簡易パルスカウンタ

概要

一般的な周波数カウンタは、1秒間または0.1秒間のパルス数をカウントして、その周波数を求めます。そのため、1分間、1時間、1日間などのように長時間に渡って発生するパルスをカウントする用途には向いていません。

しかし、実際には、次のようなカウントの要望は多くあります。

- 開店時間から閉店時間までにデパートに来場する人の数のカウント
- 朝8時から17時までの、交通量のカウント
- GM(Geiger Muelle)管を使用した放射線のカウント(1分、10分、60分)

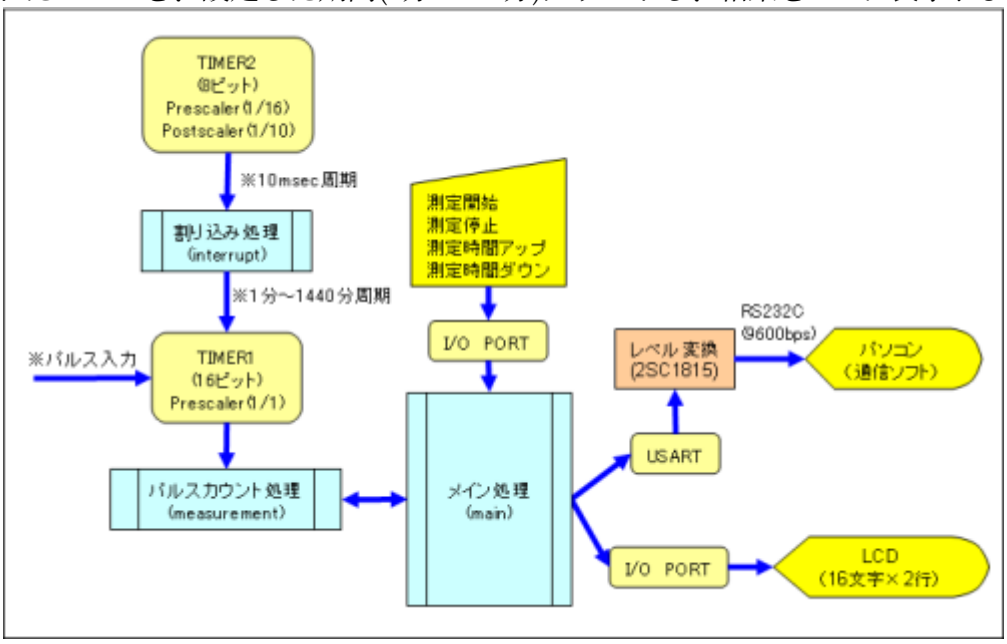
そこで今回は、仕組み的には周波数カウンタと同じですが、測定時間を分単位で設定可能なパルスカウンタを 製作しました。

<仕様>

- パルス測定時間→1分~1440分(24時間)
- 測定結果の表示□LCD表示(測定値、最大値、最小値)
- 測定結果の通信→シリアル出力(9600bps)

動作原理

入力されたパルスを、設定した期間(1分~1440分)カウントし、結果をLCDに表示およびシリアル出力します。



動作原理(ハードウェア)

◎TIMER1 TIMER1を外部クロックモードで使用して、入力されるパルスをカウントします。

◎TIMER2 パルス入力のゲート開閉のために必要となる基本インターバル(10msec)を発生させます。

◎USART 測定したデータをシリアル送信します。(通信速度=9600bps)

◎トランジスタ(2SC1815) シリアル信号電圧のレベル変換と位相反転を行います。

動作原理(ソフトウェア)

