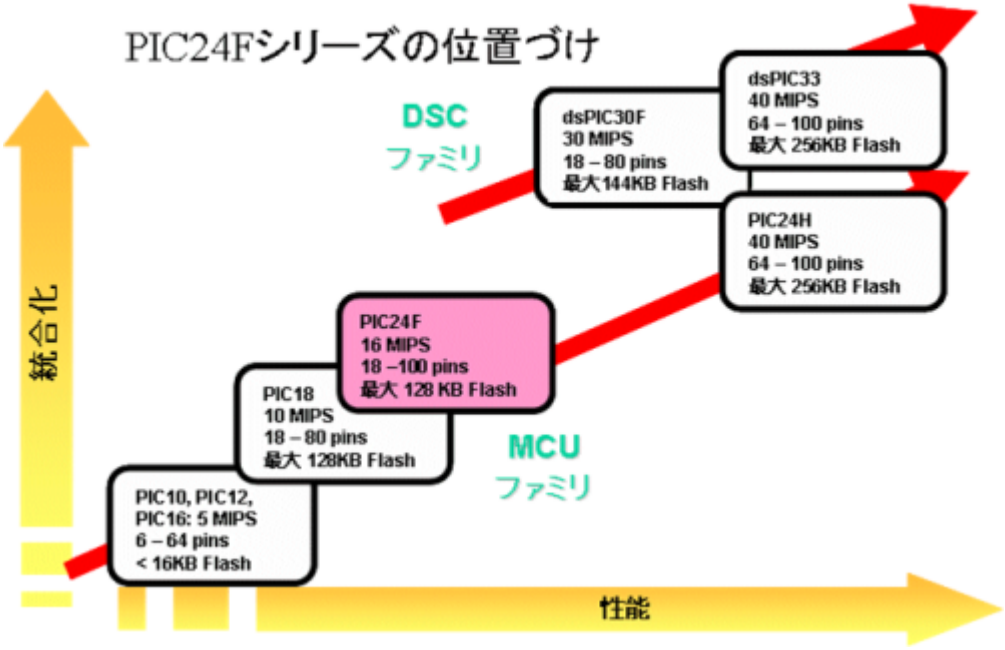
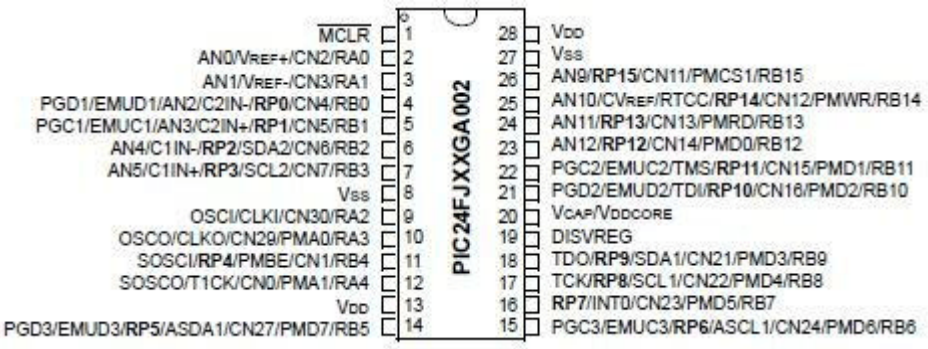
周波数カウンタV6(16ビットPIC採用)(PIC24FJ64GA002)

