周期&周波数測定ユニットV2(1Hz~100Hz)

概要

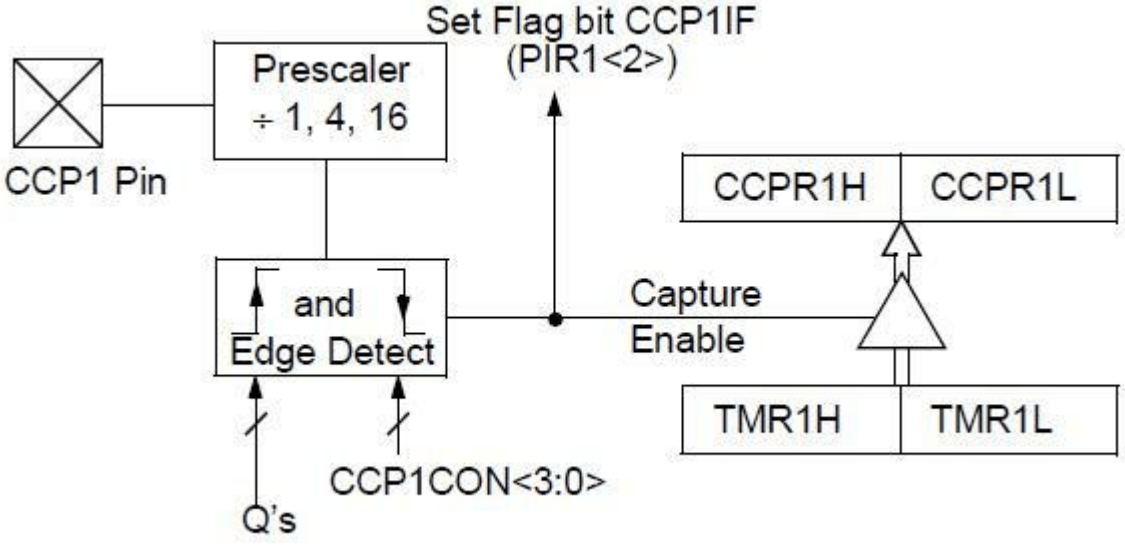
周波数をカウントするには、大きく二つの方式があります。

- 直接計数方式
  一定の時間に何回のパルスが入力されたかをカウントする方式です。(一般的)
- レシプロカル方式
  被測定パルスの立ち上がりから、次の立ち上がりまでの時間を計測パルスでカウントし、このカウント値を「時間→周波数変換」の式で計算する方式です。

直接計数方式では低い周期の測定の精度を高めるには時間がかかりますが、レシプロカル方式ですと、最低1周期分の測定で結果が得られます。

動作原理

PICのキャプチャ機能は、CCP1ピンの信号をトリガにして、その瞬間のタイマ1の値を記憶するものです。 これを利用して、波形の周期を測定します CCP1ピンの入力のエッジがプリスケーラ指定回数入力されたことをトリガにして、16ビットカウンタのタイマ1の内容をコンパレータレジスタであるCCPRに取り込んで記憶します。 それと同時に割り込み信号CCP1IFをセットします。 この時のCCPRの値より周波数へ換算します。



クロックが2MHzの時の分解能と最小測定周波数は、次のようになります。

- タイマ1のプリスケール値=1 2.0usec 7.6295Hz~
- タイマ1のプリスケール値=2 4.0usec 3.8148Hz~
- タイマ1のプリスケール値=4 8.0usec 1.9074Hz~
- タイマ1のプリスケール値=8 16.0usec 0.9537Hz~

