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。

- プリスケーラ無(1/1)で、40MHzまで測定可能(実測値)
- プリスケーラ有(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を組み合わせて使います。

- 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

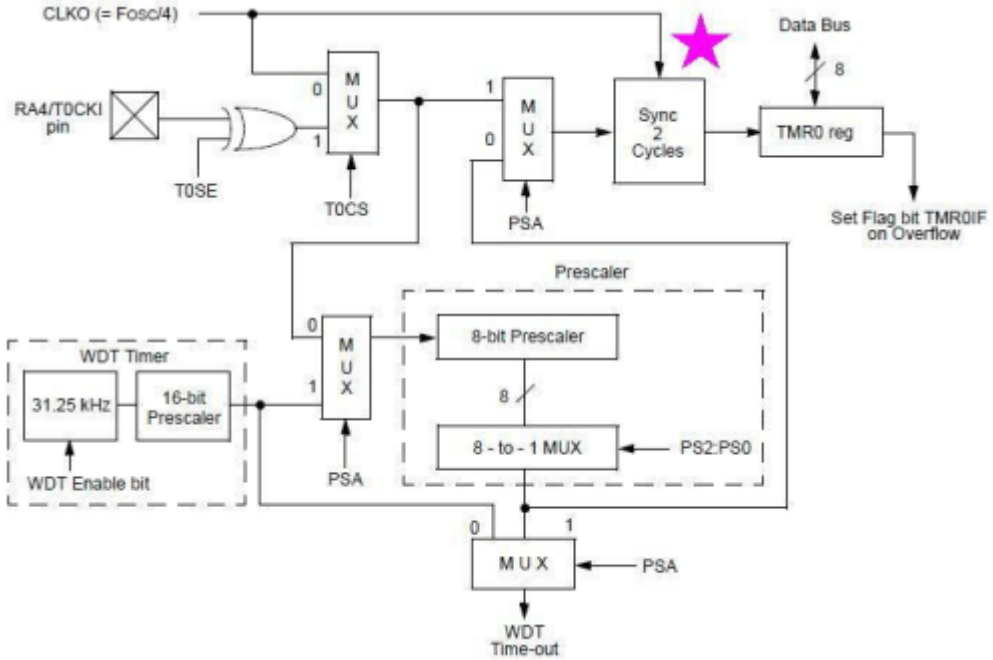
しかしTIMER0は、TIMERのON/OFF制御が出来ないため、入力ピンを強制的に出力モードにして“0“を出力する等の工夫(本来は余分な事)が必要となります。

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

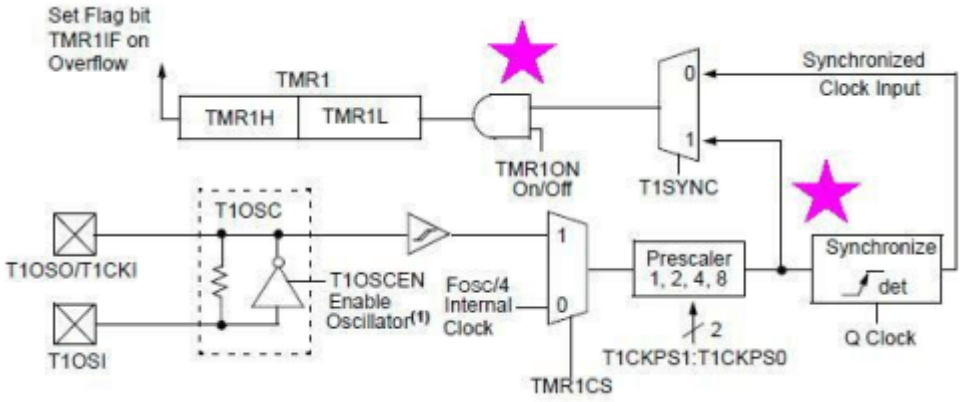
- 信号をカウントするためにTIMER1(16ビット)を使用(外部クロック入力同期制御ON/OFF可能)
- ゲートタイム(1秒、0.1秒)を得るためにTIMER2(8ビット)を使用

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のプリスケー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。

<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。



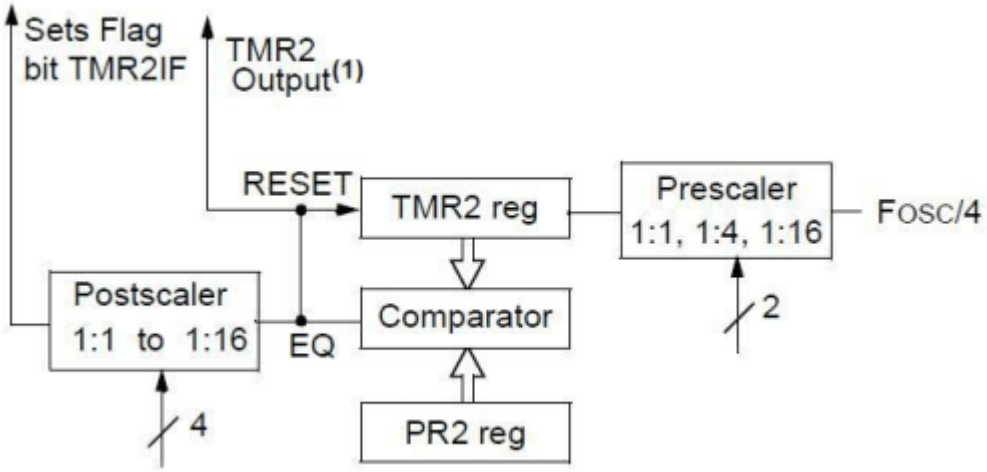
<TIMER1> 外部クロック入力同期(Synchronize)をON/OFFすることが可能 TMR1ONでTIMERを開始/停止することが可能。