◎メイン関数

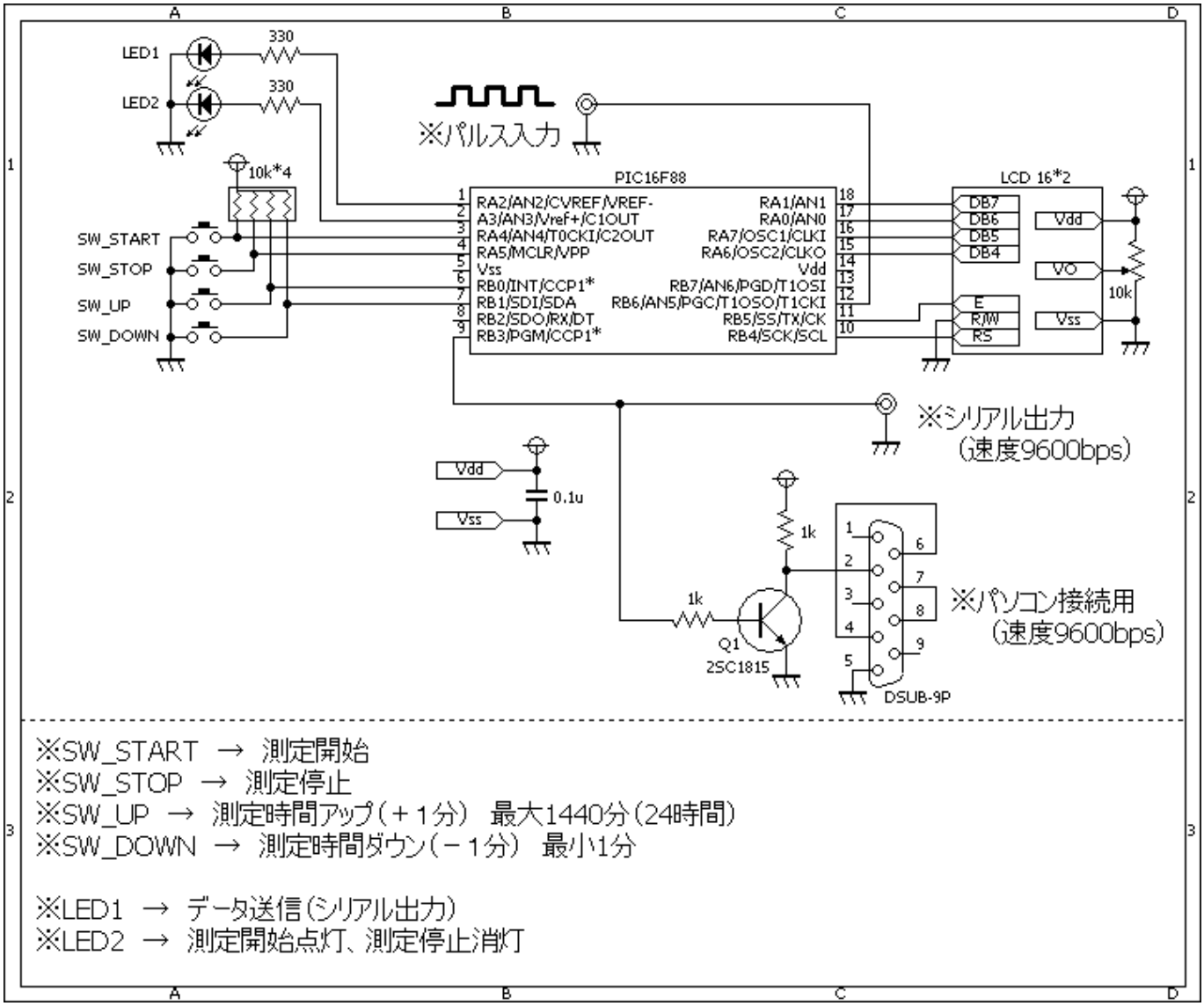
- 測定時間のEEPROMからの読み込み
  以前にEEPROMに保存された内容を読み出します。
  値が、1よりも小さいか、1440よりも大きい場合には、1に設定します。
- LCDの初期化
  LCDを初期化し、測定時間などを表示します。
- USARTの初期化
  通信速度9600bpsの、ソフトウエアUSARTとして、初期化します。
- タイマの初期化
  タイマ初期化関数を呼び出します。
- 測定開始(SW_START)
  押下されると、動作モード(mode)を“1“ にしLED2を点灯させ、変数を初期化します。
- 測定停止(SW_STOP)
  押下されると、動作モード(mode)を“0”にしLED2を消灯させます。
- 測定時間アップ(SW_UP)
  押下されると、測定時間(Interval_target_min)を“+1“ しEEPROMに書き込みます。
- 測定時間ダウン(SW_DOWN)
  押下されると、測定時間(Interval_target_min)を“-1”しEEPROMに書き込みます。
- パルスのカウント
  パルスカウント関数を呼び出します。
- 測定結果のLCDへの表示
  測定値、最大値、最小値をLCDに表示します。
- 測定結果のシリアル送信
  測定値をシリアル送信します。

