簡易信号発生ユニット(PWM)

概要

前回の簡易信号発生ユニットでは、サイン波とノコギリ波を取り上げましたが、今回はPWM波の発生ユニットを作成しました。 モーターの制御やランプの制御には欠かせませんのでユニット化しておくことにしました。 ちょっとした実験をするときに側にあれば何かと便利なのでは考えました。

動作原理

PIC内臓のPWMモジュールでは、次の図のようにPeriodで期間（周期、周波数）を設定します。 そしてDutyCycleでデューティ比を設定します。 今回のユニットでは、次の3つの項目を設定することが出来ます。

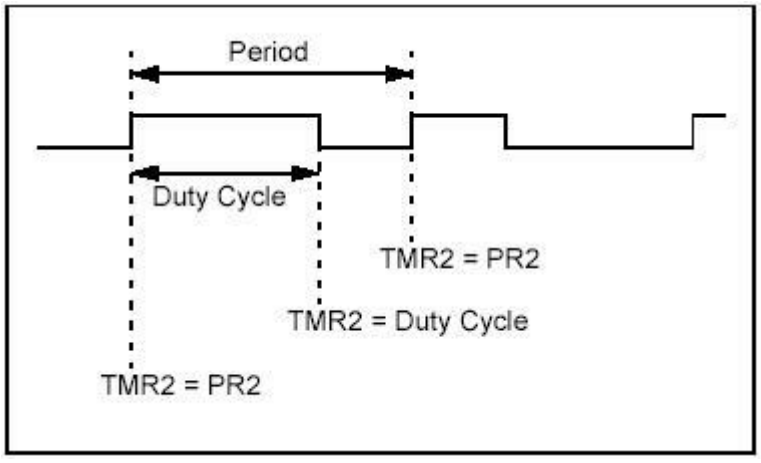
- プッシュスイッチ(SW1)で、クロック周波数(1Mhz□2Mhz□4Mhz□8Mhz)の設定
- プッシュスイッチ(SW2)で、TMR2のprescaler値（1/1、1/4、1/8）の設定
- ボリューム(VR1)で、デューティを設定

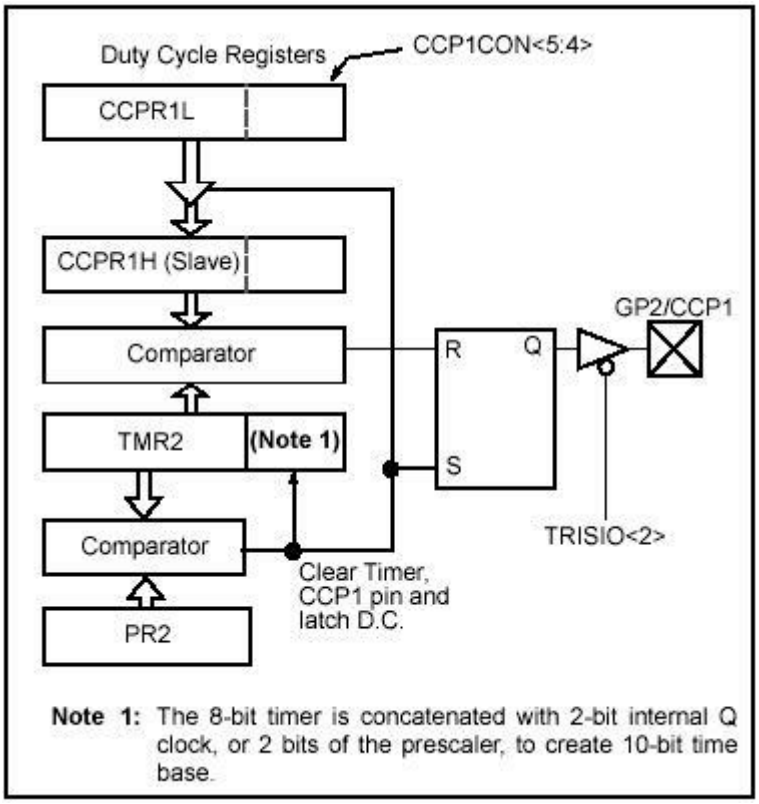
【周波数】

- 7.8Khz□1.95Khz□488Hz□クロック8Mhz□プリスケーラ1/1,1/4,1/8＞
- 3.9Khz□975Hz□244Hz□クロック4Mhz□プリスケーラ1/1,1/4,1/8＞
- 1.95Khz□488Hz□122Hz □クロック2Mhz□プリスケーラ1/1,1/4,1/8＞
- 975Hz□244Hz□61Hz□クロック1Mhz□プリスケーラ1/1,1/4,1/8＞
  ※つまり□61Hz,122Hz,244Hz,488Hz,975Hz,1.95Khz,3.9Khz,7.8Khzの8種類が設定可能です。