概要

今までは、8ビットのPICで製作をしてきましたが、今回は16ビットのPICで周波数カウンタを製作して



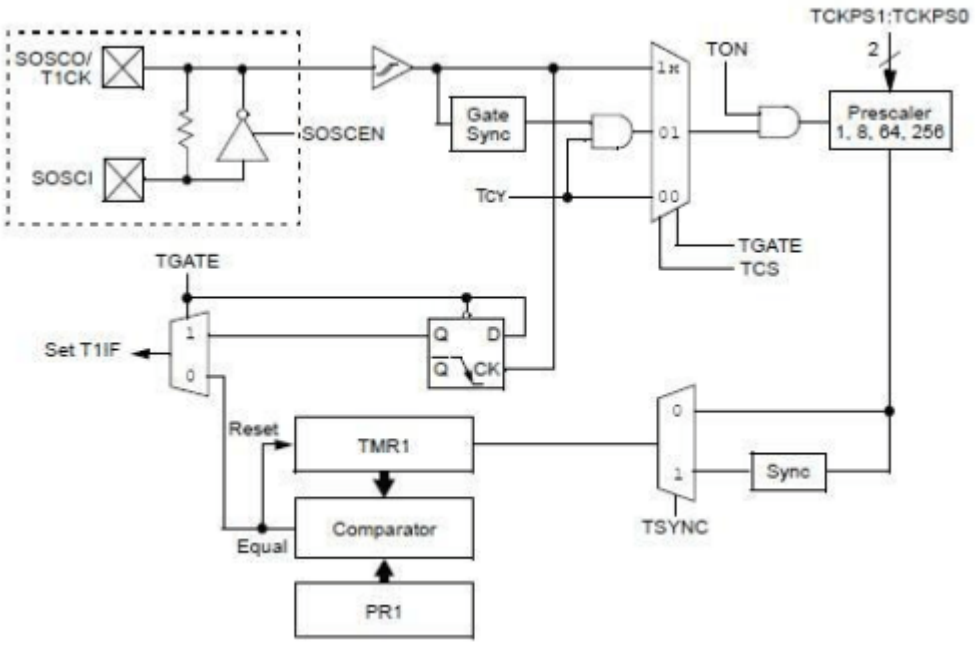
みました。



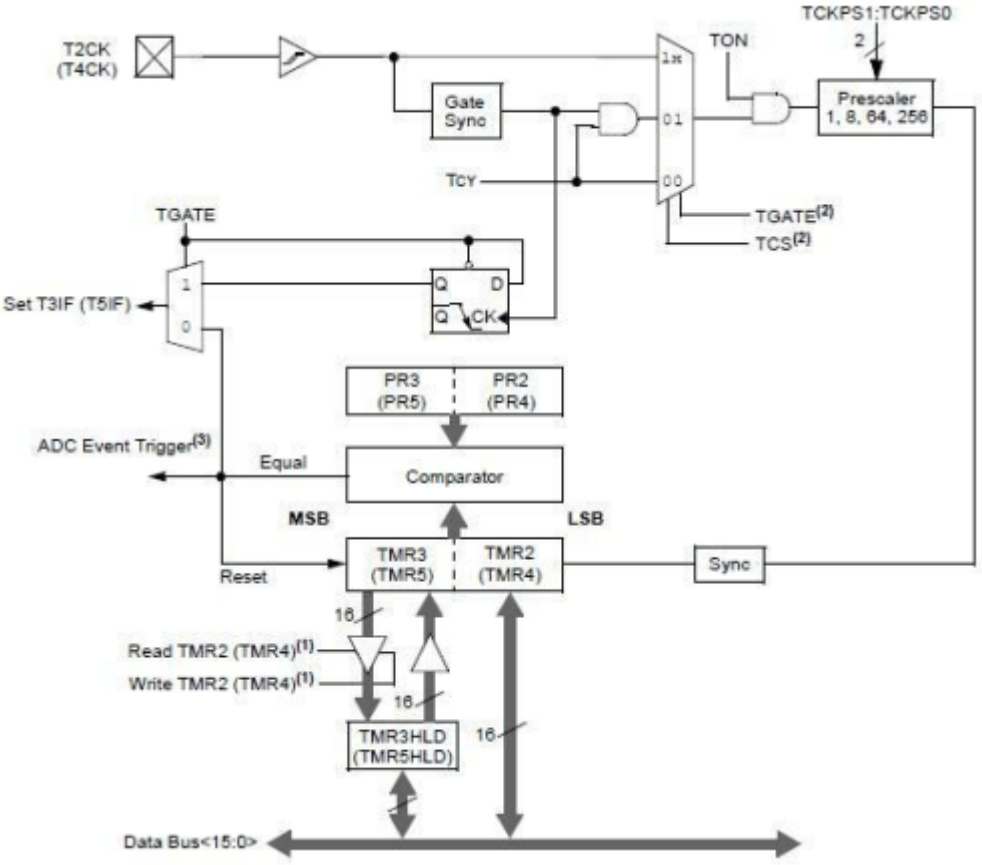
動作原理

ゲートタイマ(1秒用、0.1秒用)には□TIMER1(16ビット)を使用しました。 クロック周波数が32MHz□プリスケーラ値が256なので□TIMER1への設定値は次の式で求めます。

