周波数カウンタV9(PIC18シリーズ最小)(PIC18F1320)

概要

PIC16Fシリーズには、16ビットのタイマーモジュールが、1個しかなく、また、タイマーの制御(ON/OFF)が貧弱なため、周波数カウンタのように、

- ゲートタイム用のタイマーモジュール(ON/OFF制御)
- 周波数カウント用のタイマーモジュール(ON/OFF制御)

を実現しようとすると、結構煩雑(意にそぐわない)な処理が必要となります。

PIC18Fシリーズでは、16ビットのタイマーモジュールが、2個に強化され、且つ、タイマーの制御(ON/OFF)も可能となり、周波数カウンタの実現には、とても便利になりました。

今回は、PIC18Fシリーズでは、最もピン数の少ないPIC18F1320(18ピン)を使用し、出来るだけコンパクトに仕上げてみました。

<仕様>

- ゲートタイム切り替え(1秒/0.1秒)
- -455kHz切り替え(有/無)
- 表示切替(Hz/kHz)
- 測定範囲(10Hz~60MHz)

動作原理

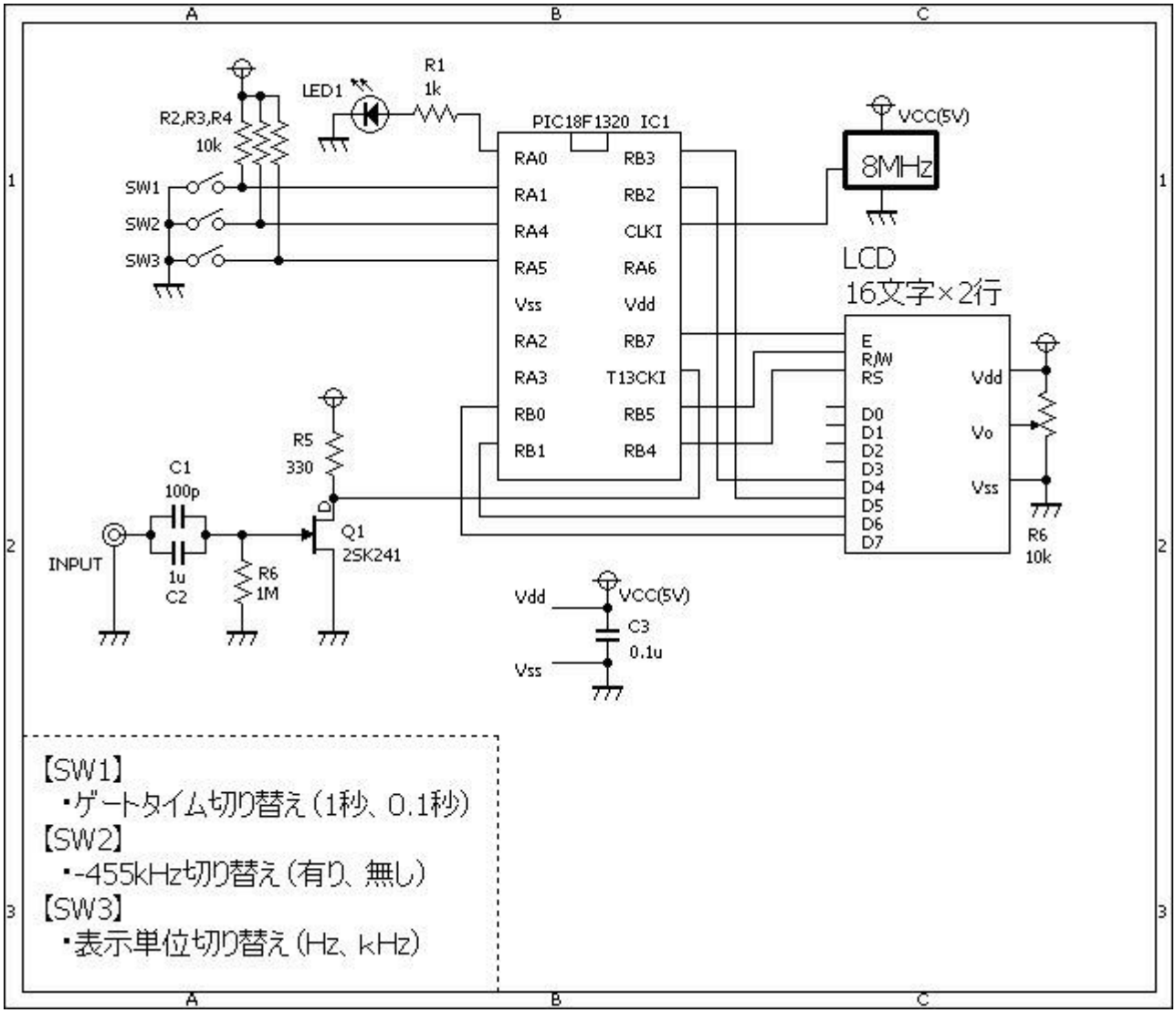
<ゲートタイム> ECCPモジュールとTIMER1を組み合わせ、コンペアモードで使用することにより、0.1秒の周期割り込みを発生させる。 割り込みを、フリーランさせることにより、測定の開始と完了の処理を単純化させる。

