周波数カウンタV8(ロガー機能搭載)(PIC18F2550)

概要

今まで製作した、周波数カウンタは、測定結果をLCDに表示させるだけでしたが、今回は測定結果を、LCDに 表示させるとともに、SDカードにも、ファイルとして保存するようにしました。 これにより、VFO等の発振器の周波数ドリフトを測定することが容易になります。

<仕様> ◆ゲートタイム(1秒、0.1秒) ◆プリスケール値(1/1、1/8) ◆表示単位(Hz、kHz) ◆測定結果表示(LCD) ◆測定結果記録(SDカード) ◆精度(1Hz、8Hz、10Hz、80Hz)

動作原理

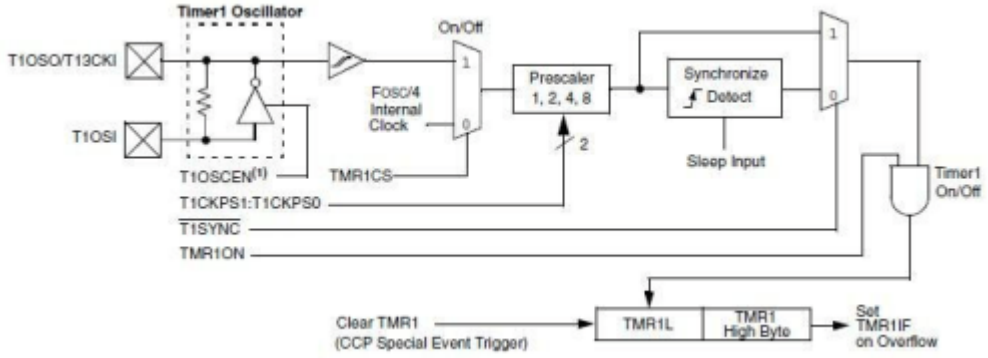
PIC16シリーズでは、16ビットTIMERが、1個だけでしたが、PIC18シリーズでは、3個に増えたので、周波数 カウンタが、作りやすくなりました。

TIMER1(16ビット)を入力周波数のカウント用に、TIMER3(16ビット)をゲート時間用に使いました。 入力周波数のカウント用には、TIMER0(16ビット)も検討してみたのですが、次の理由で使用をやめました。

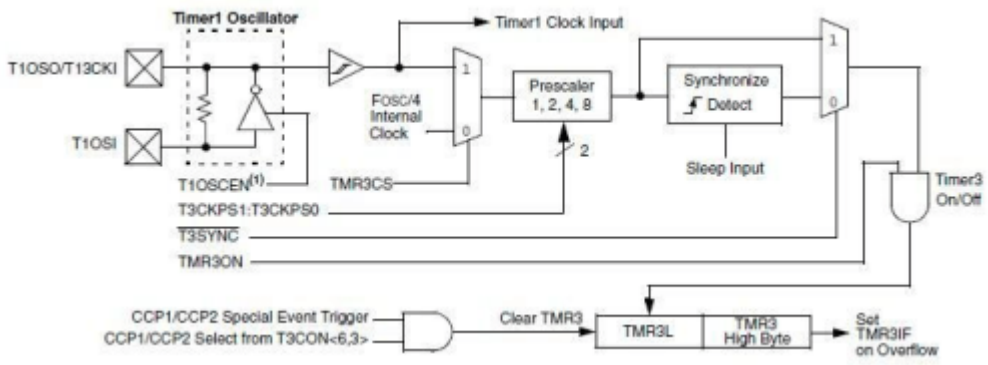
- TIMER0には、TMR1ONやTMR3ONのような、TIMER停止機能が無い。(ゲートの開閉に不向き)
- 外部クロック入力同期制御をOFFにすることが出来ない。(高い周波数測定に不向き)

TIMER1とTIMER3は、ブロックダイアグラムを見ても分かりますように、TIMER0のこれらの問題を解決することが 出来ます。両方とも殆ど同じような構造になっています。

