
# 423307 คู่มือปฏิบัติการไมโครโปรเซสเซอร์

**จัดทำโดย**

อาจารย์วิชัย ศรีสุรักษ์
ห้องปฏิบัติการไมโครโพรเซสเซอร์
สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
*(ฉบับร่าง2 ปีการศึกษา 1/2552)*

# การทดลองที่ 3

# การคำนวณทางคณิตศาสตร์และคำสั่งเชิงตรรกะ

## 3.1 วัตถุประสงค์

1. เพื่อศึกษาคำสั่งการบวก การลบ การคูณ และการหาร
2. เพื่อศึกษาคำสั่งการคำนวณเชิงตรรกะ
3. เพื่อศึกษาการแปลงตัวเลขฐานสิบหกเป็นฐานสิบ ที่เรียกว่า รหัสบีซีดี (BCD)

## 3.2 อุปกรณ์ทดลอง

1. เครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับจำลองการทำงานของ MCS-51

## 3.3 เนื้อหา ทฤษฎีที่เกี่ยวข้อง

1. การใช้คำสั่งการบวก การลบ การคูณ และการหาร

```
ADD     A, Source
ADDC    A, Source
DA      A
SUBB    A, Source
MUL     AB          ; Result BA
DIV     AB          ; Result A, Remain B
```

2. การใช้คำสั่งการคำนวณเชิงตรรกะ

```
AND     A, Source
ORL     A, Source
XRL     A, Source
CPL     A
```

3. การใช้คำสั่งเกี่ยวกับการหมุนบิต

```
RR      A
RL      A
RRC     A
RLC     A
SWAP    A
XCH     A,B
```

3. การใช้คำสั่งเพื่อศึกษาการแปลงระหว่างเลขฐานสิบหกกับลขฐานสิบ

HEX2BCD ➔ FDH (253D) {Mazidi Ex6.8}

Step1: FDH/0AH ➔ Result =19H Remain =03H

Step2: 19H/0AH ➔ Result =02H Remain =05H

Step3: 02H/0AH ➔ Result =00H Remain =02H

BCD2HEX ➔ 45H (2DH)

High Nipple * 10 + Low Nipple = Result

04H * 0AH + 05H = 2DH

## 3.4 การทดลอง

<u>ตอนที่ 1</u> คำสั่งการบวก ลบ คูณ และหาร

1. โปรแกรม "Lab31.Asm" เป็นโปรแกรมสำหรับรวมค่า R0, R1 และ R2 ทดสอบการทำงานด้วย TS-Emulator
   - ก่อนรันโปรแกรมกำหนดค่า R0=R1=R2=0FFH
   - ผลลัพท์เก็บที่แอดเดรส 20H(MSB)=02H ,21H(LSB)=FDH {FF+FF+FF=02FD}
2. จากโปรแกรม "Lab31.Asm" ให้นักศึกษาปรับปรุงโปรแกรมเพื่อรวมค่าใน R0-R7 แบบไบนารี่ เก็บผลลัพธ์ที่ได้ในหน่วยความจำภายนอกที่ 9500H-9501H และกำหนดให้โปรแกรมเริ่มต้นทำงานที่แอดเดรส 8000H
3. ให้นักศึกษาเขียนโปรแกรมเพื่อดำเนินการลบเลขฐานสิบหก จำนวน 2 ตัว ดังต่อไปนี้
   - ตัวตั้งกำหนดไว้ก่อนรันโปรแกรมที่หน่วยความจำภายนอกตำแหน่ง 8800H
   - ตัวลบกำหนดไว้ก่อนรันโปรแกรมที่หน่วยความจำภายในตำแหน่ง 31H
   - คัดลอกข้อมูลจากตำแหน่ง 8800H มาเก็บไว้ที่หน่วยความจำภายในตำแหน่ง 21H
   - ดำเนินการลบ แล้วเก็บผลลัพธ์ที่ได้ไว้ในหน่วยความจำตำแหน่ง 40H และ 41H
4. ให้นักศึกษาเขียนโปรแกรมบวกเลขฐานสอง จำนวน 4 ไบต์ แบบไบนารี่ โดยมีข้อกำหนด ดังนี้

   ตัวตั้ง = หน่วยความจำภายนอกที่ตำแหน่ง 9001H(MSB) - 9004H(LSB)

   ตัวบวก = หน่วยความจำภายนอกที่ตำแหน่ง 9011H(MSB) - 9014H(LSB)

   ผลลัพธ์ = หน่วยความจำภายนอกที่ตำแหน่ง 9020H(MSB) - 9025H(LSB)

   **หมายเหตุ:** การบวกต้องคำนึงถึงตัวทดระหว่างไบต์ และต้องมีการบันทึกผลลัพธ์ไบต์ที่ 5 ด้วย
5. เขียนโปรแกรมเพื่อทำการคูณเลข 2 จำนวน ตามข้อกำหนด ดังนี้

   ตัวตั้งเป็นเลข 2 ไบต์ที่เก็บใน R4:R3 (R4 =MSB, R3=LSB)

   ตัวคูณเป็นเลข 1 ไบต์ที่เก็บใน R6

   ผลลัพธ์ที่ได้ไปเก็บในหน่วยความจำภายนอก โดยให้ค่าไบต์ต่ำเริ่มที่ตำแหน่ง 9000H และให้โปรแกรมเริ่มต้นที่แอดเดรส 8200H

ตอนที่ 2 คำสั่งการคำนวณเชิงตรรกะและคำสั่งเกี่ยวกับการหมุนบิต

1. การคำนวณเชิงตรรกะส่วนใหญ่ใช้เพื่อต้องการตรวจสอบระดับบิตหรือการจัดการระดับบิตในกรณีที่รีจีสเตอร์ไม่สามารถจัดการระดับบิตได้
   - กรณีที่ต้องการให้รีจีสเตอร์ TMOD เฉพาะ 4 บิตล่างเป็น 0101B จะใช้ชุดคำสั่งดังนี้

     ```
     ANL    TMOD,#11110000B
     ORL    TMOD,#00000101B
     ```

   - กรณีที่ต้องการตรวจสอบว่าให้รีจีสเตอร์ Acc มีค่าเท่ากับ 10110101B หรือไม่

     ```
     XRL    A,#10110101B
     ```

     ผลการทำงานชุดคำสั่งนี้

     ถ้าผลลัพธ์ Acc = 00000000B แสดงว่าก่อนการทำคำสั่งนี้ค่า A=10110101B

     ถ้าผลลัพธ์ Acc $\neq$ 00000000B แสดงว่าก่อนการทำคำสั่งค่า A$\neq$10110101B
2. เขียนโปรแกรมตามโปรแกรม “Lab32.Asm” คอมไพล์และโหลดโปรแกรมด้วย TS-Emulator
3. ทดสอบการทำงานของโปรแกรมแบบ Single Step สังเกตการเปลี่ยนแปลงข้อมูลในหน่วยความจำที่เกี่ยวข้องและบันทึกผลการรันโปรแกรมลงในตารางที่ 3.1

ตอนที่ 3 การแปลงเลขฐานสิบหกเป็นฐานสิบ

1. เขียนนักศึกษาโปรแกรมเพื่อรวมเลขแบบ BCD ระหว่าง DPTR R0 และ R1 รวมเสร็จแล้วให้เก็บค่าผลลัพธ์ที่หน่วยความจำภายในตำแหน่งใดๆก็ได้ {ระหว่าง 30H-7FH }

   ***เช่น*** DPTR=1234H, R0=00H และ R1=11H

   Result = 1234+00+11 = 1245H เป็นต้น
2. ใส่ค่าเริ่มต้นลงในหน่วยความจำในข้อมูลตำแหน่ง 8500H เพื่อเป็นค่าที่ต้องการแปลงจากไบนารี่ (HEX) เป็นเลขบีซีดี(BCD)
3. ให้นักศึกษาเขียนโปรแกรมแปลงเลขไบนารี่ ขนาด 8 บิท ที่เก็บไว้ในหน่วยความจำ 8500H เป็นเลข Packed BCD ขนาด 2ไบต์/3หลัก เก็บไว้ในหน่วยความจำภายนอกแอดเดรส 8510H-8511H
4. ใส่ค่าเริ่มต้นลงในรีจีสเตอร์ R1 เพื่อเป็นค่าที่ต้องการแปลงจากเลขบีซีดี(BCD)เป็นเลขไบนารี่ (HEX) โดยค่านี้ต้องเป็นเลข BCD เท่านั้น
5. ให้นักศึกษาเขียนโปรแกรมแปลงเลขบีซีดีขนาด 8 บิท ที่เก็บไว้ในรีจีสเตอร์ R1 เป็นเลขไบนารี่เก็บไว้ในหน่วยความจำภายในแอดเดรส 30H

## 3.5 วิเคราะห์และอภิปรายผลการทดลอง

1. จงอธิบายการทำงานของชุดคำสั่งแอสเซมบลีที่พัฒนาขึ้นในตอนที่ 1 ข้อ 4 ทีละบรรทัด
2. จงอธิบายการทำงานของชุดคำสั่งแอสเซมบลีที่พัฒนาขึ้นในตอนที่ 1 ข้อ 5 ทีละบรรทัด
3. จงอธิบายผลการทำงานของชุดคำสั่งแอสเซมบลีในตอนที่ 2 ข้อ 3 ทีละบรรทัด
4. จงอธิบายการทำงานของชุดคำสั่งแอสเซมบลีที่พัฒนาขึ้นในตอนที่ 3 ข้อ5 ทีละบรรทัด

## 3.6 คำถามท้ายการทดลอง

1. จากโปรแกรม "Lab31.Asm" ปรับปรุงโปรแกรมเพื่อรวมค่าใน R0-R3 แบบBCD เก็บผลลัพธ์ที่ได้ในหน่วยความจำภายนอกที่ 9510H-9511H และให้โปรแกรมเริ่มต้นที่แอดเดรส 8100H

2. กำหนดให้ป้อนค่าใน Register B ได้ไม่เกิน 16H (22D) ก่อนรันโปรแกรม และให้นักศึกษาเขียนโปรแกรมเพื่อบวกค่าจาก 0 ถึงค่าที่เก็บใน Register B บันทึกค่าที่รวมได้ใน External RAM Address 9000H {ข้อสอบเก่า}

   ***เช่น*** Register B = 0FH ➔ 0 + 1 + 2 + … + 9 + A + B + C + D + E + F = 88H เป็นต้น

3. กำหนดให้ Register B มีค่าอยู่ก่อนแล้ว ให้นักศึกษาทำการนับจำนวนเลข 0 ในค่าใน Register B เก็บผลลัพธ์ที่ได้ใน R7

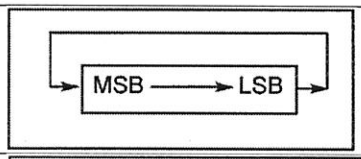
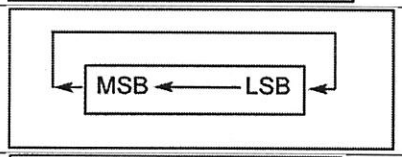
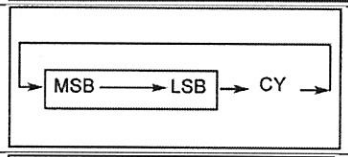
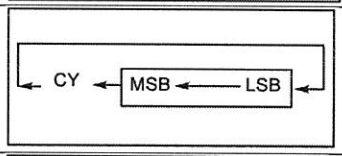
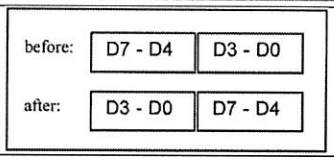
   ***เช่น*** B=F7H ➔ 1111 0111 Run แล้วได้ R7=01H

4. เขียนโปรแกรมเพื่อรวมตั้งแต่ค่าที่เก็บใน Register B จนถึง 88H กำหนดหน่วยความจำสำหรับเก็บผลลัพธ์ด้วยตัวเอง

