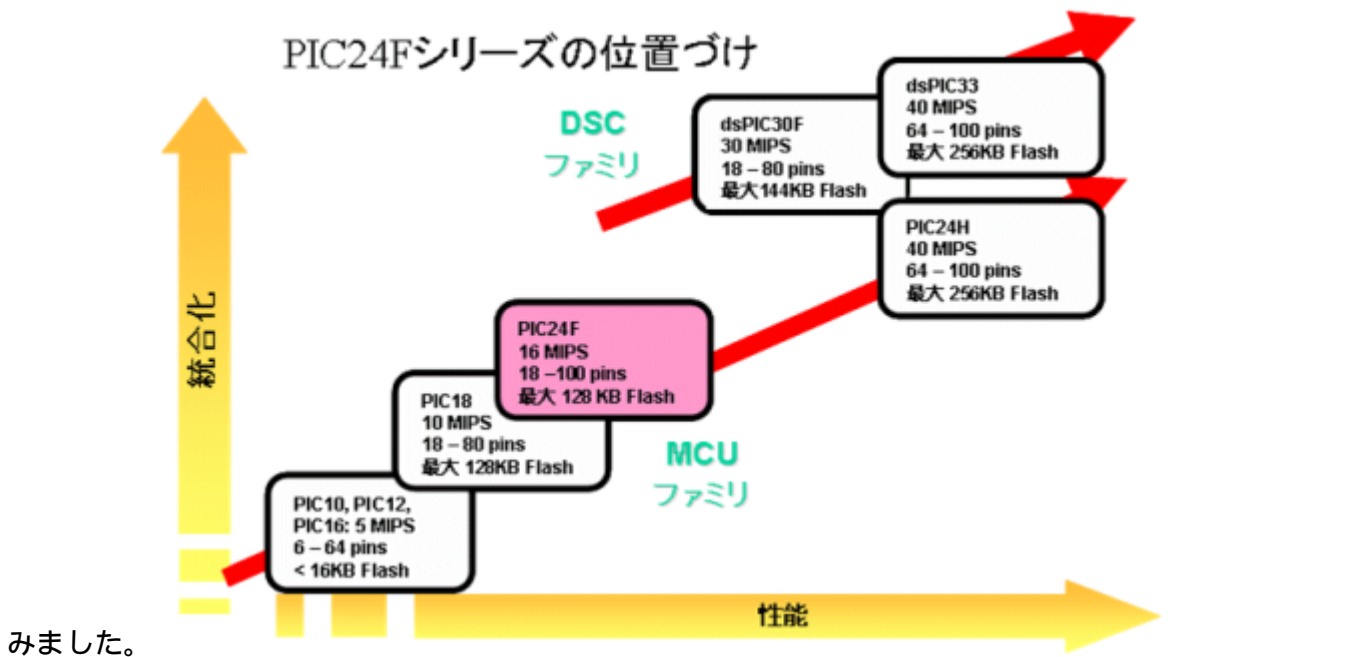


# 周波数カウンタV6(16ビットPIC採用)(PIC24FJ64GA002)

## 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

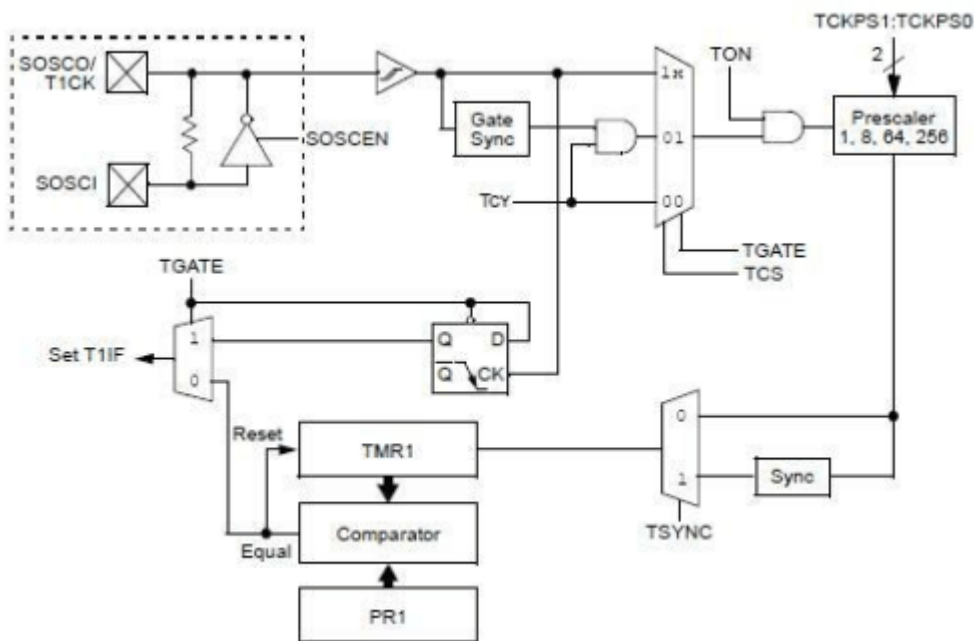


MCLR	1	28	V <sub>DD</sub>
AN0/VREF+/CN2/RA0	2	27	V <sub>SS</sub>
AN1/VREF-/CN3/RA1	3	26	AN9/RP15/CN11/PMCS1/RB15
PGD1/EMUD1/AN2/C2IN-/RP0/CN4/RB0	4	25	AN10/CVREF/RTCC/RP14/CN12/PMWR/RB14
PGC1/EMUC1/AN3/C2IN+/RP1/CN5/RB1	5	24	AN11/RP13/CN13/PMRD/RB13
AN4/C1IN-/RP2/SDA2/CN6/RB2	6	23	AN12/RP12/CN14/PMDD/RB12
AN5/C1IN+/RP3/SCL2/CN7/RB3	7	22	PGC2/EMUC2/TMS/RP11/CN15/PMD1/RB11
V <sub>SS</sub>	8	21	PGD2/EMUD2/TDI/RP10/CN16/PMDD2/RB10
OSCI/CLKI/CN30/RA2	9	20	V <sub>CAP</sub> /V <sub>DDCORE</sub>
OSCO/CLKO/CN29/PMA0/RA3	10	19	DISVREG
SOSCI/RP4/PMBE/CN1/RB4	11	18	TDO/RP9/SDA1/CN21/PMDD3/RB9
SOSCO/T1CK/CN0/PMA1/RA4	12	17	TCK/RP8/SCL1/CN22/PMDD4/RB8
V <sub>DD</sub>	13	16	RP7/INT0/CN23/PMDD5/RB7
PGD3/EMUD3/RP5/ASDA1/CN27/PMDD7/RB5	14	15	PGC3/EMUC3/RP6/ASCL1/CN24/PMDD6/RB6

