

周波数カウンタV7

概要

以前に、周波数カウンタV5(集大成版)を作成し、ほぼ終了と思っていましたが、更に検討を進めるうちに、ちょっとした工夫で、思わぬ効果を得ることが出来ました。

- プリスケアラ無(1/1)で、40MHzまで測定可能(実測値)
- プリスケアラ有(1/8)で、80MHzまで測定可能(実測値)

動作原理

PICを使用した周波数カウンタでは、一般的にはTIMER0とTIMER1を組み合わせています。

- 信号をカウントするためにTIMER0(8ビット)を使用(外部クロック入力同期ON固定)
- ゲートタイム(1秒、0.1秒)を得るためにTIMER1(16ビット)を使用

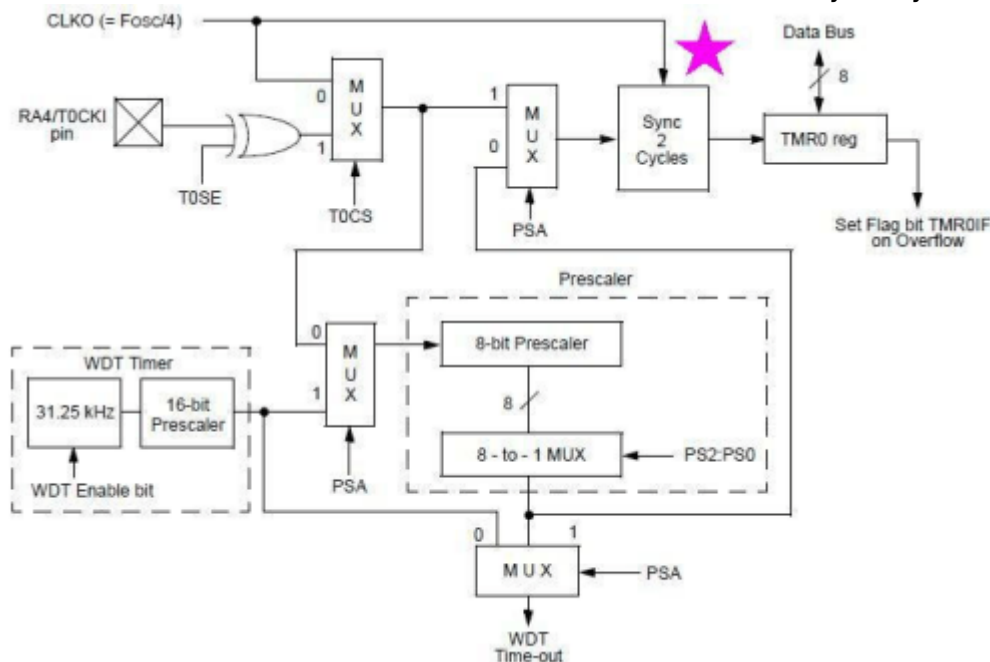
しかしTIMER0は、TIMERのON/OFF制御が出来ないため、入力ピンを強制的に出力モードにして“0”を出力する等の工夫(本来は余分な事)が必要となります。

そこでTIMERでON/OFFの出来る、TIMER1とTIMER2を組み合わせました。

- 信号をカウントするためにTIMER1(16ビット)を使用(外部クロック入力同期制御ON/OFF可能)
- ゲートタイム(1秒、0.1秒)を得るためにTIMER2(8ビット)を使用

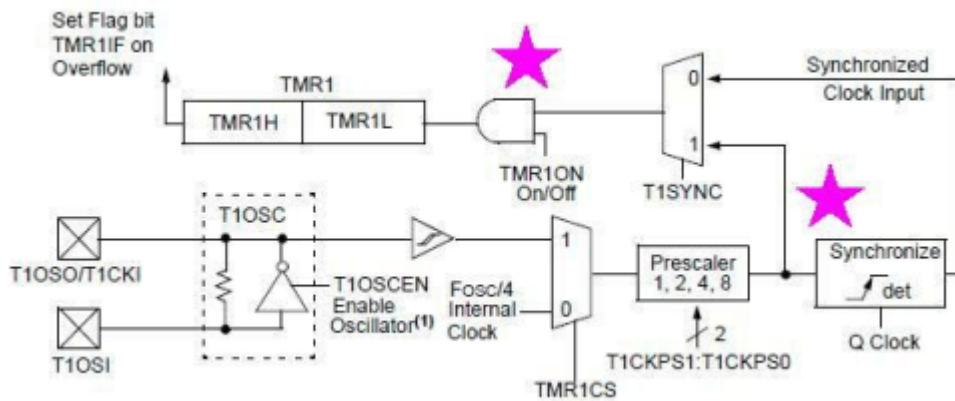
特に、外部クロック入力同期制御をOFFにすることが肝心で、もしもこれをONにすると20MHzの信号でも1/8のプリスケアラに設定しないと測定が出来ません。多分TIMER0方式でのカウントが高い周波数まで伸びないのは、外部クロック入力同期制御がON固定に原因があるのかもしれませんが。