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	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							

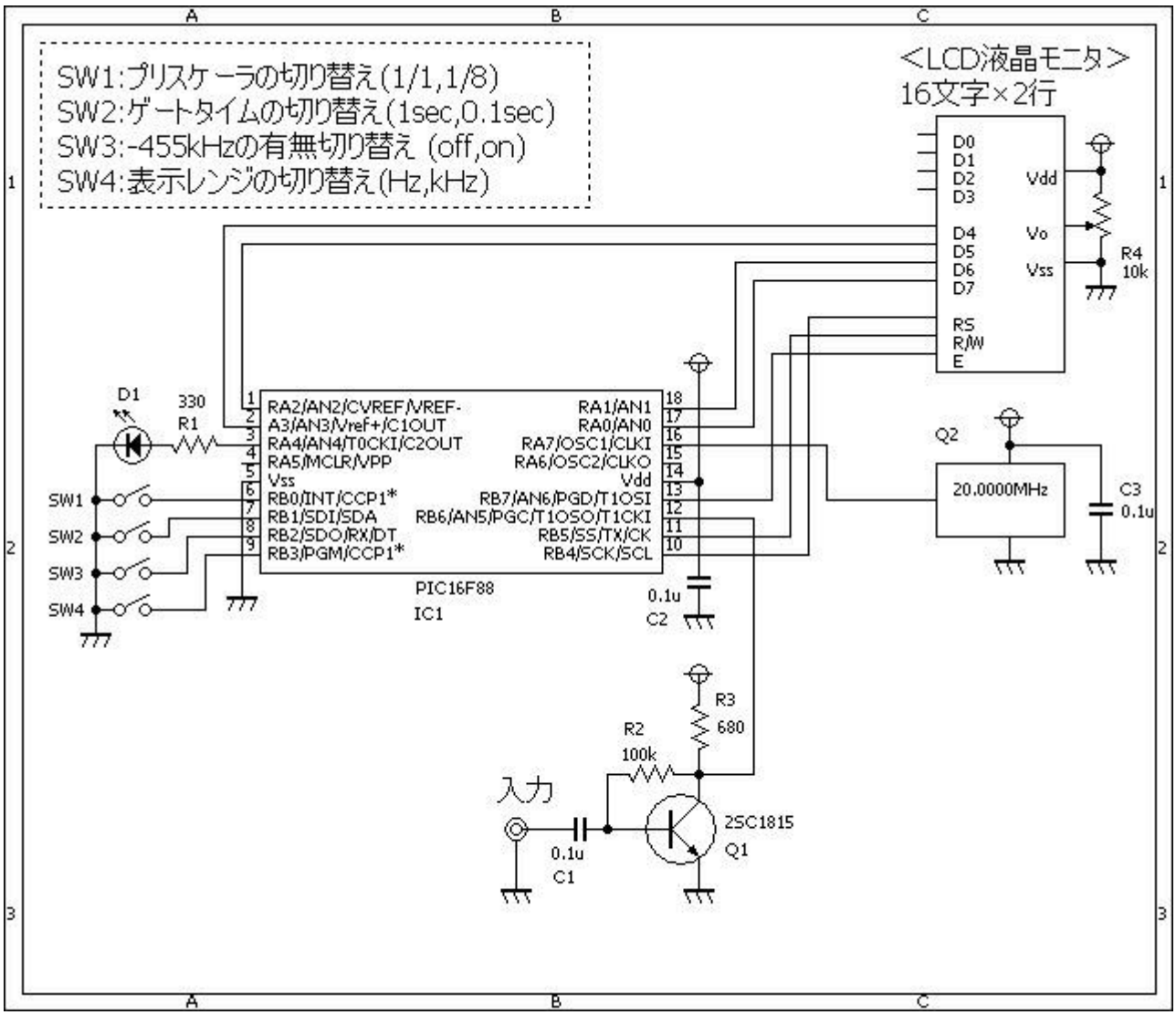
<TIMER2> TMR2ONでTIMERを開始/停止することが可能。



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=0□kHz表示
  ■コンフィグ設定
□□LVP_OFF
□□MCLR_OFF
□□WDT_OFF
□□EXTCLK
  ■ピンアサイン
□□Pin-01□LCD: D5
□□Pin-02□LCD: D4
□□Pin-03□LED
□□Pin-04□未使用
□□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: RS
□□Pin-11□LCD: WR
□□Pin-12□信号入力
□□Pin-13□LCD: EN
□□Pin-14□Vdd (+5V)
□□Pin-15□未使用
□□Pin-16□クロック入力(20MHz入力)
□□Pin-17□LCD: D7
□□Pin-18□LCD: D6
*/
//*****************************************************************************
*

