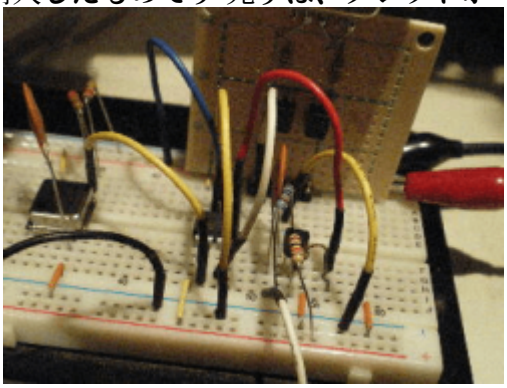
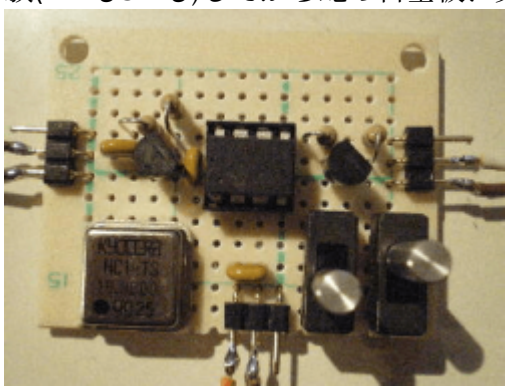
簡易周波数カウンタ

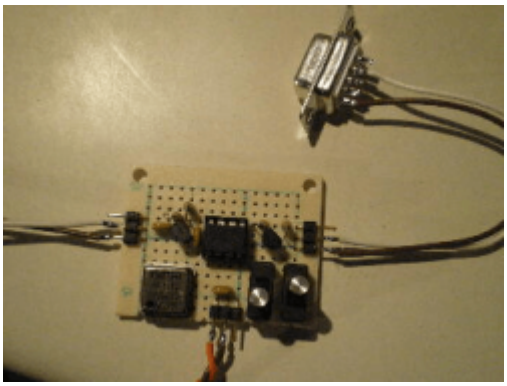
概要

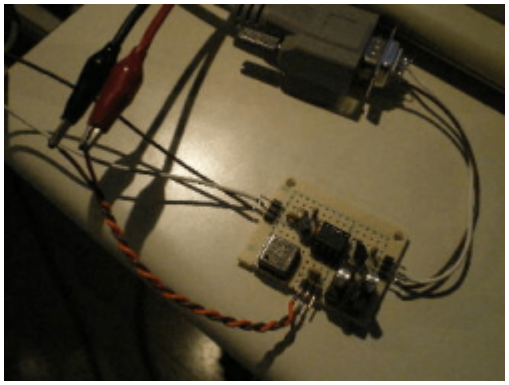
とても簡易（ハードウエアもソフトウエアも共に）な周波数カウンタを作成しました 最高2.5Mhzまで1Hz単位で、または最高20Mhzまで8Hz単位で計測できる周波数カウンタです ゲートタイムは、1秒と0.1秒の切り替えが出来るようにしました 基準となるクロックは、手持ちの16.0000Mhz□京セラ製）を使用しました これは以前にジャンク袋で大量に購入したものです 先ずは、ブレッドボードで実



験(HWもSWも)してから蛇の目基板に実装しました







処理説明

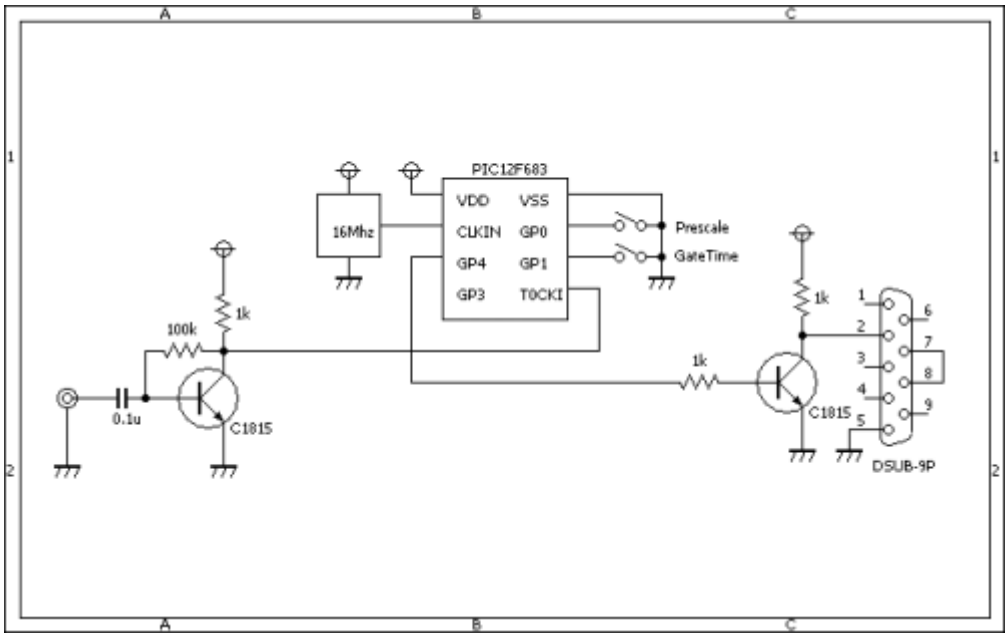
先ず、入力信号は、トランジスタ(2SC1815)で構成されたアンプで増幅されます。 増幅された信号は、GP2(T0CKI)端子に接続されTIMER0でカウントされます□ TIMER1とCCP1を使って正確な0.1秒または1秒を得ます。