◎パルスカウント関数 TIMER1を初期化し、割り込み関数にカウント開始指示を出しますTIMER1のオーバーフローフラグを監視し、パルス変数(pulse)を“+1“ します。 カウントが停止するまで繰り返します。 カウントが停止すると、パルス変数及びTIMER1のカウンタより、パルス数を求め結果を戻します。

◎割り込み関数 カウント開始指示があると、パルスを入力するためにゲートを開きます。 設定された測定時間(Interval_target_min)が経過するとゲートを閉じます。

◎タイマ初期化関数 TIMER2モジュールを、10msec周期で割り込みを発生させるために初期化します。

回路図



ソースコード

pulse_counter_v1_00.c

```
//*****************************************************************************
*
/*
    ＜簡易パルスカウンタ□□□
*/
//*****************************************************************************
*
//LCD
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D7 at RA1_bit;
sbit LCD_D6 at RA0_bit;
sbit LCD_D5 at RA7_bit;
sbit LCD_D4 at RA6_bit;
sbit LCD_RS_Direction at TRISB4_bit;
```

```
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISA1_bit;
sbit LCD_D6_Direction at TRISA0_bit;
sbit LCD_D5_Direction at TRISA7_bit;
sbit LCD_D4_Direction at TRISA6_bit;
//
#define BYTE    unsigned  short
#define WORD    unsigned  int
#define DWORD   unsigned  long
//
#define LED1            PORTA.B2
#define LED2            PORTA.B3
#define LED_ON          1
#define LED_OFF         0
//
#define SW_START        PORTA.B4
#define SW_STOP         PORTA.B5
#define SW_UP           PORTB.B0
#define SW_DOWN         PORTB.B1
//*****************************************************************
*
//■関数宣言
extern  void    main();
extern  void    init_timer();
extern  void    interrupt();
extern  long    measurement();
extern  void    Soft_UART_Write_Str(char *msg);
extern  void    WordToStrEx(unsigned input, char *output);
//*****************************************************************
*
//■■■タイマー初期化関数■■□□10msecインターバルタイマ発生用
void    init_timer()
{
        T2CON.T2CKPS1 = 1;
        T2CON.T2CKPS0 = 0;
        T2CON.TOUTPS3 = 1;
        T2CON.TOUTPS2 = 0;
        T2CON.TOUTPS1 = 0;
        T2CON.TOUTPS0 = 1;
        TMR2 = 0;
        PIE1.TMR2IE = 1;
        PIR1.TMR2IF = 0;
        PR2 = 125;
//125=10msec/((1sec/8MHz)*4*16Prescale*10Postscale)
        T2CON.TMR2ON = 1;
}
//*****************************************************************
*
//■■■割り込み関数■■■
long    Interval_counter = -1;
long    Interval_target_min = 1;
```

```
//
void    interrupt()
{
        if (PIR1.TMR2IF == 1) { //(10msecインターバルタイマ)
                PIR1.TMR2IF = 0;
                //ゲート開閉処理
                switch (Interval_counter) {
                case -1:
                        break;
                case 0:
                        T1CON.TMR1ON = 1;                //ゲートを開ける。
                        Interval_counter++;
                        break;
                default:
                        if (Interval_counter == (Interval_target_min *
6000)) {
                                T1CON.TMR1ON = 0;                //ゲー
トを閉める。
                                Interval_counter = -1;
                        } else {
                                Interval_counter++;
                        }
                        break;
                }
        }
}
//**********************************************************************
*
//■■■パルスカウント関数■■■
long    measurement()
{
        long    pulse;
        //
        pulse = 0;
        PIE1.TMR1IE = 0;
        PIR1.TMR1IF = 0;
        T1CON.T1CKPS1 = 0;
        T1CON.T1CKPS0 = 0;
        T1CON.TMR1CS = 1;
        T1CON.NOT_T1SYNC = 1;
        TMR1H = 0;
        TMR1L = 0;
        Interval_counter = 0;
        while (Interval_counter != -1) {
                if (PIR1.TMR1IF == 1) {
                        PIR1.TMR1IF = 0;
                        pulse++;
                }
                if (SW_STOP == 0) {
                        return (-1);
                }
```