   ***เช่น*** Register B = 87H ➔ 87 + 88H = 010FH เป็นต้น

## 3.7 อ้างอิง

1. Rotate Instruction

| | |
|---|---|
| **RR A ;rotate right A** |  |
| **RL A ;rotate left A** |  |
| **RRC A ;rotate right with carry** |  |
| **RLC A ;rotate left with carry** |  |
| **SWAP A** |  |
| **XCH A,B** | **Before A=0AAH, B=0BBH**<br>**After A=0BBH, B=0AAH** |

## 3.8 โปรแกรมทดสอบ

```
      ; ==== Lab31.Asm ====
Result_H    EQU   20H
Result_L    EQU   21H

      ORG   0000H
      MOV   Result_H,#00 ; Clear Result
      MOV   Result_L,#00

      MOV   A,R0
      CLR   C
      ADDC  A,Result_L
      MOV   Result_L,A
      MOV   A,#00
      ADDC  A,Result_H
      MOV   Result_H,A

      MOV   A,R1
      CLR   C
      ADDC  A,Result_L
      MOV   Result_L,A
      MOV   A,#00
      ADDC  A,Result_H
      MOV   Result_H,A

      MOV   A,R2
      CLR   C
      ADDC  A,Result_L
      MOV   Result_L,A
      MOV   A,#00
      ADDC  A,Result_H
      MOV   Result_H,A

      JMP   $
      END
```

```
; ==== Lab32.Asm ====
      ORG   0000H          ; 1
      MOV   A,#0FEH        ; 2
      ANL   A,#0F0H        ; 3
      ORL   A,#005H        ; 4

      RR    A              ; 5
      INC   A              ; 6
      RL    A              ; 7
      SETB  Acc.4          ; 8
      DEC   A              ; 9
      SWAP  A              ; 10
      CRL   C              ; 11
      ADD   A,#0FFH        ; 12
      RLC   A              ; 13

      SETB  21H            ; 14
      ANL   A,24H          ; 15
      MOV   DPTR,#8888H    ; 16
      MOVX  @DPTR,A        ; 17

      SJMP  $              ; 18
      END                  ; 19
```

### 3.9 ตารางบันทึกผล

ตอนที่ 1

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

ตอนที่ 2

ตารางที่ 3.1 ข้อมูลในหน่วยความจำก่อนและหลังการรันโปรแกรม

| Assembly instruction | | Data | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Item | Initial | PC | DPTR | Acc | PSW | | | | | |
| 1 | ORG 0000H | | | | | | | | | |
| 2 | MOV A,#0FEH | | | | | | | | | |
| 3 | ANL A,#0F0H | | | | | | | | | |
| 4 | ORL A,#005H | | | | | | | | | |
| 5 | RR A | | | | | | | | | |
| 6 | INC A | | | | | | | | | |
| 7 | RL A | | | | | | | | | |
| 8 | SETB Acc.4 | | | | | | | | | |
| 9 | DEC A | | | | | | | | | |
| 10 | SWAP A | | | | | | | | | |
| 11 | CRL C | | | | | | | | | |
| 12 | ADD A,#0FFH | | | | | | | | | |
| 13 | RLC A | | | | | | | | | |
| 14 | SETB 21H | | | | | | | | | |
| 15 | ANL A,24H | | | | | | | | | |
| 16 | MOV DPTR,#8888H | | | | | | | | | |
| 17 | MOVX @DPTR,A | | | | | | | | | |
| 18 | SJMP $ | | | | | | | | | |
| 19 | END | | | | | | | | | |

ตอนที่ 3

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

# การทดลองที่ 4
# การใช้คำสั่งกระโดด การเขียนโปรแกรมย่อย และการหน่วงเวลา

## 4.1 วัตถุประสงค์

1. เพื่อศึกษาคำสั่งการกระโดดแบบไม่มีเงื่อนไขและแบบมีเงื่อนไข
2. เพื่อศึกษาการเขียนโปรแกรมย่อย การเปิดตาราง และการสร้างโปรแกรมเพื่อหน่วงเวลา

## 4.2 อุปกรณ์ทดลอง

1. ET-8032 V2.0 MCS-51 single board microcontroller
2. เครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับเชื่อมต่อกับ ET-8032

## 4.3 เนื้อหา ทฤษฎีที่เกี่ยวข้อง

1. การกระโดด การกระโดดแบ่งออกเป็น 2 กรณี คือ

1.1 การกระโดดแบบไม่มีเงื่อนไข

เช่น **JMP Label**

คำสั่ง JMP ตัวคอมไพล์เลอร์ SXA51 จะเลือกให้เลยว่าเป็น SJMP หรือ LJMP

1.2 การกระโดดแบบมีเงื่อนไข

| | | |
|---|---|---|
| เช่น | **JZ = Jump if Acc is Zero** | **JZ Label** |
| | **JNZ = Jump if Acc is not Zero** | **JNZ Label** |
| | **DJNZ = Decrement and Jump if Not Zero** | **DJNZ R7, Loop_Label** |
| | **CJNE = Compare and Jump if Not Equal** | **CJNE A, #data, Label** |
| | **JC = Jump if Carry Set** | **JC Label** |
| | **JNC = Jump if not Carry Set** | **JNC Label** |
| | **JB = Jump if Bit Set** | **JB Px.x, Label** |
| | **JNB = Jump if not Bit Set** | **JNB 20H.x, Label** |

2. การเปิดตาราง

การเปิดตาราง คือ การนำส่วนหนึ่งในโค้ดโปรแกรมมาใช้เป็นข้อมูลในการทำงานของโปรแกรม เช่น ต้องการแสดงค่าต่างๆ ที่ 7_Segment กำหนดให้ 7_Segment มีการต่อเซ็กเมนต์ dot, g, f, e, d, c, b และ a เข้ากับพอร์ต 0E060H บิต 7, 6, 5, 4, 3, 2, 1 และ 0 ตามลำดับ

ต้องการแสดงเลข  ต้องให้เซ็กเมนต์ a, b, c, d, e, f = On และเซ็กเมนต์ g และ dot = Off เมื่อมีการถอดรหัสแล้วค่าที่ต้องแสดงบนพอร์ต 0E060H คือ 00111111B หรือ 3FH โปรแกรมการทำงานสามารถเขียนได้ดังนี้

```
        ANL   A,#0FH        ; Table Only 0 to F
        MOV   DPTR,# DATA   ; Point DPTR to DATA Table
        MOVC  A,@A+DPTR     ; Open Table
        ;
DATA:   DB    3FH, 06H, 5BH, 4FH
        DB    66H, 6DH, 7DH, 07H
        DB    7FH, 6FH, 77H, 7CH
        DB    39H, 5EH, 79H, 71H
```

3. การสร้างโปรแกรมย่อยหน่วงเวลา

โปรแกรมย่อยหน่วงเวลาเป็นโปรแกรมย่อยที่มีการใช้งานมาก การคำนวณต้องตอบคำถามให้ได้ 3 นี้ก่อนจึงจะคำนวณได้ คือ

- ระบบ MCS-51 ทำงานกี่ Machine Cycle ➔ ET-8032 V2 = ??? ขึ้นกับโค้ดโปรแกรม
- ระบบ MCS-51 ทำงานกี่ Clock ใน 1 Machine Cycle ➔ ET-8032 V2 = 12 Clock/1MC
- ระบบ MCS-51 ทำงานที่ความถี่กี่ Hz ➔ ET-8032 V2 = 11.0592MHz

เช่น โค้ดโปรแกรมต่อไปนี้ถ้าต้องการหน่วงเวลา 500 ไมโครวินาที ค่า A และ B ควรเป็นเท่าไหร่

| | | | | | |
|---|---|---|---|---|---|
| 1 | DELAY500U: | PUSH B | ; 2 MC | | |
| 2 | | PUSH ACC | ; 2 MC | | |
| 3 | | MOV A,#aa | ; 1 MC | | |
| 4 | DLY01: | MOV B,#bb | ; 1 MC | | Repeat aa time |
| 5 | DLY00: | DJNZ B,DLY00 | ; 2 MC | Repeat bb time | |
| 6 | | DJNZ A,DLY01 | ; 2 MC | | |
| 7 | | POP Acc | ; 2 MC | | |
| 8 | | POP B | ; 2 MC | | |
| 9 | | RET | ; 2 MC | | |

- ทำงานบรรทัดที่ 5 จำนวน bb ครั้ง = 2*bb MC
- ทำงานบรรทัดที่ 4,5,6 จำนวน aa ครั้ง = (1+2*bb+2)*aa MC
- ส่วนอื่นๆ ของโปรแกรม = 2+2+1+(1+2*bb+2)*aa+2+2+2 MC

  = 11+2*aa*bb + 3*aa MC
- ระบบ MCS-51 ทำงานที่ความถี่ 11.0592MHz 1 Clock ใช้เวลา = $\frac{1}{11.0592}$ µSec
- ระบบ MCS-51 ทำงาน 1 MC = 12 Clock ใช้เวลา = $\frac{12}{11.0592}$ µSec

- ถ้าต้องการหน่วงเวลา 500 µSec ใช้ $\frac{500 \times 11.0592}{12}$ MC
- หน่วงเวลา 500 µSec ใช้ 460.8 MC = 461 MC
- หน่วงเวลา 461MC = 11+2*aa*bb + 3*aa

  สมมุติ aa = 10 จะได้ bb= 21
- นั่นคือ โปรแกรมบรรทัดที่ 3, 4 กำหนด A และ B เป็น 10D และ 21D ตามลำดับเมื่อเรียกโปรแกรมย่อยนี้จะหน่วงเวลา 500 ไมโครวินาที
- หากต้องการหน่วงเวลา 0.1 วินาทีก็เขียนโปรแกรมย่อยใหม่ให้เรียก Delay500U จำนวน 200 ครั้ง

```
1  DELAY100M:PUSH 00
3            MOV  R0,#200
5  DLY03:    CALL DELAY500U
6            DJNZ R0,DLY03
8            POP  00
9            RET
```

## 4.4 การทดลอง

<u>ตอนที่ 1</u> คำสั่งการกระโดด

1. ให้นักศึกษาระบุค่า oscillator frequency ของบอร์ด ET-8032 V2 = ..... MHz
2. ให้นักศึกษาคำนวณค่า oscillator period ของบอร์ด ET-8032 V2 = ..... µSec
3. ให้นักศึกษาคำนวณค่าเวลารอบการทำงานของบอร์ด ET-8032 V2

   1 Machine Cycle (MC) = ..... µSec
4. ศึกษาการทำงานของโปรแกรม “Lab41.Asm” และ “Lab42.Asm” เป็นโปรแกรมที่ทำการย้ายค่าจากหน่วยความจำภายในตำแหน่ง 70H ถึงตำแหน่ง 7FH ไปเก็บไว้ที่หน่วยความจำภายนอกเริ่มจากตำแหน่ง 1000H เป็นต้นไป

5. ให้นักศึกษาเขียนโปรแกรมเติมค่าหน่วยความจำจากตำแหน่ง 9000H ถึงตำแหน่ง 90FFH โดยมีค่าเริ่มจาก 00H และเพิ่มขึ้นตำแหน่งละ 1 จนครบ

6. ให้นักศึกษาเขียนโปรแกรมเพื่อตรวจสอบว่าค่าข้อมูลที่อยู่ในหน่วยความจำภายในตำแหน่ง 40H-7FH ว่ามีค่าเท่ากับค่าที่เก็บในรีจีสเตอร์ B กี่ค่าโดยเก็บค่าที่นับได้ในหน่วยความจำภายนอกตำแหน่ง 9000H