#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。
```

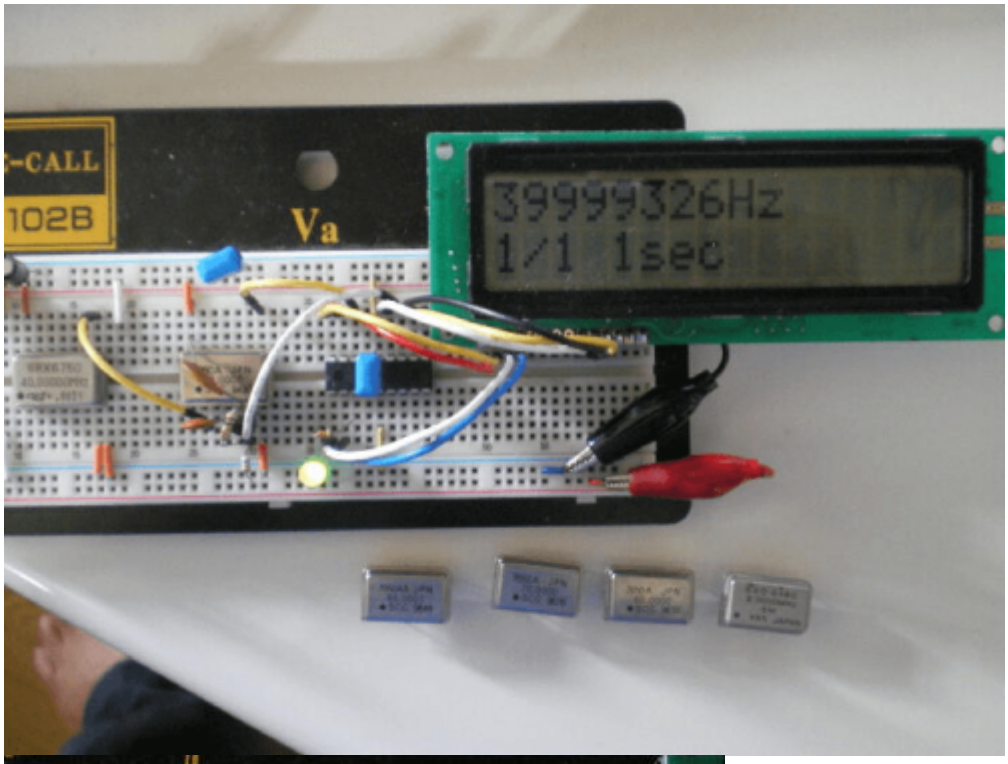
```
        T2CON.TMR2ON = 0;     // TIMER2を停止する。
    }
}

