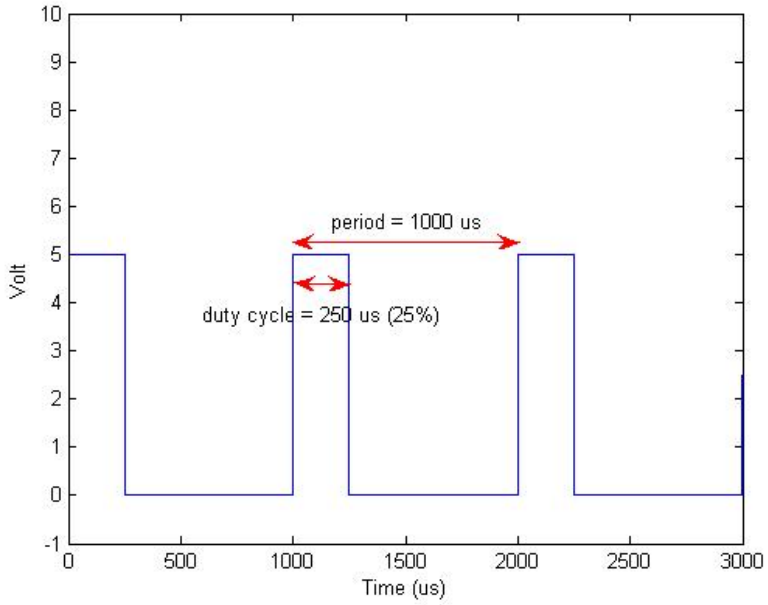
Project 4 PWM by Timer Interrupt

โจทย์

สร้างวงจรและเขียนโปรแกรม PIC เพื่อสร้างสัญญาณ PWM ดังแสดงในรูปที่ 1 โดยใช้ Interrupt ของ Timer Module ซึ่งต่างกับใน Project 2 ซึ่งใช้ PWM Module ในการสร้างสัญญาณ PWM

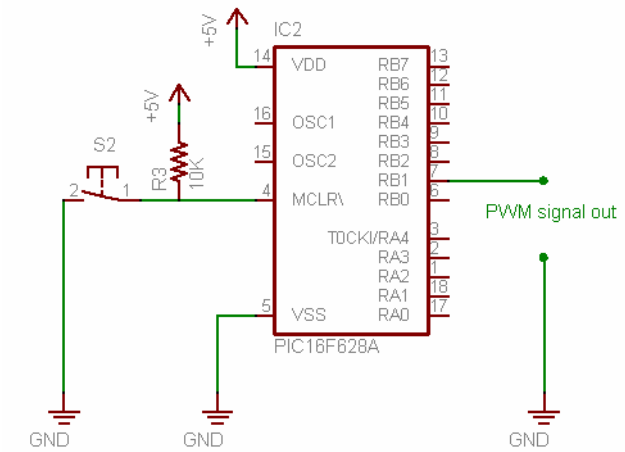


รูปที่ 1 สัญญาณ PWM ที่ต้องการสร้าง

ขั้นตอนการปฏิบัติ

1) การต่อวงจร

ให้ต่อวงจรดังแสดงในรูปที่ 2 โดยจะเห็นว่าเราใช้ขา RB0 สำหรับให้สัญญาณ PWM ซึ่งขา RB0 นี้เป็นขา I/O ธรรมดา ไม่ได้มีฮาร์ดแวร์เฉพาะที่สำหรับสร้าง PWM ดังเช่นขา RB3/CCP1 ที่เราใช้ใน Project 2



รูปที่ 2 แผนผังวงจรสำหรับ Project 4

2) การพัฒนาโปรแกรม

<u>สิ่งที่ควรทราบก่อนลงมือเขียนโปรแกรม</u>

PIC16F628 มี timer ทั้งหมด 3 ตัว (ดู data sheet หัวข้อ 6.0-8.0) ซึ่งสามารถนำมาใช้ในการทำ Power Width Modulation ได้ โดยใช้ timer เพียง 1 ตัว หรือ 2 ตัว ก็ได้ขึ้นอยู่กับการออกแบบโปรแกรม ใน project นี้จะแนะนำการใช้ timer 2 ตัว โดยเลือกใช้ Timer0 และ Timer1 โดยมีขั้นตอนดังนี้

1. กำหนดให้ Timer0 เป็นส่วนควบคุมระยะ duty cycle

Timer0 เป็นตัวนับขนาด 8 bit, ใน timer mode จะ increment ด้วยความเร็ว Fosc/4 = 1 us ดังนั้นในกรณีที่ไม่มีการกำหนดค่า prescale value timer0 จะสามารถจับเวลาได้นานที่สุด 255 us หากต้องการจับเวลานานกว่านี้สามารถกำหนดค่าผ่าน OPTION<3:1> register (PS2:PS0) ตามตารางในหัวข้อ 3.2.2.2 หน้า 20 และจะต้องเปิดการใช้งาน prescale value โดยกำหนดให้ PSA=0 ด้วย