<TIMER0> TIMERのON/OFF制御が出来ない。外部クロック入力同期(Sync2Cycles)がONに固定されて



いる。

<TIMER1> 外部クロック入力同期(Synchronize)をON/OFFすることが可能 □ TMR1ONでTIMERを開始/停



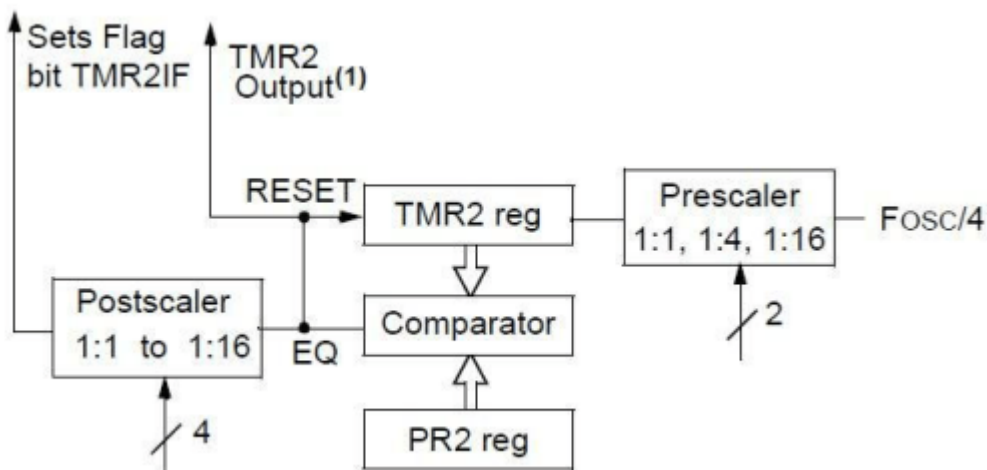
止することが可能。

T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
bit 7							



<TIMER2> TMR2ONでTIMERを開始/停止することが可能。

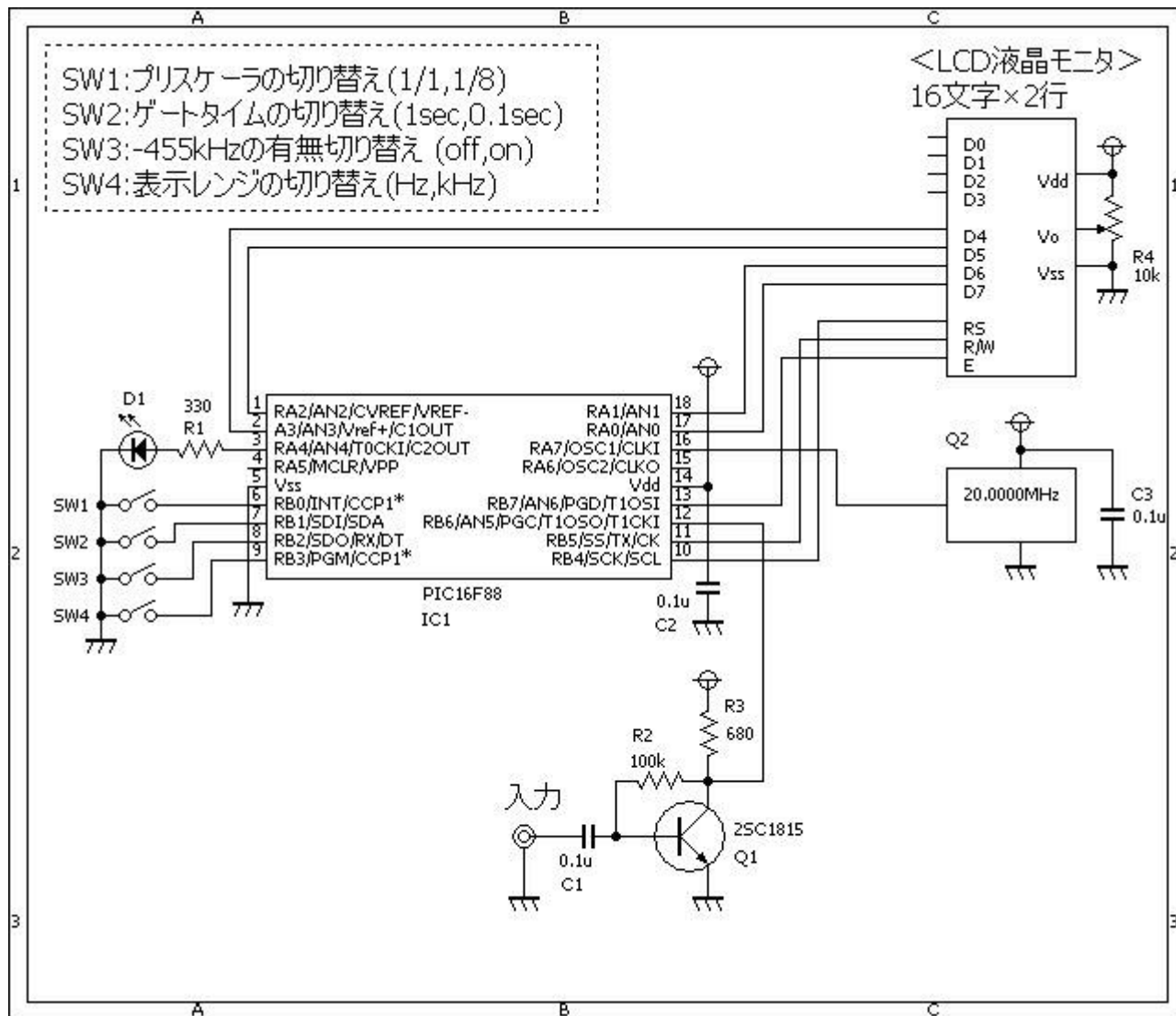


T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0



回路図



ソースコード

[FreqCounterV7.c](#)

```
//*****
*
/*
  <周波数カウンター>
  機能
  [] 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=0kHz表示
    コンフィグ設定
□□LVP_OFF
□□MCLR_OFF
□□WDT_OFF
□□EXTCLK
    ピンアサイン
□□Pin-01□□□□□□□□
□□Pin-02□□□□□□□□
□□Pin-03□□□□□
□□Pin-04□未使用
□□Pin-05□□□□□□□□□□
□□Pin-06□プリスケアラの切替□□
□□Pin-07□ゲートタイムの切替□□
□□Pin-08□□□□□□□□の切替□□
□□Pin-09□表示レンジの切替□□
□□Pin-10□□□□□□□□
□□Pin-11□□□□□□□□
□□Pin-12□信号入力
□□Pin-13□□□□□□□□
□□Pin-14□□□□□□□□□□
□□Pin-15□未使用
□□Pin-16□クロック入力□□□□□□入力)
□□Pin-17□□□□□□□□
□□Pin-18□□□□□□□□
*/
//*****
*

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

#define      LED      PORTA.F4

#define      GATETIME_100MSEC    10
#define      GATETIME_1SEC      1

//*****
*

static unsigned    int    MeasurementCnt;

void    interrupt()
{
    PIR1.TMR2IF = 0;
    //
    MeasurementCnt--;
    if (MeasurementCnt == 0) {
        T1CON.TMR1ON = 0;    // ゲートを閉める。
    }
}

