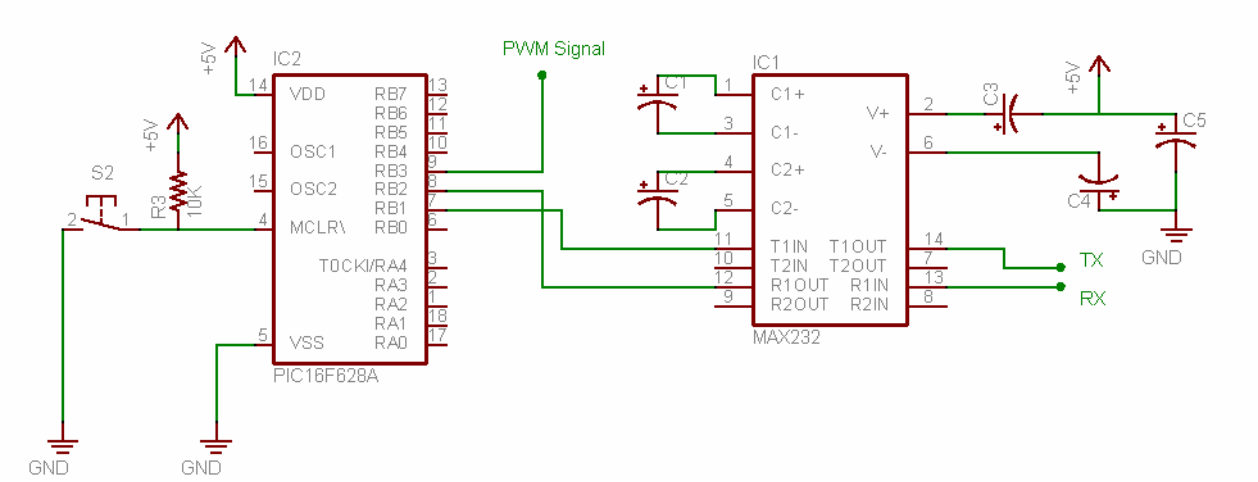
Project 5 USART (Serial Communication)

โจทย์

สร้างวงจรสำหรับติดต่อสื่อสารแบบอนุกรมโดยใช้ IC MAX232 โดยจะให้ PIC ติดต่อกับคอมพิวเตอร์ผ่าน serial port (RS232) เมื่อใช้โปรแกรม Hyper Terminal ที่มีอยู่ในคอมพิวเตอร์ส่งตัวอักษร a, b, c หรือ d ไป ให้ PIC รับข้อมูลตัวอักษรดังกล่าว แล้วตอบสนองโดยการสร้างสัญญาณ PWM ที่มี duty cycle ที่ 25%, 50%, 75% และ 100% ตามลำดับโดยจะใช้วิธีการสร้าง PWM แบบใดก็ได้ (ใน Project นี้ใช้วิธีแบบ Project ที่ 2)

ขั้นตอนการปฏิบัติ

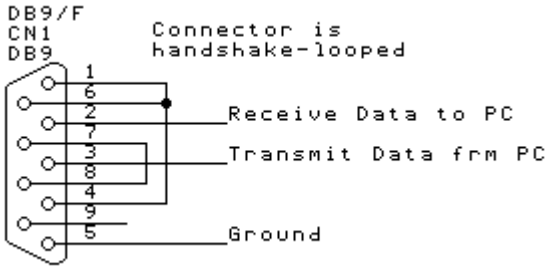
1) การออกแบบวงจร



รูปที่ 1 แผนผังวงจรควบคุม PWM ด้วยสัญญาณอนุกรมจากคอมพิวเตอร์

ในแผนผังวงจรในรูปที่ 1 เป็นการออกแบบวงจรเพิ่มเติมจากวงจรใน Project ที่ 2 โดยเพิ่มส่วนเชื่อมต่อสัญญาณอนุกรม เราใช้ IC MAX232 ช่วยในการปรับแรงดันสัญญาณให้เป็นไปตามมาตรฐาน IEEE XXX (มาตรฐานแรงดันไฟ = 10-0V)