จากโจทย์เราต้องการให้มี duty cycle = 250 us ดังนั้นจะต้อง increment เป็นจำนวน = 250 us/1 us = 250 ครั้ง ซึ่งยังไม่เกิน 255 จึงไม่จำเป็นต้องใช้งาน prescale value

Register ที่เกี่ยวข้องและจำเป็นต้องกำหนดค่ามีดังนี้

GIE = 1 เพื่อเปิดให้ interrupt ทั้งหมดสามารถใช้งานได้

T0CS = 0 เพื่อเลือกให้ timer0 ทำงานใน timer mode (1 คือ counter mode)

PSA = 1 เมื่อไม่ต้องการใช้งาน prescale value

PS2:PS0 กำหนดเมื่อใช้งาน prescale value เช่น 010 = 1:8 เป็นต้น

TMR0 = 0x05 (5) ตัวเลขตั้งต้นในการเริ่มนับ (จากโจทย์ 255-250 = 5)

T0IE ปิด/เปิดการใช้งาน timer0 interrupt

T0IF = 0 clear flag ที่เกิดขึ้นจากการเกิด timer0 interrupt

2. กำหนดให้ Timer1 เป็นส่วนควบคุมระยะ period

Timer1 module ประกอบด้วย TMR1H และ TMR1L register จึงมีขนาดรวมกันถึง 16 bit, ใน timer mode สามารถเลือกใช้ external clock ได้ แต่ถ้าต้องการใช้ internal clock จะต้องกำหนดให้ TMR1CS = 0 จะทำให้ increment ด้วยความเร็ว = Fosc/4 = 1 us ดังนั้นในกรณีที่ค่า prescale value = 1:1 timer1 จะสามารถจับเวลาได้นานที่สุด 65535 us หากต้องการจับเวลานานกว่านี้สามารถกำหนดค่าผ่าน T1CON<5:4> register (T1CKPS1:T1CKPS0) ในหน้า 46

จากโจทย์เราต้องการให้มี period = 1000 us ดังนั้นจะต้อง increment เป็นจำนวน = 1000 us/1 us = 1000 ครั้ง ซึ่งยังไม่เกิน 65535 จึงกำหนดค่า prescale value เพียง 1:1 โดยให้ T1CKPS1=0 และ T1CKPS0=0

Register ที่เกี่ยวข้องและจำเป็นต้องกำหนดค่ามีดังนี้

PEIE = 1 เพื่อเปิดให้อุปกรณ์ในส่วน peripheral interrupt สามารถใช้งานได้

TMR1IE = 1 เปิดการใช้งาน timer1 interrupt

TMR1CS = 0 เพื่อเลือกให้ timer1 ทำงานใน timer mode (1 คือ counter mode)

T1SYNC = 1 เมื่อไม่ต้องการให้ใช้สัญญาณนาฬิกาจากภายนอก

T1CON<5:4> = 00 กำหนดค่า prescale value (00 = 1:1)

TMR1L = 0x17 ตัวเลขตั้งต้นในการเริ่มนับ สำหรับ 8 bit ล่าง

TMR1H = 0xfc ตัวเลขตั้งต้นในการเริ่มนับ สำหรับ 8 bit บน

TMR1ON = 1 เมื่อต้องการให้เริ่มจับเวลา

3. สร้างฟังก์ชัน main เพื่อใช้ในการกำหนดค่าเบื้องต้นให้กับ Timer0 และ Timer1 ในตอนเริ่มโปรแกรม และ infinity loop เพื่อรองรับการทำงานในขณะที่ไม่มีการเกิด interrupt
4. สร้างฟังก์ชันสำหรับรองรับการเกิด interrupt (ดูตัวอย่างได้จาก project 3) แต่ในครั้งนี้จะต้องมี if clause เพื่อตรวจสอบว่าเป็นสัญญาณ interrupt จากแหล่งใดจะต้องทำงานอะไรเมื่อเกิด interrupt นั้น