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

unsigned   long    FreqMeasurement(unsigned char gateTime)
{
    static    unsigned    long    freq;
    // TIMER1の設定
    PIR1.TMR1IF = 0;
    TMR1L = 0;
    TMR1H = 0;
    // TIMER2の設定
    PIR1.TMR2IF = 0;
    switch (gateTime) {
    case GATETIME_1SEC:
        MeasurementCnt = 1221;
        TMR2 = 0x4C;         // 312500=1/((1/20000000) * 4 * 16)
                             // 0x4C=256-(312500-(256*1220))
        break;
    case GATETIME_100MSEC:
        MeasurementCnt = 123;
        TMR2 = 0xEE;         // 31250=0.1/((1/20000000) * 4 * 16)
                             // 0xEE=256-(31250-(256*122))
        break;
    }
    //
    freq = 0;
    //割り込みを許可する。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //開始
    T2CON.TMR2ON = 1;     //タイマを開始する。
    //    Delay
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
```

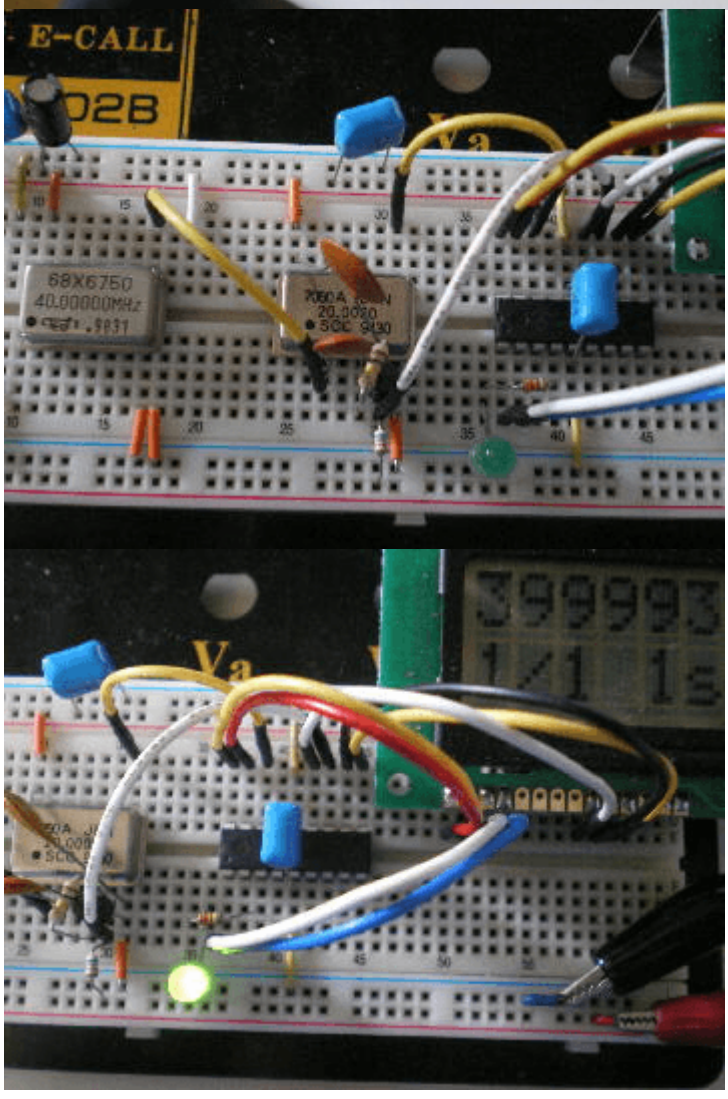
```
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    asm    nop;
    //
    T1CON.TMR1ON = 1;     //ゲートを開ける。
    //測定
    while (T2CON.TMR2ON != 0) {
        if (PIR1.TMR1IF == 1) {
            PIR1.TMR1IF = 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.T1OSCEN = 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 {
```

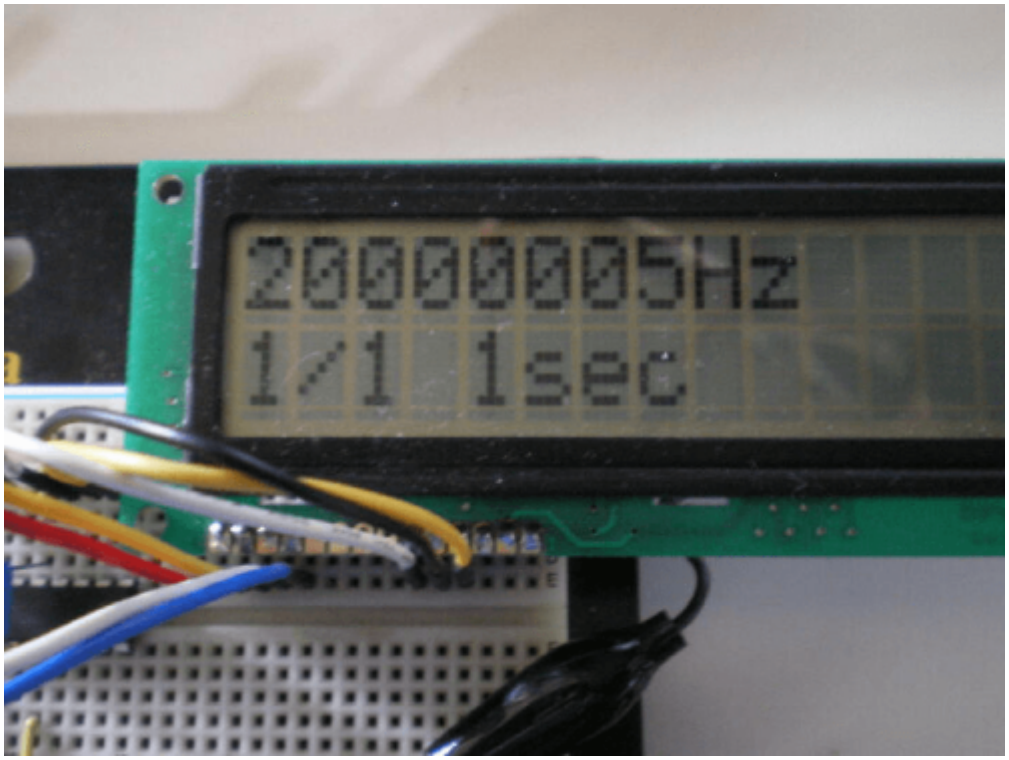
```
            gateTime = GATETIME_100MSEC;
            msg = "0.1sec ";
        }
        Lcd_Custom_Out(2, 5, msg);
        // □□□□□Hzの有無
        if (sw3 == 0) {
            freq -= 455000;
            msg = "-455k";
        } else {
            msg = "     ";
        }
        Lcd_Custom_Out(2, 12, 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