ตอนที่ 2 การเขียนโปรแกรมย่อย การเปิดตาราง และการหน่วงเวลา

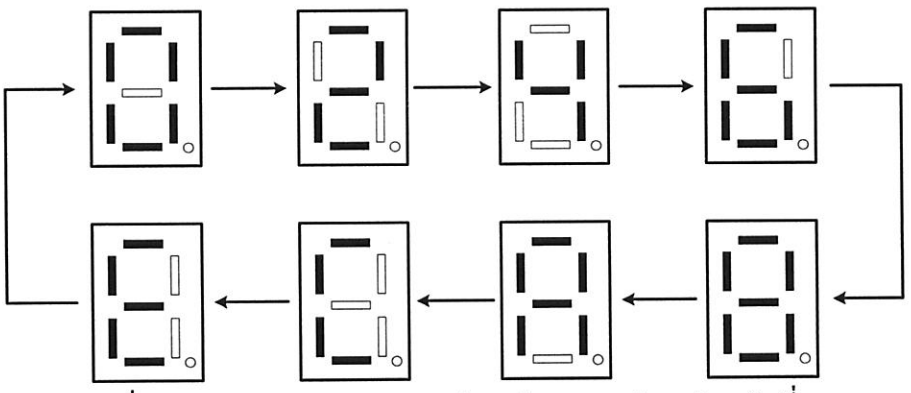
1. โปรแกรม "Lab43.Asm" เป็นโปรแกรมสำหรับแสดงค่า 0 ถึง F ที่พอร์ต 7_Segment ตำแหน่ง 0E060H ทดสอบการทำงานด้วยการใช้ HyperTerminal ควบคุมให้บอร์ด ET-8032 ทำงาน
2. จากชุดคำสั่งในข้อที่ 1 ให้นักศึกษาคำนวณจำนวน machine cycle ที่ใช้ในการเรียกโปรแกรมย่อย DELAY หนึ่งครั้ง จำนวน machine cycle = .....
3. ให้คำนวณระยะเวลาในการหน่วงเวลาดังกล่าว delay time = ..... วินาที
4. ให้นักศึกษาดำเนินการปรับแก้โปรแกรมในข้อที่ 1 เพื่อให้การแสดงผลตัวอักษรที่ตัวแสดงผล 7 ส่วน (7-segment display) เกิดการหน่วงเวลาประมาณ 3 วินาที

## 4.5 วิเคราะห์และอภิปรายผลการทดลอง

1. จงแสดงวิธีการคำนวณ oscillator period เวลาการทำงาน 1 Machine cycle ของ ET-8032
2. จงอธิบายการทำงาน ข้อจำกัดและความแตกต่างระหว่างการใช้คำสั่ง SJMP AJMP และ LJMP
3. จงอธิบายการทำงาน ข้อจำกัดและความแตกต่างระหว่างการใช้คำสั่ง ACALL และ LCALL
4. จงอธิบายการทำงานของชุดคำสั่งแอสเซมบลีที่พัฒนาขึ้นในตอนที่ 1 ข้อ 4 ทีละบรรทัด
5. จงอธิบายการทำงานของชุดคำสั่งแอสเซมบลีที่พัฒนาขึ้นในตอนที่ 1 ข้อ 6 ทีละบรรทัด
6. อธิบายการคำนวณค่าในรีจีสเตอร์ที่ใช้กรณีคำนวณการหน่วงเวลาประมาณ 3 วินาทีในตอนที่ 2 ข้อ 4
7. จงอธิบายการสร้างรหัสคำสั่งเพื่อควบคุมตัวแสดงผล 7 ส่วน เพื่อแสดงค่าตั้งแต่ค่า 0-F

## 4.6 คำถามท้ายการทดลอง

1. ให้นักศึกษาเขียนโปรแกรมเพื่อแสดงผลตัวเลข 7 ส่วน (สามารถเข้าถึงได้ที่หน่วยความจำภายนอกตำแหน่ง E060H) โดยกำหนดให้แสดงผลเฉพาะเลขคู่ 0 → 2 → 4 → 6 → 8 → A → C → E → 0 ดังรูปที่ 4.1 การแสดงผลตัวเลขให้เกิดการหน่วงเวลา 5 วินาที

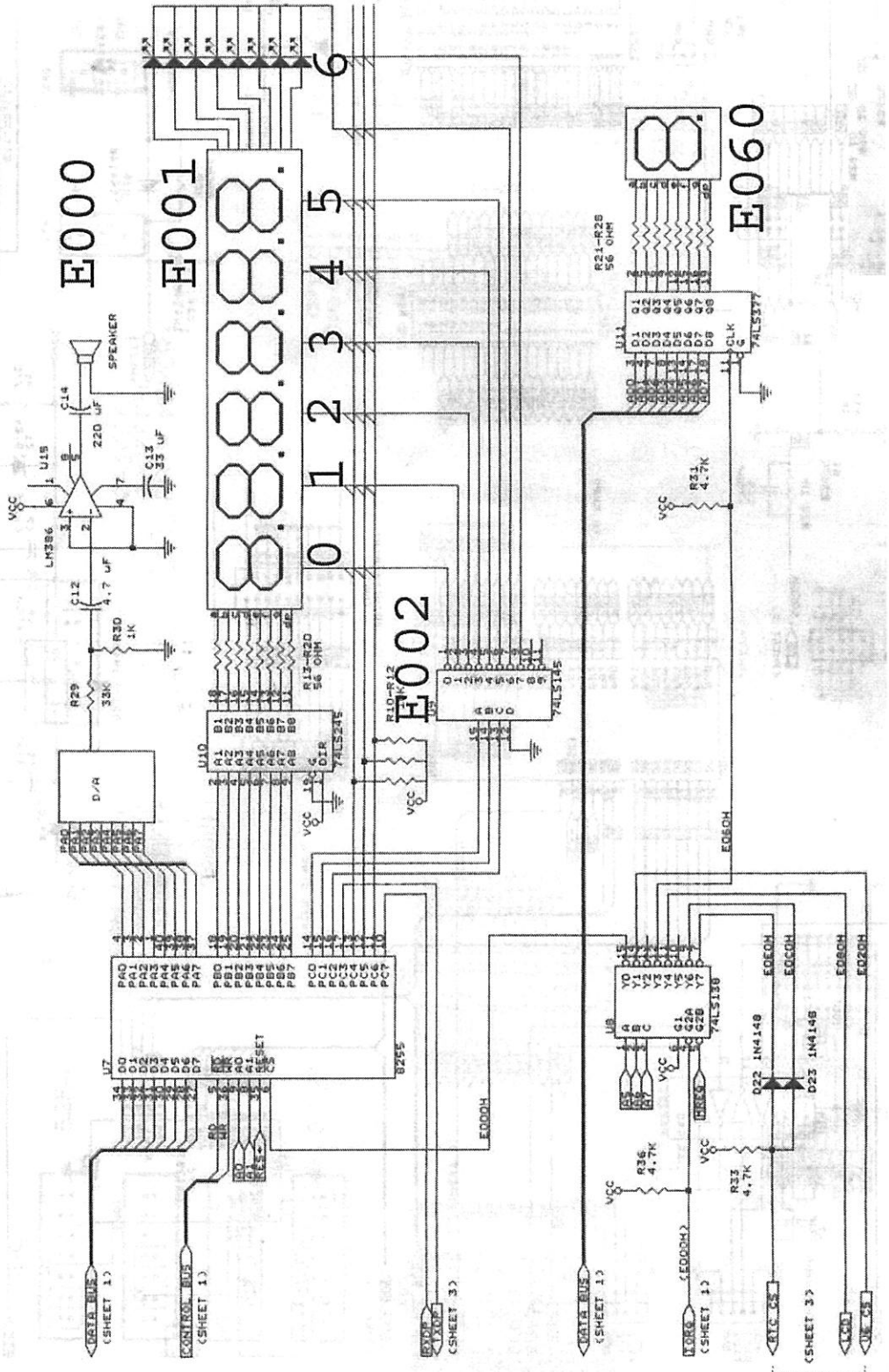


รูปที่ 4.1 รูปแบบการแสดงผลเลขคู่ของตัวเลขฐานสิบหลักหลักเดี่ยว

2. จงเขียนโปรแกรมเพื่อยกกำลังสองตัวเลขที่เก็บอยู่ในหน่วยความจำตำแหน่ง 8500H ซึ่งกำหนดให้มีค่าระหว่าง 00H – 0FH เท่านั้น โดยกำหนดให้เขียนผลลัพธ์ที่เป็นเลขฐานสิบหกไว้ที่ตำแหน่ง 8501H นอกจากนี้ ให้แปลงตัวเลขผลลัพธ์ฐานสิบหกให้เป็นรหัส BCD โดยเก็บหลักร้อยไว้ที่ตำแหน่ง 8502H หลักสิบและหลักหน่วยที่ตำแหน่ง 8503H

3. ให้นักศึกษาเขียนโปรแกรมเพื่อหาค่าต่ำสุดและค่าสูงสุดของข้อมูลดังต่อไปนี้
   ก. ตำแหน่ง 8600H – 8601H เก็บค่าตำแหน่งเริ่มต้นของบล็อกข้อมูลในการค้นหา
   ข. ตำแหน่ง 8602H – 8603H เก็บค่าตำแหน่งสุดท้ายของบล็อกข้อมูลในการค้นหา
   ค. ตำแหน่ง 8700H – 8701H เก็บค่าตำแหน่งที่พบค่าสูงสุด
   ง. ตำแหน่ง 8702H เก็บค่าสูงสุดที่พบ
   จ. ตำแหน่ง 8800H – 8801H เก็บค่าตำแหน่งที่พบค่าต่ำสุด
   ฉ. ตำแหน่ง 8803H เก็บค่าต่ำสุดที่พบ

## 4.7 อ้างอิง


E000
E001
E002
E060
0 1 2 3 4 5 6
SPEAKER
LM386
D/A
8255
74LS245
74LS145
74LS377
74LS138
R21-R28
56 OHM
R13-R20
56 OHM
R10-R12
1K
R29
33K
R30
1K
R31
4.7K
R33
4.7K
R36
4.7K
C12
4.7 uF
C13
33 uF
C14
220 uF
D22 1N4148
D23 1N4148
VCC
E060H
E000H
E020H
DATA BUS
CONTROL BUS
(SHEET 1)
(SHEET 3)
RTC_CS
LCD
LC_CS

## 4.8 โปรแกรมทดสอบ

```
;==== LAB41.ASM =====
        ORG   8000H
        MOV   R0,#70H       ;Start Read RAM
        MOV   R1,#10H       ;Stop-Start+1
        MOV   DPTR,#9000H;Save RAM
LOOP:   MOV   A,@R0
        MOVX  @DPTR,A
        INC   R0
        INC   DPTR
        DJNZ  R1,Loop

        JMP   $
        END
```

For

```
;==== LAB42.ASM =====
        ORG   8000H
        MOV   R0,#70H       ;Start Read RAM
        MOV   DPTR,#9000H;Save RAM
LOOP:   MOV   A,@R0
        MOVX  @DPTR,A
        INC   R0
        INC   DPTR
        MOV   A,R0
        CJNE  A,#80H,Loop  ;Stop 7FH+1

        JMP   $
        END
```

Repeat Until

```
;==== LAB43.ASM =====
                ORG   8000H                ; 1
C7SEG           EQU   0E060H               ; 2

START:          MOV   R0,#0                ; 3
MAIN:           MOV   A,R0                 ; 4
                INC   R0                   ; 5
                ACALL TAB01                ; 6
                MOV   DPTR,#C7SEG          ; 7
                MOVX  @DPTR,A              ; 8
                ACALL DELAY                ; 9
                CJNE  R0,#10H,MAIN         ; 10
                SJMP  START                ; 11

TAB01:          MOV   DPTR,#DATA           ; 12
                MOVC  A,@A+DPTR            ; 13
                RET                        ; 14

DELAY:          MOV   R5,#04               ; 15
DELAY1:         MOV   R6,#200              ; 16
DELAY2:         MOV   R7,#200              ; 17
DELAY3:         DJNZ  R7,DELAY3            ; 18
                DJNZ  R6,DELAY2            ; 19
                DJNZ  R5,DELAY1            ; 20
                RET                        ; 21

DATA:           DB    3FH,06H,5BH,4FH      ; 22
                DB    66H,6DH,7DH,07H      ; 23
                DB    7FH,6FH,77H,7CH      ; 24
                DB    39H,5EH,79H,71H      ; 25

                END                        ; 26
```