2) การต่อสายสัญญาณอนุกรมเข้ากับหัวเชื่อมชนิด DB9



รูปที่ 2 แผนผังการต่อสายสัญญาณอนุกรมเข้ากับหัวเชื่อมชนิด DB9

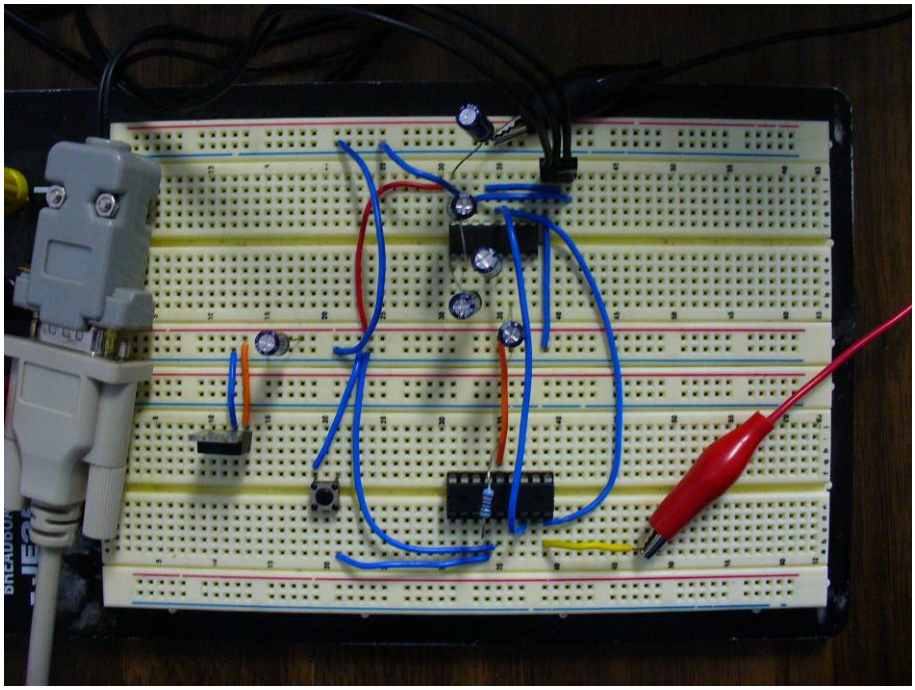
จุดสำคัญคือ

ขาที่ 2 นำไปต่อกับสัญญาณ TX จาก PICa

ขาที่ 3 นำไปต่อกับสัญญาณ RX จาก PIC

ขาที่ 5 เป็นกราวร่วม

3) การต่อวงจร



รูปที่ 3 รูปแสดงการต่อวงจร

4) การเขียนโปรแกรม

การเขียนโปรแกรมเริ่มจากการกำหนดค่าเริ่มต้นที่จำเป็นได้แก่การกำหนด I/O ของพอร์ตที่จะใช้งาน การกำหนด SFR register สำหรับ PWM module, Interrupt logic (หัวข้อ 14.6 หน้า 101) และ USART module

การเขียนโปรแกรมเพื่อสื่อสารด้วยสัญญาณอนุกรมสามารถทำได้ผ่าน USART module (data sheet หัวข้อ 12 หน้า 67) ใน Project นี้จะใช้งานในลักษณะ Asynchronous Mode มีการรับและส่งข้อมูลได้พร้อมกัน และใช้ interrupt ช่วยในการรับและส่งข้อมูล จะต้องกำหนด SFR register ดังนี้

TXSTA สำหรับควบคุมและแสดงสถานะ การส่งข้อมูล

- TX9 = 0 ส่งข้อมูลแบบ 8-bit
- TXEN = 1 อนุญาตให้สามารถส่งข้อมูลออกได้
- SYNC = 0 เลือกเป็น Asynchronous mode
- BRGH = 1 ส่งข้อมูลด้วยความเร็วสูง