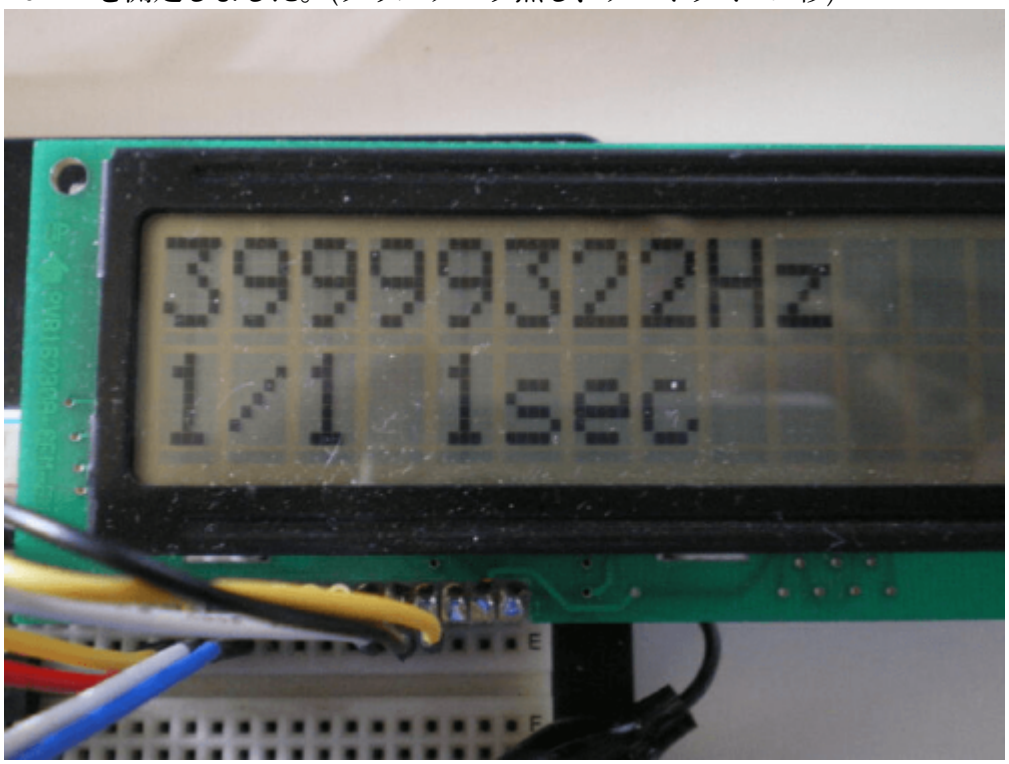




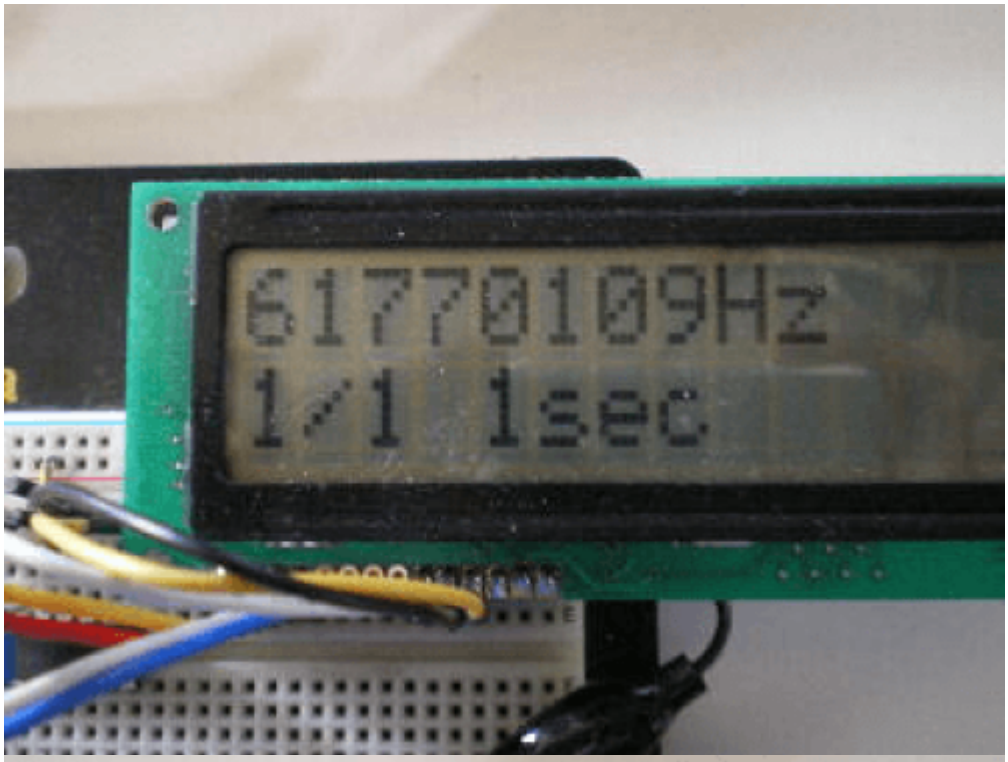
20MHzを測定しました。(プリスケーラ無し、ゲートタイム1秒)