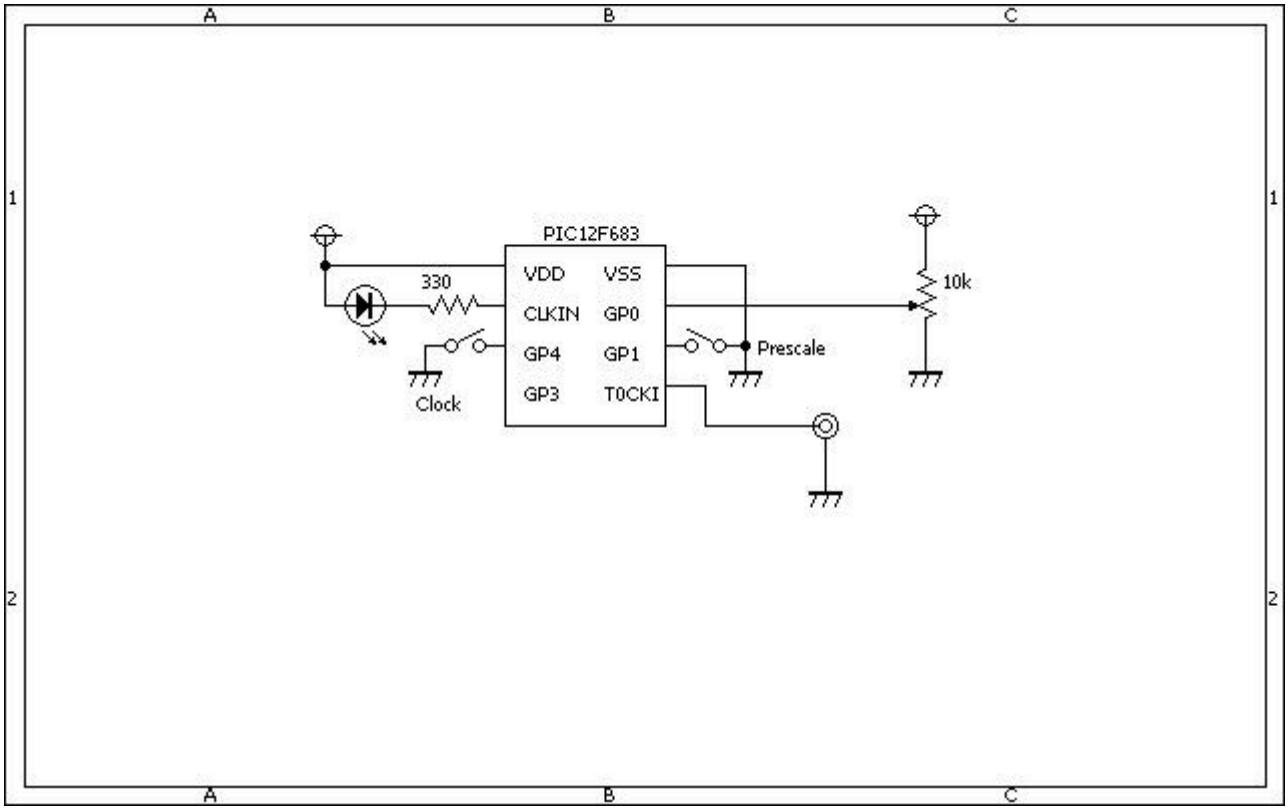
【デューティ比】

- 1024段階




Duty Cycle Registers
CCP1CON<5:4>
CCPR1L
CCPR1H (Slave)
Comparator
TMR2
(Note 1)
Comparator
PR2
R
Q
S
GP2/CCP1
TRISIO<2>
Clear Timer,
CCP1 pin and
latch D.C.
Note 1: The 8-bit timer is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create 10-bit time base.