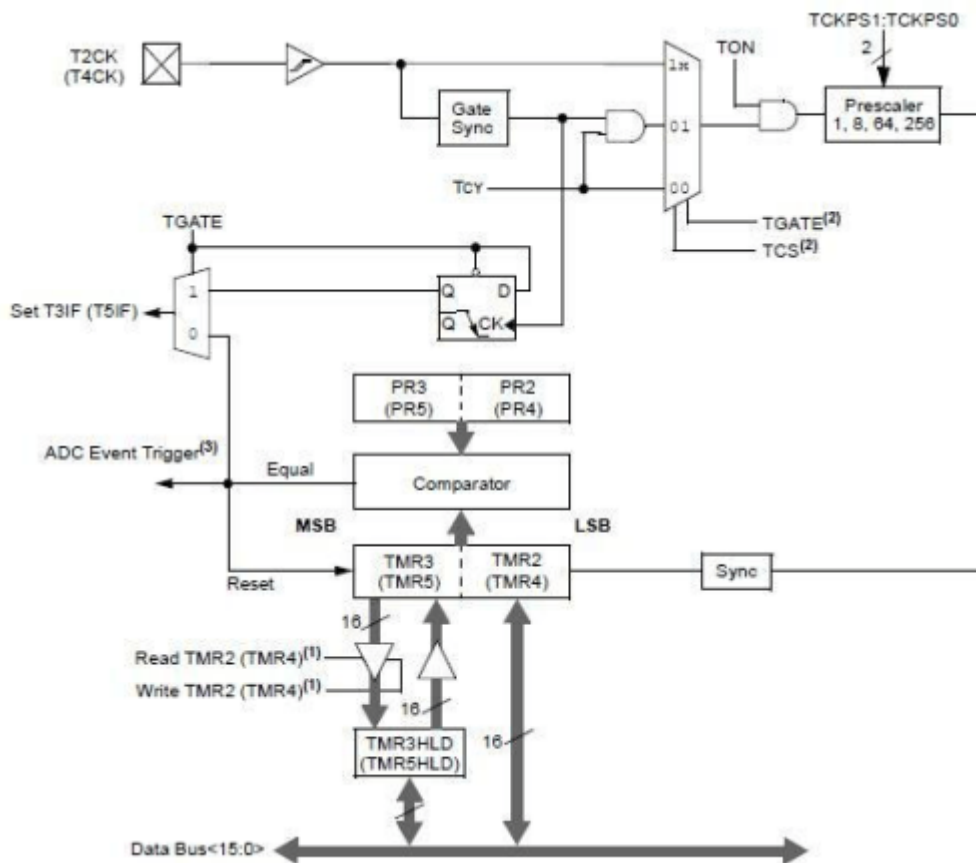
## 動作原理

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、プリスケアラ値が256なのでTIMER1への設定値は次の式で求めます。

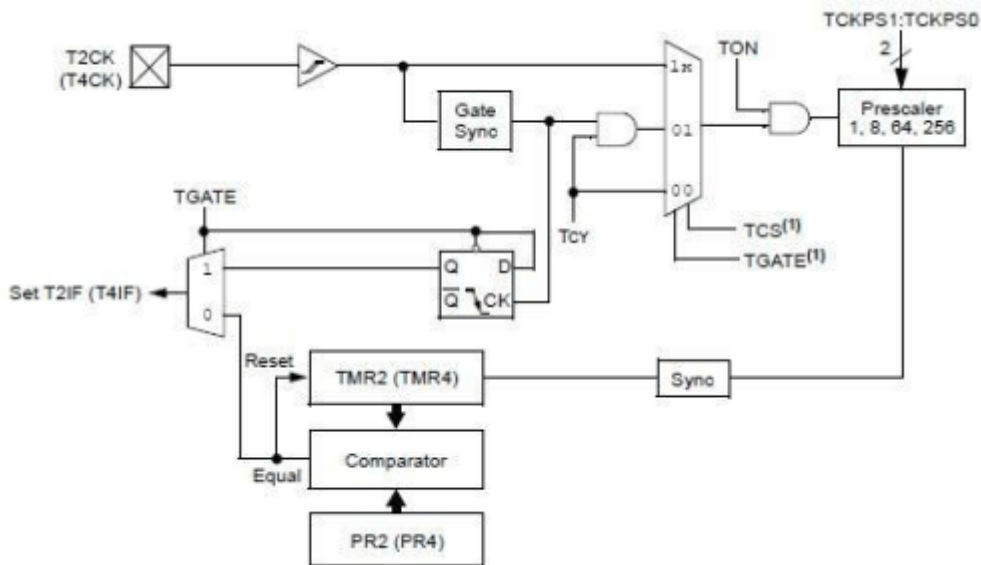
$$0xBDC = 65536 - (1 / ((1 / (32\text{MHz} / 2)) * 256))$$



入力される信号のカウンタ用には、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までのカウンタが可能となります。これでソフトウェアの仕組み