<TIMER1のブロックダイアグラム>



<TIMER3のブロックダイアグラム>



測定した結果を、SDカードへ記録する方法は、

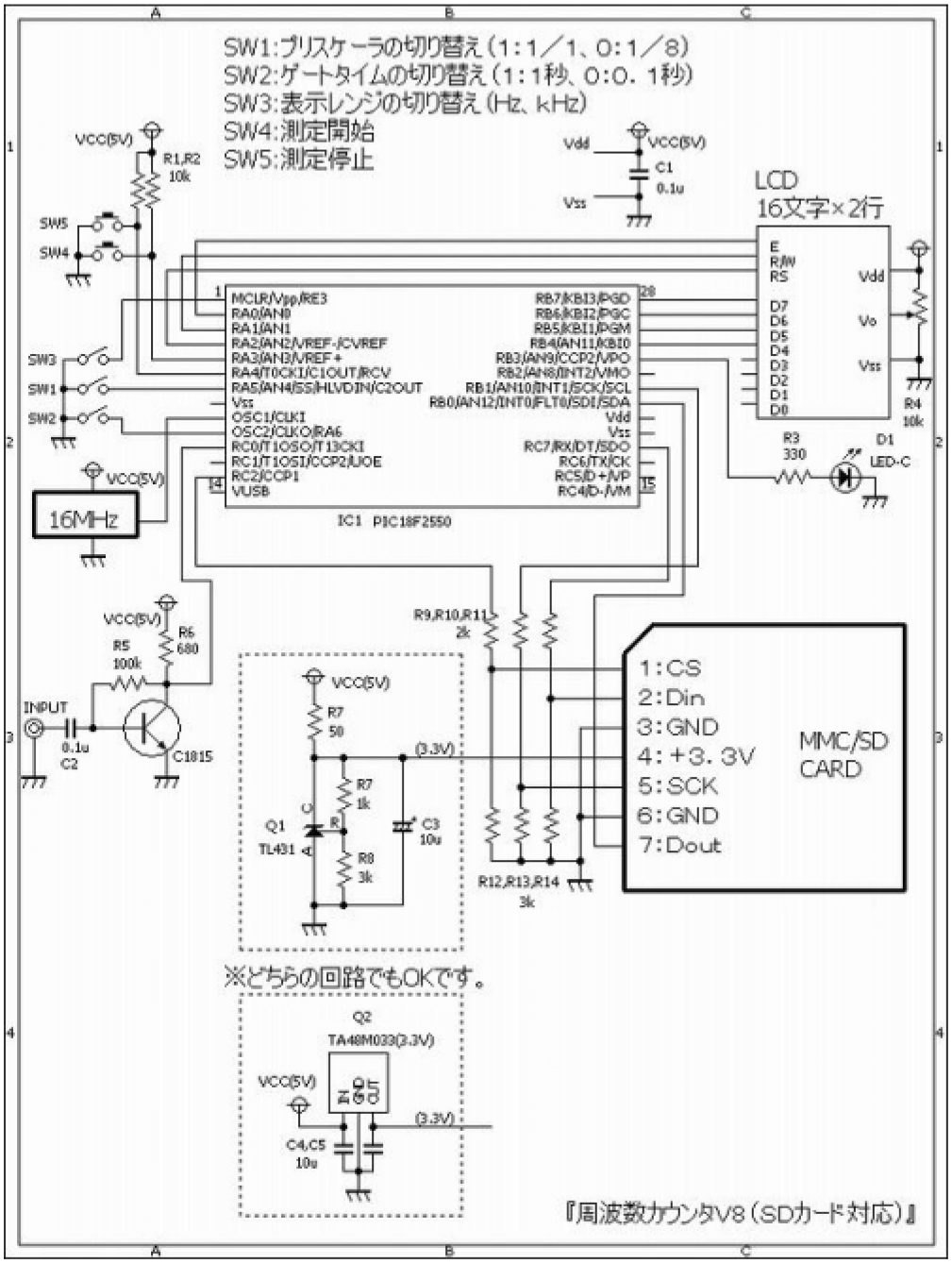
- アナログデータロガ□V3(SDカード対応)
- アナログデータロガ□V4(SDカード対応)
- SDカードReader&Writer(RS232C)

等で説明していますので、ここでは説明を省略します。

<処理の流れ>

1. ポートやLCD□MMC等を初期化する。
2. 開始スイッチが押されるのを待つ。
3. SDカードにファイル"log.csv"をリライトモードでアサインする。
4. 周波数を測定する。
5. SW1(プリスケーラの切り替え)をチェックし、1/1または1/8を設定する。
6. SW2(ゲートタイムの切り替え)をチェックし、1秒または0.1秒を設定する。
7. SW3(表示レンジの切り替え)をチェックし、HzまたはkHzを設定する。
8. 測定結果をLCDに表示する。
9. 測定結果をファイル"log.csv"に書き込む。
10. 停止スイッチが押されるまで、4.~9.の処理を繰り返す。
11. 停止スイッチが押されると、2.へ戻る。