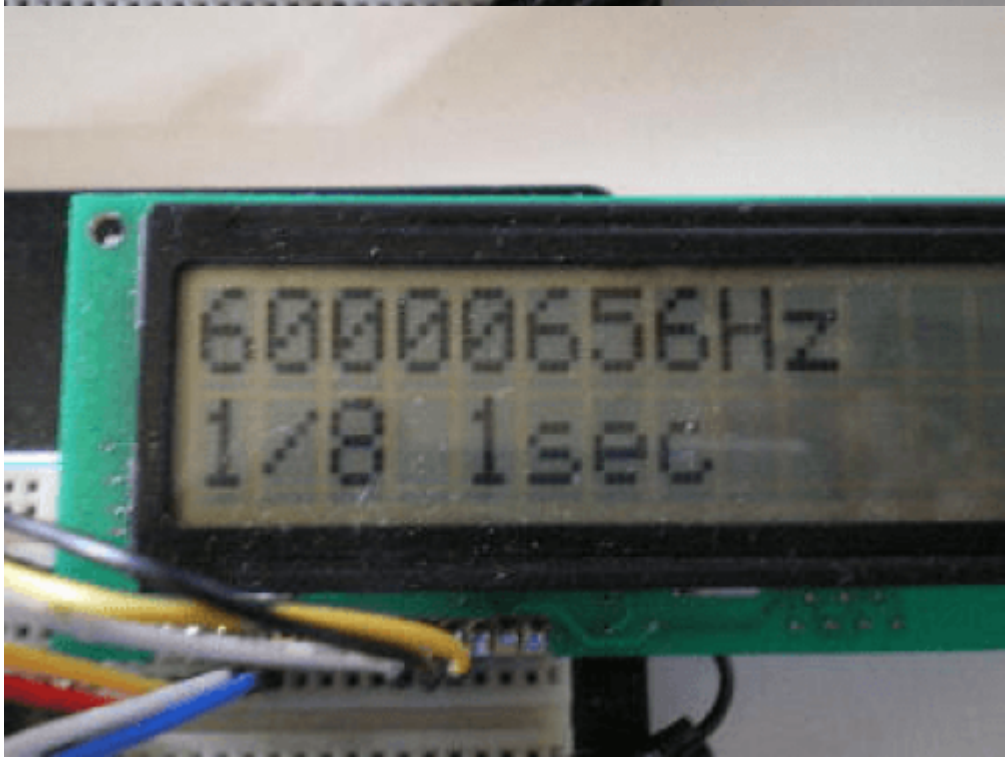


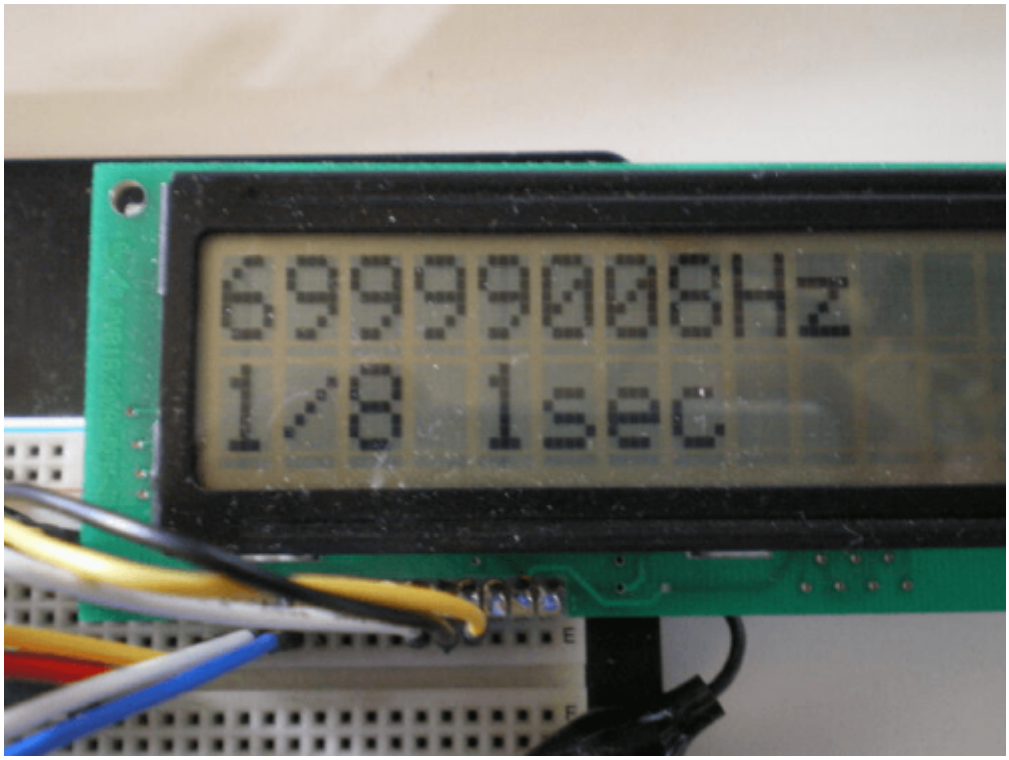
40MHzを測定しました。(プリスケーラ無し、ゲートタイム1秒)