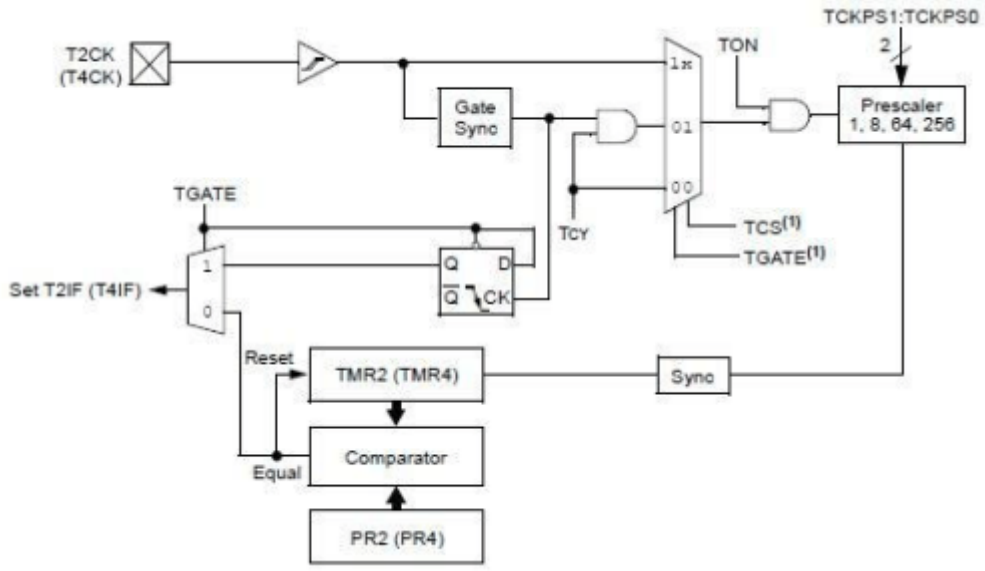
```
0xBDC = 65536 - (1 / ((1 / (32MHz / 2)) * 256))
```



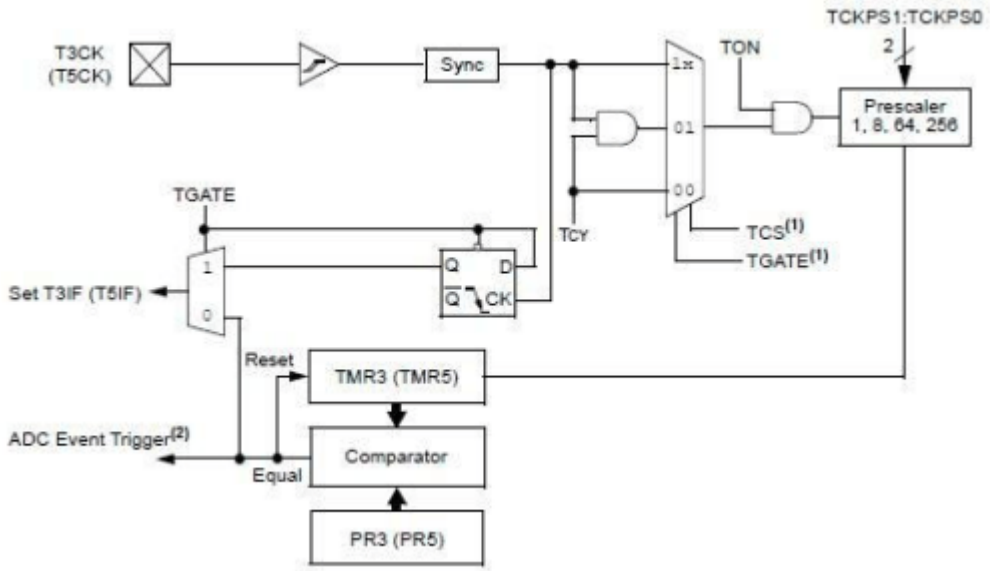
入力される信号のカウント用には□TIMER2とTIMER3を結合した、32ビットのカウンタを使用します。
従って、論理上は、4,294,967,296Hzまでのカウントが可能となります。 これでソフトウエアの仕組み