RCSTA สำหรับควบคุมและแสดงสถานะ การรับข้อมูล

- SPEN = 1 ให้ใช้ขาที่ 7 และ 8 ของ PIC เป็นขาสัญญาณอนุกรม
- RX9 = 0 รับข้อมูลแบบ 8-bit
- CREN = 1 อนุญาตให้สามารถรับข้อมูลได้อย่างต่อเนื่อง

SPBRG = 12 ให้ความเร็วในการรับส่งข้อมูล 19200 bps

การคำนวณค่า SPBRG เพื่อให้ได้ความเร็วในการรับส่งข้อมูลตามที่ต้องการ หาได้จากสูตร

ความเร็วที่ต้องการ = F_{osc} / (64 x (SPBRG+1)) (BRGH=0)

ความเร็วที่ต้องการ = F_{osc} / (16 x (SPBRG+1)) (BRGH=1)

สำหรับใน Project นี้ใช้ Oscillator ภายใน มีการตั้งให้ BRGH = 1 เมื่อต้องการให้มีความเร็วในการรับส่งข้อมูลที่ 19200 bps สามารถคำนาณได้ดังนี้

$$19200 = (4x10^6) / (16 \times (SPBRG+1))$$

$$SPBRG = ((4x10^6) / (16 \times 19200)) -1$$

$$SPBRG = 12.02$$

จึงให้ SPBRG = 12

เมื่อคำนวณกลับจะได้ ค่าความเร็ว = 19230.76 bps จึงมี error = (19230.76-19200) x 100/19200 = 0.16 %

ทั้งนี้จะต้องระวังไม่ให้มี error มากเกินไป จะทำให้การสื่อสารผิดพลาด ใน Project นี้เลือกใช้ที่ความเร็ว 19200 bps ซึ่งเป็นความเร็วสูงสุดที่เป็นไปได้ ซึ่งไม่ทำให้ข้อมูลเกิดความผิดพลาด

ส่วนของ interrupt service routine จะมีเงื่อนไขสำหรับตรวจสอบ interrupt flag 2 ส่วนด้วยกันคือ สำหรับการรับข้อมูล (RCIF flag) และสำหรับการส่งข้อมูล (TXIF flag) แต่เนื่องจากเวลาโดยส่วนใหญ่ buffer สำหรับส่งข้อมูลจะว่างจากการส่งข้อมูล ดังนั้น TXIF flag จึงถูก set อยู่เกือบตลอดเวลาอันจะทำให้เกิด interrupt ขึ้นตลอดเวลาเช่นกัน จึงต้องตรวจสอบร่วมกับ TXIE register ว่าขณะนั้นได้มีการเปิดใช้งาน interrupt สำหรับส่งข้อมูลหรือไม่

```
void interrupt isr(void)      //interrupt service routine
{
        if(RCIF)              //if recive flag is set
        {
                ...//โปรแกรมที่ต้องการให้ microcontroller ทำเมื่อ buffer สำหรับรับข้อมูลเต็ม
        }
        if(TXIF&&TXIE)        //if transmit flag is set
        {
                ...//โปรแกรมที่ต้องการให้ microcontroller ทำเมื่อ buffer สำหรับส่งข้อมูลว่าง
        }
}
```

โปรแกรม project05.c

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

char Rdata,*Tdata;
int prompt=1;
int i=0;
void setup(void);
void transmit(char str[]);

void main(void)
{
   setup();
   while(1)                        //start infinity loop
   {
      if (prompt)                  //if prompt is signaled
      {
         char str[]="\n\rcommand prompt: ";
         prompt=0;                 //clear prompt status
         transmit(str);            //transmit "command prompt: " to PC
      }
   }
}
void transmit(char str[])
```