โปรแกรม project04.c

```
#include<pic.h>
__CONFIG(UNPROTECT & LVPDIS & BOREN & MCLREN & PWRTDIS & WDTDIS & INTIO );

main(void)
{
   TRISB1=0;
   GIE=1;     //Enable Global Interrupt

   //T0IE=0;         //Disable Timer0 Interrupt (first time)
   T0CS=0;    //Set Timer0 Module as timer mode
   PSA=1;     //Disable prescale value option

   PEIE=1;    //Enable Peripheral Interrupt
   TMR1IE=1;         //Enable Timer1 Interrupt
   TMR1CS=0;         //Use internal clock (Fosc)
   T1SYNC=1;         //Do not synchronize external clock input
   T1CKPS0=0;        //Set Prescale value as 1:1
   T1CKPS1=0;        //           ~
   TMR1L=0x17;       //Set Start value of timer1 (Low Byte)
   TMR1H=0xfc;       //           ~                    (Hi Byte)
   TMR1ON=1;         //Start counting of timer1

   while(1)
   {
   }
}
static void interrupt isr(void)
{
   if(T0IF==1)
   {
      RB1=0;         //Stop duty period
      T0IE=0;        //Disable Timer0 Interrupt
   }
   if(TMR1IF==1)
   {
      TMR1IF=0;             //Clear timer1 interrupt flag
      TMR1L=0x17;    //Set Start value of timer1 (Low Byte)
      TMR1H=0xfc;    //            ~                    (Hi Byte)
      TMR0=0x05;            //Set Start value of timer0
      T0IF=0;        //Clear timer1 interrupt flag
      T0IE=1;        //Enable timer0 interrupt
      RB1=1;         //Start duty period
   }
}
```

คำอธิบายโปรแกรม

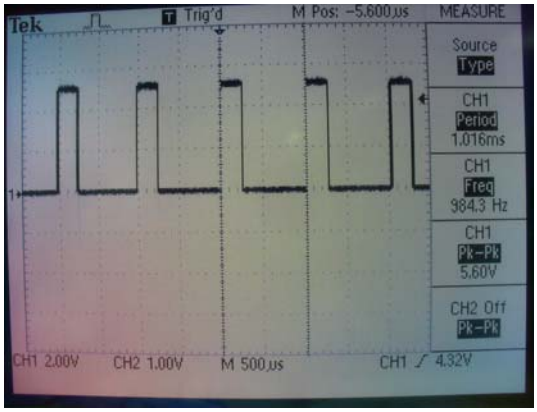
ในโปรแกรมนี้จะเริ่มต้นในฟังก์ชัน main ซึ่งจะเป็นส่วนกำหนดค่าเริ่มต้นต่างๆ ได้แก่ port output สำหรับสัญญาณ PWM ที่ต้องการ, กำหนดค่าที่จำเป็นให้กับ register ที่เกี่ยวข้องกับ Timer0 และ Timer1 รวมถึง infinity loop ด้วย ในส่วนของ interrupt routine function จะถูกแบ่งด้วย if clause ว่าเป็น interrupt จาก Timer0 หรือ Timer1 ถ้าเป็นของ Timer0 โปรแกรมจะปรับสัญญาณ PWM ให้เป็น 0 และยกเลิกการทำงานของ

Timer0 interrupt เนื่องจาก Timer0 ไม่มี register ที่ใช้ควบคุมการเปิด-ปิดวงจรนับ ถ้าไม่ยกเลิกจะทำให้เมื่อ Timer0 นับครบรอบแล้ว T0IF จะมีค่าเป็น 1 ทำให้ต้องเข้ามาทำงานใน interrupt routine อีก

ถ้าเป็นส่วนของ Timer1 ปรแกรมจะปรับสัญญาณ PWM ให้เป็น 1 และทำการตั้งเวลา Timer0 และ Timer1 ให้กลับมาเริ่มต้นตามที่คำนวณไว้อีกครั้ง จากนั้นจึงสั่งให้เริ่มจับเวลาใหม่ โดยมีข้อสังเกตว่าจะต้อง clear flag ของ Timer0 interrupt เสียก่อนจะทำการเปิดใช้งาน T0IE เพื่อให้แน่ใจว่าไม่มี flag ค้างไว้ ซึ่งจะทำให้วนกลับมาทำงานใน interrupt routine อีกในทันที

3) ผลการทำงานของโปรแกรม

รูปที่ 3 แสดงสัญญาณ PWM ที่สร้างขึ้นโดยใช้ timer interrupt ซึ่งให้ค่า period และ duty cycle ถูกต้องตามที่เราต้องการ



รูปที่ 3 สัญญาณ PWM ที่ได้

ข้อดีของการใช้ timer interrupt คือทำให้เราสามารถสร้างสัญญาณ PWM มากกว่าหนึ่งช่องสัญญาณได้ เนื่องจาก PIC ส่วนมากจะมีโมดูล PWM เพียงอันเดียว แต่มักจะมี timer 3 ตัว ซึ่งจะทำให้สามารถสร้างสัญญาณ PWM 2 ช่องที่มี period เดียวกัน แต่มี duty cycle ไม่เท่ากันได้