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

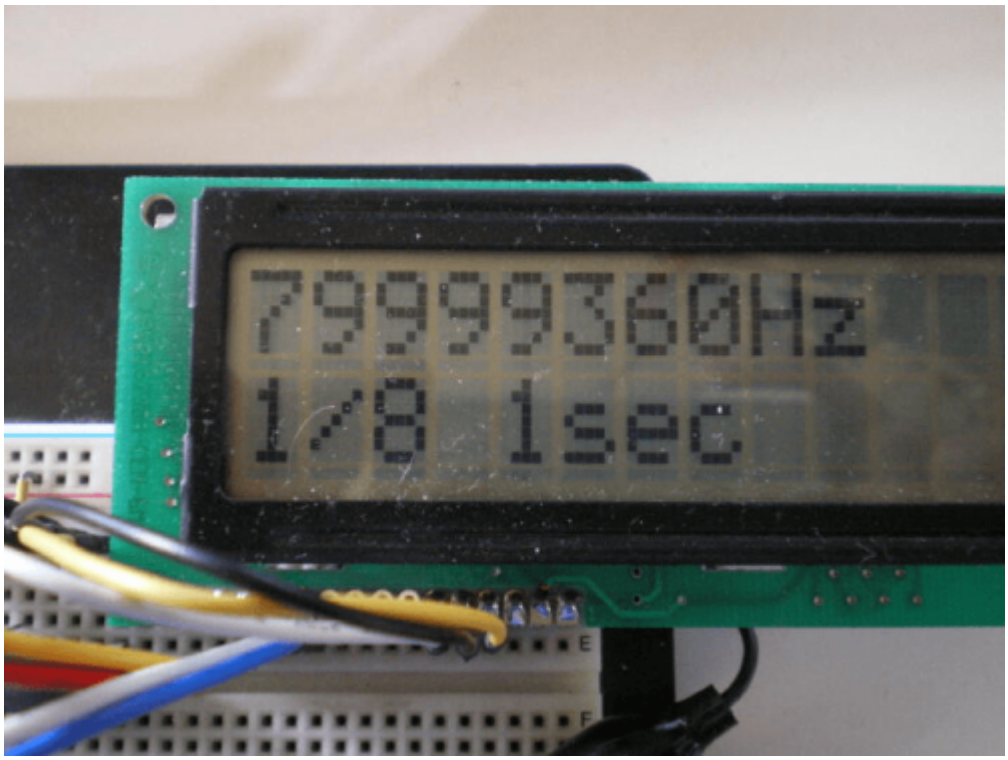

61770109Hz
1/1 1sec


60000656Hz
1/8 1sec



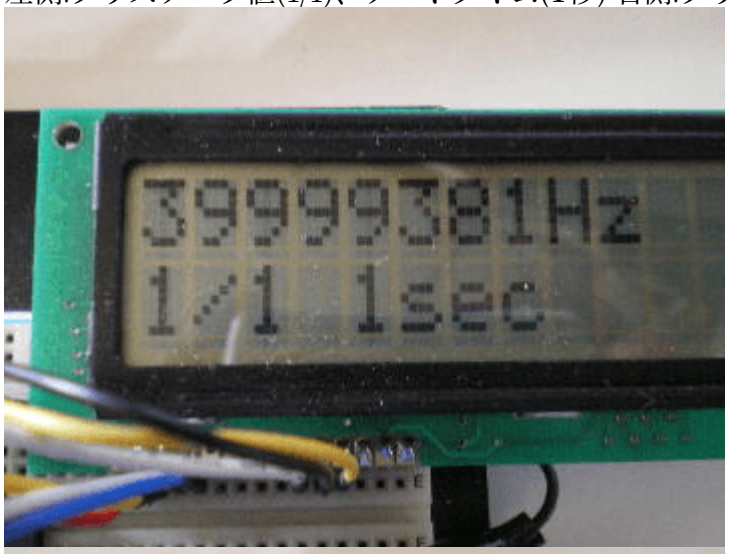
70MHzです。

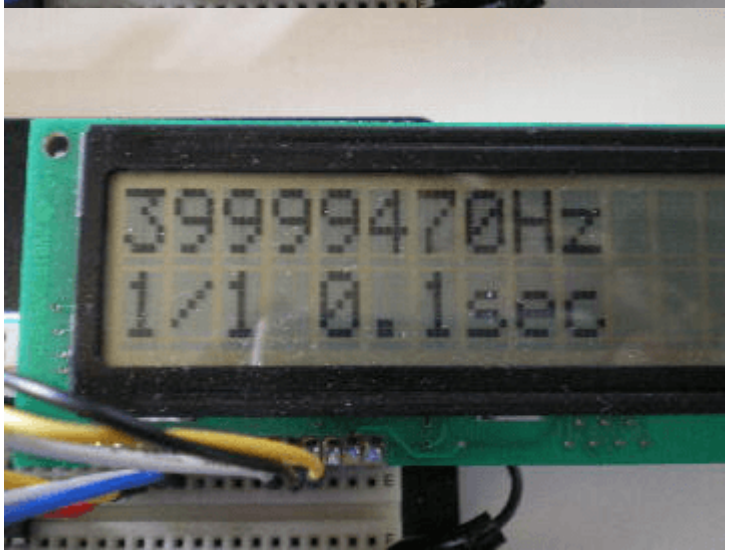
80MHzです。ここまで測れるとは。。。!?