- 測定の開始(測定処理→割り込み処理へ指示)
- 測定の完了(割り込み処理処理→測定処理へ指示)

<周波数カウント> TIMER3と32ビットの変数(freq)を組み合わせ、周波数をカウントするTIMER3でオーバーフローが発生する都度に、freq変数をインクリメント(+1)する。測定完了後に次式により周波数を求める。

```
freq=(freq×65536)+(TMR3H×256)+TMR3L
```

回路図



ソースコード

FreqCounterV9.c

```
//*****************************************************************************
*
/*
    ＜周波数カウンタ□□□
*/
//*****************************************************************************
*

#define       LED                           PORTA.F0

#define       GATE_TIME_1SEC               10
#define       GATE_TIME_100MSEC           1

#define       SW1                           PORTA.F1
#define       SW2                           PORTA.F4
#define       SW3                           PORTA.F5
```

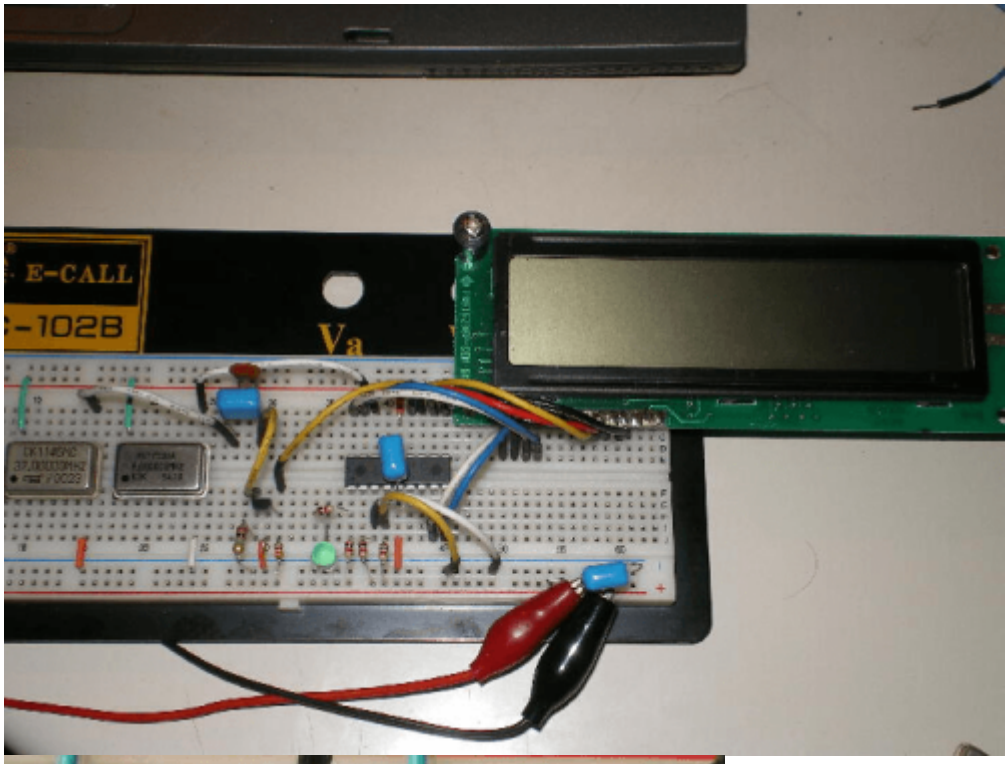
```
//*********************************************************************
*

short  fc_flg, gateTime;

void   interrupt()
{
    if (PIR1.CCP1IF == 1) {
        PIR1.CCP1IF = 0;
        //
        //周波数カウンタ処理
        switch (fc_flg) {
        case -1:
            break;
        case 0:
            T3CON.TMR3ON = 1;     //ゲートを開ける。
            fc_flg++;
            LED = 1;
            break;
        default:
            if (gateTime == fc_flg) {
                T3CON.TMR3ON = 0;     //ゲートを閉める。
                fc_flg = -1;
                LED = 0;
            } else {
                fc_flg++;
            }
            break;
        }
    }
}

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

unsigned   long    FreqMeasurement()
{
    static    unsigned    long    freq;
    //
    T3CON.TMR3ON = 0;     //ゲートを閉める。
    PIR2.TMR3IF = 0;
    TMR3L = 0;
    TMR3H = 0;
    freq = 0;
    //
    fc_flg = 0;             //測定開始
    //測定
    while (fc_flg != -1) {
        if (PIR2.TMR3IF == 1) {
            PIR2.TMR3IF = 0;
```