### 4.9 ตารางบันทึกผล

ตอนที่ 1

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

ตอนที่ 2

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

# การทดลองที่ 5

# การโปรแกรมพอร์ตอินพุตและเอาต์พุตสำหรับ ET-8032

## 5.1 วัตถุประสงค์

1. เพื่อศึกษาการเชื่อมต่อ ET-8032 กับอุปกรณ์ภายนอก
2. เพื่อศึกษาการเขียนโปรแกรมควบคุมพอร์ตของ ET-8032
3. เพื่อศึกษาการสร้างสัญญาณเสียงโดยใช้ ET-8032

## 5.2 อุปกรณ์ทดลอง

1. ET-8032 V2.0 MCS-51 single board microcontroller
2. เครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับเชื่อมต่อกับ ET-8032

## 5.3 เนื้อหา ทฤษฎีที่เกี่ยวข้อง

- การใช้งานพอร์ตกรณีเป็นเอาต์พุต จะทำเพื่อต้องการแสดงผลลัพท์เมื่อไมโครคอนโทรลเลอร์ทำงานเสร็จ การควบคุมพอร์ตเป็นเอาต์พุตทำได้ 2 กรณี คือ

การเข้าถึงแบบบิต เช่น

```
SETB    P3.2            ; Define P3.2 is One
CPL     P1.0            ; Complement P1.0
MOV     P3.2,C          ; Send Bit status in carry to P3.2
```

การเข้าถึงแบบไบต์ เช่น

```
MOV     P1,#055H        ; Define P1 = 01010101B
CPL     P1              ; Complement P1
ORL     P1,#00FH        ; Define Only P1 Lower Bit = 1111B
```

- การใช้งานพอร์ตกรณีเป็นอินพุต จะทำเมื่อต้องการรับค่าให้ไมโครคอนโทรลเลอร์ตัดสินใจทำงานต่างๆตามที่ได้โปรแกรมไว้ การควบคุมพอร์ตเป็นอินพุต***ก่อนอ่านค่าจำเป็นต้องทำให้พอร์ตนั้นเป็น 1 ก่อน*** การอ่านค่าพอร์ตอินพุตทำได้ 2 กรณี คือ

การเข้าถึงแบบบิต เช่น

```
SETB    P3.2            ; Initial Input Port
MOV     C,P3.2          ; Send Bit status to carry flag
JB      P3.2,Label      ; Check P3.2 is One Jump to Label
JNB     P3.2,Label      ; Check P3.2 is not One Jump to Label
```

การเข้าถึงแบบไบต์ เช่น

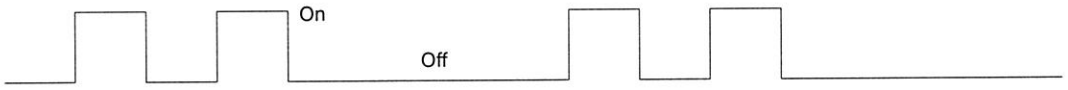
```
MOV     P1,#0FFH        ; Define P1 = 01010101B
MOV     A,P1            ; Read P1 to Acc
```

- การสร้างสัญญาณเสียงออกที่ลำโพง 0E000H สามารถทำได้โดยการส่งสัญญาณสี่เหลี่ยม (square wave) ออกที่พอร์ตลำโพงในกรณีบอร์ด ET-8032 V2 มีแอดเดรสเป็น 0E000H ขั้นตอนการคำนวณทำได้ดังนี้

  - ต้องการสร้างความถี่ = 1,000 Hz
  - คาบของสัญญาณ $= \frac{1}{1,000} x1,000,000 = 1,000\ \mu Sec$
  - การสร้างสัญญาณต้องมีการส่งสัญญาณ On + Delay On และ Off + Delay Off ทำซ้ำไปเรื่อยๆ กรณีที่ดิวตี้ไซเคิล (duty cycle) 50% หรือ T_On = 50% จะได้เวลาสำหรับ Delay On = Delay Off $= 1,000x\frac{50}{100} = 500\ \mu Sec$
  - กรณีใช้บอร์ดที่ XTAL=11.0592MHz และไมโครคอนโทรลเลอร์ 12 Clock ต่อ 1 MC จะได้ว่า 1 MC ใช้เวลา $= \frac{1}{11.0592} x1,000,000x12 = 1.085\ \mu Sec$
  - หน่วงเวลา 500 µSec $= \frac{500}{1.085} = 460.8$ MC
  - หากใช้รูปแบบโปรแกรมตามการทดลองที่ 4 หน้า 39 ก็จะสามารถหาค่าของรีจีสเตอร์ต่างๆ ได้

## 5.4 การทดลอง

1. ทดสอบโปรแกรม "Lab51.ASM" อธิบายผลที่ได้อ้างอิงกับคู่มือบอร์ดหน้า 96
2. จากโปรแกรม "Lab51.ASM" ให้ปรับปรุงโปรแกรมเพื่อกำหนด LED P3.2 ให้กระพริบดังนี้




3. ทดสอบโปรแกรม "Lab52.ASM" เพื่อสร้างสัญญาณเสียง ออกที่พอร์ต E000H และพอร์ต 1
4. จากโปรแกรม "Lab52.ASM" ให้ปรับปรุงเพื่อสร้างสัญญาณเสียง 1,000 Hz ออกที่พอร์ต E000H และพอร์ต 1
5. ทดสอบโปรแกรม "Lab53.ASM" โดยโปรแกรมรับค่าจาก P1 แสดงผลที่เป็น Logic 0 ที่ 7_Segment Display Address E060H เฉพาะ P1.0 และ P1.1

6. จากโปรแกรม "Lab53.ASM" โดยโปรแกรมรับค่าจาก P1 แสดงผลที่เป็น Logic 0 ที่ 7_Segment Display Address E060H ในจำนวนทั้ง 8 พอร์ต และกรณีไม่มีพอร์ตใดเป็น 0 ให้แสดงเครื่องหมาย –(ลบ)

7. ทดสอบโปรแกรม "Lab54.ASM" เป็นโปรแกรมไฟวิ่งที่ LED Display ขนาดเล็ก Value Address E001H ,Position =06H Address E002H
8. จากโปรแกรม "Lab54.ASM" โดยโปรแกรมรับค่าจาก P3 แสดงผลเป็นไฟวิ่งรูปแบบต่างๆ ที่ LED Display ขนาดเล็ก Value Address E001H ,Position =06H Address E002H ดังนี้
   - ถ้า P3.3 P3.4 = 00 ให้ LED ติดค้างทุกดวง
   - ถ้า P3.3 P3.4 = 10 ให้ไฟวิ่งจากขวาไปซ้าย
   - ถ้า P3.3 P3.4 = 01 ให้ไฟวิ่งจากซ้ายไปขวา
   - ถ้า P3.3 P3.4 = 11 ให้ LED ดับทุกดวง

## 5.5 วิเคราะห์และอภิปรายผลการทดลอง

1. จงอธิบายหลักการใช้งานของพอร์ตทั้ง 4 ของ ET-8032
2. ผู้ใช้สามารถเข้าถึงพอร์ต P0 P2 และ P3 ของ ET-8032 ได้หรือไม่ จงอธิบาย
3. จงพัฒนาอัลกอริทึมการสร้างสัญญาณเสียงที่ความถี่ 1,000 Hz และมีค่า duty cycle = 80%
4. จงอธิบายการทำงานของชุดคำสั่งแอสเซมบลีที่ดำเนินการทดลองและได้ออกแบบไว้ในการทดลองข้อ 8 ทีละบรรทัด

## 5.6 คำถามท้ายการทดลอง

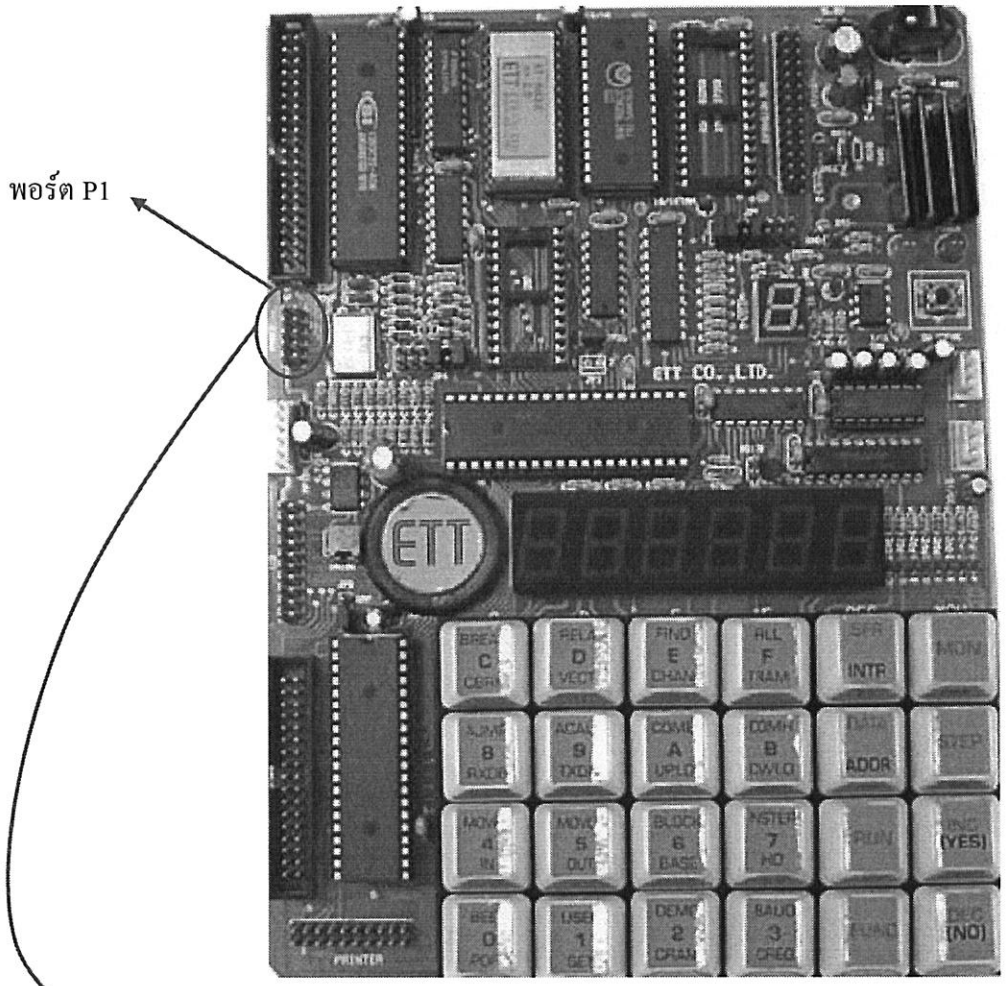
1. ให้นักศึกษาออกแบบโปรแกรมเพื่อให้รับอินพุตจากสวิตช์ที่ต่อเชื่อมกับพอร์ต P1.0 – P1.3 รวมทั้งสิ้น 4 ตัว และให้เชื่อมต่อ P1.4 – P1.7 กับไดโอเปล่งแสง จำนวนทั้งสิ้น 4 ตัวเช่นกัน โดยโปรแกรมจะดำเนินงานดังนี้
   ก. ถ้าสวิตช์ P1.0=0 ให้ LED P1.4 สว่าง และ 7-segment 0E060H แสดงตัวเลข 0
   ข. ถ้าสวิตช์ P1.1=0 ให้ LED P1.5 สว่าง และ 7-segment 0E060H แสดงตัวเลข 1
   ค. ถ้าสวิตช์ P1.2=0 ให้ LED P1.6 สว่าง และ 7-segment 0E060H แสดงตัวเลข 2
   ง. ถ้าสวิตช์ P1.3=0 ให้ LED P1.7 สว่าง และ 7-segment 0E060H แสดงตัวเลข 3
   จ. ถ้าเปิดสวิตช์มากกว่า 1 ตัว ให้ LED สว่างทุกตัว และให้ 7-segment แสดงตัวเลข E