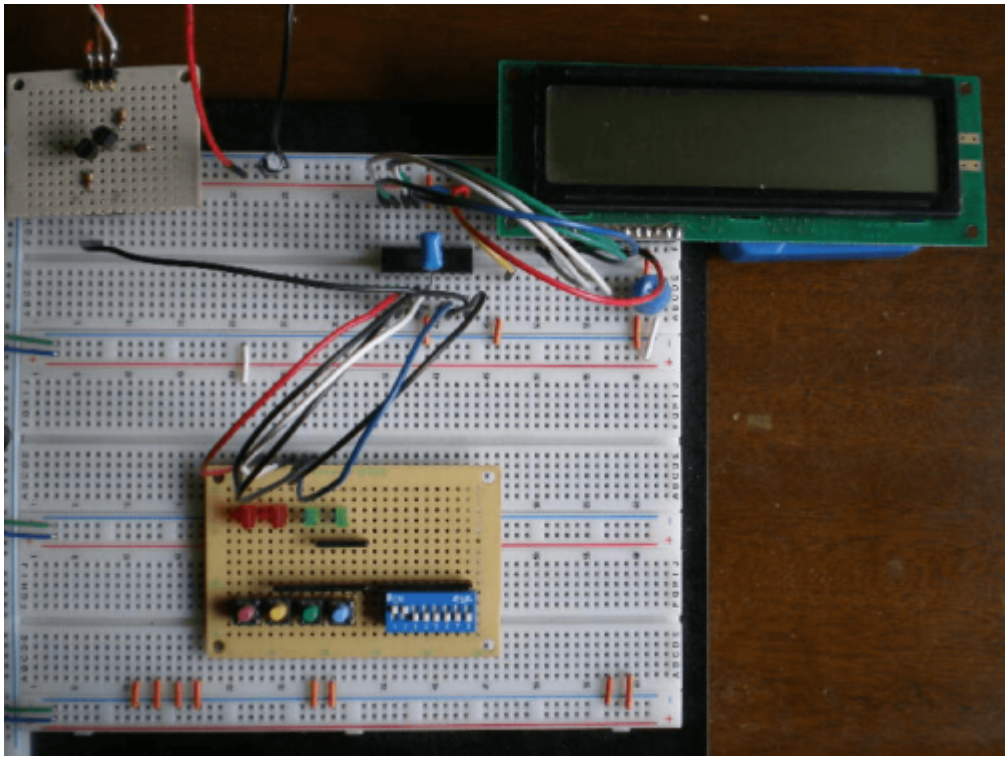
```
        }
        if (PIR1.TMR1IF == 1) {
                PIR1.TMR1IF = 0;
                pulse++;
        }
        return ((pulse * 65536) + ((long)TMR1H * 256) + (long)TMR1L);
}
//*********************************************************************
*
//■■■USART文字列送信関数■■■
void    Soft_UART_Write_Str(char *msg)
{
        while (*msg != 0x00) {
                Soft_UART_Write(*msg);
                msg++;
        }
}
//*********************************************************************
*
//■■■文字列変換関数■■■
void    WordToStrEx(unsigned input, char *output)
{
        WordToStr(input, output);
        while (*output != 0x00) {
                *output = (*output == ' ') ? '0' : *output;
                output++;
        }
}
//*********************************************************************
*
//■■■メイン関数■■■
void    main()
{
        long    pulse, max, min;
        char    buf[16];
        short   mode;
        //
        OSCCON = 0b01110000;
        ANSEL  = 0b00000000;
        TRISA  = 0b00110000;
        TRISB  = 0b11110111;
        //
        LED1 = LED_OFF;
        LED2 = LED_OFF;
        //
        Interval_target_min = EEPROM_Read(3);
        Interval_target_min = (Interval_target_min << 8) +
EEPROM_Read(2);
        Interval_target_min = (Interval_target_min << 8) +
EEPROM_Read(1);
        Interval_target_min = (Interval_target_min << 8) +
```

```
EEPROM_Read(0);
        if ((Interval_target_min < 1) || (Interval_target_min > 1440))
{
