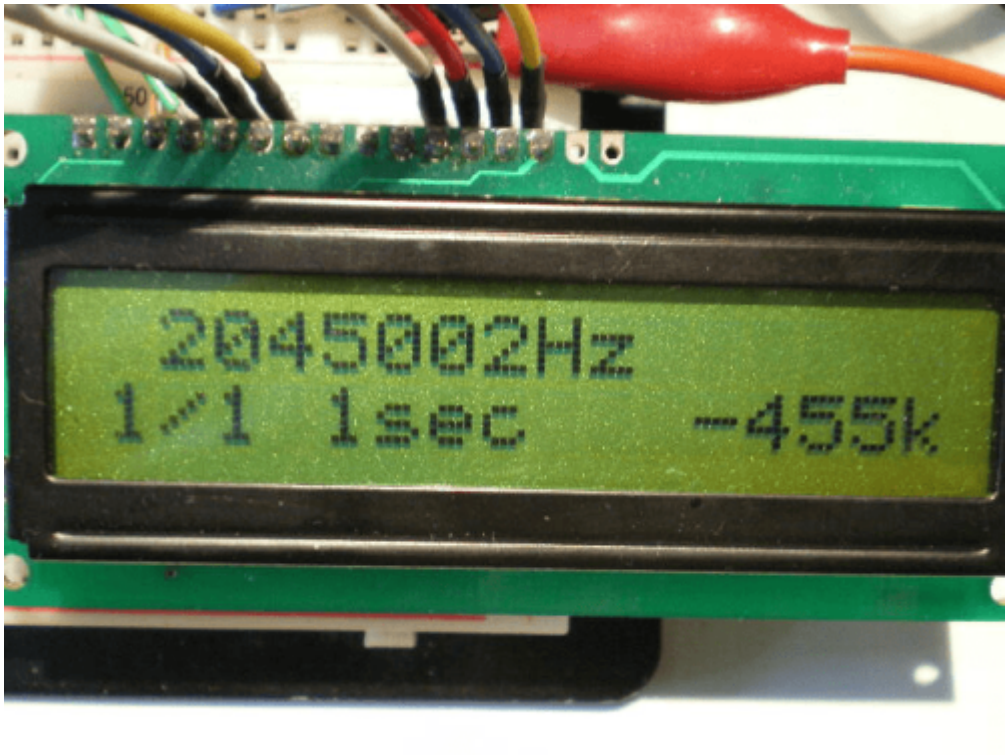
プリスケーラ→1/8ゲートタイム→1秒



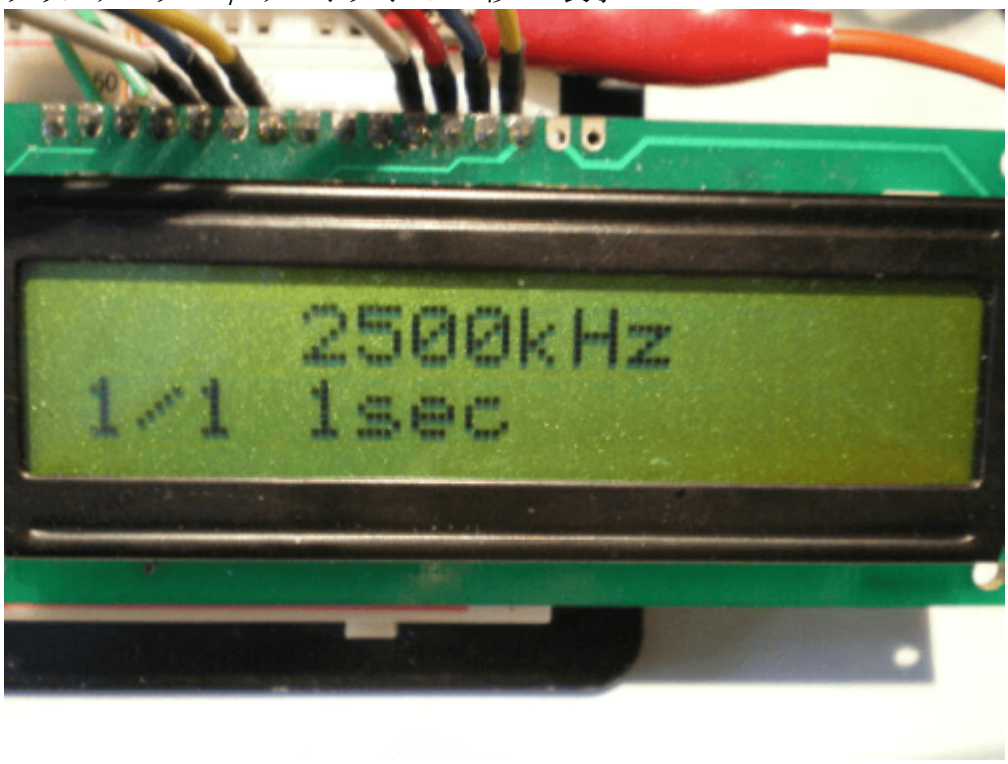
プリスケーラ→1/1ゲートタイム→0.1秒

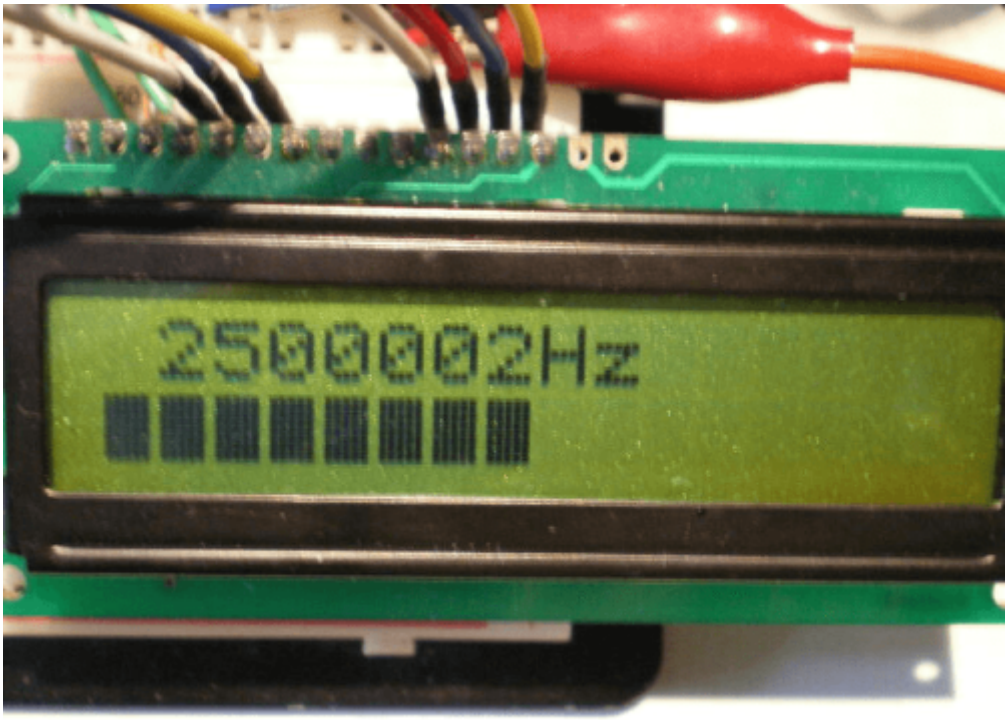


プリスケーラ→1/1ゲートタイム→1秒-455kHz



プリスケーラ→1/1ゲートタイム→1秒kHz表示





アナログデータのバー表示