```



```
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
asm    nop;
//
T1CON.TMR10N = 1;    //ゲートを開ける。
// 測定
while (T2CON.TMR20N != 0) {
    if (PIR1.TMR1IF == 1) {
        PIR1.TMR1IF = 0;
        freq++;
    }
}
if (PIR1.TMR1IF == 1) {
    PIR1.TMR1IF = 0;
    freq++;
}
//換算
freq = freq * 65536;
freq = freq + ((unsigned)TMR1H * 256) + (unsigned)TMR1L;
//
return (freq);
}

//*****
*

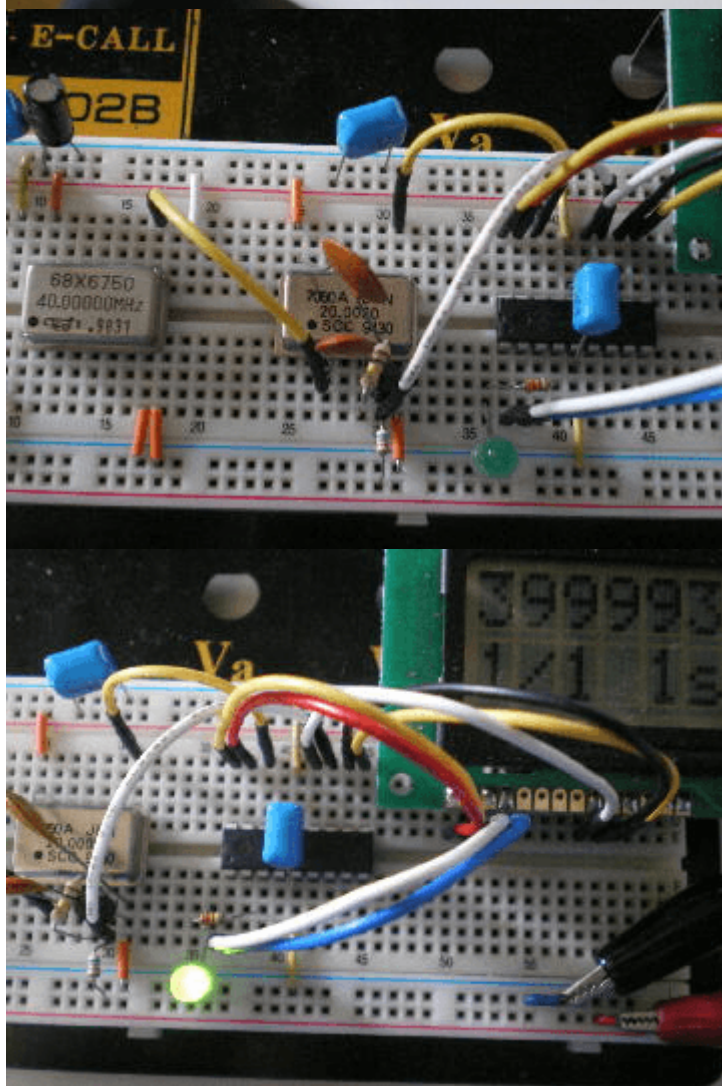
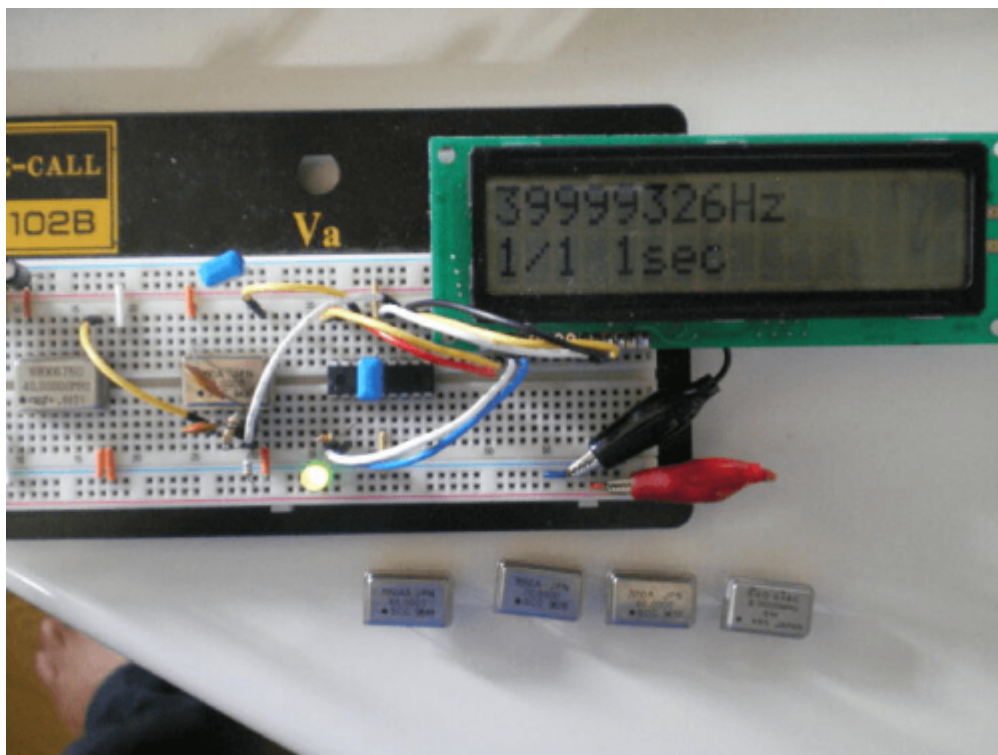
void main()
{
    static    char*          msg;
    static    unsigned long  freq, temp;    // 0...4294967295
    static    unsigned char  buf[20], prescaler, gateTime;
    // アナログの設定
    ANSEL = 0b00000000;    // 使用しない。
    // ポートの設定
    TRISA = 0b11100000;
    TRISB = 0b01001111;
    OPTION_REG.F7 = 0;    // PORTBをプルアップする。
    // TIMER2の設定
    PIE1.TMR2IE = 1;
    PIR1.TMR2IF = 0;
    T2CON.TOUTPS0 = 0;
    T2CON.TOUTPS1 = 0;
    T2CON.TOUTPS2 = 0;
```

```
T2CON.TOUTPS3 = 0;