                Interval_target_min = 1;
        }
        //LCDを初期化します。
        Lcd_Init();
        Lcd_Cmd(_LCD_CURSOR_OFF);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "Pulse Counter");
        Delay_ms(1000);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 1, "CPM(0000)");
        WordToStrEx(Interval_target_min, buf);
        Lcd_Out(1, 5, &buf[1]);
        Lcd_Out(1, 16, ">");
        Lcd_Out(2, 16, "<");
        //USARTを初期化します。
        Soft_UART_Init(&PORTB, 2, 3, 9600, 0);
        Soft_UART_Write_Str("\r\nPulse Counter v1.00\r\n");
        //タイマーを初期化します(10msecインターバルタイマ)
        init_timer();
        //割り込みを許可します。
        INTCON.PEIE = 1;
        INTCON.GIE = 1;
        //
        mode = 0;
        max = 0;
        min = 99999999;
        //
        while (1) {
                //測定開始スイッチ確認
                if ((mode == 0) && (SW_START == 0)){
                        mode = 1;
                        LED2 = LED_ON;
                        max = 0;
                        min = 99999999;
                }
                //測定停止スイッチ確認
                if ((mode == 1) && (SW_STOP == 0)){
                        mode = 0;
                        LED2 = LED_OFF;
                }
                //測定時間アップスイッチ確認
                if ((mode == 0) && (SW_UP == 0)){
                        Delay_ms(100);
                        if (Interval_target_min < 1440) {  //最大24時間（60分×24時間）
                                Interval_target_min++;
                        }
                        //測定時間をLCDに表示する。
```

