PWM発振ユニットV2

概要

モータの制御やDC/DCコンバータ(昇圧、降圧)などを行う際には、PWM(Pulse Width Modulation)方式が良く採用されます。 電子工作に置いても、簡単な実験をする際には、PWMの周期やデューティを手軽に設定できるユニットが手元にあれば何かと便利です。 以前にも、簡易信号発生ユニット(PWM)を製作しましたが、周期やデューティの値を視覚的(LCDなどで)に知る事が出来ず不便を強いられていました。

そこで今回は、周期やデューティの値をLCDに表示することにより利便性を向上させました。

<仕様>

- 周期は、128usec(7.81kHz)、512usec(1.95kHz)、2048usec(488.3Hz)の3種を提供する。
- デューティは、0~1023(0x3FF)の1024段階とする。
- デューティの設定は、ボリュームによるスムーズな操作と、スイッチによる微調整を可能とする。

動作原理

基本的な原理は、簡易信号発生ユニット(PWM)を参照してください。

<ボリュームによるデューティの設定> SW1を押下したまま、ボリューム(R5)を廻すことにより、デューティの値を0%(0)~99.9%(1023)までスムーズに設定することが出来ます。

※A/D変換の精度は、10ビットなので、

- 上位8ビットを、CCPR1Lレジスタにセットし、
- 下位2ビットを、CCP1CONのCCP1XビットとCCP1Yビットにセットします。

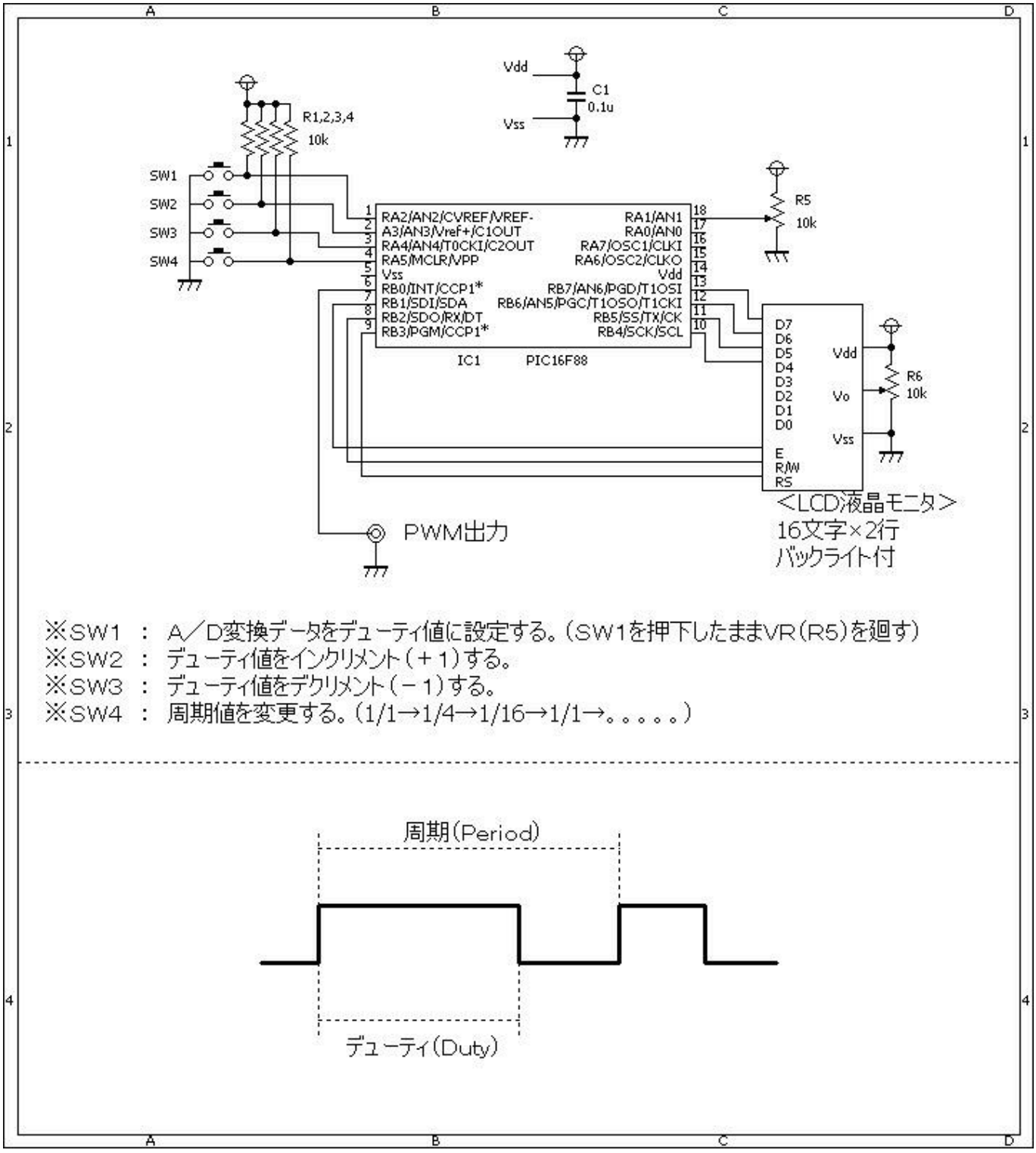
<スイッチによるデューティの設定> SW2を押下する毎に、デューティの値がインクリメント(+1)されていきます。 SW3を押下する毎に、デューティの値がデクリメント(-1)されていきます。