T2CON.TMR2ON = 0;
T2CON.T2CKPS0 = 1;
T2CON.T2CKPS1 = 1;
TMR2 = 0;
// TIMER1の設定
PIE1.TMR1IE = 0;
PIR1.TMR1IF = 0;
T1CON.T1RUN = 0;
T1CON.T1CKPS0 = 0;
T1CON.T1CKPS1 = 0;
T1CON.T10SCEN = 0;
T1CON.NOT_T1SYNC = 1;
T1CON.TMR1CS = 1;
T1CON.TMR1ON = 0;
TMR1L = 0;
TMR1H = 0;
// 変数の初期化
prescaler = 1;
gateTime = GATETIME_1SEC;
// □□□□液晶モニタ)の初期化
Lcd_Custom_Config(&PORTA,0,1,2,3,&PORTB,4,5,7);
Lcd_Custom_Cmd(LCD_CURSOR_OFF);
Lcd_Custom_Out(1, 1, "FreqCounter V7");
Delay_ms(1000);
Lcd_Custom_Cmd(LCD_CLEAR);
//
while (1) {
    // 周波数の測定
    LED = 1;
    freq = FreqMeasurement(gateTime);
    LED = 0;
    // 換算
    freq = freq * prescaler * gateTime;
    // プリスケーラの切り替え
    if (sw1 == 1) {
        T1CON.T1CKPS0 = 0;
        T1CON.T1CKPS1 = 0;
        prescaler = 1;
        msg = "1/1 ";
    } else {
        T1CON.T1CKPS0 = 1;
        T1CON.T1CKPS1 = 1;
        prescaler = 8;
        msg = "1/8 ";
    }
    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 ";
    }
}
```