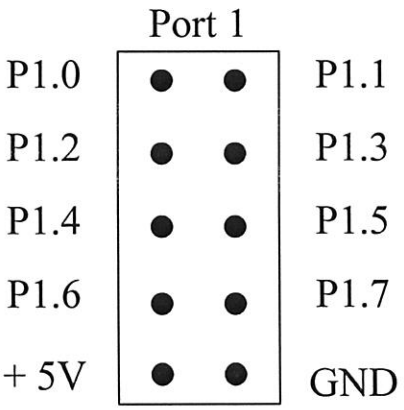
2. ให้นักศึกษาออกแบบโปรแกรมไฟกระพริบ กำหนดให้มี LED ต่อกับ P3.2 และสวิตซ์ต่อกับ P1.0 และให้กระพริบดังนี้
   - ถ้าสวิตช์ P1.0=0 ให้ LED P3.2 ดับ
   - ถ้าสวิตช์ P1.0=1 ให้ LED P3.2 สว่าง

3. จากข้อ 2 ให้นักศึกษาออกแบบโปรแกรมใหม่มีข้อกำหนดการกระพริบดังนี้
   - ถ้าสวิตช์ P1.0=0 ให้ LED P3.2 ติดสว่างค้าง
   - ถ้าสวิตช์ P1.0=1 ให้ LED P3.2 ติดกระพริบ(On/Off สลับกัน)

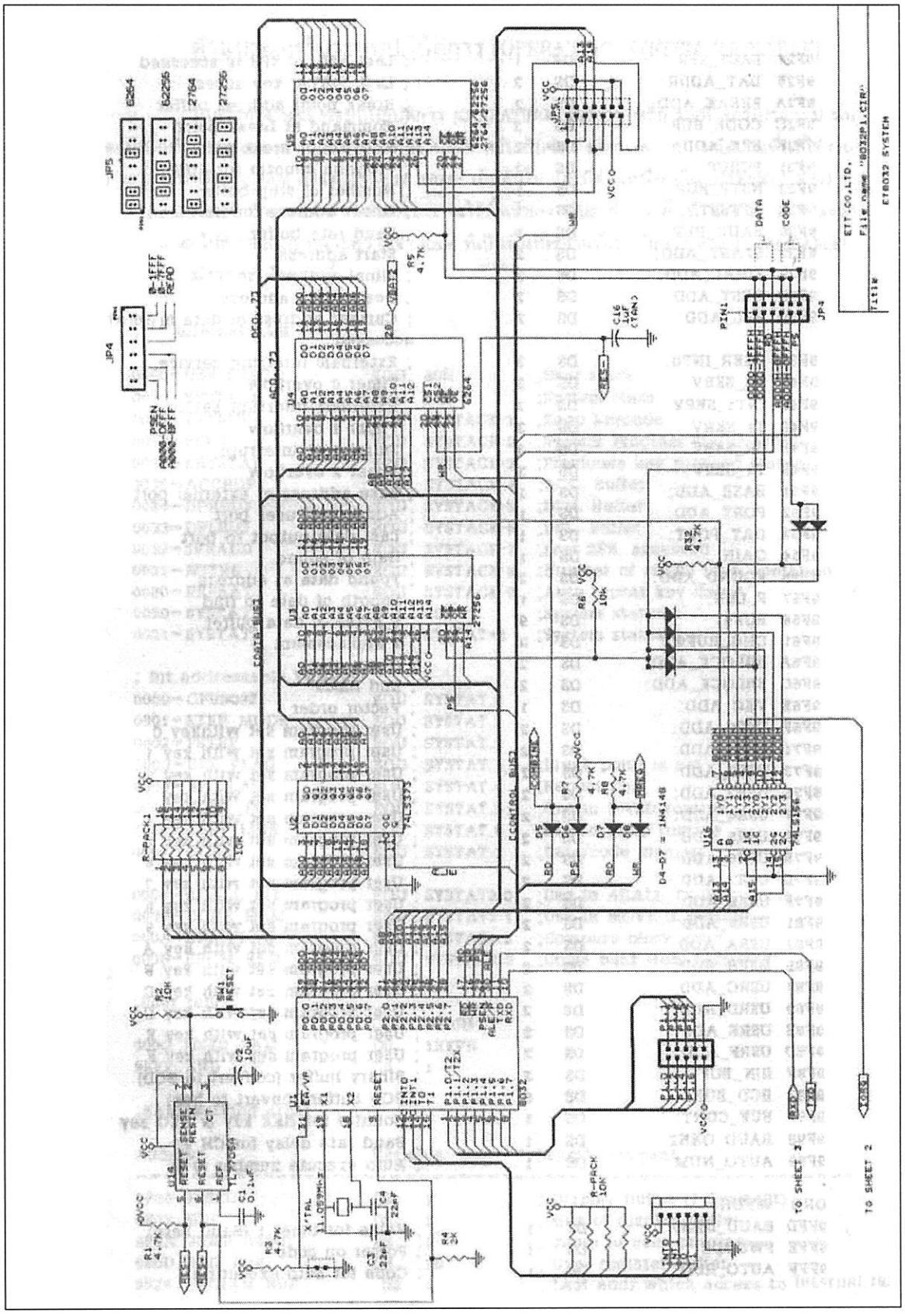
4. จากข้อ 2 ให้นักศึกษาออกแบบโปรแกรมใหม่มีข้อกำหนดการกระพริบ ดังนี้
   - สถาณะปกติสวิตช์ P1.0 จะเป็น 1 c
   - สภาวะเริ่มต้นให้ LED P3.2 ติดสว่างค้าง
   - ถ้าสวิตช์ P1.0=0 ครั้งแรก ให้ LED P3.2 ติดสว่างค้าง
   - ถ้าสวิตช์ P1.0=0 อีกครั้ง ให้ LED P3.2 ติดกระพริบ(On/Off สลับกัน)
   - ถ้าสวิตช์ P1.0=0 อีกครั้ง ให้ LED P3.2 ติดสว่างค้าง
   - ถ้าสวิตช์ P1.0=0 อีกครั้ง ให้ LED P3.2 ติดกระพริบ(On/Off สลับกัน)
   - ถ้าสวิตช์ P1.0=0 อีกครั้ง ให้ LED P3.2 ติดสว่างค้าง
   - {ทำซ้ำระหว่าง LED ค้าง กับ LED กระพริบ ไปเรื่อยๆ }