※約0.1%の単位でデューティを設定できます。

<スイッチによる周期の切り替え> SW4を押下する毎に、周期が【128usec→512usec→2048usec→128usec→】というふうに切り替わっていきます。

※128usec:TIMER2のプリスケーラを、1/1にセットします。 ※512usec:TIMER2のプリスケーラを、1/4にセットします。 ※2048usec:TIMER2のプリスケーラを、1/16にセットします。

回路図



ソースコード

pwmOsc_v2.c

```
//**********************************************************************
*
/*
□□□□□□発振ユニット＞
*/
```

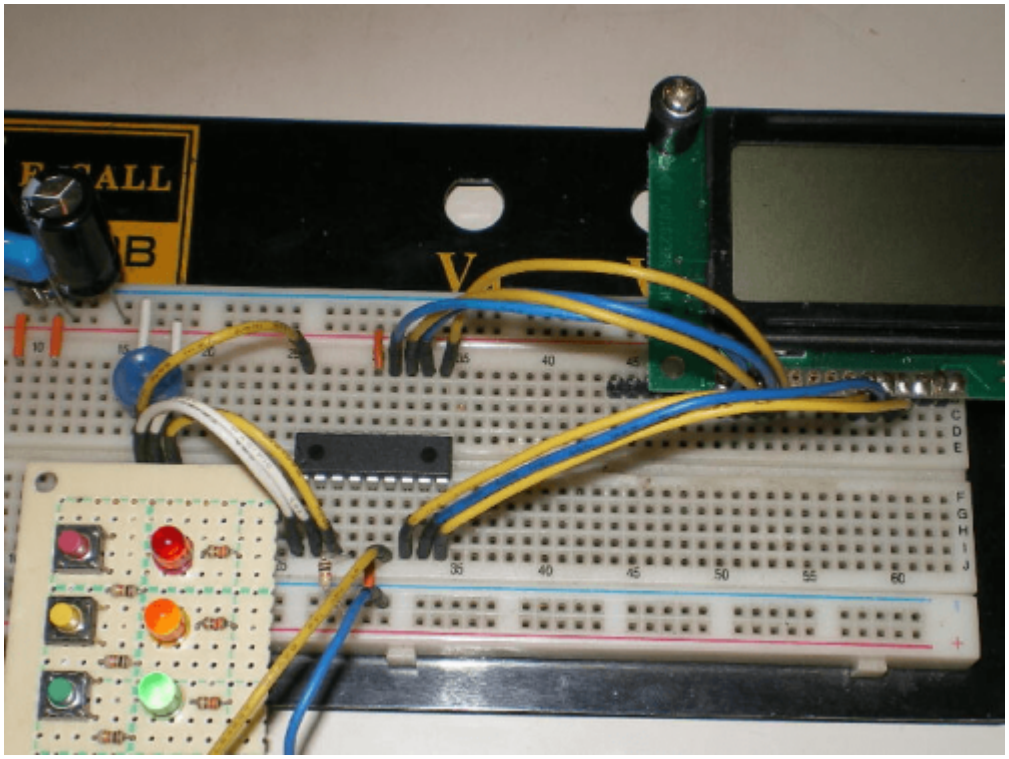
```
//*****************************************************************************
*

#define       ON            0
#define       OFF            1

#define       TRUE        255
#define       FALSE        0

#define       SW1            &PORTA, 2
#define       SW2            &PORTA, 3
#define       SW3            &PORTA, 4
#define       SW4            &PORTA, 5

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

void   Pwm_Change_DutyEx(unsigned int duty_ratio)
{
    CCPR1L = duty_ratio >> 2;
    CCP1CON.CCP1Y = (duty_ratio & 0b0000000000000001) == 0 ? 0 : 1;
    CCP1CON.CCP1X = (duty_ratio & 0b0000000000000010) == 0 ? 0 : 1;
}

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

static char    buf[16];

void   display(short Prescale, int duty)
{
    static    double    tmp;
    //
    switch (Prescale) {
    case 0:
        Lcd_Custom_Out(2, 10, "7.81kHz");
        tmp = (256.0 * 4 * 0.000000125 * 1) * 1000000;
        WordToStr(tmp, buf);
        Lcd_Custom_Out(2, 1, &buf[1]);
        Lcd_Custom_Out(2, 5, "usec");
        break;
    case 1:
        Lcd_Custom_Out(2, 10, "1.95kHz");