これより、各周波数における測定精度(有効桁数)は、次のようになります。

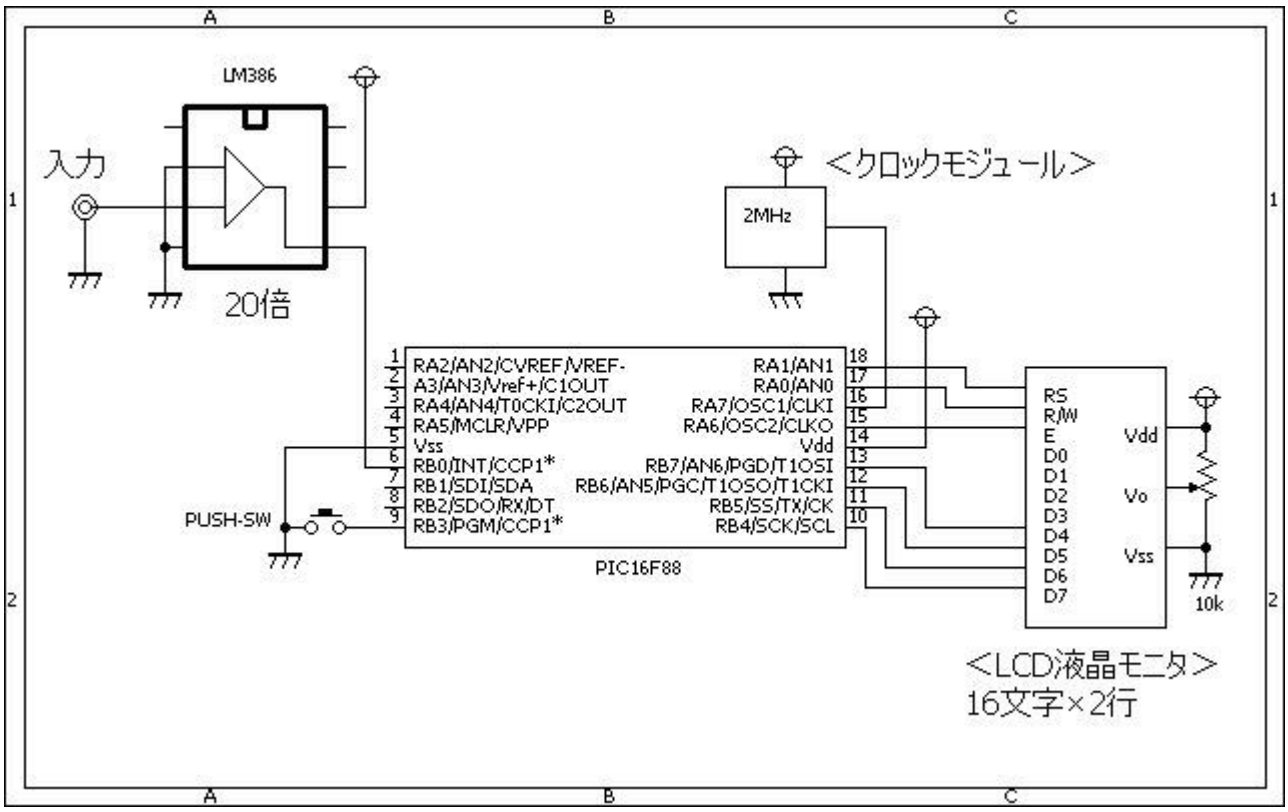
- 1Hzの時の精度→【小数点4桁】タイマ1のプリスケール値=8

1.00000=1÷((62500× 16.0)÷1000000)
0.99998=1÷((62501× 16.0)÷1000000)
0.99997=1÷((62502× 16.0)÷1000000)

- 10Hzの時の精度→【小数点3桁】**タイマ1のプリスケール**値=1
10.0000=1÷((50000× 2.0)÷1000000)
9.9998=1÷((50001× 2.0)÷1000000)
9.9996=1÷((50002× 2.0)÷1000000)
- 50Hzの時の精度→【小数点2桁】**タイマ1のプリスケール**値=1
50.0000=1÷((10000× 2.0)÷1000000)
49.9950=1÷((10001× 2.0)÷1000000)
49.9900=1÷((10002× 2.0)÷1000000)
- 100Hzの時の精度→【小数点1桁】**タイマ1のプリスケール**値=1
100.0000=1÷((5000× 2.0)÷1000000)
99.9800=1÷((5001× 2.0)÷1000000)
99.9600=1÷((5002× 2.0)÷1000000)