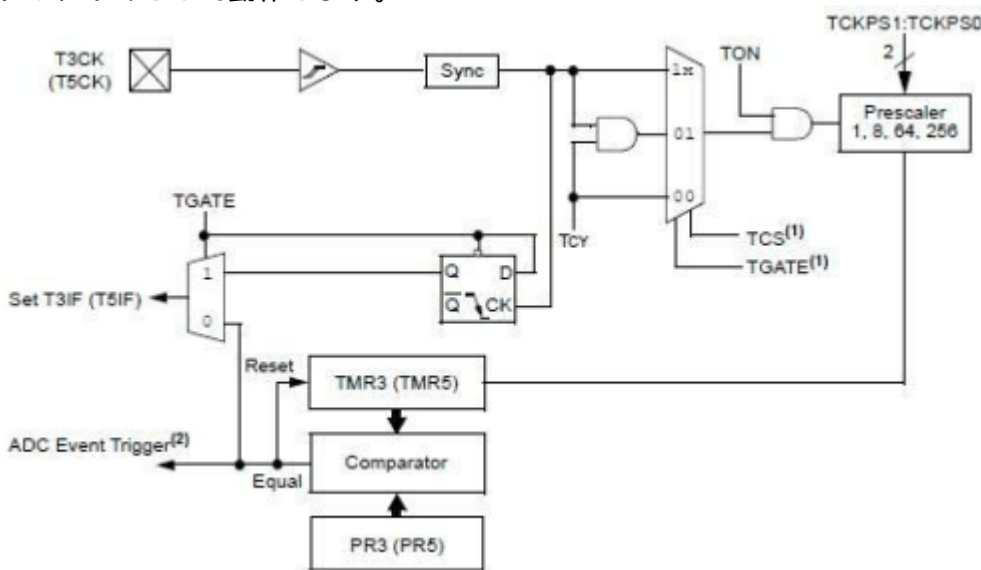


も簡単になりました。  
TIMER2は単体では、16ビットのタイマとして動作します。



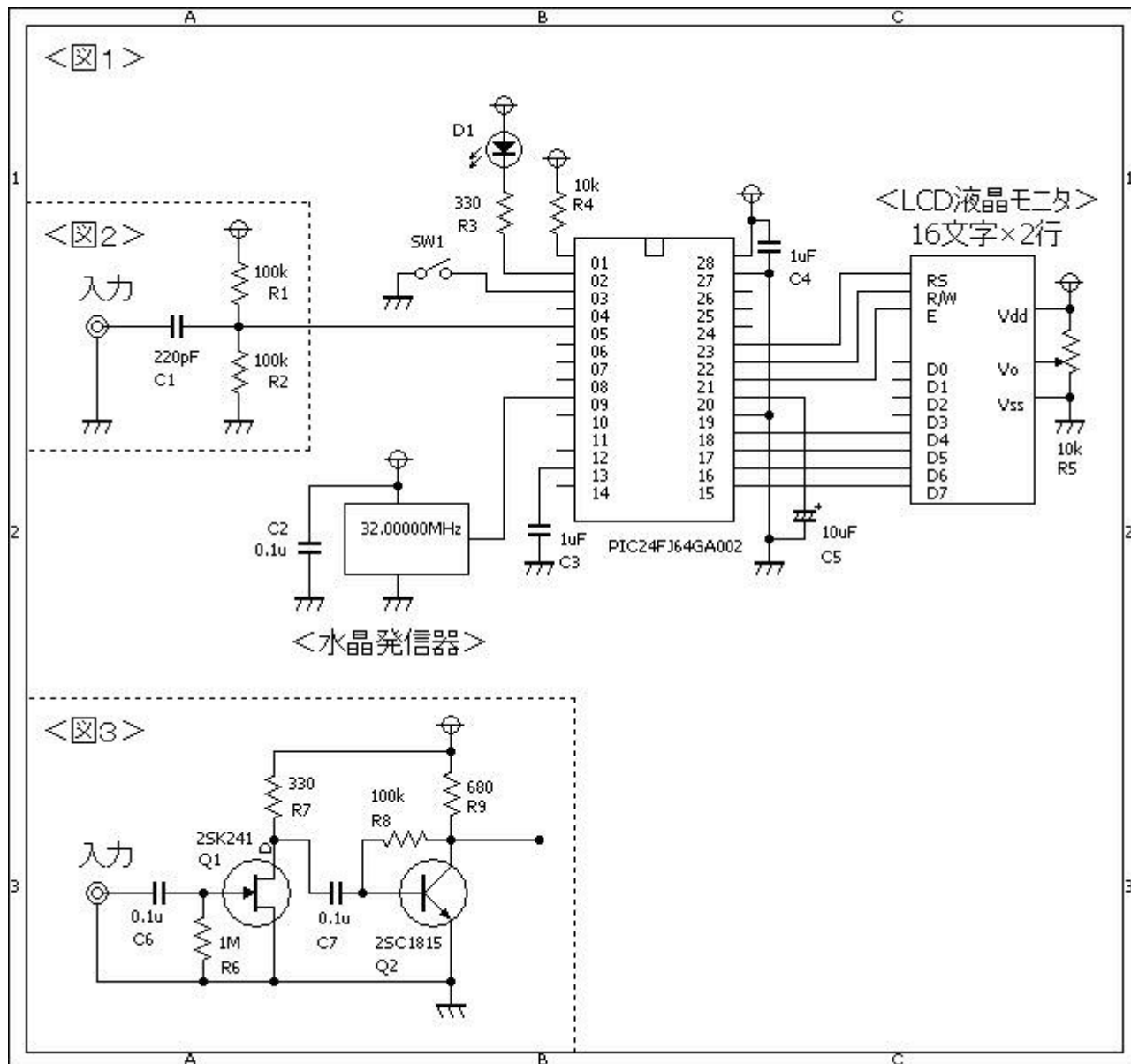
TIMER3は単体では、16ビット

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、プリスケアラの値を、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
```

```

__POSXMOD_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;
}

//*****
*

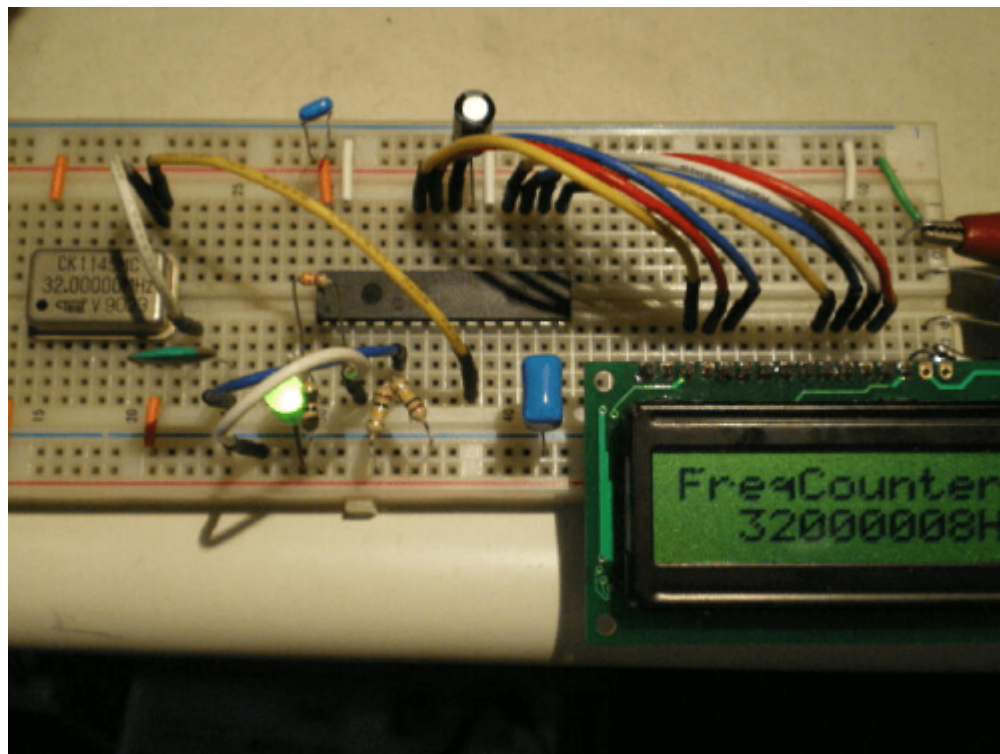
void main()
{
    unsigned    long    cnt;
    char        buf[20];
    //
    ADPCFG = 0xFFFF;
    TRISA  = 0b11111110;
    TRISB  = 0x0002;
    // timer1(16bits)ゲートタイム制御
    IPC0bits.T1IP0 = 1;