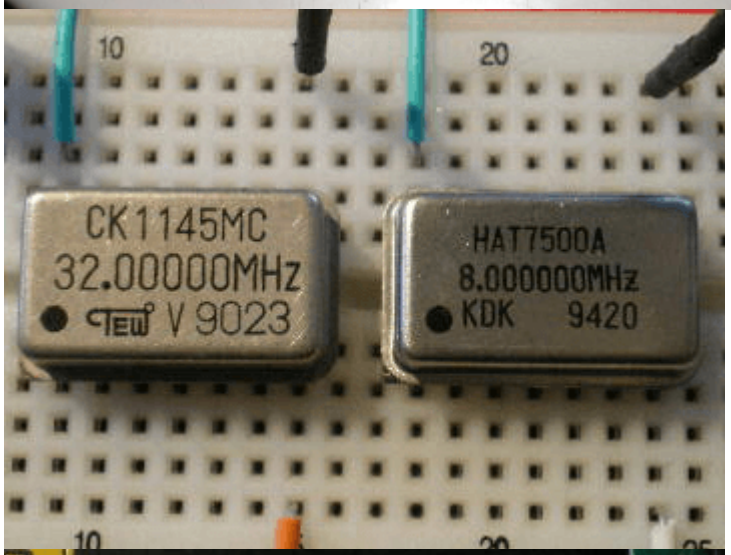
```
            freq++;
        }
    }
    if (PIR2.TMR3IF == 1) {
        PIR2.TMR3IF = 0;
        freq++;
    }
    freq = (freq * 65536) + (unsigned long)((TMR3H << 8) | TMR3L);
    return (freq);
}

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

void   main()
{
    static    unsigned    char    buf[20], cnt;
    static    unsigned    long    freq, temp;
    //
//    OSCCON.IRCF2 = 1;
//    OSCCON.IRCF1 = 1;
//    OSCCON.IRCF0 = 1;
    ADCON1 = 0b11111111;
    TRISA = 0b11111110;
    TRISB = 0b01000000;
    //TIMER3の設定
    PIE2.TMR3IE = 0;
    PIR2.TMR3IF = 0;
    T3CON.RD16 = 0;
    T3CON.T3CKPS0 = 0;
    T3CON.T3CKPS1 = 0;
    T3CON.T3CCP1 = 0;
    T3CON.NOT_T3SYNC = 1;
    T3CON.TMR3CS = 1;
    T3CON.TMR3ON = 0;
    TMR3L = 0;
    TMR3H = 0;
    //TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1CKPS1 = 1;
    T1CON.NOT_T3SYNC = 1;
    T1CON.TMR1CS = 0;
    T1CON.TMR1ON = 0;
    TMR1L = 0;
    TMR1H = 0;
    //CCPの設定
    PIE1.CCP1IE = 1;
    PIR1.CCP1IF = 0;
    CCP1CON.CCP1M3 = 1;
```