        tmp = (256.0 * 4 * 0.000000125 * 4) * 1000000;
        WordToStr(tmp, buf);
        Lcd_Custom_Out(2, 1, &buf[1]);
        Lcd_Custom_Out(2, 5, "usec");
        break;
    case 2:
        Lcd_Custom_Out(2, 10, "488.3Hz");
        tmp = (256.0 * 4 * 0.000000125 * 16) * 1000000;
```

```
        WordToStr(tmp, buf);
        Lcd_Custom_Out(2, 1, &buf[1]);
        Lcd_Custom_Out(2, 5, "usec");
        break;
    }
    //
    WordToStr(duty, buf);
    Lcd_Custom_Out(1, 9, &buf[1]);
    tmp = (duty / 1024.0) * 1000;
    WordToStr(tmp, buf);
    buf[6] = 0x00;
    buf[5] = buf[4];
    buf[4] = '.';
    Lcd_Custom_Out(1, 1, &buf[1]);
    Lcd_Custom_Out(1, 6, "%");
}

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

void main()
{
    static    short    cnt, Prescale;
    static    int        duty;
    static    double    tmp;
    //
    OSCCON = 0b01110000;     //クロックを8Mhzに設定する。
    ANSEL  = 0b00000001;      //□□□変換は使用しない。
    CMCON  = 0b00000111;     //コンパレータを使用する。
    TRISA  = 0b11111111;     //□□□ポートを設定する。
    TRISB  = 0b11111111;     //□□□ポートを設定する。
    //□□□の初期化
    Lcd_Custom_Config(&PORTB,7,6,5,4,&PORTB,3,2,1);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    Lcd_Custom_Out(1, 1, "PWM OSC v2");
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(2, cnt + 1, 0xFF);
        Delay_ms(100);
    }
    Lcd_Custom_Cmd(LCD_CLEAR);
    //□□□の設定
    Pwm_Init(1000);    // 1kHz
    PR2 = 0xFF;
    duty = 512;
    Pwm_Change_DutyEx(duty);
    Prescale = 0;
    T2CON.T2CKPS1 = 0;
    T2CON.T2CKPS0 = 0;
    Pwm_Start();
```