プログラム上で少し工夫**をしてみました。タイマ1のオーバーフローを**検出し、**プリスケール**値を**自動**的に最適な値に設定します。例えば□3Hzを測定する場合に、**プリスケール**値が、1および2ではタイマー**がオーバーフローしてしまいますが、これを**自動的に**プリスケール**値を4に**設定します。

回路図



ソースコード

CycleMeasure.c

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

#define       CLOCK2MHZ

#ifdef        CLOCK2MHZ        // min1Hz
#define       PS1        2.00    // 2.00 = (1 / 2000000Hz) * 4 * 1 *
1000000 -> ≒8Hz□
#define       PS2        4.00    // 4.00 = (1 / 2000000Hz) * 4 * 2 *
1000000 -> ≒4Hz□
#define       PS4        8.00    // 8.00 = (1 / 2000000Hz) * 4 * 4 *
1000000 -> ≒2Hz□
#define       PS8        16.00    // 16.00 = (1 / 2000000Hz) * 4 * 8 *
1000000 -> ≒1Hz□
/*
□□□の時の精度□PS9の時） 小数点第４桁
□□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□の時の精度□PS1の時） 小数点第３桁
□□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□
□□□□の時の精度□PS1の時） 小数点第２桁
□□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□の時の精度□PS1の時） 小数点第１桁
□□□□□□□□□□□□□÷□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□の時の精度□PS1の時） 小数点第０桁
□□□□□□□□□□□□□÷□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□の時の精度□PS1の時） 小数点第０桁
□□□□□□□□□□□□□÷□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□× □□□ □÷□□□□□□□□□
□□□□□□□□□□□□□÷□□□□□□× □□□ □÷□□□□□□□□□
*/
#endif

#ifdef        CLOCK16MHZ    // min8Hz
#define       PS1        0.25    // 0.25 = (1 / 16000000Hz) * 4 * 1 *
1000000 -> ≒62Hz□
#define       PS2        0.50    // 0.50 = (1 / 16000000Hz) * 4 * 2 *
1000000 -> ≒31Hz□
#define       PS4        1.00    // 1.00 = (1 / 16000000Hz) * 4 * 4 *
1000000 -> ≒16Hz□
#define       PS8        2.00    // 2.00 = (1 / 16000000Hz) * 4 * 8 *
1000000 -> ≒8Hz□
```

```
/*
最高精度（PS1の時） 小数点第３桁
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□
□□□□の時の精度（PS8の時） 小数点第３桁
□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□ □÷□□□□□□□□
□□□□□の時の精度（PS1の時） 小数点第２桁
□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□の時の精度（PS1の時） 小数点第１桁
□□□□□□□□□□□□÷□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□の時の精度（PS1の時） 小数点第０桁
□□□□□□□□□□□□÷□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□× □□□□ □÷□□□□□□□□□
□□□□□□□□□□□□÷□□□□□□□× □□□□ □÷□□□□□□□□□
*/
#endif