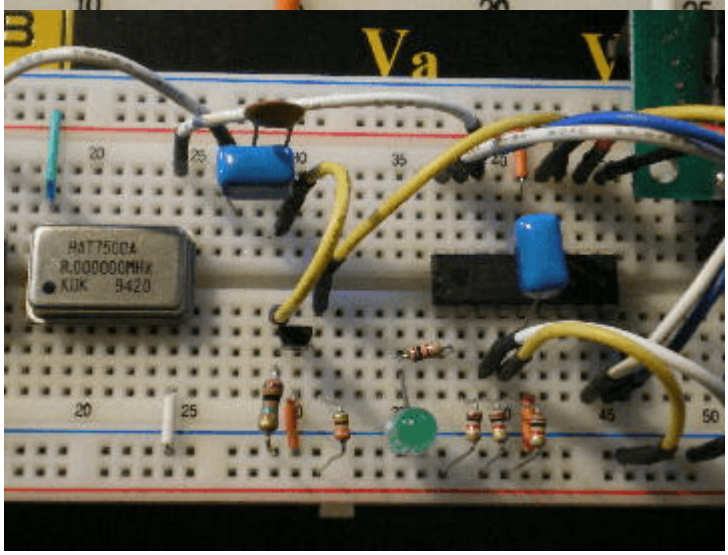
```
    CCP1CON.CCP1M2 = 0;
    CCP1CON.CCP1M1 = 1;
    CCP1CON.CCP1M0 = 1;
    CCPR1L = 0xA8;    // 0.1sec...(1÷8000000)*4*8*25000
    CCPR1H = 0x61;    //
    //LCDの設定
    Lcd_Custom_Config(&PORTB,0,1,3,2,&PORTB,4,5,7);
    Lcd_Custom_Cmd(Lcd_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    Lcd_Custom_Out(1, 1, "FreqCounterV9");
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(2, cnt + 1, 0x00);
        Delay_ms(100);
    }
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    fc_flg = -1;
    gateTime = GATE_TIME_1SEC;
    //割り込みを許可する。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    T1CON.TMR1ON = 1;