    IFS0bits.T1IF = 0;
    IEC0bits.T1IE = 1;
    T1CONbits.TCKPS0 = 1;
    T1CONbits.TCKPS1 = 1;
    T1CON.F15 = 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;
}

```

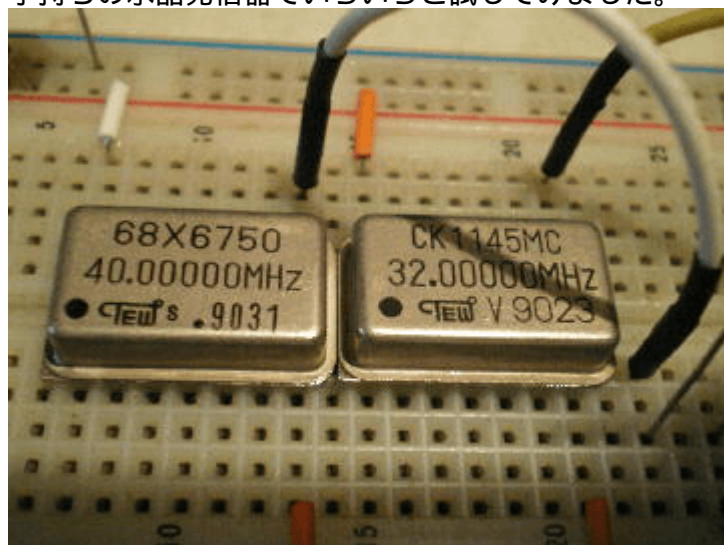
```
T2CON.F15 = 0;
//
Lcd_Custom_Config(&PORTB, 6, 7, 8, 9, &PORTB, 12, 11, 10);
Lcd_Custom_Cmd(LCD_CURSOR_OFF);
Lcd_Custom_Out(1, 1, "FreqCounter V6");
//
RPINR3bits.T2CKR0 = 1;
RPINR3bits.T2CKR1 = 0;
RPINR3bits.T2CKR2 = 0;
RPINR3bits.T2CKR3 = 0;
RPINR3bits.T2CKR4 = 0;
//
while(1) {
    if (PORTA.F1 == 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