回路図



ソースコード

FreqCounterV8.c

```
//*****************************************************************************
*
/*
        『周波数カウンタV8(SDカード対応)』
  ■概要
□□TIMER1(16ビット)をカウント用、TIMER3(16ビット)をゲート時間用
    に使用することにより、全体の処理を簡易化。
    測定結果は、LCDへの表示とSDカードへの保存(ロガー)を可能とした。
    これにより、VFO等の発振器の周波数ドリフトを測定することが容易となる。
  ■仕様
```

```
    ・ゲートタイム：1秒、0．1秒
    ・プリスケール：1／1、1／8
    ・表示単位□□□□□□□
*/
//*****************************************************************
*

#define       LED                     PORTB.F3

#define       SW_START            PORTA.F3    //測定開始
#define       SW_STOP                 PORTA.F4    //測定停止

#define       SW1                     PORTA.F5    //プリスケーラの切り替え
#define       SW2                     PORTA.F6    //ゲートタイムの切り替え
#define       SW3                     PORTE.F3    //表示レンジの切り替え

#define       ON                      1
#define       OFF                      0

#define       CR                      0x0d
#define       LF                      0x0a

#define       GATETIME_100MSEC    10
#define       GATETIME_1SEC        1

#define       preScale_1            1
#define       preScale_2            2
#define       preScale_4            4
#define       preScale_8            8

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

static unsigned    short    MeasurementCnt;

void   interrupt()
{
    if (PIR2.TMR3IF == ON) {
        PIR2.TMR3IF = OFF;
        //
        MeasurementCnt--;
        if (MeasurementCnt == 0) {
            T1CON.TMR1ON = OFF;    //ゲートを閉める。
            T3CON.TMR3ON = OFF;    // TIMER3を停止する。
        }
    }
}

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

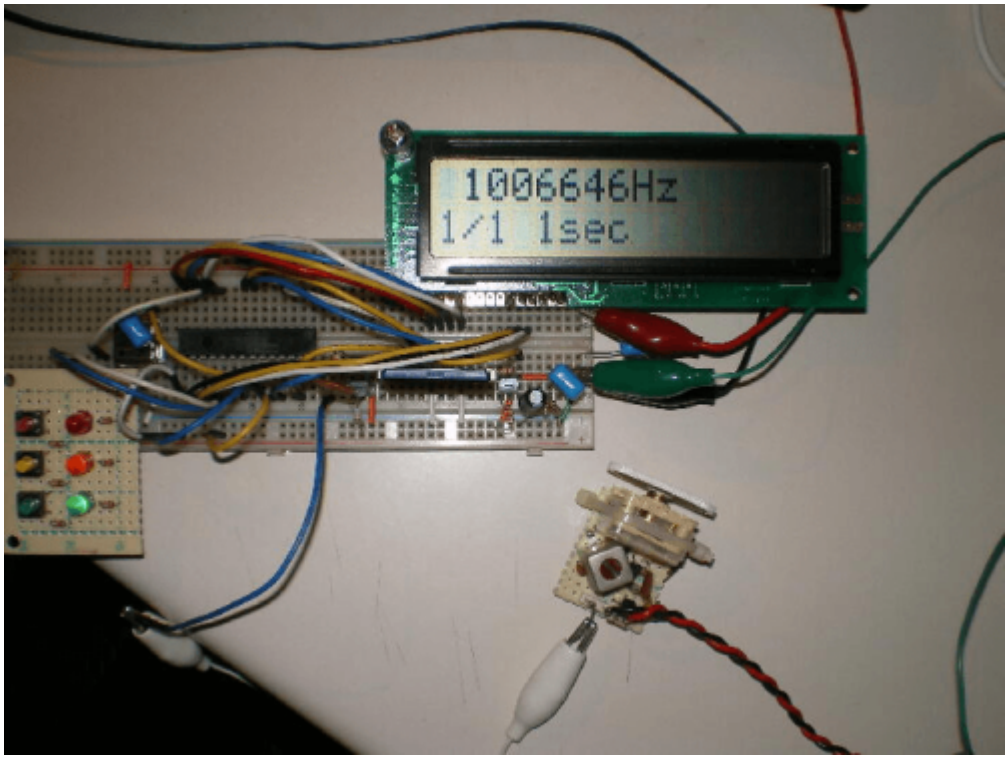
```
unsigned   long    FreqMeasurement(short gateTime, short preScale)
{
    static    unsigned    long    freq;
    // TIMER1の設定
    PIE1.TMR1IE = OFF;
    PIR1.TMR1IF = OFF;
    T1CON.T1RUN = 0;
    switch (preScale) {
    case preScale_1:
        T1CON.T1CKPS0 = 0;
        T1CON.T1CKPS1 = 0;
        break;
    case preScale_2:
        T1CON.T1CKPS0 = 0;
        T1CON.T1CKPS1 = 1;
        break;
    case preScale_4:
        T1CON.T1CKPS0 = 1;
        T1CON.T1CKPS1 = 0;
        break;
    case preScale_8:
        T1CON.T1CKPS0 = 1;
        T1CON.T1CKPS1 = 1;
        break;
    }
    T1CON.T1OSCEN = 0;
    T1CON.NOT_T1SYNC = 1;
    T1CON.TMR1CS = 1;
    T1CON.TMR1ON = OFF;
    TMR1L = 0;
    TMR1H = 0;
    // TIMER3の設定
    PIE2.TMR3IE = ON;
    PIR2.TMR3IF = OFF;
    T3CON.T3CKPS1 = 1;
    T3CON.T3CKPS0 = 1;
    T3CON.NOT_T3SYNC = 1;
    T3CON.TMR3CS = 0;
    T3CON.TMR3ON = OFF;
    switch (gateTime) {
    case GATETIME_1SEC:
        MeasurementCnt = 8;
        TMR3L = 0xE0;         // 500000=1/((1/16000000) * 4 * 8)
        TMR3H = 0x5E;         // 0x5EE0=65536-(500000-(65536*7))
        break;
    case GATETIME_100MSEC:
        MeasurementCnt = 1;
        TMR3L = 0xB0;         // 50000=0.1/((1/16000000) * 4 * 8)
        TMR3H = 0x3C;         // 0x3CB0=65536-50000
        break;
    }
```