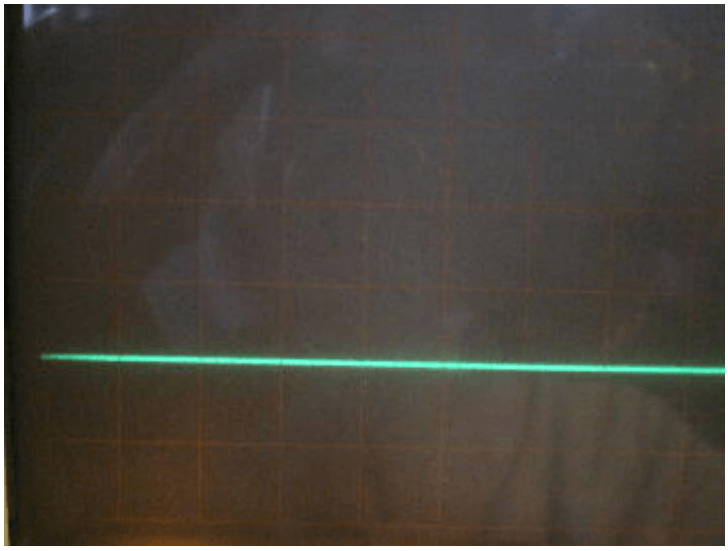
```
    //
    display(Prescale, duty);
    //
    while (1) {
        //□□□変換データをduty値として使用する。
        if (Button(SW1, 10, ON) == TRUE) {
            duty = Adc_Read(1);
            display(Prescale, duty);
            Pwm_Change_DutyEx(duty);
        }
        //duty値をインクリメントする
        if (Button(SW2, 10, ON) == TRUE) {
            while (Button(SW2, 10, OFF) == FALSE)
                ;
            //
            duty++;
            duty = (duty < 1024) ? duty : 0;
            display(Prescale, duty);
            Pwm_Change_DutyEx(duty);
        }
        //duty値をデクリメントする
        if (Button(SW3, 10, ON) == TRUE) {
            while (Button(SW3, 10, OFF) == FALSE)
                ;
            //
            duty--;
            duty = (duty >= 0) ? duty : 1023;