も簡単になりました。
TIMER2は単体では、16ビットのタイマとして動作します。

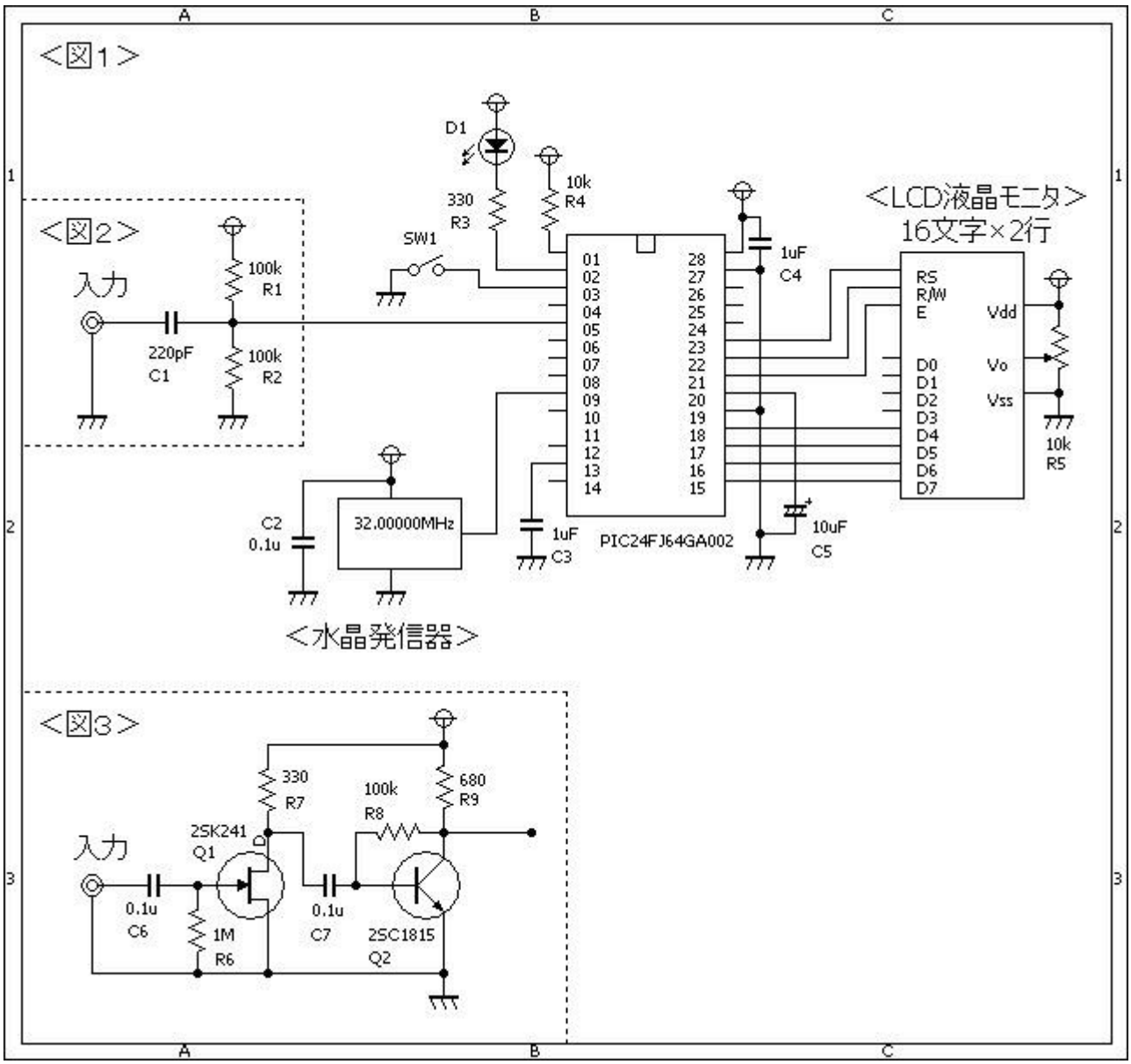


TIMER3は単体では、16ビットのタイマとして動作します。



回路図

動作確認は、図1と図2で実施しました。 実用(感度、インピーダンス等)を考慮するのであれば、図2の箇所を図3のアンプに置き換えてください SW1で、プリスケーラの値を、1/1、1/8の切替を可能としました。



※注意事項 PIC24FJ64GA002の電源電圧は、規格では2.0V~3.6Vなのですが、今回は無理させて5.0Vを加えて います。実用的な回路にするには、もう少し工夫が必要です。