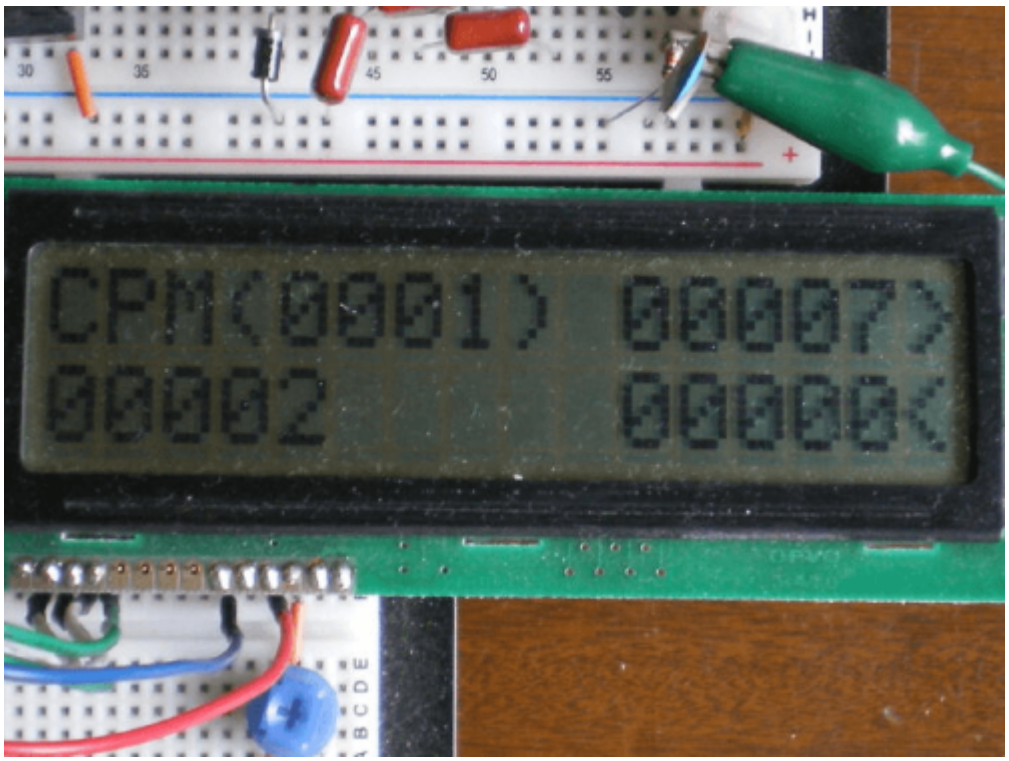
```
                        WordToStrEx(Interval_target_min, buf);
                        Lcd_Out(1, 5, &buf[1]);
                        //測定時間をEEPROMに書き込む。
                        EEPROM_Write(0, Interval_target_min & 0xFF);
                        EEPROM_Write(1, (Interval_target_min >> 8) &
0xFF);
                        EEPROM_Write(2, (Interval_target_min >> 16) &
0xFF);
                        EEPROM_Write(3, (Interval_target_min >> 24) &
0xFF);
                }
                //測定時間ダウンスイッチ確認
                if ((mode == 0) && (SW_DOWN == 0)){
                        Delay_ms(100);
                        if (Interval_target_min > 1) {      //最小1分
                                Interval_target_min--;
                        }
                        //測定時間をLCDに表示する。
                        WordToStrEx(Interval_target_min, buf);
                        Lcd_Out(1, 5, &buf[1]);
                        //測定時間をEEPROMに書き込む。
                        EEPROM_Write(0, Interval_target_min & 0xFF);
                        EEPROM_Write(1, (Interval_target_min >> 8) &
0xFF);
                        EEPROM_Write(2, (Interval_target_min >> 16) &
0xFF);
                        EEPROM_Write(3, (Interval_target_min >> 24) &
0xFF);
                }
                //測定
                if (mode == 1) {
                        //パルスをカウントしLCDに表示します。
                        pulse = measurement();
                        if (pulse != -1) {
                                max = pulse > max ? pulse : max;
                                min = pulse < min ? pulse : min;
                                WordToStrEx(pulse, buf);
                                Lcd_Out(2, 1, buf);
                                WordToStrEx(max, buf);
                                Lcd_Out(1, 11, buf);
                                WordToStrEx(min, buf);
                                Lcd_Out(2, 11, buf);
                                //カウント値をUSARTで送信します。
                                LED1 = LED_ON;
                                INTCON.GIE = 0;
                                WordToStr(pulse, buf);
                                Soft_UART_Write_Str(buf);
                                Soft_UART_Write_Str("\r\n");
                                INTCON.GIE = 1;
                                LED1 = LED_OFF;
```

```
                        } else {
                                mode = 0;
                                LED2 = LED_OFF;
                        }
                }
        }
}
//**************************************************************************
*
```

動作確認



左上=測定時間(1分~1440分の設定が可能) 左下=測定値、右上=測定最大値、右下=測定最小値



パソコンのハイパーターミナルで受信した測定値です。(パルス源は、GM管の出力パルスを使用)