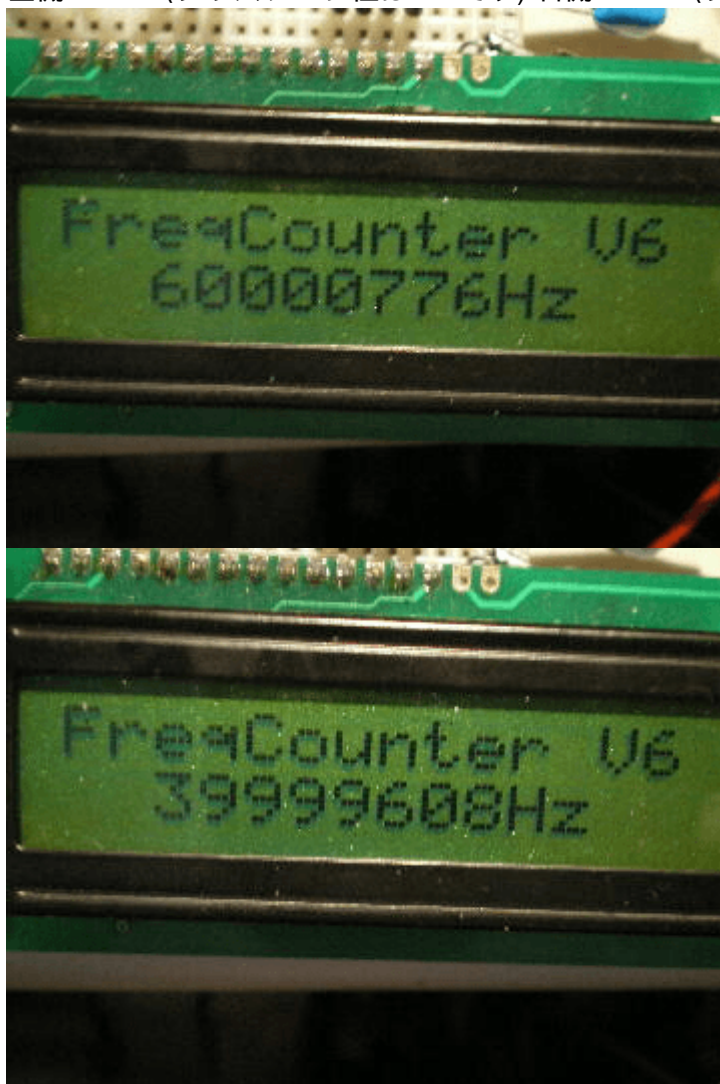


手持ちの水晶発信器でいろいろと試してみました。





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



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



左側:12MHz(プリスケータ値は1/8です) 右側:6MHz(プリスケータ値は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。。。{ 😊 }!

From:  
<http://www.deepsky.jp/wiki/> - うごくといいな

Permanent link:  
<http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:otherpic:171&rev=1588238922>

Last update: **2025/10/17 14:27**