回路図


A
B
C
1
2
PIC12F683
VDD
VSS
CLKIN
GP0
GP4
GP1
GP3
T0CKI
330
10k
Prescale
Clock

ソースコード

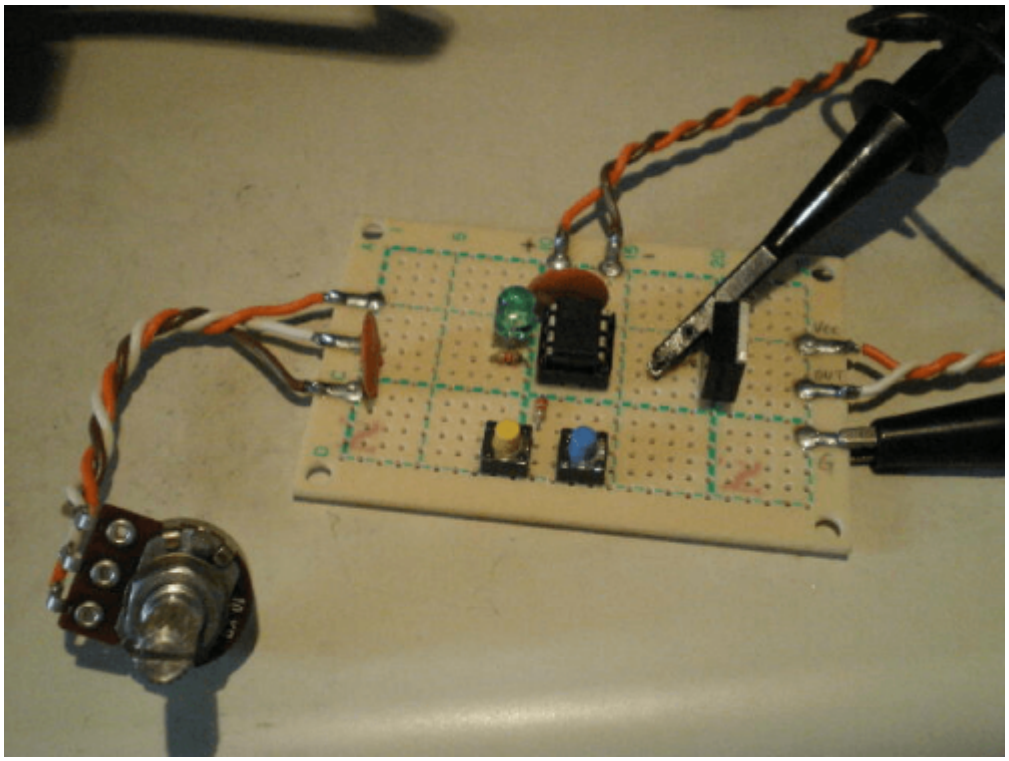
PWMunit.c

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

#define       swPrescaler        GPIO.F1
#define       swClock            GPIO.F4

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

void   interrupt()
{
    if (PIR1.TMR1IF == 1) {
        PIR1.TMR1IF = 0;
        GPIO.F5 = ~GPIO.F5;
    }
}

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

void   Pwm_Change_DutyEx(unsigned int duty_ratio)
{
    CCPR1L = duty_ratio >> 2;
    CCP1CON.F6 = duty_ratio & 0b00000001;
    CCP1CON.F7 = (duty_ratio & 0b00000010) >> 1;
}

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

static unsigned    char    clockData;