- TIMER1を内部クロックで1/8プリスケーラで動作させます。
- するとTIMER1のクロックは、16MHz÷4÷8□500kHzとなります。
- CCP1をTIMER1とのコンペアモードで使い、一致する値を50000とします。

- すると10Hz周期で一致割り込みが発生し、正確な0.1秒を得ることが出来ます。
- 更にこれを10回カウントすると、正確な1秒を得ることが出来ます。

プリスケール値は1/1、1/8を選択できるようにしました。 最終的にカウントした値を文字列に変換して□RS232Cで送信します。

回路図



ソースコード

freqCounter.c

```
/*
  ＜簡易周波数カウンター＞
*/

static short    endFlag;
static short    gateTime;

void   interrupt(){
    if (PIR1.CCP1IF == 1) {
        PIR1.CCP1IF = 0;
        //
        endFlag++;
        if (endFlag == gateTime) {
            TRISIO.F2 = 0;          //ゲートを閉める。
            GPIO.F2 = 0;
            T1CON.TMR1ON = 0;     // TIMER1を停止する。
        }
    }
```

```
}

void   Soft_Uart_Write_String(char *buf)
{
    short     len, i;
    len = strlen(buf);
    for (i = 0; i < len; i++) {
        INTCON.GIE = 0;
        Soft_Uart_Write(buf[i]);
        INTCON.GIE = 1;
    }
}

void main()
{
    static    unsigned    long    freq;    // 0...4294967295
    static    unsigned    char    buf[20];
    static    short                prescaler;
    //クロックの設定→今回は外付けの16Mhzクロックモジュールを使用する。
    OSCCON = 0b01110000;
    //コンパレータの設定→今回は使用しない。
    CMCON0 = 0b00000111;
    //アナログの設定→今回は使用しない。
    ANSEL  = 0b00000000;
    //ポートの設定
/*
    GPIO.F0 = sw0(INPUT)
    GPIO.F1 = sw1(INPUT)
    GPIO.F2 = freq(INPUT)
    GPIO.F3 = RS232C(RECV:INPUT)
    GPIO.F4 = RS232C(SEND:OUTPUT)
    GPIO.F5 = ExternalClock(16Mhz:INPUT)
*/
    TRISIO = 0b00101111;
    OPTION_REG.F7 = 0;
    //入力割り込みの設定→今回は使用しない。
    INTCON.INTE = 0;
    INTCON.INTF = 0;
    OPTION_REG.INTEDG = 0;
    //入力割り込み（変化）の設定→今回は使用しない。
    INTCON.GPIE = 0;
    INTCON.GPIF = 0;
    // CCPの設定
    PIE1.CCP1IE = 1;
    PIR1.CCP1IF = 0;
    CCP1CON = 0b00001011;
    CCPR1L = 0x50;    // 0.1sec...10hz...クロックが16Mhzの時
    CCPR1H = 0xC3;    // 0.1sec...(1÷16000000)*4*8*50000
    // TIMER0の設定
    INTCON.T0IE = 0;
    INTCON.T0IF = 0;
```

```
    TMR0 = 0;
    OPTION_REG.T0CS = 1;
    OPTION_REG.T0SE = 0;
    OPTION_REG.PSA  = 1;
    OPTION_REG.PS0  = 0;
    OPTION_REG.PS1  = 0;
    OPTION_REG.PS2  = 0;
    // TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    TMR1L = 0;
    TMR1H = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1CKPS1 = 1;
    T1CON.TMR1ON = 0;
    // TIMER2の設定→今回は使用しない。
    PIE1.TMR2IE = 0;
    PIR1.TMR2IF = 0;
    T2CON.TMR2ON = 0;
    T2CON.T2CKPS0 = 0;
    T2CON.T2CKPS1 = 0;
    TMR2 = 0;
    //
    Soft_Uart_Init(GPIO, 3, 4, 9600, 0);
    Soft_Uart_Write_String("PIC12F683 Frequency Counter R1.00\r\n");
    //割り込み(全体)の設定
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    while (1) {
        //初期化
        T1CON.TMR1ON = 0;
        TRISIO.F2 = 0;
        GPIO.F2 = 0;
        TMR0 = 0;
        INTCON.T0IF = 0;
        TMR1L = 0;
        TMR1H = 0;
        PIR1.TMR1IF = 0;
        endFlag = 0;
        freq = 0;
        if (GPIO.F0 == 1) {     //プリスケーラの切り替え
            OPTION_REG.PSA  = 1;
            OPTION_REG.PS0  = 0;
            OPTION_REG.PS1  = 0;
            prescaler = 1;
        } else {
            OPTION_REG.PSA  = 0;
            OPTION_REG.PS0  = 0;
            OPTION_REG.PS1  = 1;
            prescaler = 8;
```

```
        }
        if (GPIO.F1 == 1) {    // Gate Timeの切り替え
            gateTime = 1;
        } else {
            gateTime = 10;
        }
        //開始
        T1CON.TMR1ON = 1;
        Delay_Cyc(10);
        TRISIO.F2 = 1;    //ゲートを開ける。
        //測定
        while (endFlag != gateTime) {
            if (INTCON.T0IF == 1) {
                INTCON.T0IF = 0;
                freq++;
            }
        }
        freq *= 256;
        freq += TMR0;
        //補正
        if (prescaler == 8)
            freq *= 8;
        if (gateTime == 1)
            freq *= 10;
        //表示
        LongToStr(freq, buf);
        Soft_Uart_Write_String(buf);
        Soft_Uart_Write_String("Hz\r\n");
        //
        Delay_ms(10);
    }
}
```

動作確認

PCのハイパータミナルを使って動作確認をしました。