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

unsigned   int    measurement()
{
    unsigned    int        dat;
    //
    T1CON.TMR1ON = 0;                           //タイマーオフ
    TMR1H = 0;                                      //タイマー値クリア
    TMR1L = 0;                                      //タイマー値クリア
    PIR1.CCP1IF = 0;                           //キャプチャフラグクリア
    asm {
    loop_001:                                           //キャプチャフラグ確認
(アセンブラで高速処理)
        btfss        PIR1, 2
        goto        loop_001
    }
//    while (PIR1.CCP1IF == 0)    // キャプチャフラグ確認
//        ;
    T1CON.TMR1ON = 1;                           //タイマーオン
    PIR1.CCP1IF = 0;                           //キャプチャフラグクリア
    asm {
    loop_002:                                           //キャプチャフラグ確認
(アセンブラで高速処理)
        btfss        PIR1, 2
        goto        loop_002
    }
```

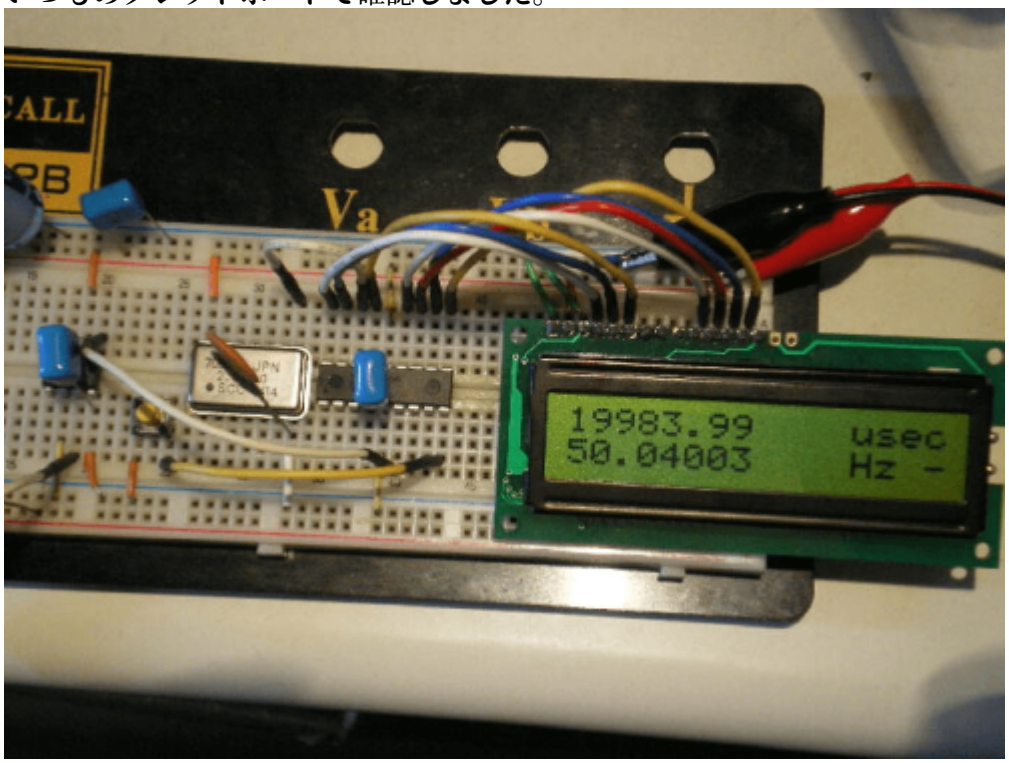
```
//    while (PIR1.CCP1IF == 0)    // キャプチャフラグ確認
//        ;
    T1CON.TMR1ON = 0;                          //タイマーオフ
    dat = CCPR1H << 8;
    dat |= CCPR1L;
    //
    return (dat);
}

void   main()
{
    static    char*        msg1 = "Cycle Measure v1";
    static    char*        msg2 = "JF3SFB{^_^}chan!";
    static    char*        msg3 = "                ";
    static    unsigned    char    buf[15];
    unsigned    int        dat;
    unsigned    char    prescale;
    double                    prescaled,tmp;
    //
//    OSCCON = 0b01110000;        // クロックは8MHz
    CMCON  = 0b00000111;        //コンパレータは使用しない。
    ANSEL  = 0b00000000;        // A/D変換は使用しない。
    TRISA  = 0b10111100;
    TRISB  = 0b00001111;
    OPTION_REG.F7 = 0;            // PORTBをプルアップする。
    //
    T1CON.TMR1CS = 0;
    T1CON.T1CKPS0 = 0;
    T1CON.T1CKPS1 = 0;
    T1CON.TMR1ON = 0;
    TMR1H = 0;
    TMR1L = 0;
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    prescale = 1;
    prescaled = PS1;
    //
    CCP1CON.CCP1M3 = 0;
    CCP1CON.CCP1M2 = 1;
    CCP1CON.CCP1M1 = 0;
    CCP1CON.CCP1M0 = 1;
    CCPR1H = 0;
    CCPR1L = 0;
    PIE1.CCP1IE = 0;
    PIR1.CCP1IF = 0;
    //
    Lcd_Custom_Config(&PORTB, 4, 5, 6, 7, &PORTA, 1, 0, 6);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Out(1, 1, msg1);
    Lcd_Custom_Out(2, 1, msg2);
    Delay_ms(500);
```