void   clockSet()
{
    while (1) {
        if (swClock == 0) {
            Delay_ms(1);
            continue;
        }
        switch (clockData) {
        case 1:         // 1Mhz
            clockData = 2;
            OSCCON.F4 = 1;
            OSCCON.F5 = 0;
            OSCCON.F6 = 1;
            break;
        case 2:         // 2Mhz
            clockData = 4;
            OSCCON.F4 = 0;
```

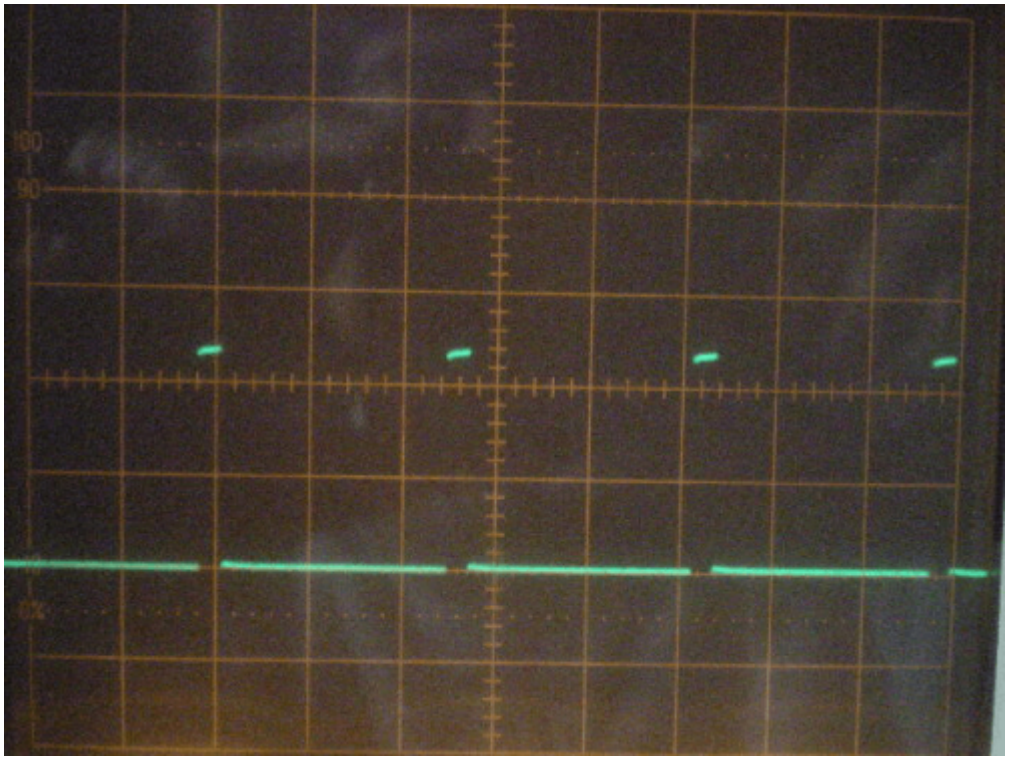
```
            OSCCON.F5 = 1;
            OSCCON.F6 = 1;
            break;
        case 4:         // 4Mhz
            clockData = 8;
            OSCCON.F4 = 1;
            OSCCON.F5 = 1;
            OSCCON.F6 = 1;
            break;
        case 8:         // 8Mhz
            clockData = 1;
            OSCCON.F4 = 0;
            OSCCON.F5 = 0;
            OSCCON.F6 = 1;
            break;
        }
        return;
    }
}

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

static unsigned    char    prescalerData;

void   prescalerSet()
{
    while (1) {
        if (swPrescaler == 0) {
            Delay_ms(1);
            continue;
        }
        switch (prescalerData) {
        case 1:         // 1/1
            prescalerData = 4;
            T2CON.F0 = 1;
            T2CON.F1 = 0;
            break;
        case 4:         // 1/4
            prescalerData = 16;
            T2CON.F0 = 1;
            T2CON.F1 = 1;
            break;
        case 16:    // 1/16
            prescalerData = 1;
            T2CON.F0 = 0;
            T2CON.F1 = 0;
            break;
        }
        return;
```

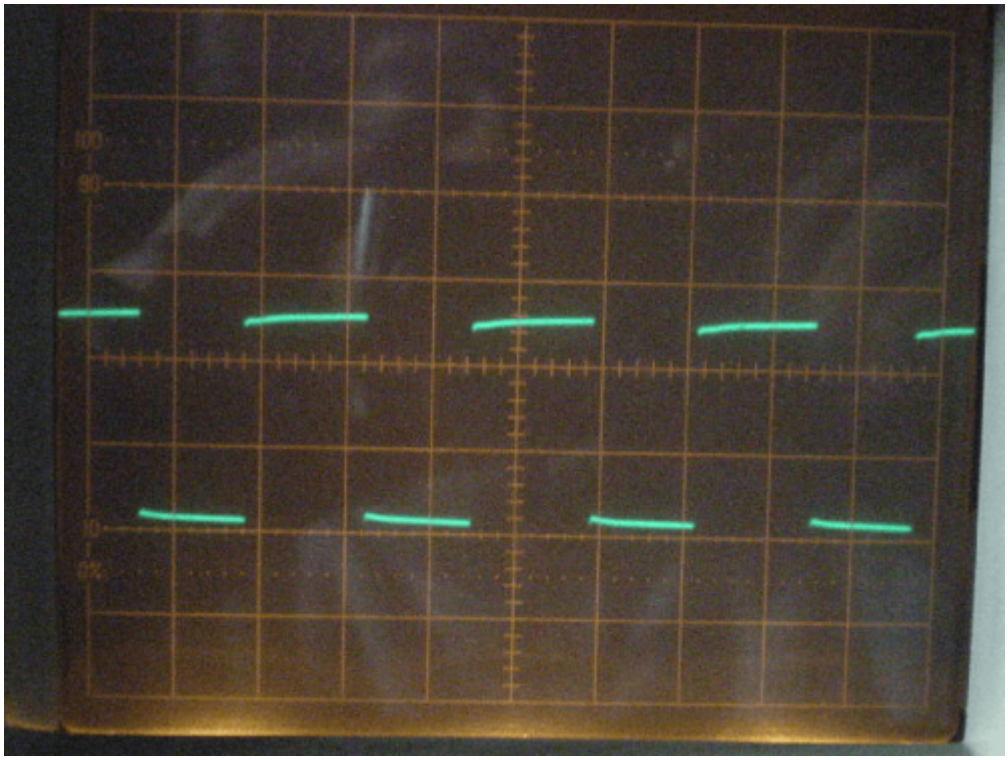
```
    }
}