```
{
   Tdata=str;                      //set string-pointer to transmit-buffer_memmory
   TXIE=1;                         //Enable transmit flag
}
void interrupt isr(void)           //interrupt service routine
{
   if(RCIF)                        //if receive flag is set
   {
      Rdata=RCREG;                 //move data from buffer to memory
      TXREG=Rdata;                 //repeat input signal
      switch(Rdata)
      {
         case 'a' :  CCPR1L=0x3E;  //set register for PWM duty cycle bit 2-9
                     CCP1X=1;      //                  ~             bit 1
                     CCP1Y=0;      //                  ~             bit 0
                     break;

         case 'b' :  CCPR1L=0x7D;
                     CCP1X=0;
                     CCP1Y=0;
                     break;

         case 'c' :  CCPR1L=0xBB;
                     CCP1X=1;
                     CCP1Y=0;
                     break;

         case 'd' :  CCPR1L=0xFA;
                     CCP1X=0;
                     CCP1Y=0;
                     break;
      }
      prompt=1;                    //signal to show command prompt
   }
   if(TXIF&&TXIE)                  //if transmit flag is set
   {
      TXREG=*Tdata;                //put next character to transmit register
      if(!*Tdata++) TXIE=0;        /*increment to next character,
                                     if end of string -> disable transmit interrupt*/
   }
}
void setup(void)
{
   // I/O port
         PORTB=0x00;        //clear port B
         TRISB=0x02;        //set bit RB1 to be input and RB2-RB3 output
   // USART interrupt
         PIE1 =0x00;        //Enable peripheral interrupt
         RCIE =1;           //Enable receive interrupt flag
         GIE  =1;           //Enable global interrupt
         PEIE =1;           //Enable
   // setup USART
         SPBRG=12;          //set baud rates to 19200 bps
         TXSTA=0x24;        //Enable 8-bit asynchronous transmition with high baud rates
         RCSTA=0x90;        /*set RB1-RB2 as USART mode and
                              enable 8-bit asynchronous continuous receiving*/
   // setup PWM mode
         PR2=249;           //set register for PWM period
         CCP1M3=1;          //set CCP1CON as PWM mode bit 3
         CCP1M2=1;          //                        ~             bit 2
         T2CKPS1=0;         //set Timer2 prescale value at T2CON bit 1
         T2CKPS0=1;         //                              ~
bit 0