    //
    while (1) {
        freq = FreqMeasurement();
        if (gateTime == GATE_TIME_100MSEC)
            freq *= 10;
        //ゲートタイム切替
        if (SW1 == 1) {
            gateTime = GATE_TIME_1SEC;
            Lcd_Custom_Out(2, 1, "  1sec");
        } else {
            gateTime = GATE_TIME_100MSEC;
            Lcd_Custom_Out(2, 1, "0.1sec");
        }
        //□□□□□□□切替
        if (SW2 == 0) {
            freq = freq - 455000;
            Lcd_Custom_Out(2, 8, "-455kHz");
        } else {
            Lcd_Custom_Out(2, 8, "       ");
        }
        //□□□□□□切替
        if (SW3 == 1) {
            LongToStr(freq, buf);
            buf[16] = 0x00;
            buf[15] = ' ';
            buf[14] = 'z';
            buf[13] = 'H';
            buf[12] = buf[10];
```

```
            buf[11] = buf[9];
            buf[10] = buf[8];
            buf[9] = buf[7] != ' ' ? ',': ' ';
            buf[8] = buf[7];
            buf[7] = buf[6];
            buf[6] = buf[5];
            buf[5] = buf[4] != ' ' ? ',': ' ';
            Lcd_Custom_Out(1, 1, &buf[2]);
        } else {
            temp = freq / 1000;
            if ((freq - (temp * 1000)) >= 500)
                temp++;
            LongToStr(temp, buf);