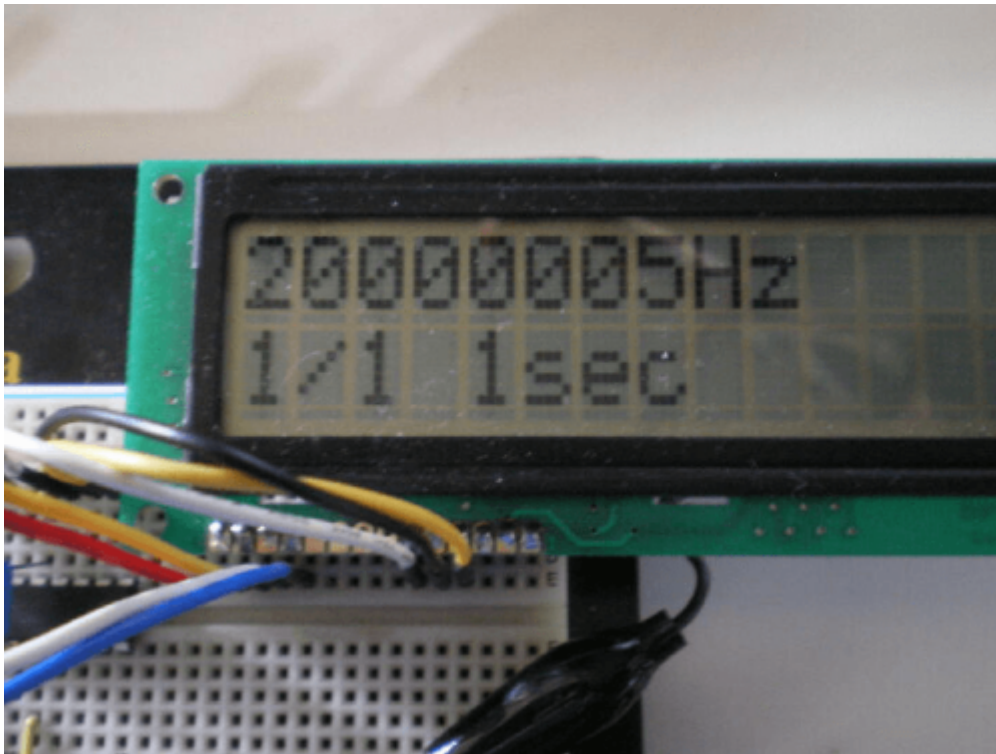
```
    } else {
        gateTime = GATETIME_100MSEC;
        msg = "0.1sec ";
    }
    Lcd_Custom_Out(2, 5, msg);
    // □□□□Hzの有無
    if (sw3 == 0) {
        freq -= 455000;
        msg = "-455k";
    } else {
        msg = "      ";
    }
    Lcd_Custom_Out(2, 12, msg);
    // 表示レンジの切り替え
    if (sw4 == 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]);
    //
    Delay_ms(500);
}
}

//*****
*
```