ソースコード

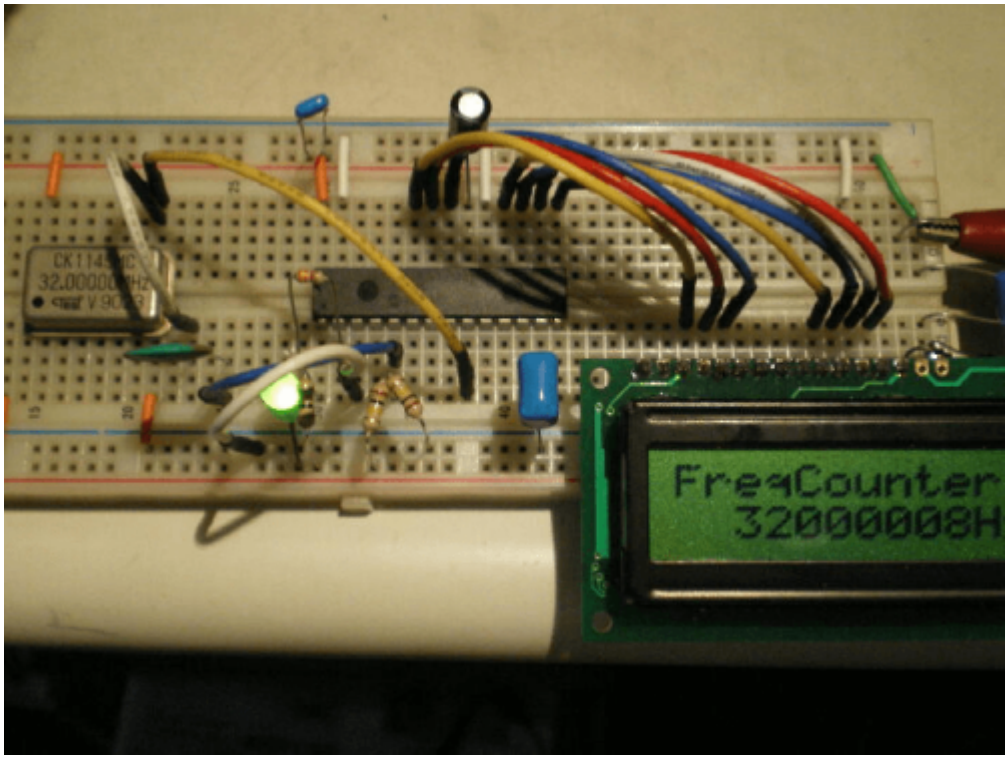
FreqCounterV6.c

```
//*****************************************************************************
*
/*
    「周波数カウンタ」

□□DeviceFlags□
□□_IESO_OFF
□□_FNOSC_PRI
□□_FCKSM_CSDCMD
□□_OSCIOFNC_OFF
```