```
    //
    freq = 0;
    //割り込みを許可する。
    INTCON.PEIE = ON;
    INTCON.GIE = ON;
    //開始
    T3CON.TMR3ON = ON;     //タイマを開始する。
    //    Delay
    Delay_Cyc(1);
    asm    nop;
    asm    nop;
    //
    T1CON.TMR1ON = ON;     //ゲートを開ける。
    //測定
    while (T3CON.TMR3ON != OFF) {
        if (PIR1.TMR1IF == ON) {
            PIR1.TMR1IF = OFF;
            freq++;
        }
    }
    if (PIR1.TMR1IF == ON) {
        PIR1.TMR1IF = OFF;
        freq++;
    }
    //換算
    freq = freq * 65536;
    freq = freq + ((unsigned)TMR1H * 256) + (unsigned)TMR1L;
    freq = freq * preScale * gateTime;
    //
    return (freq);
}

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

void   main()
{
    //変数の定義
    static    char                buf[50], *msg;
    static    unsigned    long    freq, temp;
    static    unsigned    short    cnt, preScale, gateTime;
    //□□□変換の設定
    ADCON1.PCFG3 = 1;
    ADCON1.PCFG2 = 1;
    ADCON1.PCFG1 = 1;
    ADCON1.PCFG0 = 1;
    //ポートの設定
    TRISA = 0b11111000;
    TRISB = 0b00000000;
    TRISC = 0b00000001;
```

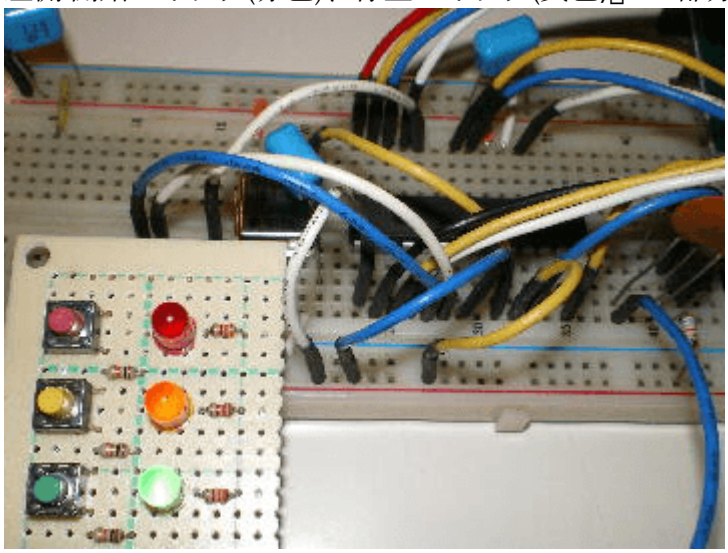
```
    //変数の初期化
    preScale = preScale_1;
    gateTime = GATETIME_1SEC;
    //□□□の初期化
    Lcd_Custom_Config(&PORTB,7,6,5,4,&PORTA,2,1,0);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    for (cnt = 1; cnt <= 16; cnt++) {
        Lcd_Custom_Chr(1, cnt, 0xFF);
        LED = ON;
        Delay_ms(50);
        LED = OFF;
        Delay_ms(50);
    }
    for (cnt = 1; cnt <= 16; cnt++) {
        Lcd_Custom_Chr(2, cnt, 0xFF);
        LED = ON;
        Delay_ms(50);
        LED = OFF;
        Delay_ms(50);
    }
    Lcd_Custom_Cmd(LCD_CLEAR);
    //□□□の初期化
    LED = ON;
    Spi_Init_Advanced(MASTER_OSC_DIV64, DATA_SAMPLE_MIDDLE,
CLK_IDLE_LOW, LOW_2_HIGH);
    if (Mmc_Fat_Init(&PORTC, 2)) {
        Lcd_Custom_Out(1, 1, "MMC error!");
        while (1) {
            LED = ON;
            Delay_ms(50);
            LED = OFF;
            Delay_ms(50);
        }
    }
    Spi_Init_Advanced(MASTER_OSC_DIV16, DATA_SAMPLE_MIDDLE,
CLK_IDLE_LOW, LOW_2_HIGH);
    LED = OFF;
    //
    while (1) {
        //開始スイッチが押されるのをチェックする。
        while (SW_START == 1) {
            Delay_ms(10);
        }
        //□□□のファイルのオープン
        Mmc_Fat_Assign("log.csv", 0xA0);
        Mmc_Fat_Rewrite();
        Mmc_Fat_Write("$START\r\n", 8);
        //停止スイッチが押されるまで処理を繰り返す。
        while (SW_STOP == 1) {
            //周波数の測定
```

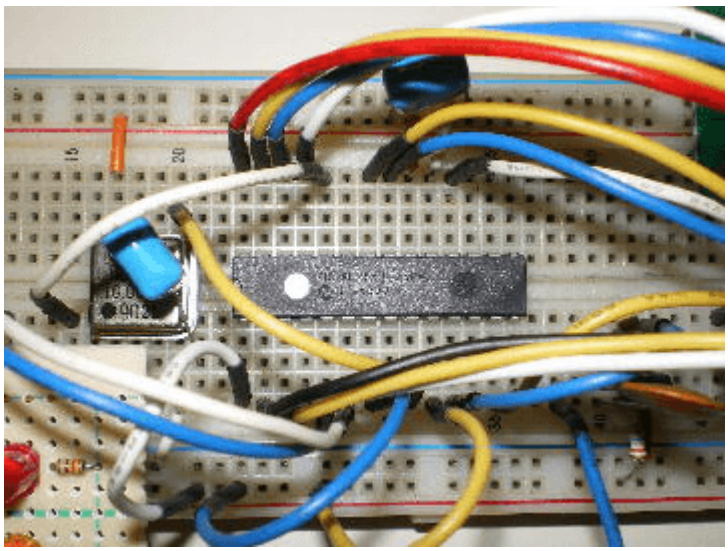
```
            LED = ON;
            freq = FreqMeasurement(gateTime, preScale);
            LED = OFF;
            //プリスケーラの切り替え
            if (SW1 == 1) {
                preScale = preScale_1;
                msg = "1/1 ";
            } else {
                preScale = preScale_8;
                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);
            //表示レンジの切り替え
            if (SW3 == 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);
            //周波数の表示と□□□への書き込み
            Lcd_Custom_Out(1, 1, &buf[3]);
            buf[11] = CR;
            buf[12] = LF;
            Mmc_Fat_Write(&buf[3], 10);
        }
        Mmc_Fat_Write("$STOP\r\n", 7);
    }
}//~!