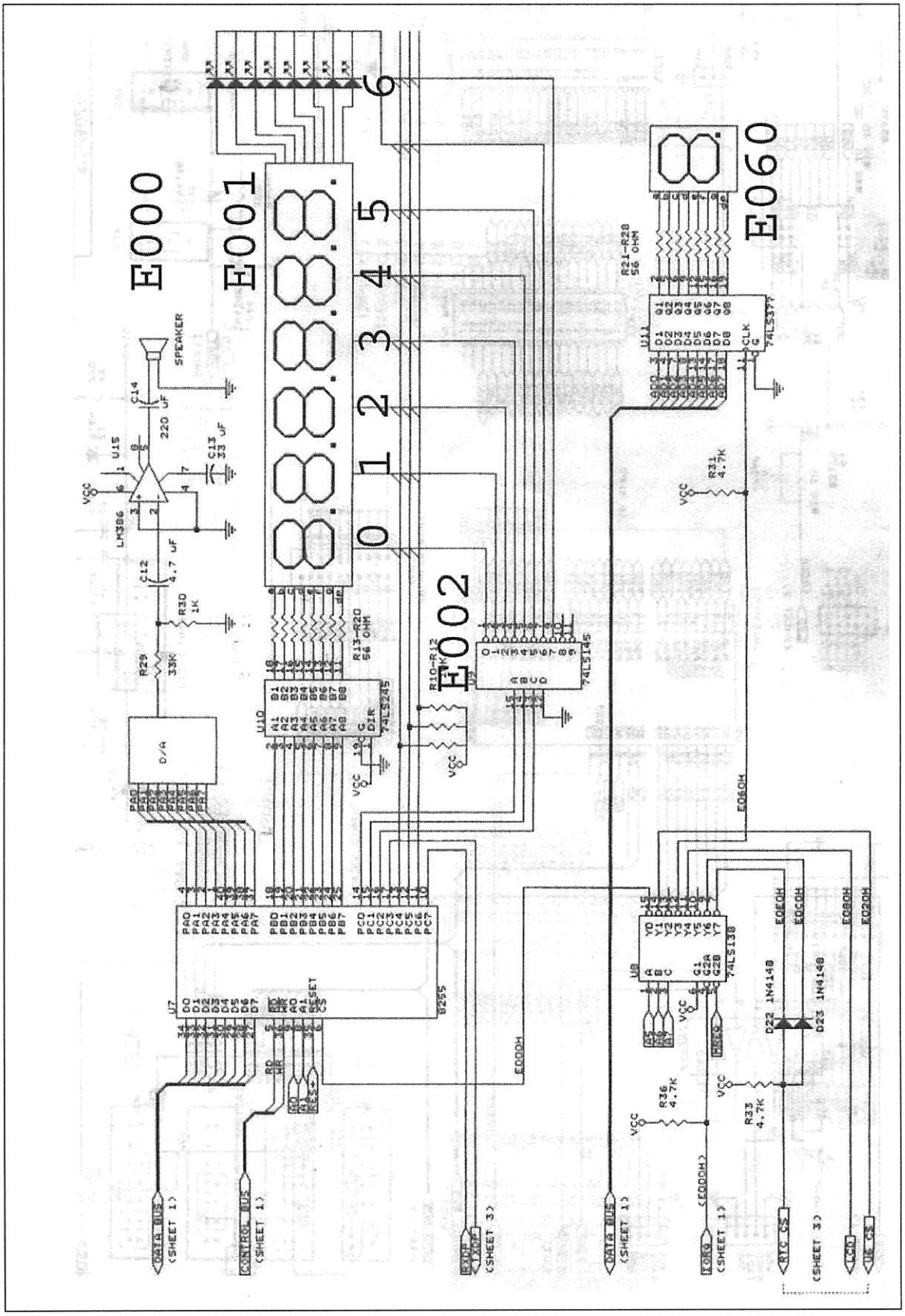
**5.6 อ้างอิง**

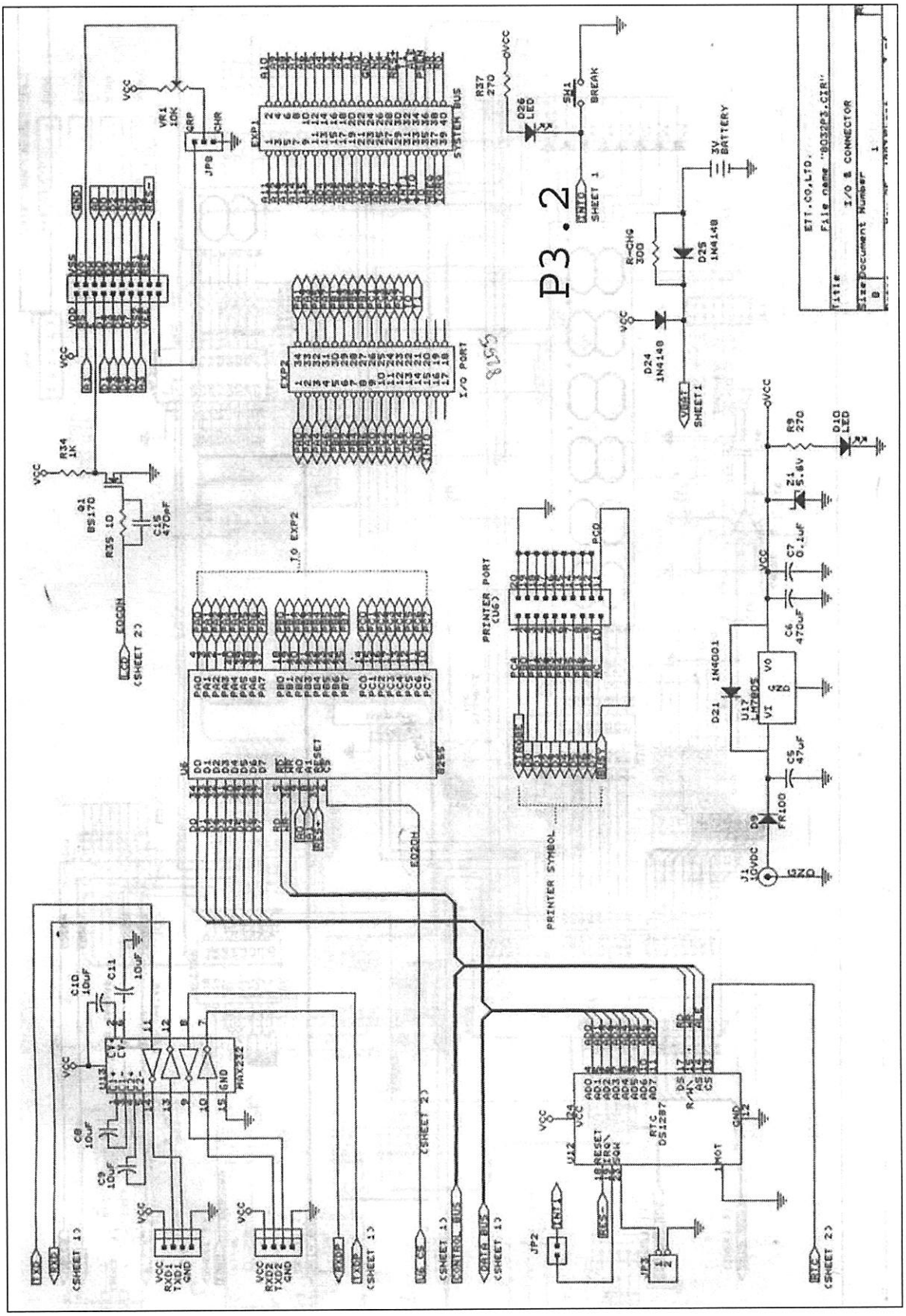


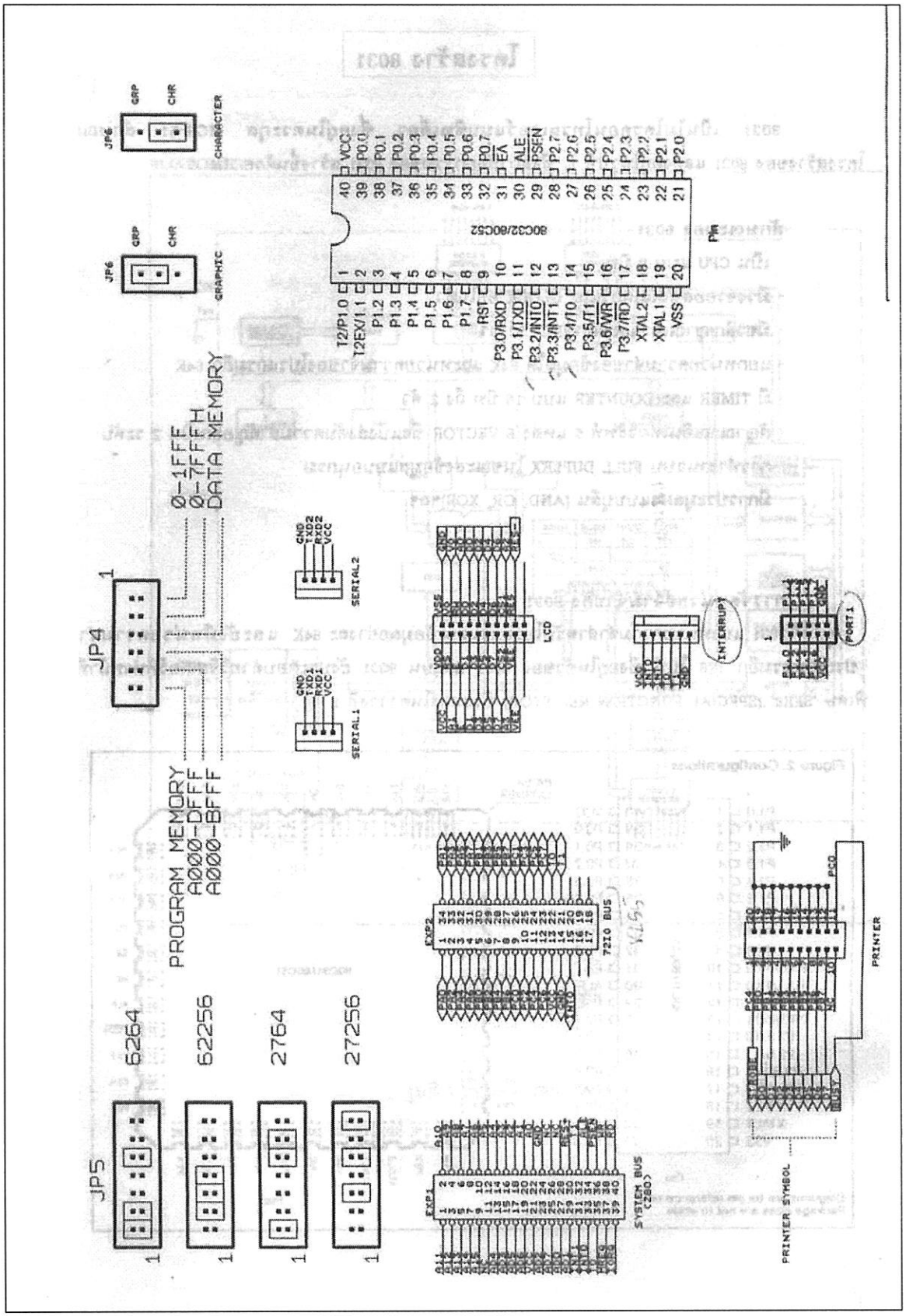




การใช้งาน Port 1 นี้ ต้องดำเนินการต่อแหล่งจ่ายไฟจากภายนอก + 5V และเชื่อมกราวด์เข้ากับตำแหน่งพินดังแสดงในรูป


ET-CO,LTD.
ET8032 SYSTEM
DATA
CODE
TO SHEET 3
TO SHEET 2


E000
E001
E002
E060
0 1 2 3 4 5 6
SPEAKER
LM386
U15
C14
220 uF
C13
33 uF
C12
4.7 uF
R30
1K
R29
33K
D/A
R13-R20
56 OHM
R10-R12
1K
74LS245
74LS145
R21-R28
56 OHM
74LS377
R31
4.7K
E060H
E000H
E020H
E040H
E050H
E080H
74LS138
1N4148
D22
D23
R36
4.7K
R33
4.7K
8255
DATA BUS
(SHEET 1)
CONTROL BUS
(SHEET 1)
(SHEET 3)
(E000H)
IOR#
RTC CS
LCD
UE CS


P3.2
I/O PORT
SYSTEM BUS
PRINTER PORT
(U6)
PRINTER SYMBOL
I/O & CONNECTOR
ET.CO,LTD.
BATTERY
BREAK
RTC
DS1287
MAX232
IN4001
IN4148
LM7805
FR100
BS170


JP6
GRP
CHR
CHARACTER
GRAPHIC
80C32/80C52
VCC
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
EA
ALE
PSEN
P2.7
P2.6
P2.5
P2.4
P2.3
P2.2
P2.1
P2.0
T2/P1.0
T2EX/1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
RST
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
XTAL2
XTAL1
VSS
0-1FFF
0-7FFFH
DATA MEMORY
JP4
SERIAL2
SERIAL1
GND
TXD2
RXD2
VCC
TXD1
RXD1
LCD
INTERRUPT
PORT1
PROGRAM MEMORY
A000-DFFF
A000-BFFF
JP5
6264
62256
2764
27256
EXP2
EXP1
7210 BUS
SYSTEM BUS
(Z80)
PRINTER
PRINTER SYMBOL

## 5.8 โปรแกรมทดสอบ

**Lab51.ASM**

```
        ORG     8000H       ; 1
        CLR     EA          ; 2
LOOP:   SETB    P3.2        ; 3
        CALL    Delay       ; 4
        CLR     P3.2        ; 5
        CALL    Delay       ; 6
        JMP     LOOP        ; 7

Delay:  CLR     A           ; 8
        MOV     B,A         ; 9
_DLY00: DJNZ    Acc,_DLY00  ; 10
        DJNZ    B,_DLY00    ; 11
        RET                 ; 12
```

**Lab52.ASM**

```
        ORG     8000H           ; 1
        MOV     DPTR,#0E000H    ; 2
LOOP:   MOV     A,#000H         ; 3
        CALL    Send_Port       ; 4
        MOV     A,#0FFH         ; 5
        CALL    Send_Port       ; 6
        JMP     LOOP            ; 7

Send_Port: MOVX @DPTR,A         ; 8
        MOV     P1,A            ; 9
Delay:  CLR     A               ; 10
        MOV     B,#5            ; 11
_DLY00: DJNZ    Acc,_DLY00      ; 12
        DJNZ    B,_DLY00        ; 13
        RET                     ; 14
```

**Lab53.ASM**

```
        ORG     8000H       ; 1
        MOV     P1,#0FFH    ; 2

LOOP:   MOV     A,#3FH      ; 3
        JNB     P1.0,ShowData ; 4
        MOV     A,#06H      ; 5
        JNB     P1.1,ShowData ; 6
        JMP     LOOP        ; 7

ShowData: MOV   DPTR,#0E060H ; 8
        MOVX    @DPTR,A     ; 9
        JMP     LOOP        ; 10
```

**Lab54.ASM**

```
        ORG     8000H           ; 1

LOOP:   CALL    LED_R2L         ; 2
        JMP     LOOP            ; 3

LED_R2L: MOV    A,#10000000B    ; 4
        CALL    SendData        ; 5
        MOV     A,#01000000B    ; 6
        CALL    SendData        ; 7
        MOV     A,#00100000B    ; 8
        CALL    SendData        ; 9
        MOV     A,#00010000B    ; 10
        CALL    SendData        ; 11
        MOV     A,#00001000B    ; 12
        CALL    SendData        ; 13
        MOV     A,#00000100B    ; 14
        CALL    SendData        ; 15
        MOV     A,#00000010B    ; 16
        CALL    SendData        ; 17
        MOV     A,#00000001B    ; 18
        CALL    SendData        ; 19
        RET                     ; 20

SendData: MOV   DPTR,#0E001H    ; 21
        MOVX    @DPTR,A         ; 22
        MOV     A,#6            ; 23
        MOV     DPTR,#0E002H    ; 24
        MOVX    @DPTR,A         ; 25
Delay:  CLR     A               ; 26
        MOV     B,A             ; 27
_DLY00: DJNZ    Acc,_DLY00      ; 28
        DJNZ    B,_DLY00        ; 29
        RET                     ; 30
```

## 5.9 ตารางบันทึกผล

ตอนที่ 1

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

ตอนที่ 2

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

# การทดลองที่ 6

# การใช้งานตัวจับเวลาและตัวนับของ ET-8032

## 6.1 วัตถุประสงค์

1. เพื่อศึกษาการใช้งานตัวจับเวลาของ ET-8032
2. เพื่อศึกษาการใช้งานตัวนับของ ET-8032