```
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    while (1) {
        dat = measurement();
        //
        tmp = (double)dat * prescaled;    // usec変換
        FloatToStr(tmp, buf);
        Lcd_Custom_Out(1, 1, msg3);
        Lcd_Custom_Out(1, 1, buf);
        Lcd_Custom_Out(1, 13, "usec");
        //
        tmp = 1000000.0 / tmp;                    //周波数変換
        FloatToStr(tmp, buf);
        Lcd_Custom_Out(2, 1, msg3);
        Lcd_Custom_Out(2, 1, buf);
        Lcd_Custom_Out(2, 13, "Hz");
        //
        if (PIR1.TMR1IF == 0) {         //オーバーフローチェック
            Lcd_Custom_Out(2, 16, "-");
        } else {
            PIR1.TMR1IF = 0;
            Lcd_Custom_Out(2, 16, "*");
            switch (prescale) {         //プリスケーラ自動調整
            case 1:
                T1CON.T1CKPS0 = 1;
                T1CON.T1CKPS1 = 0;
                prescale = 2;
                prescaled = PS2;
                break;
            case 2:
                T1CON.T1CKPS0 = 0;
                T1CON.T1CKPS1 = 1;
                prescale = 4;
                prescaled = PS4;
                break;
            case 4:
                T1CON.T1CKPS0 = 1;
                T1CON.T1CKPS1 = 1;
                prescale = 8;
                prescaled = PS8;
                break;
            case 8:
                break;
            }
        }
        //
        Delay_ms(500);
        //
        if (PORTB.F3 == 0) {         //プリスケーラリセット（無し：1／1）
            T1CON.T1CKPS0 = 0;
```

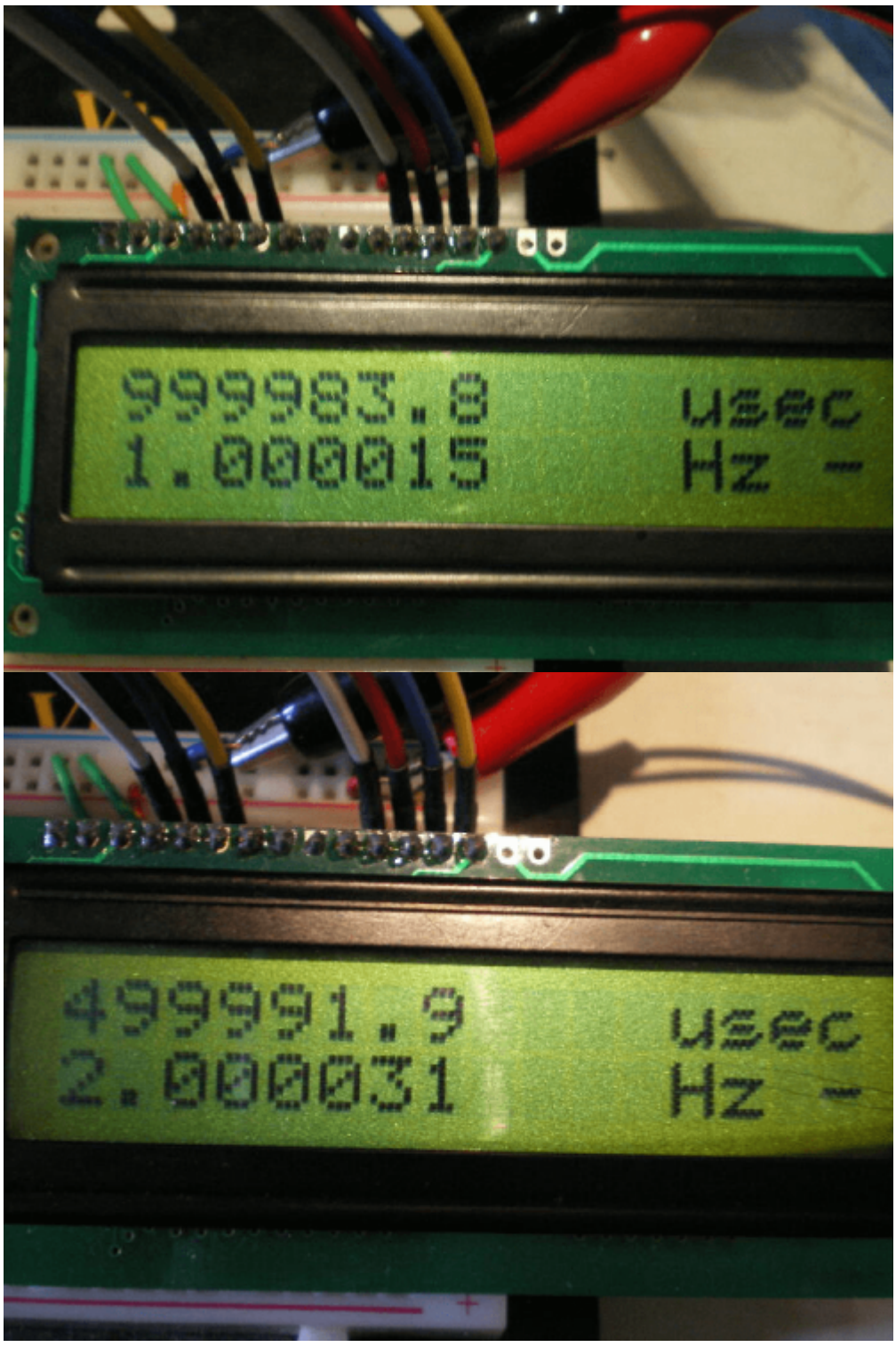
```
            T1CON.T1CKPS1 = 0;
            prescale = 1;
            prescaled = PS1;
        }
    }
}