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

動作確認

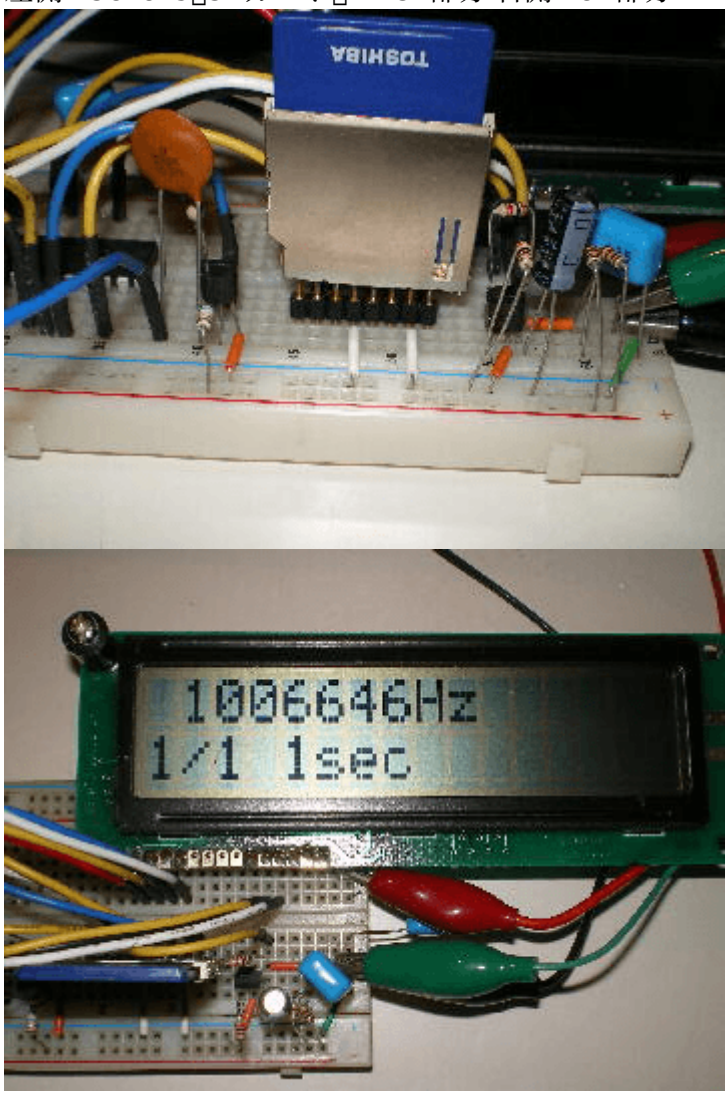


左側:開始スイッチ(赤色)、停止スイッチ(黄色)□LED部分 右側:16MHz水晶モジュール□PIC18F2550部分

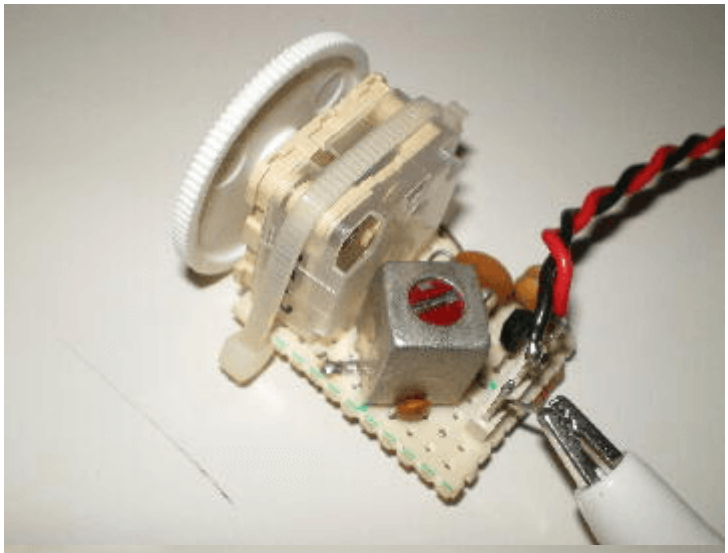