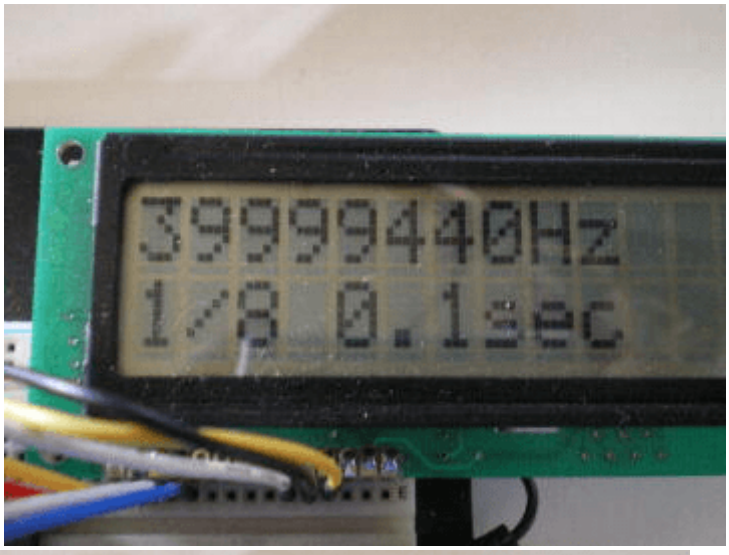
測定したクロックモジュールです。

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)

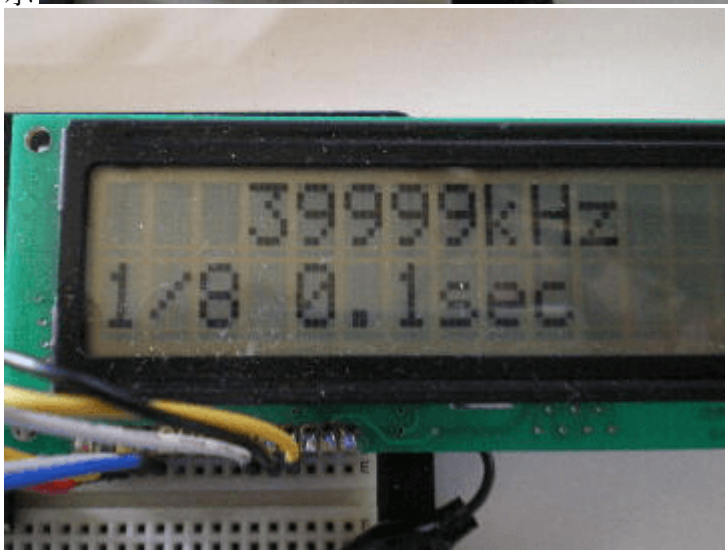




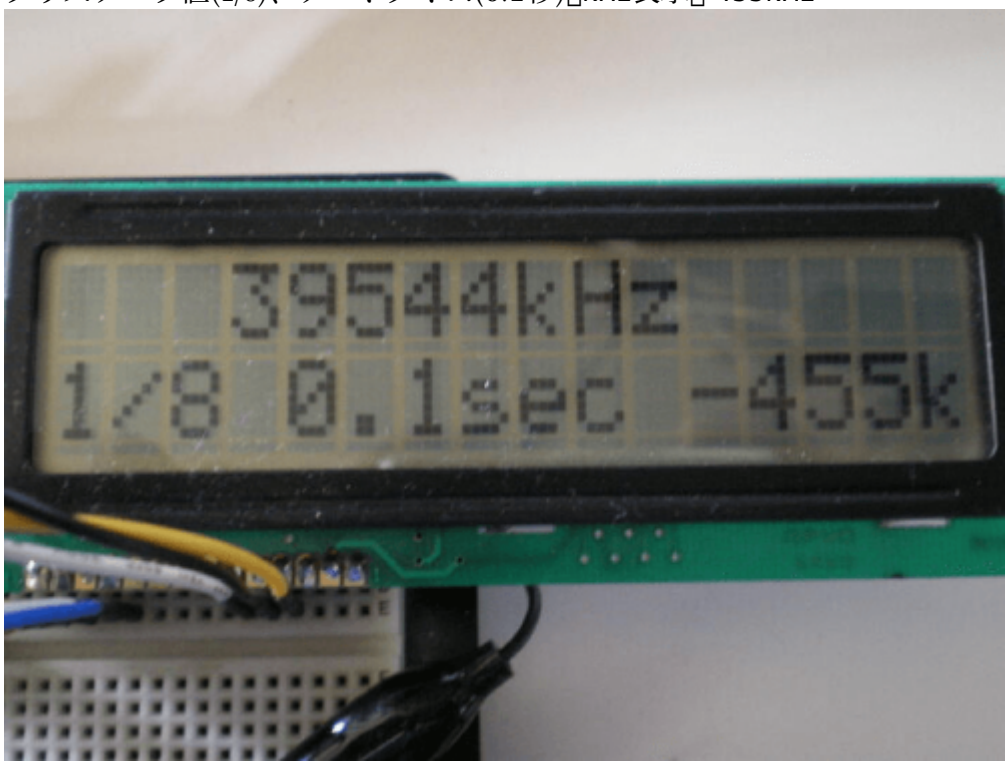
左側:**プリスケーラ**値(1/8)、**ゲートタイム(0.1**秒) 右側:**プリスケーラ**値(1/1)、**ゲートタイム(0.1**秒)□kHz表



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**。