         TMR2ON=1;          //start pulse
}
```

<u>คำอธิบายโปรแกรม</u>

ในโปรแกรมนี้แบ่งได้เป็น 4 ส่วน ดังนี้

ส่วนแรกคือการปรับตั้งค่า SFR register ต่างๆ โดยใช้ฟังก์ชัน setup(); ประกอบด้วย การเปิด port I/O, การตั้งค่า interrupt, การตั้งค่าสำหรับ USART module และการตั้งค่าสำหรับ PWM mode

ในฟังก์ชัน main(); จะประกอบด้วย infinity loop ซึ่งจะตรวจสอบสถานะว่าพร้อมจะรับคำสั่งใหม่หรือไม่ ก็จะเรียกฟังก์ชันสำหรับส่งข้อมูล transmit(char str[]); เพื่อส่งข้อความ command prompt : เพื่อบอกให้ผู้ใช้ทราบว่าเครื่องพร้อมจะรับคำสั่งแล้ว

ฟังก์ชัน transmit ก็จะส่ง address(ในหน่วยความจำ) ของข้อความที่ต้องการส่งไปยัง *Tdata ซึ่งกำหนดให้ใช้เสมือนเป็นตัวกลางสำหรับส่งข้อมูลไปยัง TXREG register พร้อมทั้งตั้งให้ TXIE = 1 เพื่อให้เริ่มส่งข้อมูลได้ทันทีหาก TXREG register ว่าง (TXIF=1)

ในส่วน interrupt service routine ก็จะตรวจสอบว่าเป็น flag สำหรับการรับหรือการส่งข้อมูล หากเป็นการรับข้อมูล (RCIF=1) จะต้องอ่านข้อมูลจาก RCREG เสียก่อนที่ข้อมูลจะหายไปโดยในโปรแกรมนำไปเก็บไว้ที่ตัวแปร Rdata พร้อมทั้งส่งกลับข้อมูลนั้นเพื่อเป็นการแจ้งให้ผู้ส่งทราบว่ารับข้อมูลอะไรมา จากนั้นจึงตรวจสอบว่าข้อมูลที่รับเข้ามานั้นตรงกับคำสั่งใดหรือไม่ (ข้อมูลเป็นตัวอักษร a, b, c หรือ d) ก็จะทำตามคำสั่งนั้นโดยการเปลี่ยนค่า duty cycle ของ PWM module ตามที่โจทย์สั่งในตอนต้น