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

void main()
{
    static    unsigned    int        ad0;
    //
    OSCCON = 0b01110000;    //クロックは8Mhz
    CMCON0 = 0b00000111;    //コンパレータは使用しない。
    ANSEL = 0b00000001;        // AN0を使用する。
    TRISIO = 0b00011011;
    GPIO = 0b00000000;
    OPTION_REG = 0b00000000;
    PIE1.TMR1IE = 1;
    PIR1.TMR1IF = 0;
    T1CON = 0b00000001;
    INTCON = 0b01000000;
    T2CON.F0 = 0;
    T2CON.F1 = 0;
    //
    Pwm_Init(1000);
    PR2 = 0xFF;
    Pwm_Change_DutyEx((PR2 * 4) / 2);
    Pwm_Start();
    //
    WPU.F1 = 1;
    WPU.F4 = 1;
    //
    clockData = 8;
    prescalerData = 1;
    //
    INTCON.GIE = 1;    //これ以降の処理で割り込みを許可する。
    //
    while (1) {
        if (swClock == 0) {
            clockSet();
        }
        if (swPrescaler == 0) {
            prescalerSet();
        }
        ad0 = Adc_Read(0);
        Pwm_Change_DutyEx(ad0);
    }
}

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

動作確認

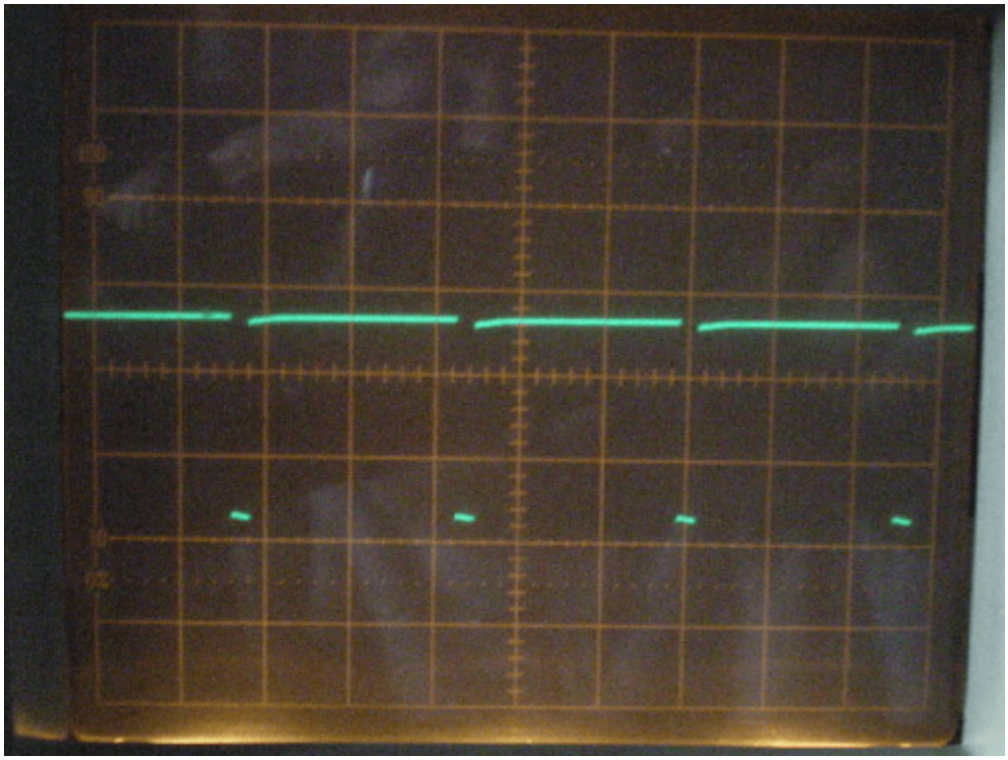


デューティ（小）





デューティ（中）



デューティ（大）

最大周波数は約7.8Khzでした。