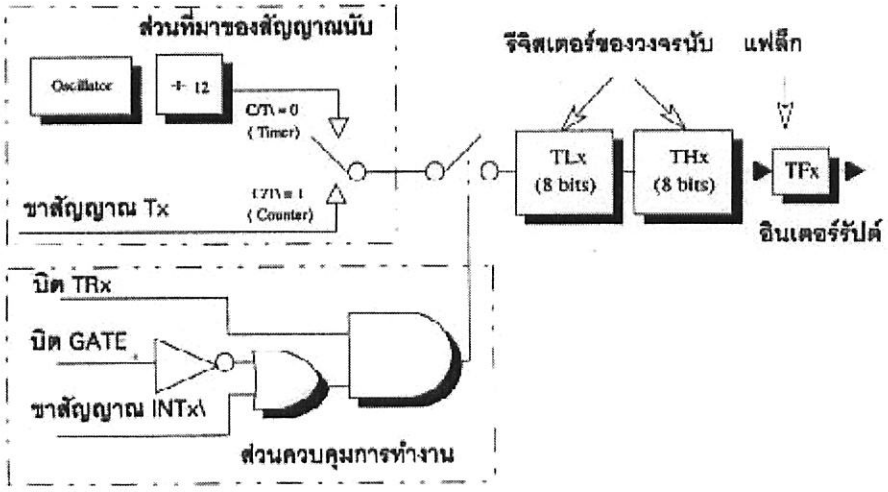
## 6.2 อุปกรณ์ทดลอง

1. ET-8032 V2.0 MCS-51 single board microcontroller
2. CPDL Logic Board
3. เครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับเชื่อมต่อกับ ET-8032

## 6.3 เนื้อหา ทฤษฎีที่เกี่ยวข้อง

1. รีจีสเตอร์ที่สำคัญ

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **TMOD** | **G(T1)** | **C/T** | **M1** | **M0** | **G(T0)** | **C/T** | **M1** | **M0** |
| **TCON** | **TF1** | **TR1** | **TF0** | **TR0** | **IE1** | **IT1** | **IE0** | **IT0** |
| **SCON** | **SM0** | **SM1** | **SM2** | **REN** | **TB8** | **RB8** | **TI** | **RI** |
| **PCON** | **SMOD1** | **SMOD0** | **-** | **POF** | **GF1** | **GF0** | **PD** | **IDL** |
| **T2CON** | **TF2** | **EXF2** | **RCLK** | **TCLK** | **EXEN2** | **TR2** | **C/T2** | **CP/RL2** |




ส่วนประกอบการใช้งานไทม์เมอร์โหมด 1 (Timer 16 Bit)

2. ตัวอย่างการใช้งาน

- **การใช้ Timer ในการหน่วงเวลา**

**เช่น** ต้องการสร้างความถี่ 10 kHz → Period =0.1mSec → Period/2 = 50uSec

Delay 50uSec ที่บอร์ด Crystal 11.0592MHz → 1MC = 12/11.0592 = 1.085uSec

→ 50uSec = 46.08 MC

```
      Load Timer0 = 0-46 = -46 = FFD2H
MOV   TMOD,#01    ; Put Timer 0 in 16-bit mode
MOV   TH0,#0FFH   ; High byte of -46(FFD2H)
MOV   TL0,#0D2H   ; Low byte of -46(FFD2H)
SETB  TR0         ; Make Timer 0 Start Counting
CLR   TF0
JNB   TF0, $      ; If TF0 is not set, jump back to this same
                  ;  instruction
```

- **การใช้ Timer ในการหน่วงเวลาสำหรับ Serial Communication**

Using Timer1, Mode2

http://www.intel.com/design/mcs51/applnots/270490.htm

Serial Mode1,3 และใช้ Timer 1Mode2 → $\text{BaudRate} = \frac{2^{\text{SMOD1}} \times \text{Oscillater Frequency}}{32 \times 12 \times [256 - (\text{th1})]}$

9600 @18.432MHz → SMOD1=0, TH1=251(FBH)

9600 @11.0592MHz → SMOD1=0, TH1=253(FDH)

```
MOV   SCON,#050H  ; Serial Port Mode 3
ANL   PCON,#7FH   ; SMOD 1=2
MOV   TMOD,#20H   ; Timer1 Mode2
MOV   TH1,#0FDH   ; Reload value for desired baud
SETB  TR1         ; Turn on Timer1
```

Using Timer2

$$\text{Buad rate} = \frac{\text{Oscillater Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

9600 @18.432MHz → RCAP2H,RCAP2L = 65476 = FFC4H

9600 @11.0592MHz → RCAP2H,RCAP2L = 65500 = FFDCH

```
MOV   SCON,#01010000B  ; Mode1 Serial Port
MOV   RCAP2H,#0FFH     ; Reload values for desired
MOV   RCAP2L,#0DCH     ; baud rate 9600@11.0592
MOV   T2CON,#00110100B ; Timer 2 as baud rate generator
                       ; and turn on Timer 2
```

- **การใช้ในโหมด Counter**

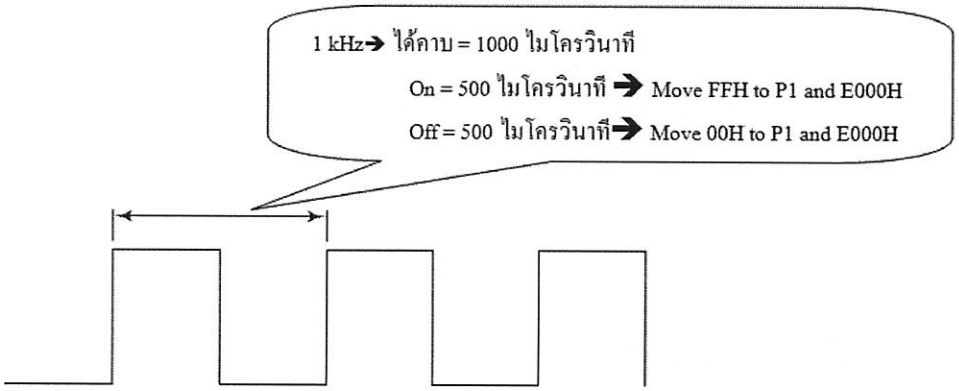
```
        MOV   TMOD,#00000101B  ; Using Counter Mode 1 16 Bit
        MOV   TH0,#0           ; Initial Value = 0000
        MOV   TL0,#0
        SETB  TR0              ; Start Counter
Loop:   {Show TH0TL0}          ; Loop for Show Value of TH0TL0
        JMP   Loop
```

## 6.4 การทดลอง

ตอนที่ 1 การใช้งานตัวจับเวลาของ ET-8032

1. ศึกษาการทำงานของโปรแกรมหน่วงเวลาเพื่อสร้างสัญญาณเสียงความถี่ 1.000 kHz ให้มี Errorได้ ± 0.010kHz นำสัญญาณเสียงส่งออกที่ Port 1 และ E000H ให้ใช้ Timer 0 Mode 1 ในการกำหนดฐานเวลา (***Lab6_Timer.Asm***)

1/2




Q11: จากตัวอย่างเมื่อคำนวณดีเลย์ได้ 460 MC ทำไมค่าที่ใช้จึงเป็น -460 และทำไมต้องบวกด้วย 17

Q12: ถ้าต้องการสร้างความถี่เป็น 1.5 kHz จะปรับแก้โปรแกรมอย่างไร

Q13: ถ้าต้องการสร้างความถี่เป็น 1.0 kHz แต่ดิวตี้ไซเคิลเป็น 66% จะปรับแก้โปรแกรมอย่างไร

2. หากใช้ Port P3.2, P3.3, P3.4 และ P3.5 กดเพื่อเลือกความถี่ 1 kHz, 2 kHz, 4 kHz และ 8 kHz ตามลำดับจะทำอย่างไร

3. ให้เขียนโปรแกรมตาม ***Lab6_Counter1.Asm*** เป็นโปรแกรมสำหรับการนับค่าด้วยเทคนิคการวนรอรับสัญญาณที่ขา P3.4(T0) ทดสอบด้วยการแตะสัญญาณ GND ที่ขา T0 สังเกตผลการนับที่ได้

Q31: เกิดอะไรขึ้นถ้าสัญญาณอินพุตเข้ามาเร็วมาก แก้ไขโปรแกรมอย่างไร

Q32: ถ้าลด (หรือเพิ่ม) Delay จะเกิดอะไรขึ้นกับโปรแกรม

4. ให้เขียนโปรแกรมตาม ***Lab6_Counter2.Asm*** เป็นโปรแกรมสำหรับการนับค่าด้วยการใช้ Counter0 เพื่อนับสัญญาณที่ขา P3.4(T0) ทดสอบด้วยการแตะสัญญาณ GND ที่ขา T0 สังเกตผลการนับที่ได้

2/2

Q41: หากต้องการรับสัญญาณที่ขา T1 จะปรับปรุงแก้ไขโปรแกรมอย่างไร

Q42: โปรแกรมบรรทัดที่ 7 เป็น MOV TMOD,#00000001B เมื่อทดสอบโปรแกรมจะเกิดอะไรขึ้น

Q43: หากต้องการให้โปรแกรมนับจาก 0 ถึง 9 แล้วหยุดการนับ จะปรับปรุงแก้ไขโปรแกรมอย่างไร

Q44: เปลี่ยนการทดสอบจากการแตะสัญญาณ GND ที่ขา T0 เป็นป้อน Clock 1Hz จาก Logic Trainer Board จะเกิดอะไรขึ้น

5. ให้เขียนโปรแกรมตาม ***Lab6_Counter3.Asm*** เป็นโปรแกรมสำหรับการนับค่าด้วยการใช้ Counter0 เพื่อนับสัญญาณที่ขา P3.4(T0) และแสดงผลการนับ TH0-TL0 ที่ 7_Segment 6หลัก ทดสอบด้วยการป้อน Clock 1Hz จาก Logic Trainer Board ที่ขา T0 สังเกตผลการนับที่ได้

Q51: Address E060H, E001H และ E002H ในโปรแกรมคือพอรต์อะไร

Q52: หากต้องการให้โปรแกรมนับจาก 0000 ถึง 0123H แล้วหยุดการนับ จะปรับปรุงแก้ไขโปรแกรมอย่างไร

## 6.5 วิเคราะห์และอภิปรายผลการทดลอง

1. จงอธิบายหลักการใช้งานตัวจับเวลาและตัวนับของ ET-8032
2. จงอธิบายความแตกต่างของโหมดการทำงานต่าง ๆ ทั้งในกรณีของตัวจับเวลาและตัวนับ
3. จงอธิบายการทำงานของโปรแกรมที่ดำเนินการทดลองและตอบคำถามของท้ายการทดลองข้อ 1
4. จงอธิบายการทำงานของโปรแกรมที่ดำเนินการทดลองและตอบคำถามของท้ายการทดลองข้อ 4

## 6.6 คำถามท้ายการทดลอง

1. กำหนดให้ใช้ Timer1 Mode1 ทำงานเป็น Timer เพื่อสร้างโปรแกรมย่อยหน่วงเวลา 25 msec