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

動作確認

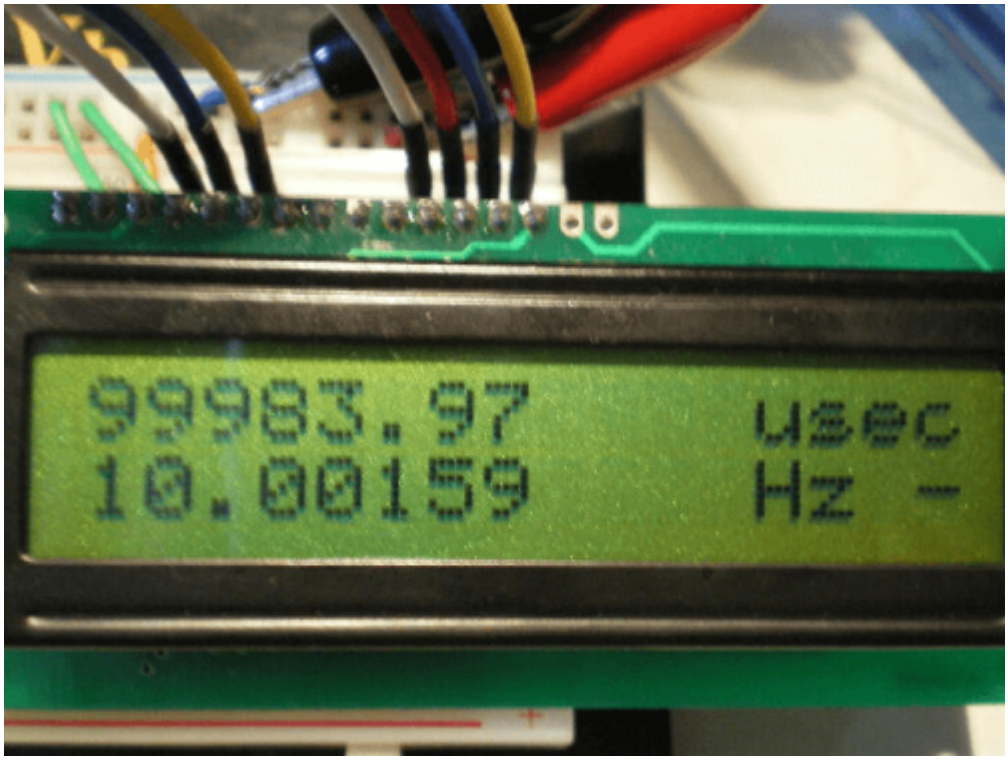
いつものブレッドボードで確認しました。



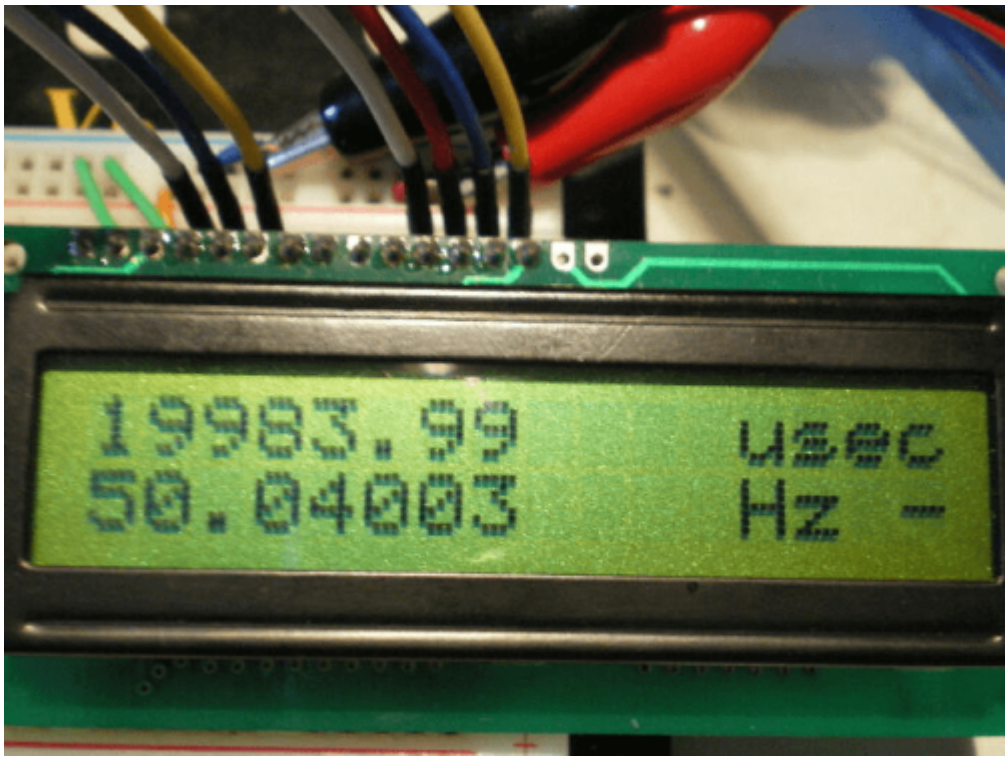
1Hzの信号を測定してみました。<有効桁数は小数点4桁>



10Hzの信号を測定してみました。<有効桁数は小数点3桁>



50Hzの信号を測定してみました。<有効桁数は小数点2桁>



100Hzの信号を測定してみました。<有効桁数は小数点1桁>