เมื่อเสร็จแล้วก็ส่งสัญญาณให้ทราบว่าพร้อมจะทำคำสั่งถัดไปแล้วด้วยการตั้งให้ตัวแปร prompt=1 ถ้าเป็น flag สำหรับส่งข้อมูล (TXIF=1) จะต้องตรวจสอบคู่กับ TXIE ตามที่กล่าวไปแล้วด้านบน เพื่อเป็นการยืนยันว่าเป็น flag สำหรับส่งข้อมูลจริง โปรแกรมในส่วนนี้จะเป็นการนำข้อมูลใน address ที่ตัวแปร Tdata ชี้อยู่ ส่งให้ TXREG register จากนั้นทำการ increment Tdata แล้วตรวจสอบว่าสิ้นสุดข้อความ(หรือข้อมูล) แล้วหรือยัง หากสิ้นสุดแล้วให้ยกเลิก interrupt สำหรับการส่งข้อมูลโดยให้ TXIE=0 เพื่อว่าหากมีการเข้า interrupt service routine อีกในคราวหน้าจะไม่มีการส่งข้อมูลใดออกไปผ่านการทำงานของ interrupt อีก ซึ่งข้อมูลที่ส่งออกไปอาจเป็นข้อมูลใดๆ ที่ตัวแปร Tdata ชี้อยู่ก็ได้

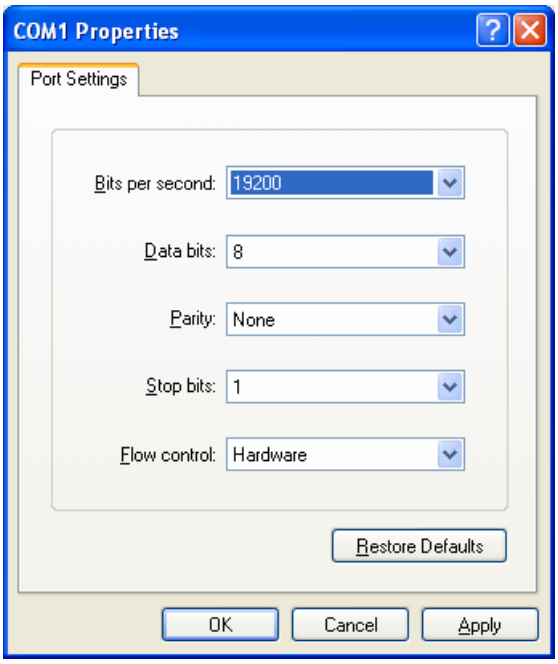
5) การทดสอบการทำงาน

ต่อวงจรตามแผนผังวงจรข้างต้นดังแสดงในรูปที่ 1 เปิดโปรแกรม Hyper Terminal โดยเข้าไปที่ Start -> All Programs -> Accessories -> Communications -> Hyper Terminal เมื่อเปิดโปรแกรมได้แล้วให้ทำตามขั้นตอนต่อไปนี้

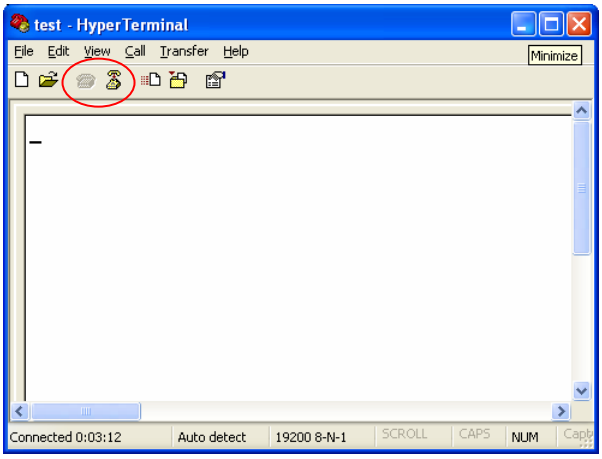
1. ใส่ชื่อสำหรับการติดต่อครั้งนี้ เป็นชื่ออะไรก็ได้, กด OK

2. เลือก serial port ที่ต้องการ (บางเครื่องที่มี 2 หรือมากกว่า 2 port จะต้องเลือกให้ตรงกับ port ที่เราจะใช้เชื่อมต่อกับ PIC เช่น COM2 COM3 ...), กด OK

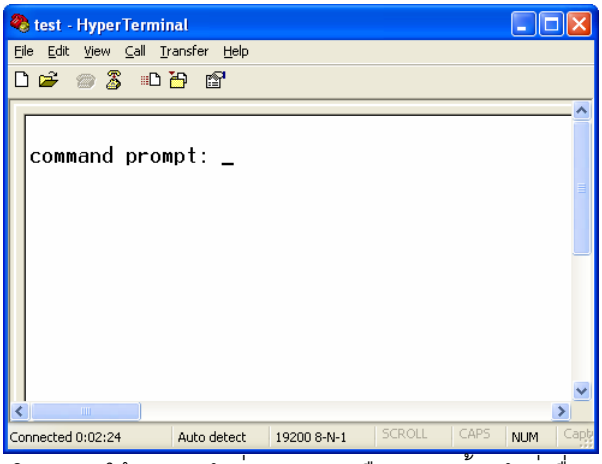