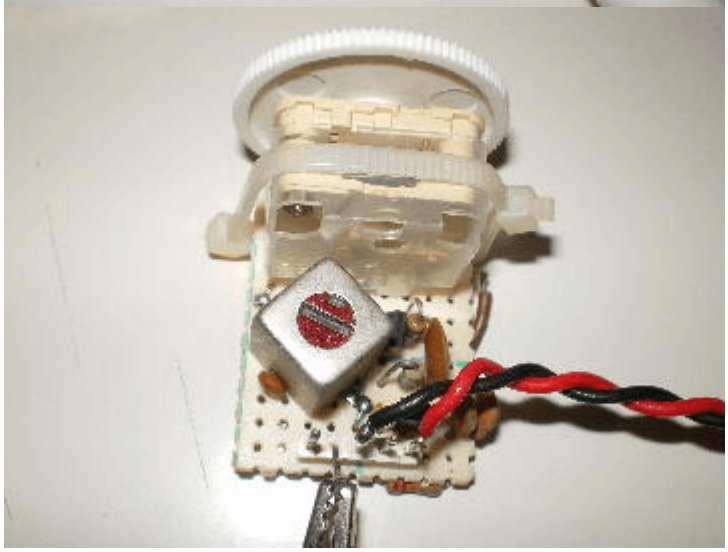


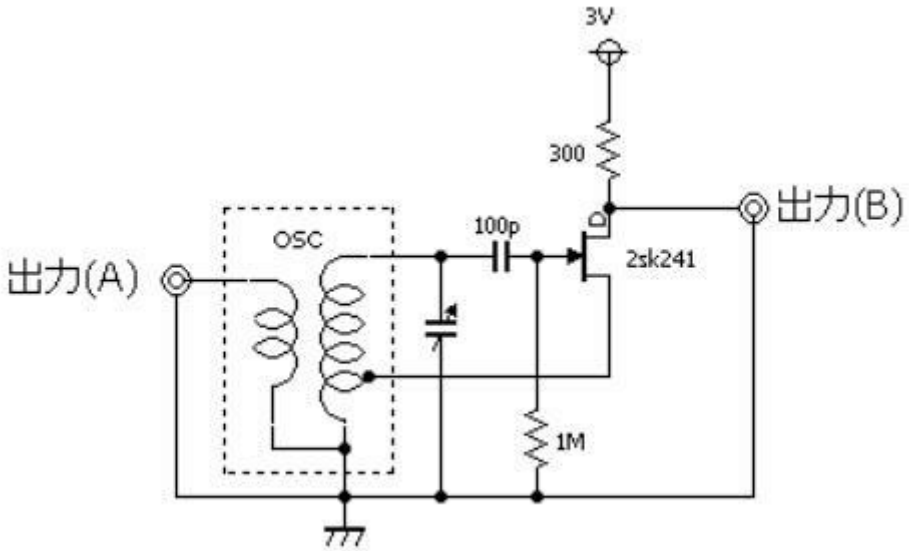
左側:2SC1815□SDカード□TL431部分 右側:LCD部分



測定するために、作った簡易発振器です□(2SK241□ラジオ用局発コイル、ポリバリコン等)