            buf[16] = 0x00;
            buf[15] = 'z';
            buf[14] = 'H';
            buf[13] = 'k';
            buf[12] = buf[10];
            buf[11] = buf[9];
            buf[10] = buf[8];
            buf[9] = buf[7] != ' ' ? ',': ' ';
            buf[8] = buf[7];
            buf[7] = buf[6];
            buf[6] = buf[5];
            buf[5] = buf[4] != ' ' ? ',': ' ';
            Lcd_Custom_Out(1, 1, &buf[2]);
        }
    }
}

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

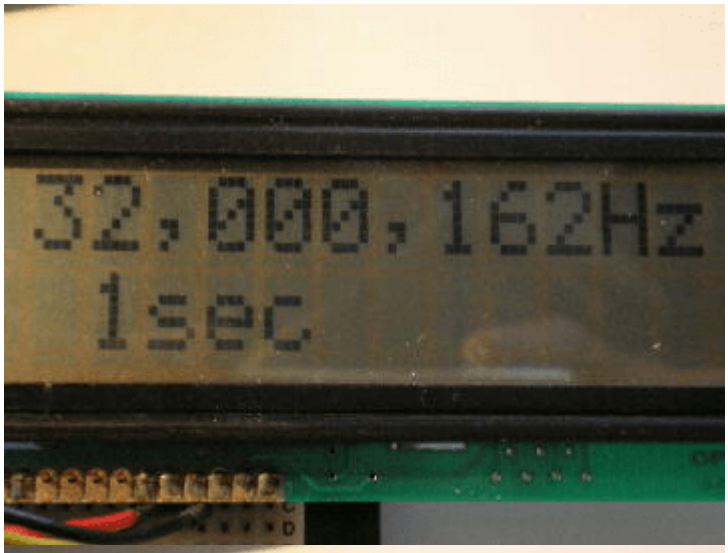
動作確認



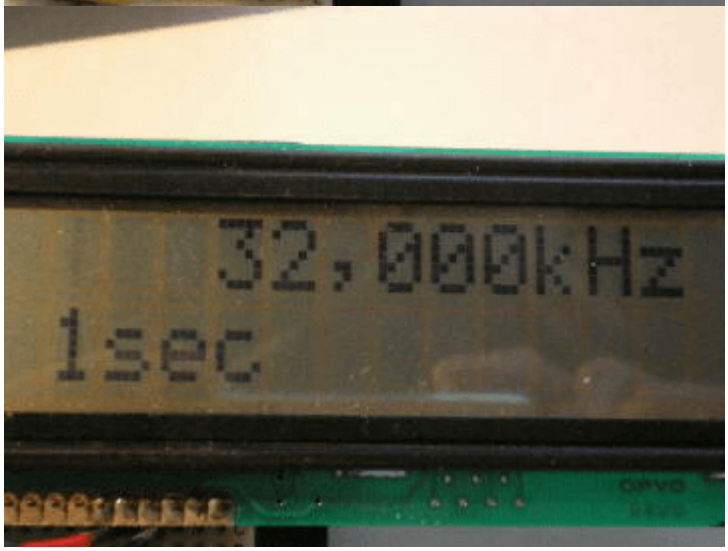


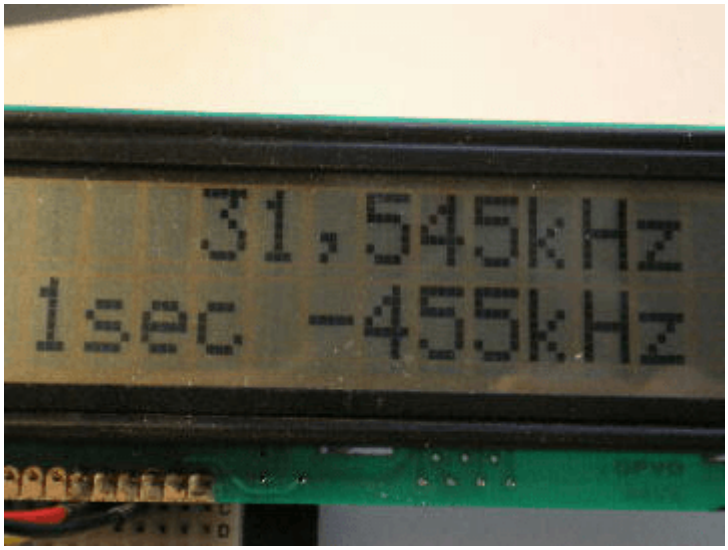