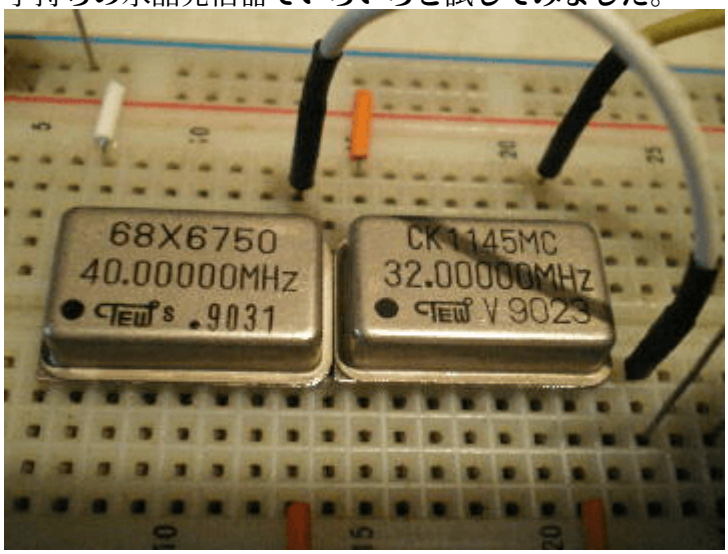
```
□□_POSCMOD_EC
□□_JTAGEN_OFF
□□_GCP_OFF
□□_GWRP_OFF
□□_BKBUG_OFF
□□_COE_OFF
□□_ICS_PGx1
□□_FWDTEN_OFF
*/
//*********************************************************************
*

#define       LED        PORTA.F0

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

void Timer1Int() org 0x1A
{
    Delay_Cyc(0, 215);
    T1CON.F15 = 0;
    T2CON.F15 = 0;
    IFS0bits.T1IF = 0;
}

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

void main()
{
    unsigned    long    cnt;
    char                buf[20];
    //
    ADPCFG = 0xFFFF;
    TRISA  = 0b11111110;
    TRISB  = 0x0002;
    // timer1(16bits)□ゲートタイム制御
    IPC0bits.T1IP0 = 1;
    IFS0bits.T1IF = 0;
    IEC0bits.T1IE = 1;
    T1CONbits.TCKPS0 = 1;
    T1CONbits.TCKPS1 = 1;
    T1CON.F15 = 0;
    // timer2+3(32bits)□周波数カウント制御
    IPC2bits.T3IP0 = 1;
    IFS0bits.T3IF = 0;
    IEC0bits.T3IE = 0;
    T2CONbits.TCKPS0 = 0;
    T2CONbits.TCKPS1 = 0;
    T2CONbits.T32 = 1;
    T2CONbits.TCS = 1;
```

```
    T2CON.F15 = 0;
    //
    Lcd_Custom_Config(&PORTB, 6, 7, 8, 9, &PORTB, 12, 11, 10);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Out(1, 1, "FreqCounter V6");