3. เลือก Bits per second ให้ตรงกับ Baud rate ที่เราได้คำนวณไว้ในขั้นตอนการเขียนโปรแกรม (19200 bps), กด OK



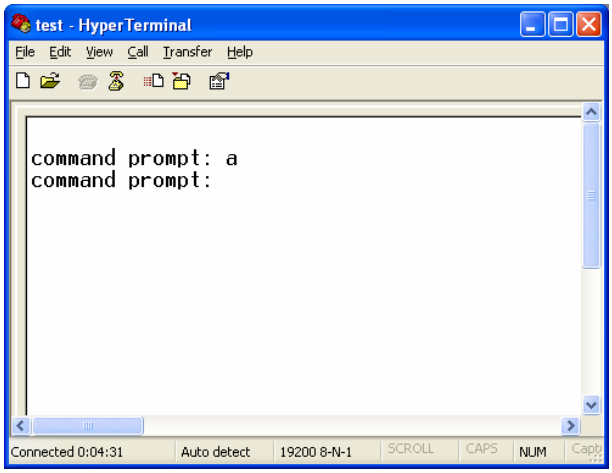
4. เมื่อตั้งค่าต่างๆ เสร็จเรียบร้อย จะได้หน้าต่าง test – Hyper Terminal ขึ้นมา ซึ่งโปรแกรมได้ทำการ connect ให้แล้วโดยอัตโนมัติ โดยสังเกตได้จากรูปโทรศัพท์บนกล่องเครื่องมือ

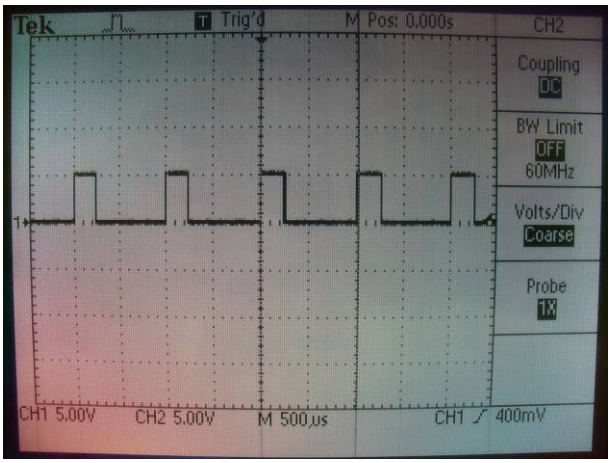


5. เสียบสายสัญญาณระหว่าง PC และ PIC ในกรณีที่ PIC มีการส่งข้อความมายัง PC จะแสดงข้อความนั้นในหน้าต่างของโปรแกรม แต่การส่งสัญญาณออกจาก PC จะไม่แสดงผลบนหน้าต่างนอกจากอุปกรณ์ปลายทางจะมีการทวนสัญญาณกลับมายัง PC จึงจะเห็นข้อความบนหน้าต่างโปรแกรม สำหรับ Project นี้ได้เขียนโปรแกรมให้ PIC ส่งข้อความมายัง PC ทันทีเมื่อพร้อมรับคำสั่ง เมื่อเสียบสายสัญญาณจึงได้ผลดังนี้

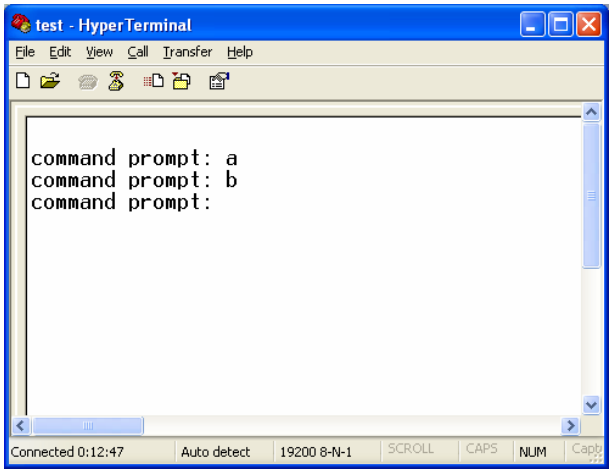


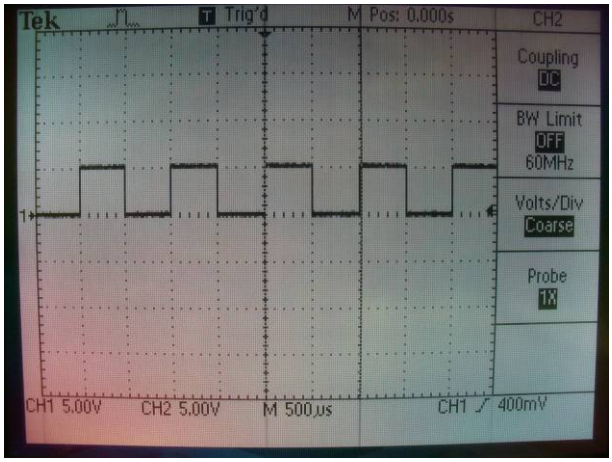
6. สำหรับ Project นี้ได้เขียนโปรแกรมให้รอบรับคำสั่ง a, b, c หรือ d เท่านั้น คำสั่งอื่นๆ ยังคงทวนสัญญาณกลับแต่จะไม่มีการทำงานใดๆ เกิดขึ้น ซึ่งได้ผลการทดลองดังนี้



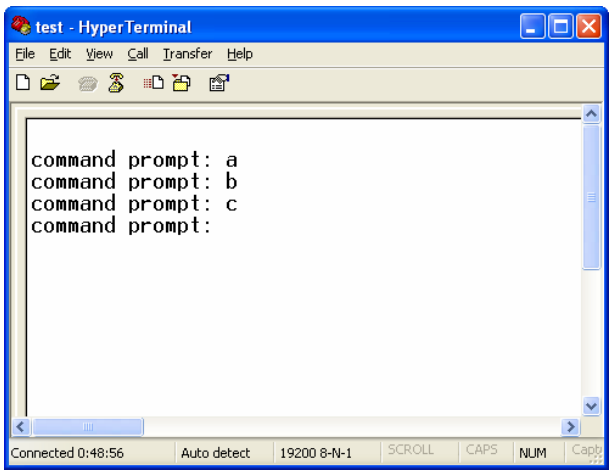


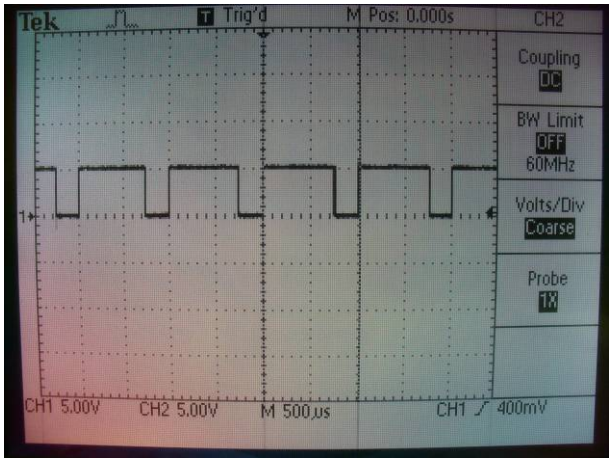
รูปที่ 4 รูปแสดงผลจากคำสั่ง a



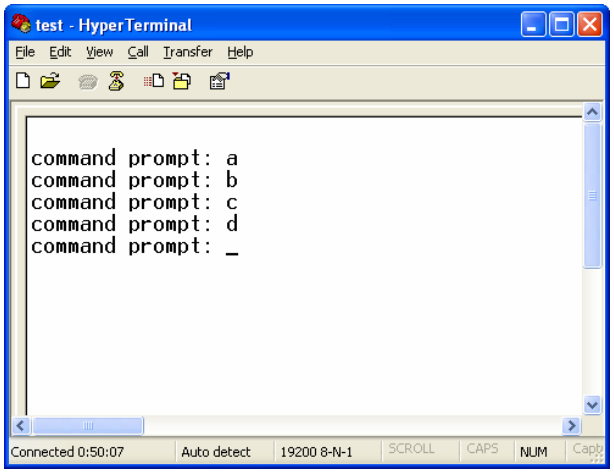


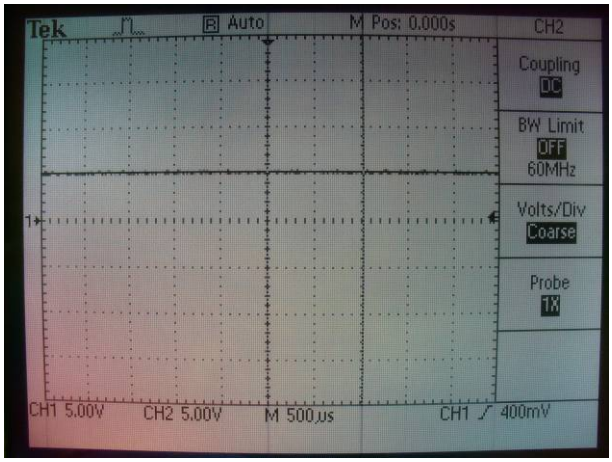
รูปที่ 5 รูปแสดงผลจากคำสั่ง b





รูปที่ 6 รูปแสดงผลจากคำสั่ง c





รูปที่ 7 รูปแสดงผลจากคำสั่ง d