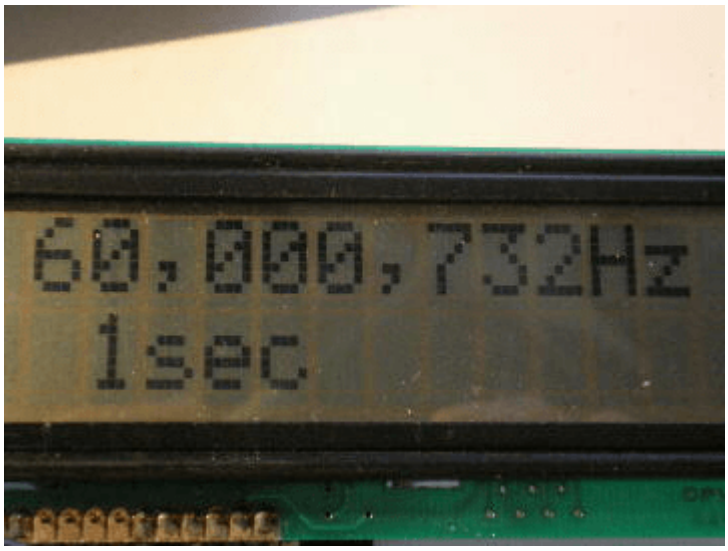
32MHzを測定してみました。 左上:**ゲートタイム=1**秒、Hz表示 右上:**ゲートタイム=0.1**秒、Hz表示 左下:**ゲートタイム=1**秒、kHz表示 右下:**ゲートタイム=1**秒、kHz表示-455kHz


32,000,162Hz
1sec

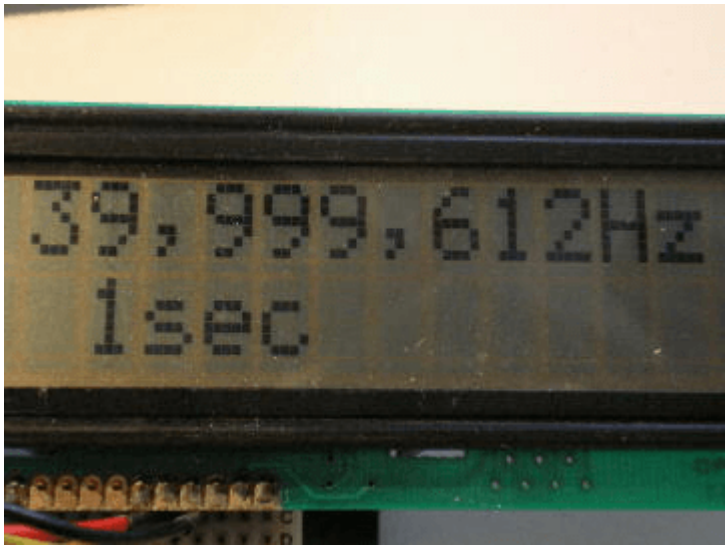

32,000,650Hz
0.1sec

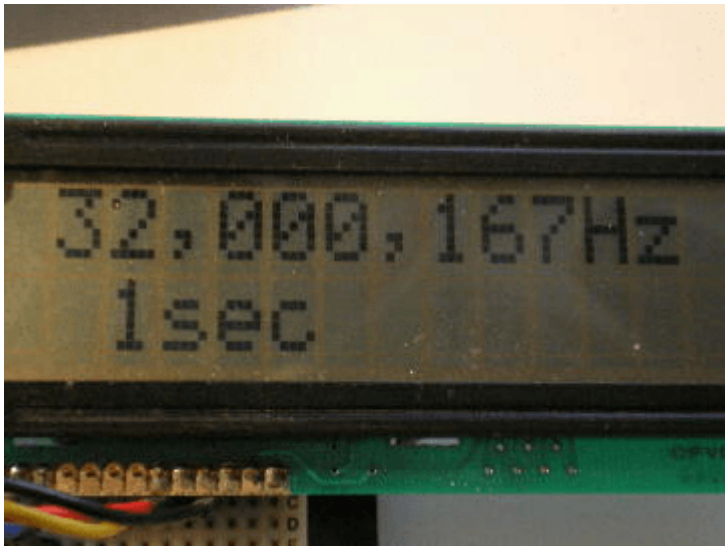

32,000kHz
1sec



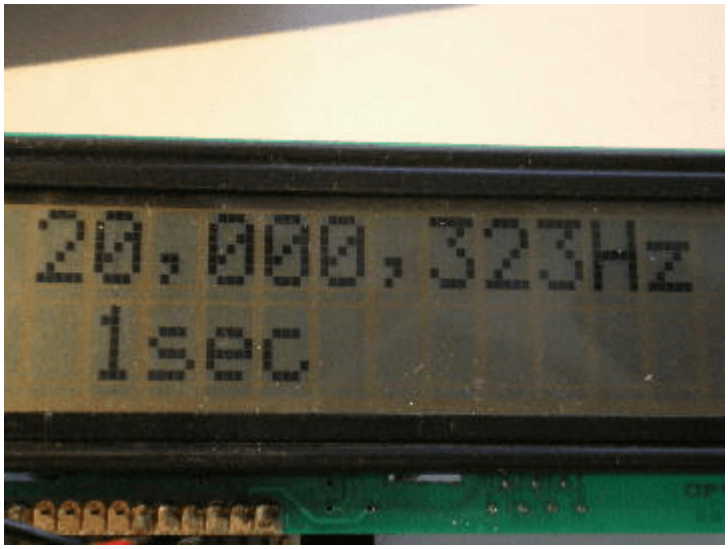


左側:60MHz 右側:40MHz

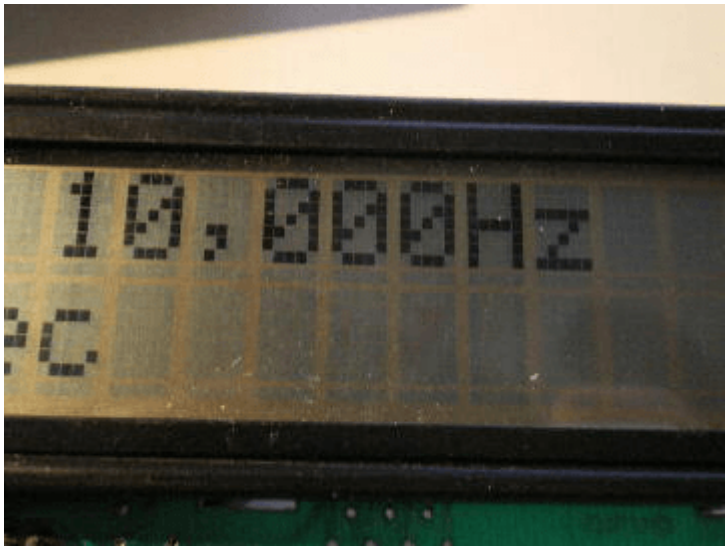


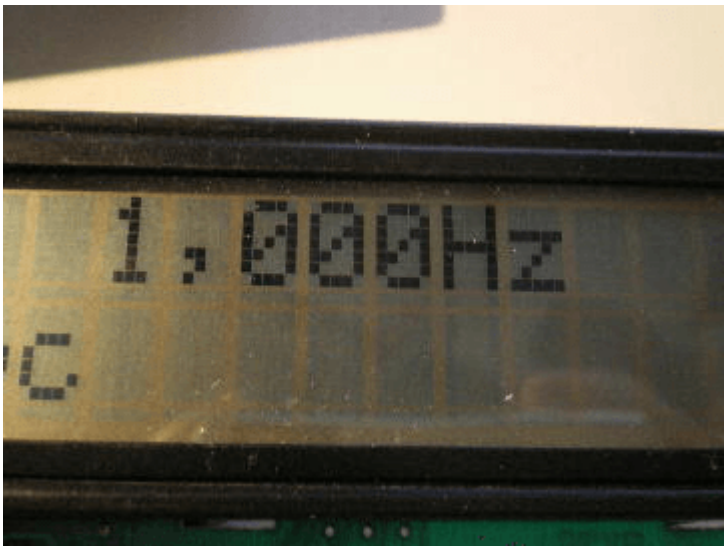


左側:32MHz 右側:20MHz

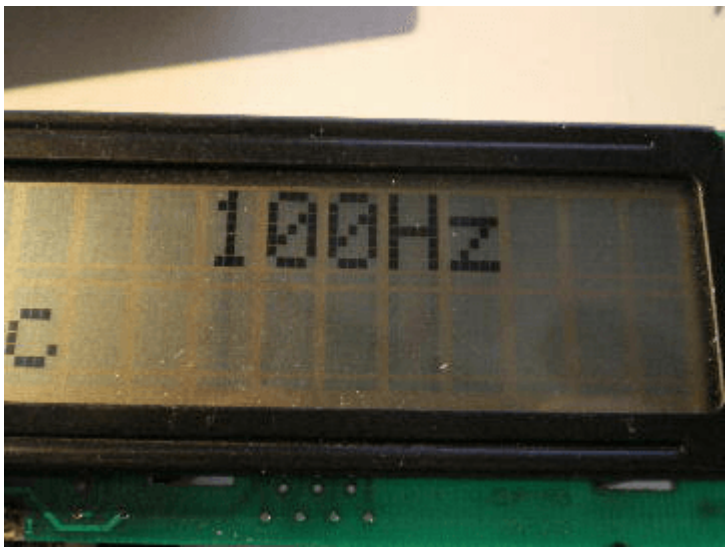


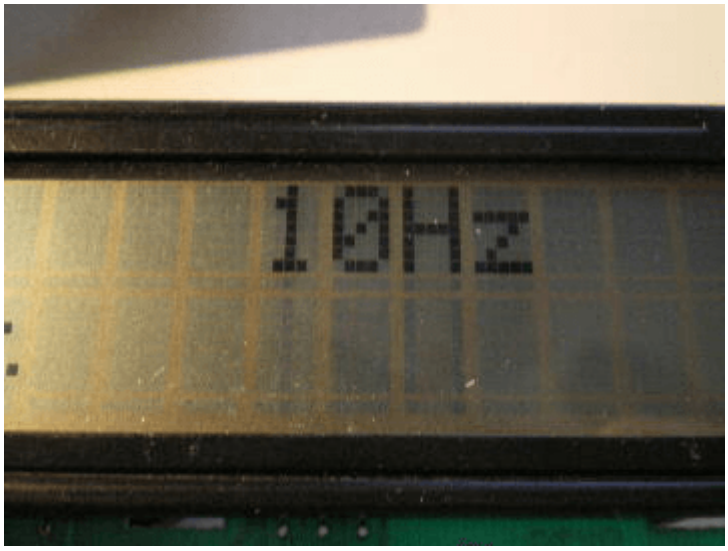
左側:100kHz 右側:10kHz





左側:1kHz 右側:100Hz





左側:10Hz 右側:5Hz

如何ですか? ハードもソフトも、かなりシンプルに仕上がりましたね。