    //
    RPINR3bits.T2CKR0 = 1;
    RPINR3bits.T2CKR1 = 0;
    RPINR3bits.T2CKR2 = 0;
    RPINR3bits.T2CKR3 = 0;
    RPINR3bits.T2CKR4 = 0;
    //
    while(1) {
        if (PORTA.F1 == 1)
            T2CONbits.TCKPS0 = 0;    // 1/1
        else
            T2CONbits.TCKPS0 = 1;    // 1/8
        //
        TMR1 = 0xBDC;    // 65536 - (1 / ((1 / 16MHz) * 256))
        TMR2 = 0;
        TMR3 = 0;
        T1CON.F15 = 1;
        T2CON.F15 = 1;
        //
        LED = 0;
        while (T1CON.F15 == 1)
            ;
        LED = 1;
        //
        cnt = TMR3;
        cnt = cnt << 16;
        cnt = cnt | TMR2;
        if (PORTA.F1 != 1)
            cnt *= 8;
        //
        LongWordToStr(cnt, buf);
        Lcd_Custom_Out(2, 1, buf);
        Lcd_Custom_Out(2, 11, "Hz");
        //
        Delay_ms(500);
    }
}

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

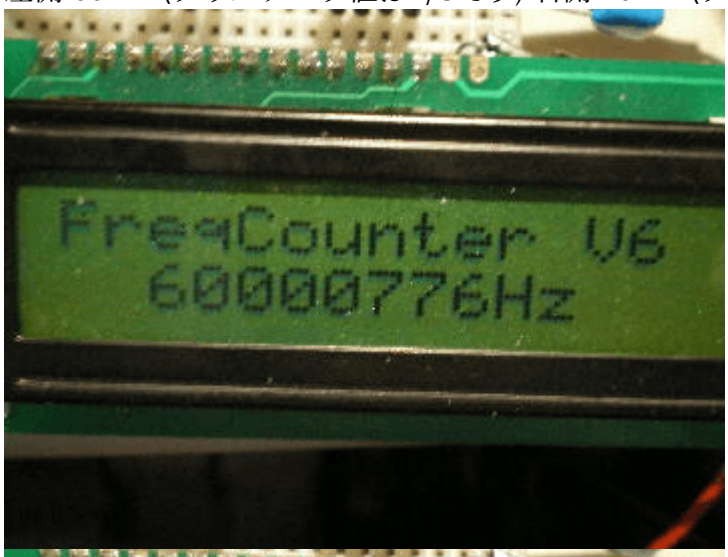
動作確認



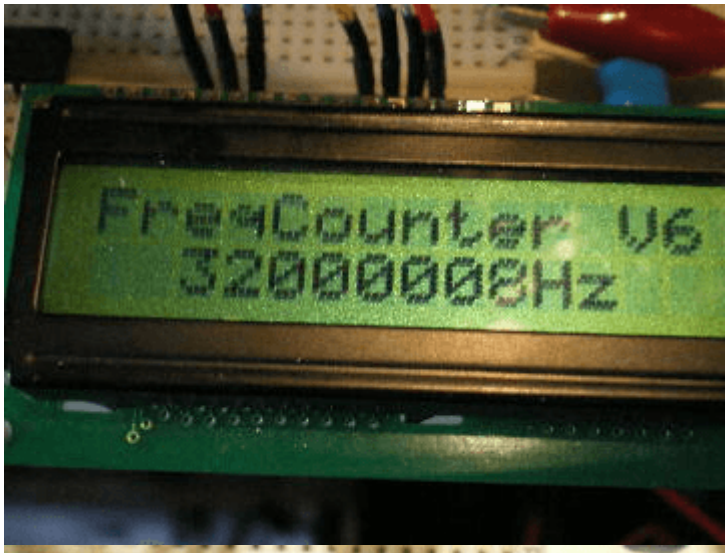
手持ちの水晶発信器でいろいろと試してみました。



左側:60MHz(プリスケーラ値は1/8です) 右側:40MHz(プリスケーラ値は1/8です)



左側:32MHz(プリスケーラ値は1/8です) 右側:20MHz(プリスケーラ値は1/8です)



左側:12MHz(プリスケーラ値は1/8です) 右側:6MHz(プリスケーラ値は1/1です)

如何ですか? こんな簡単な回路で60MHzまでを測定できるんですねえ。。。{ }!