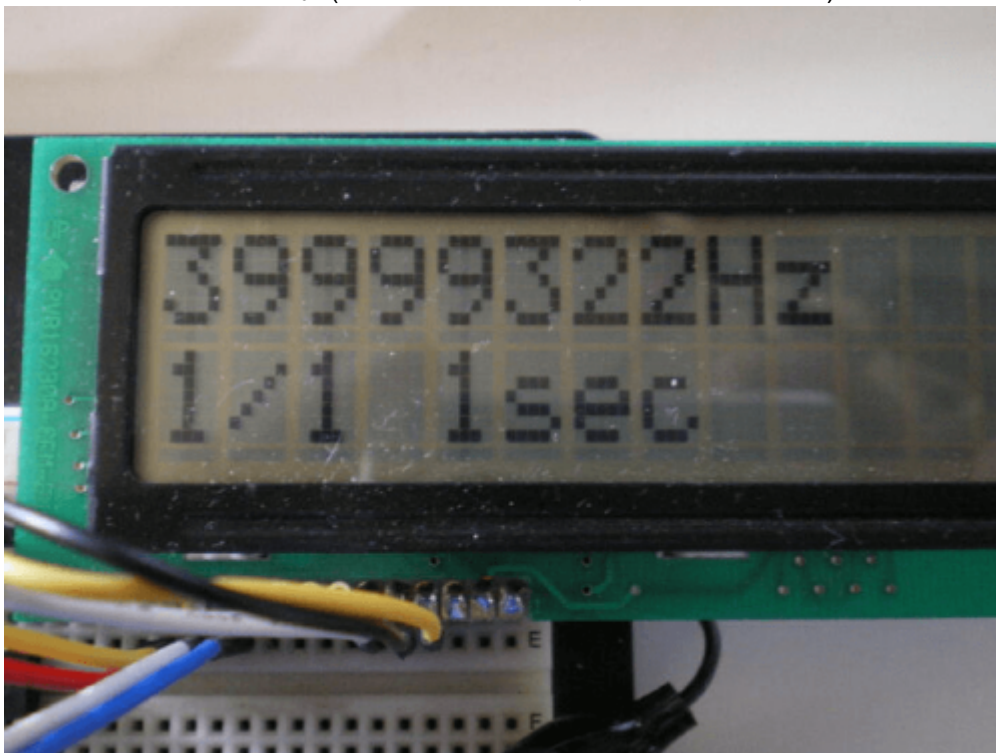
動作確認



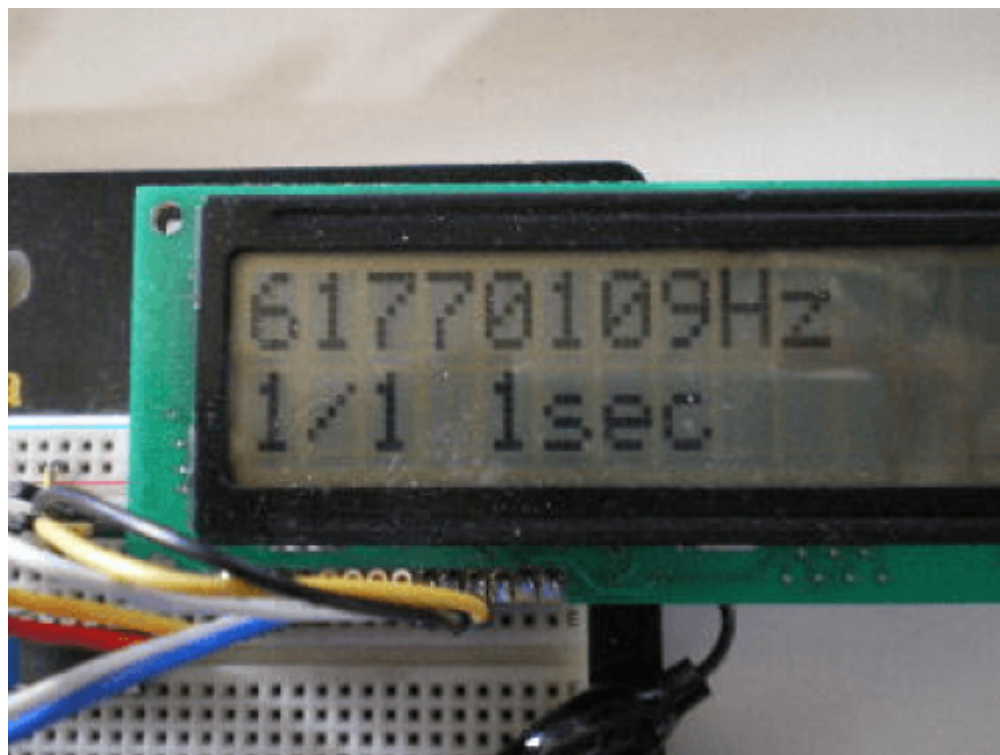
20MHzを測定しました。(プリスケアラ無し、ゲートタイム1秒)

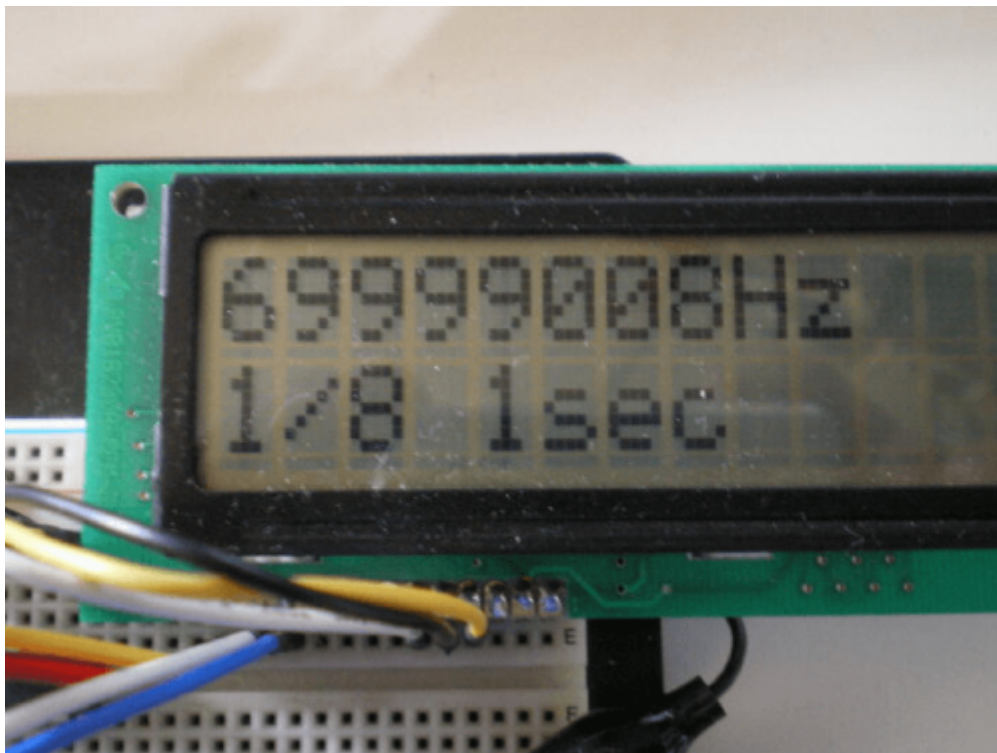


40MHzを測定しました。(プリスケアラ無し、ゲートタイム1秒)