            display(Prescale, duty);
            Pwm_Change_DutyEx(duty);
        }
        //TIMER2のプリスケール値を変更する。
        if (Button(SW4, 10, ON) == TRUE) {
            while (Button(SW4, 10, OFF) == FALSE)
                ;
            //
            Prescale++;
            Prescale = (Prescale < 3) ? Prescale : 0;
            switch (Prescale) {
            case 0:
                T2CON.T2CKPS1 = 0;
                T2CON.T2CKPS0 = 0;
                break;
            case 1:
                T2CON.T2CKPS1 = 0;
                T2CON.T2CKPS0 = 1;
                break;
            case 2:
                T2CON.T2CKPS1 = 1;
                T2CON.T2CKPS0 = 0;
                break;
            }
```

```
                display(Prescale, duty);
            }
        }
}

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

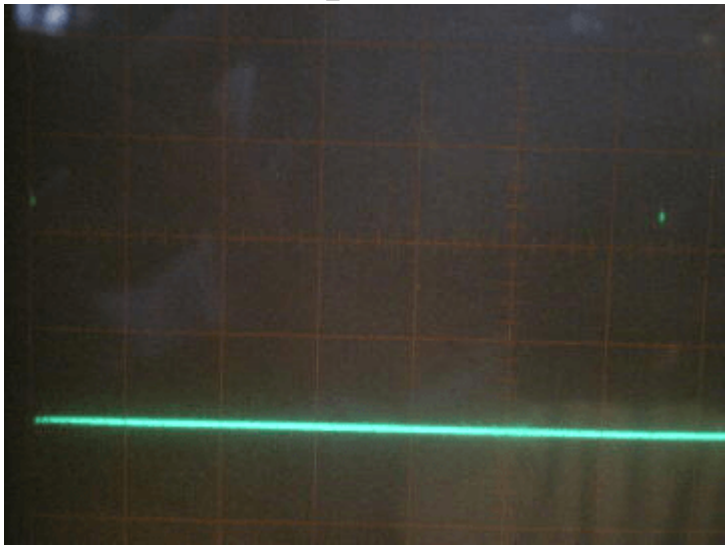
動作確認

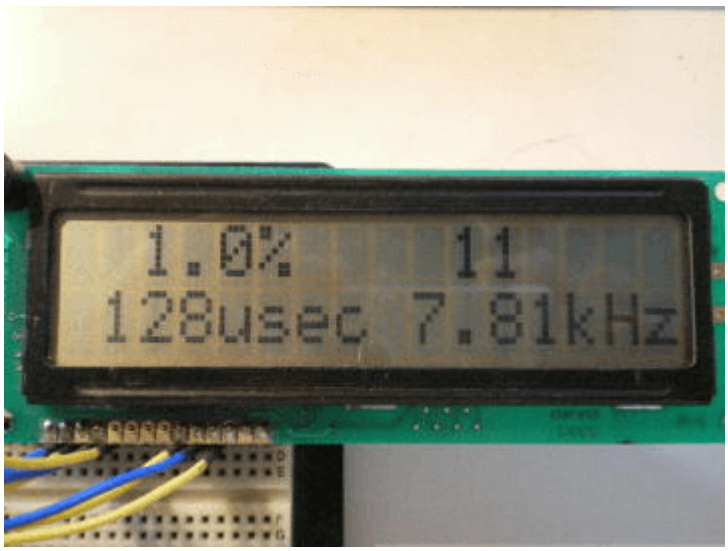


周期=128usec(7.81kHz)□デューティ=0% LCDの左上=デューティ比率 LCDの右上=デューティ値 LCDの左下=周期(時間) LCDの右下=周期(周波数)

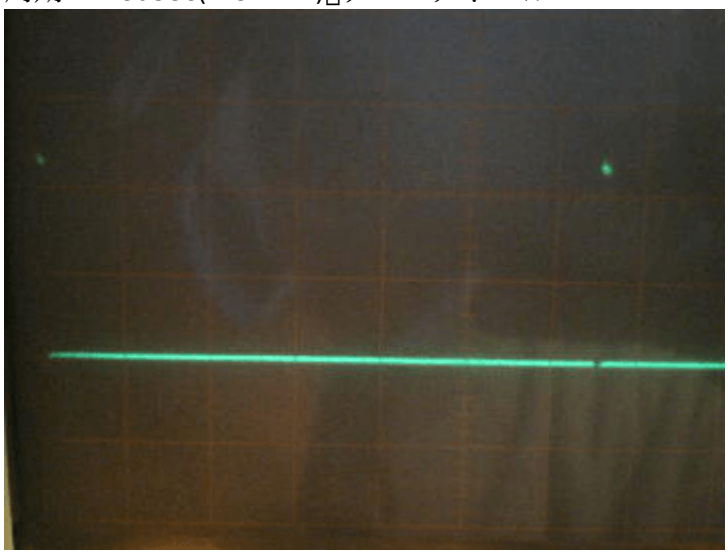


周期=128usec(7.81kHz)□デューティ=0.1%

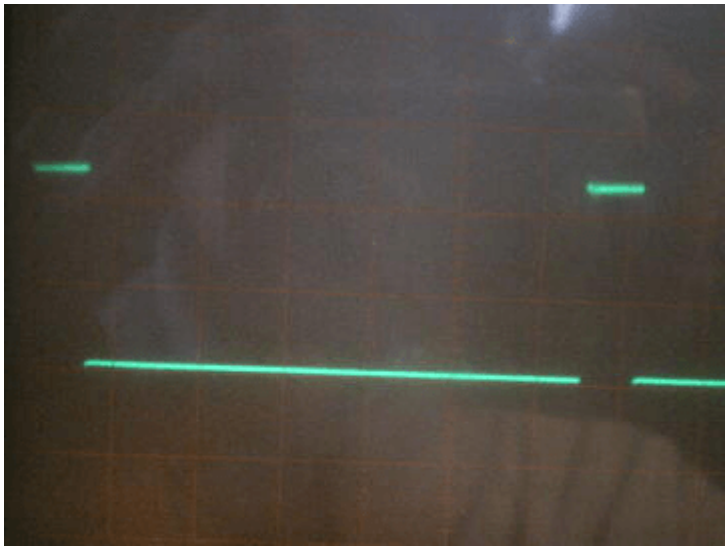




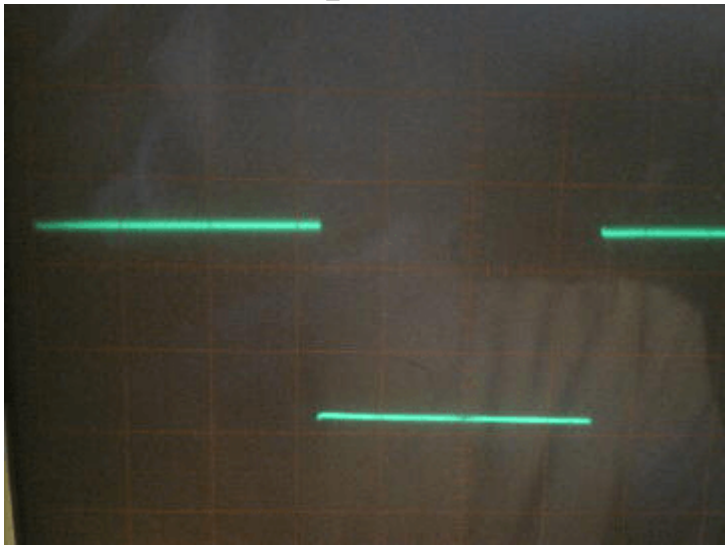
周期=128usec(7.81kHz)、デューティ=1%



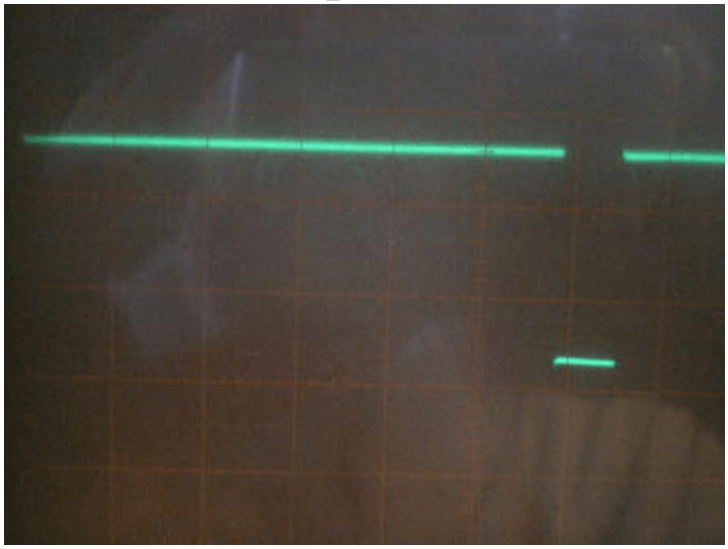
周期=128usec(7.81kHz)、デューティ=10%



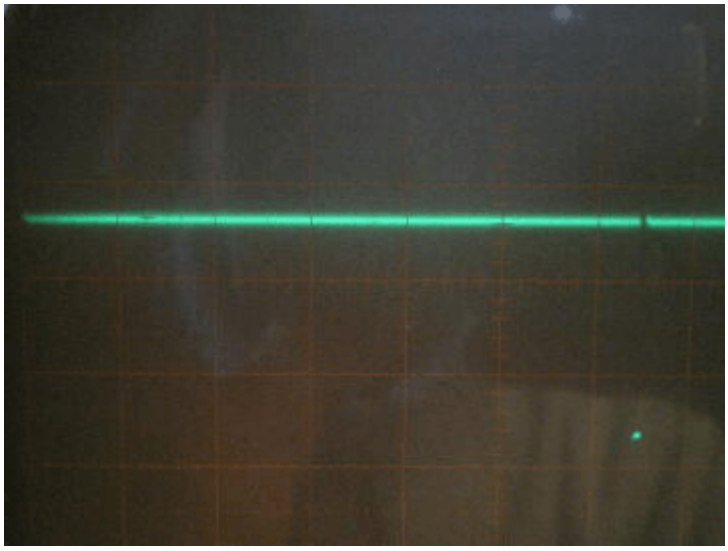
周期=128usec(7.81kHz)□デューティ=50%



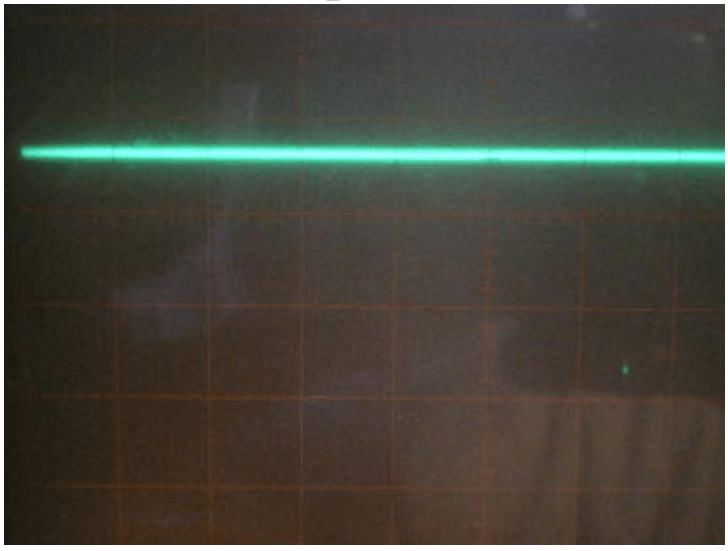
周期=128usec(7.81kHz)、デューティ=90%



周期=128usec(7.81kHz)、デューティ=99%



周期=128usec(7.81kHz)□デューティ=99.9%



左側:周期=512usec(1.95kHz)□デューティ=50% 右側:周期=2048usec(488.3Hz)□デューティ=50%