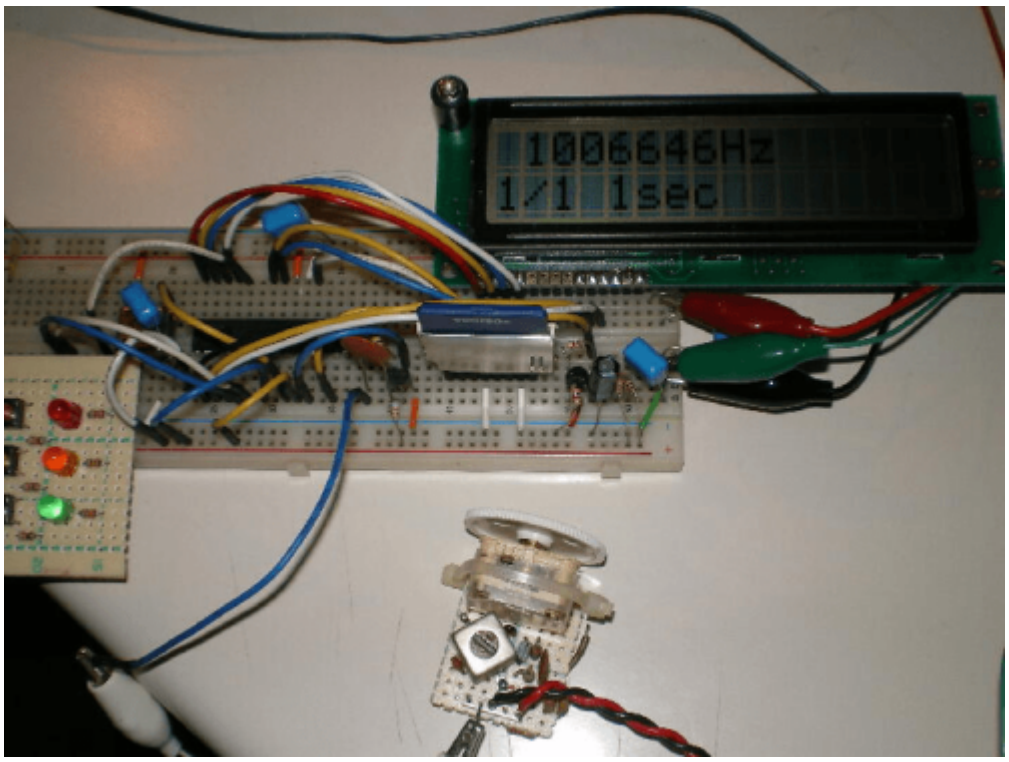




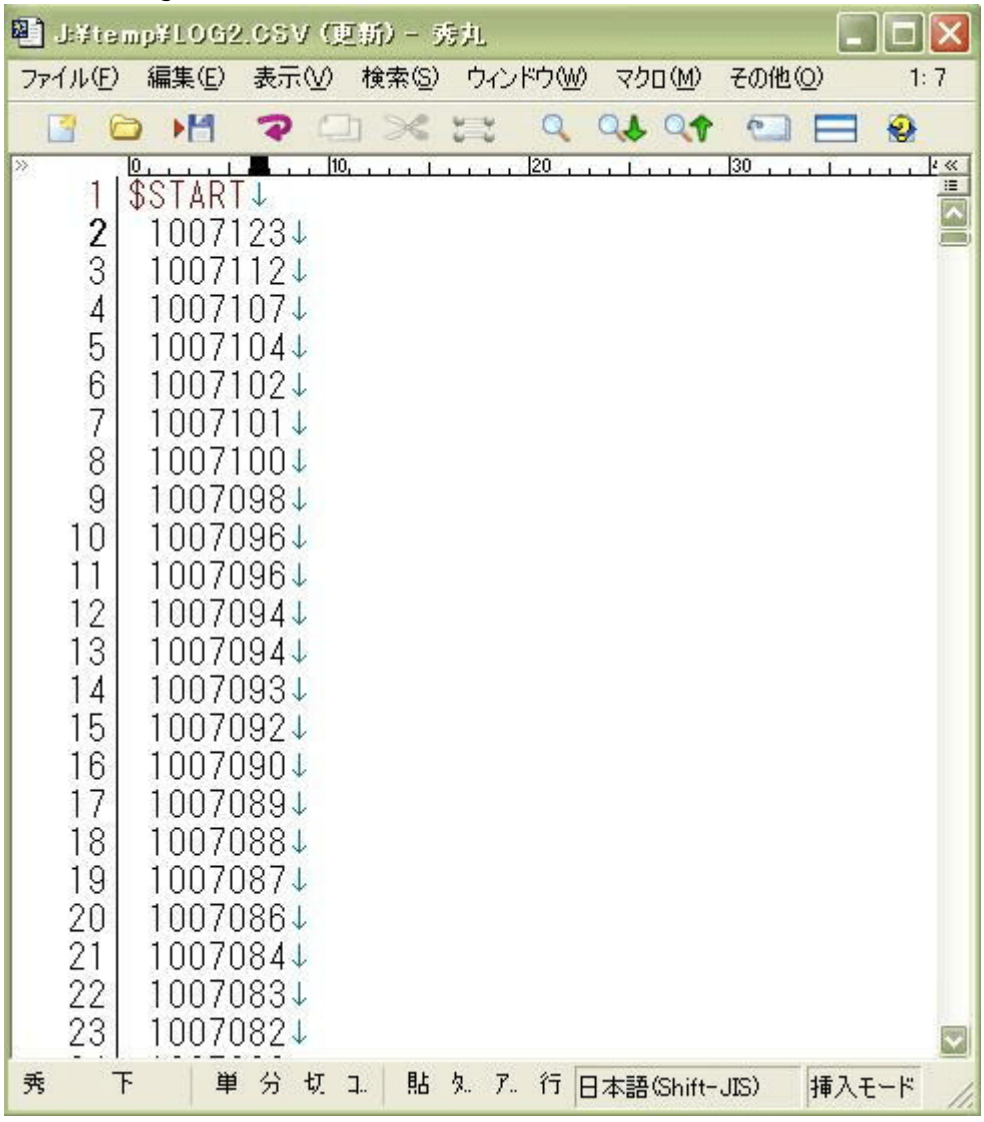


簡易発振器の回路図です。

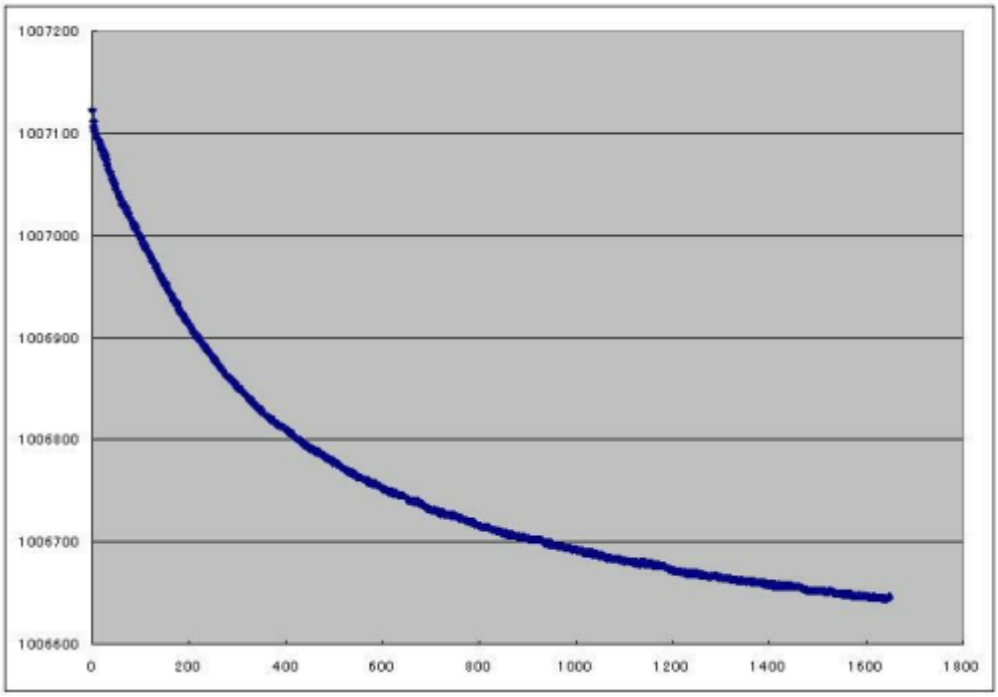
プリスケール値(1/1)、ゲートタイム(1秒)で、30分ほど測定してみました。



ファイル"log.csv"に記録された内容です。



記録結果を、Excelでグラフ表示させて見ました。 時間の経過とともに、周波数が低くなり、安定(変動が緩やか)してきます。



如何ですか? 無線機を自作される方には、とても重宝するのではと思います。!{ }!