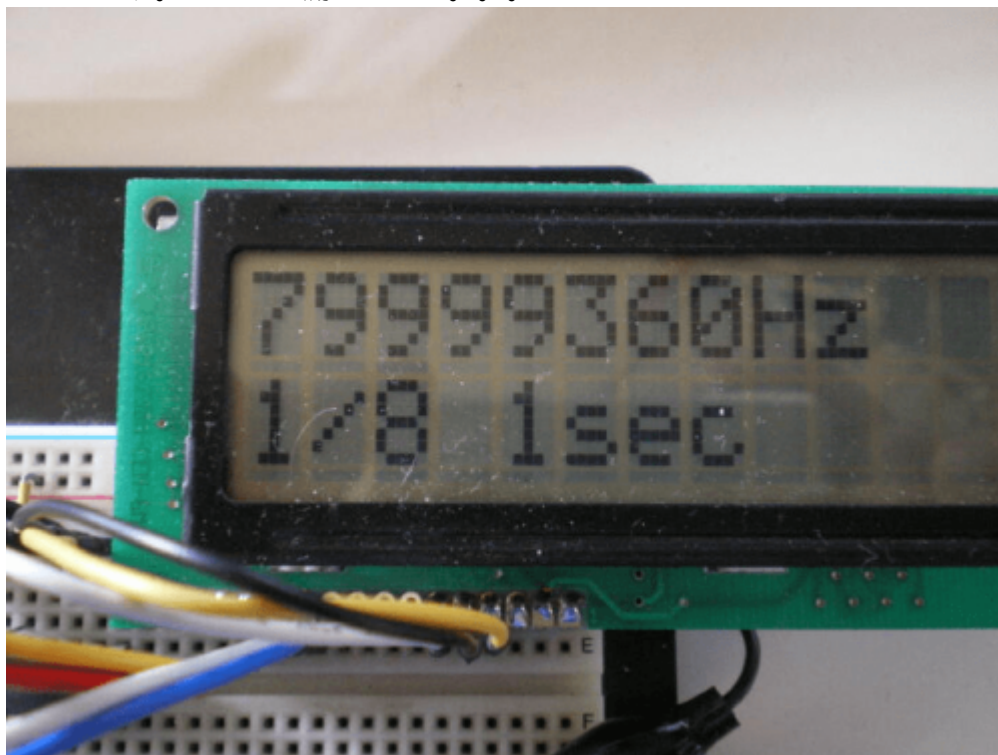
左側:60MHzを測定しました。(プリスケアラ無し、ゲートタイム1秒)測定不能 右側:60MHzを測定しました。(プリスケアラ有り、ゲートタイム1秒)測定可能





70MHzです。

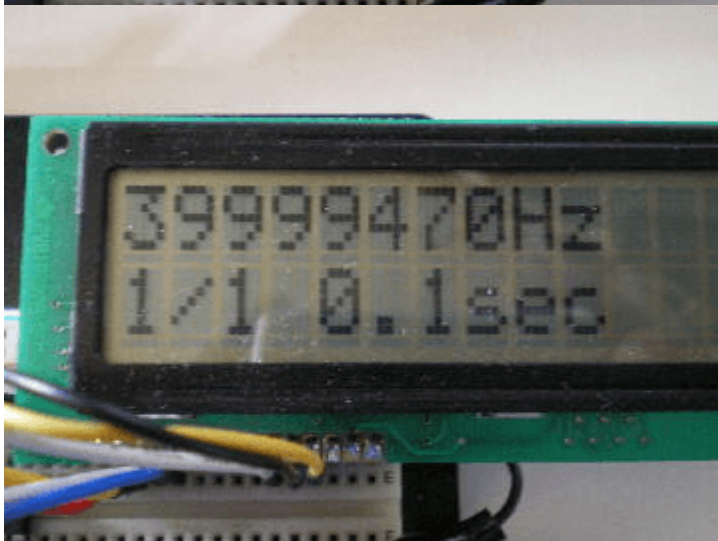
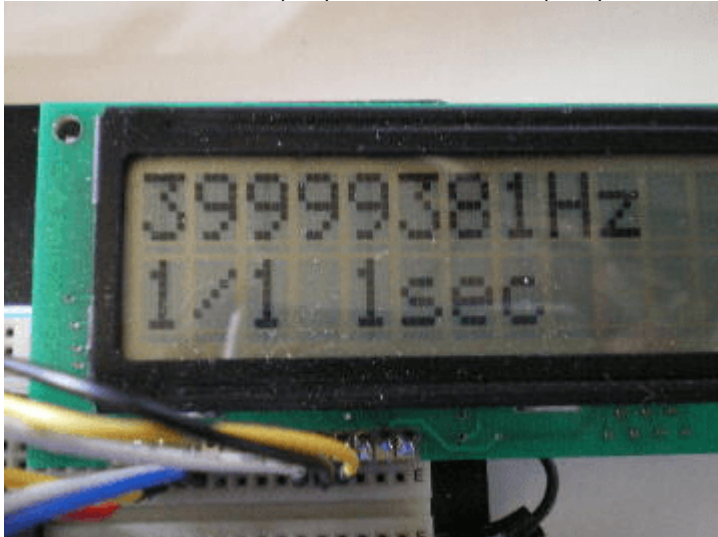
80MHzです。ここまで測れるとは。。。!?



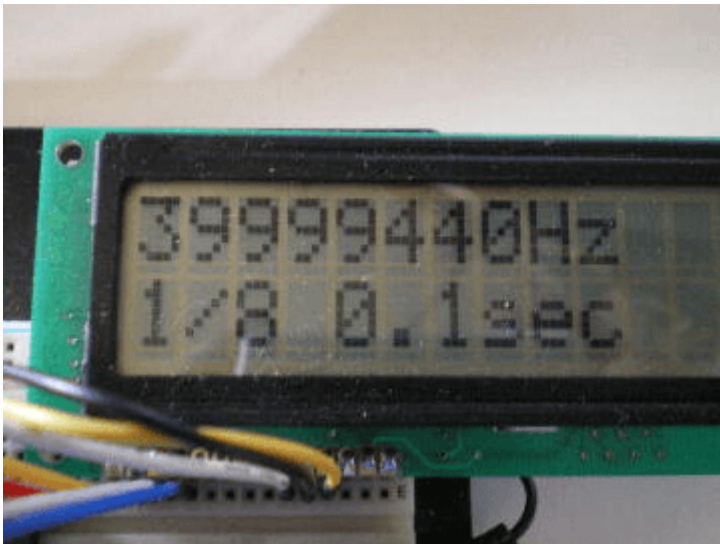
測定したクロックモジュールです。



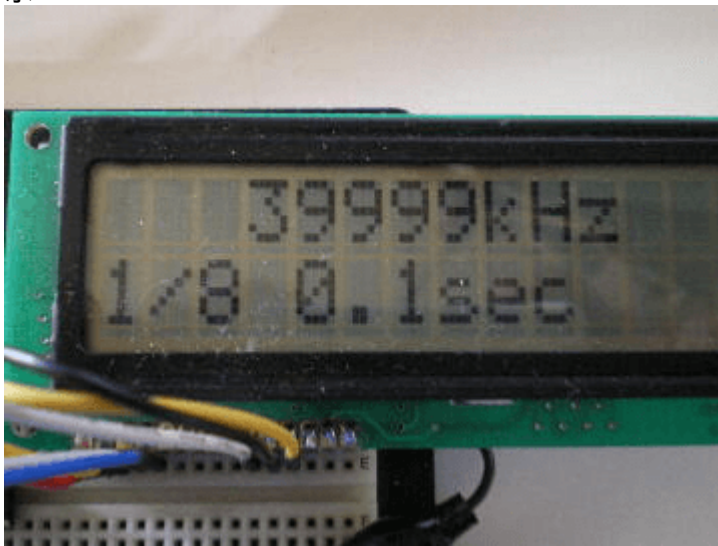
左側:プリスケーラ値(1/1)、ゲートタイム(1秒) 右側:プリスケーラ値(1/1)、ゲートタイム(0.1秒)



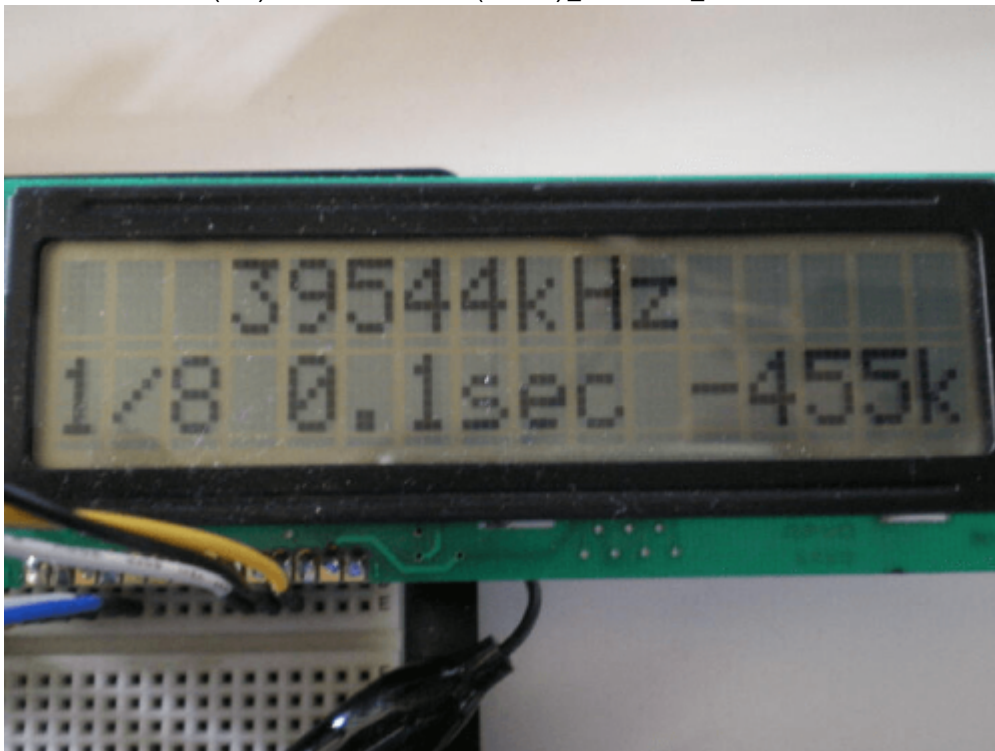
左側:プリスケーラ値(1/8)、ゲートタイム(0.1秒) 右側:プリスケーラ値(1/1)、ゲートタイム(0.1秒) kHz表



示



プリスケーラ値(1/8)、ゲートタイム(0.1秒) kHz表示 -455kHz



如何ですか? 少し視点を変えるだけで、大きな回路変更もなく 80MHzの周波数が測定できました。



著作権表示 **copyright notice**

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。[詳細](#) This page is a copy of Mr. Inasaki's closed website, and the copyright is held by him.[Details](#)

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:pic16f88:85&rev=1588324663>

Last update: **2025/10/17 14:28**