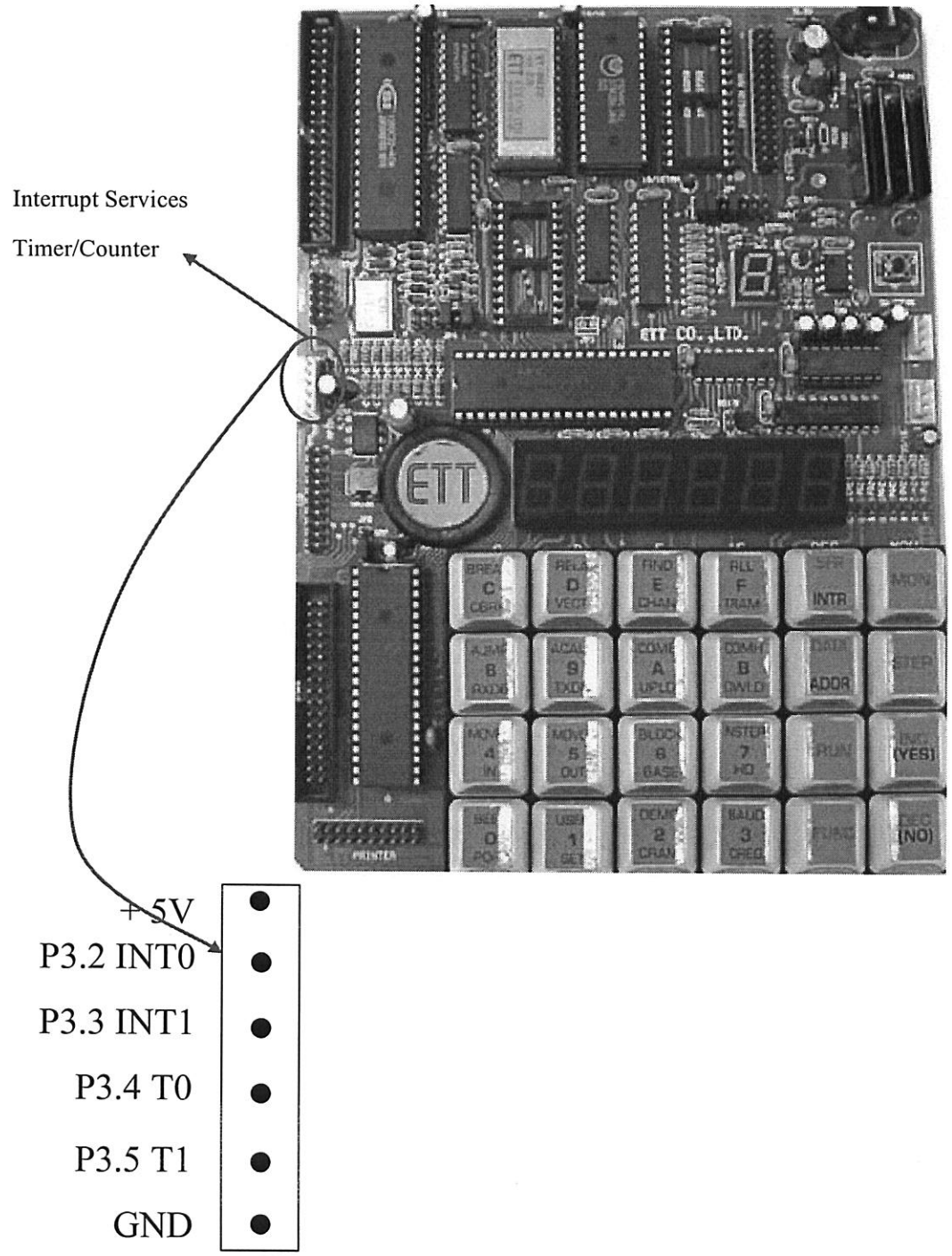
Step 1: ให้เขียนโปรแกรมหลักให้ LED ขนาดเล็ก 8 ตัวติดกระพริบให้ติด 300 msec ดับ 200 msec

Step 2: ให้ใช้ Port P1.0 ในการควบคุมการกระพริบโดย
กดครั้งที่หนึ่งให้ LED กระพริบ และกดอีกครั้งให้ LED ดับ ทำเช่นนี้สลับกันไปเรื่อยๆ

2. ให้เขียนโปรแกรมสร้างสัญญาณ Square Wave โดยใช้ Timer1 Mode 2 เท่านั้น โดยมีกำหนดดังนี้

- สร้างความถี่ 12,123±10 Hz ออกทาง P1 และลำโพง E000H ตลอดเวลา พร้อมแสดงวิธีคำนวณ
- กำหนด Duty cycle เท่ากับ 40%

## 6.7 อ้างอิง


Interrupt Services
Timer/Counter
ETT CO.,LTD.
ETT
+5V
P3.2 INT0
P3.3 INT1
P3.4 T0
P3.5 T1
GND

## 6.8 โปรแกรมทดสอบ

- ***Lab6_Timer.Asm***

สร้างความถี่ 1 kHz = 1000 MicroSec

Duty Cycle 50% Ton=Toff = 500 MicroSec

= 500*11.0592/12 MC

= 460.8 MC

```
        ORG   8000H             ; Lab6_Timer.Asm        1
        MOV   TMOD,#00010001B    ; Timer0,1 Mode1 16Bit  2
                                 ;                       3
T_OFF:  CLR   TR1                ; Stop Timer            4
        MOV   A,#00H             ; Out Pulse             5
        MOV   P1,A               ; Out P1                6
        MOV   DPTR,#0E000H       ; Out Sound             6
        MOVX  @DPTR,A            ;                       7
        MOV   DPTR,#-460+17      ; Reload Data           8
        MOV   TH1,DPH            ; For TH1               9
        MOV   TL1,DPL            ; For TL1               10
        CLR   TF1                ; Initial Start         11
        SETB  TR1                ; Start Timer1          12
        JNB   TF1,$              ; Wait Until TF On      13
                                 ;                       14
T_ON:   CLR   TR1                ; 1MC                   15
        MOV   A,#0FFH            ; 1MC                   16
        MOV   P1,A               ; 1MC                   17
        MOV   DPTR,#0E000H       ; 2MC                   18
        MOVX  @DPTR,A            ; 2MC                   19
        MOV   DPTR,#-460+17      ; 2MC                   20
        MOV   TH1,DPH            ; 2MC                   21
        MOV   TL1,DPL            ; 2MC                   22
        CLR   TF1                ; 1MC                   23
        SETB  TR1                ; 1MC                   24
        JNB   TF1,$              ; 2MC                   25
                                 ;                       26
        JMP   T_OFF              ;                       27
        END                      ;                       28
```

- ***Lab6_Counter1.Asm***

โปรแกรมสำหรับอ่านค่าการนับสัญญาณพัลส์ที่ P3.4 โดยไม่ใช้เทคนิค Counter

```
P_TestIn    EQU  P3.4
Counter     EQU  40H

            ORG  8000H            ; Lab6_Counter1.Asm
            CLR  EA
            SETB P_TestIn         ; Input Port
            MOV  Counter,#0
Main_Loop:  CALL Display
            JB   P_TestIn,$
            JNB  P_TestIn,$
            INC  Counter
            CALL Delay_XXX
            JMP  Main_Loop

Display:    MOV  A,Counter
            ANL  A,#0FH
            MOV  DPTR,#CODE_TABLE
            MOVC A,@A+DPTR
            MOV  DPTR,#0E060H
            MOVX @DPTR,A
            RET

CODE_TABLE: DB   3FH,06H,5BH,4FH
            DB   66H,6DH,7DH,07H
            DB   7FH,6FH,77H,7CH
            DB   79H,5EH,79H,71H

Delay_XXX:  CLR  A
            MOV  B,#200
_Dly00:     DJNZ Acc,_Dly00
            DJNZ B,_Dly00
            RET

            END
```

- ***Lab6_Counter2.Asm***

โปรแกรมสำหรับอ่านค่าการนับสัญญาณพัลส์ที่ P3.4 (T0) โดยใช้เทคนิค Counter

```
P_TestIn    EQU  P3.4
Counter     EQU  40H

            ORG  8000H              ; Lab6_Counter2.Asm
            CLR  EA
            SETB P_TestIn
            MOV  TMOD,#00000101B
            MOV  TH0,#0
            MOV  TL0,#0
            SETB TR0

Main_Loop:  MOV  Counter,TL0
            CALL Display
            JMP  Main_Loop

Display:    MOV  A,Counter
            ANL  A,#0FH
            MOV  DPTR,#CODE_TABLE
            MOVC A,@A+DPTR
            MOV  DPTR,#0E060H
            MOVX @DPTR,A
            RET

CODE_TABLE: DB   3FH,06H,5BH,4FH
            DB   66H,6DH,7DH,07H
            DB   7FH,6FH,77H,7CH
            DB   79H,5EH,79H,71H

Delay_XXX:  CLR  A
            MOV  B,#200
_Dly00:     DJNZ Acc,_Dly00
            DJNZ B,_Dly00
            RET

            END
```

- ***Lab6_Counter3.Asm***

โปรแกรมสำหรับอ่านค่าการนับสัญญาณพัลส์ที่ P3.4 (T0) โดยใช้เทคนิค Counter

```
;======= Lab6_Counter3.Asm ,Long=======
        ORG     8000H
        CLR     EA
        MOV     TMOD,#00000101B; Counter 0 Mode 1
        MOV     TH0,#00H
        MOV     TL0,#00H  ; Start TH0TL0 at 0000
        SETB    TR0             ; Start Counter_0
Main_LOOP:
        MOV     DPH,TH0
        MOV     DPL,TL0
        CALL    DISPLAY_DPTR    ; Show TH0TL0
        JMP     Main_LOOP

DISPLAY_DPTR:
        MOV     R0,#1   ; Position
        MOV     R6,DPH  ; DPTR = 5678H
        MOV     R7,DPL
        MOV     A,R6
        SWAP    A
        CALL    _Display        ; Send 5
        MOV     A,R6
        CALL    _Display        ; Send 6
        MOV     A,R7
        SWAP    A
        CALL    _Display        ; Send 7
        MOV     A,R7
        CALL    _Display        ; Send 8
        RET

_Display:
        ANL     A,#0FH
        MOV     DPTR,#CODE_TABLE
        MOVC    A,@A+DPTR
        MOV     DPTR,#0E001H
        MOVX    @DPTR,A
        MOV     A,R0
        MOV     DPTR,#0E002H
        MOVX    @DPTR,A
        INC     R0
        CLR     A
        DJNZ    Acc,$
        RET

CODE_TABLE:
        DB      3FH,06H,5BH,4FH
        DB      66H,6DH,7DH,07H
        DB      7FH,6FH,77H,7CH
        DB      79H,5EH,79H,71H

        END
```

```
;======= Lab6_Counter3.Asm ,Short=======
        ORG     8000H
        CLR     EA
        MOV     TMOD,#00000101B; Counter 0 Mode 1
        MOV     TH0,#00H
        MOV     TL0,#00H; Start TH0TL0 at 0000
        SETB    TR0     ; Start Counter_0
Main_LOOP:
        MOV     DPH,TH0
        MOV     DPL,TL0
        CALL    DISPLAY_DPTR    ; Show TH0TL0
        JMP     Main_LOOP

DISPLAY_DPTR:
        MOV     R0,#1   ; Position
        PUSH    DPL
        MOV     A,DPH
        CALL    _Display00
        POP     Acc
_Display00:
        PUSH    Acc
        SWAP    A
        CALL    _Display01
        POP     Acc
_Display01:
        ANL     A,#0FH
        MOV     DPTR,#CODE_TABLE
        MOVC    A,@A+DPTR
        MOV     DPTR,#0E001H
        MOVX    @DPTR,A
        MOV     A,R0
        MOV     DPTR,#0E002H
        MOVX    @DPTR,A
        INC     R0
        CLR     A
        DJNZ    Acc,$
        RET

CODE_TABLE:
        DB      3FH,06H,5BH,4FH
        DB      66H,6DH,7DH,07H
        DB      7FH,6FH,77H,7CH
        DB      79H,5EH,79H,71H

        END
```

## 6.9 ตารางบันทึกผล

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 4. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 5. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 6. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

# การทดลองที่ 7

# การสื่อสารผ่านพอร์ตอนุกรมของ ET-8032

## 7.1 วัตถุประสงค์

1. เพื่อศึกษาการติดต่อสื่อสารระหว่าง ET-8032 และ PC ได้

## 7.2 อุปกรณ์ทดลอง

1. ET-8032 V2.0 MCS-51 single board microcontroller
2. เครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับเชื่อมต่อกับ ET-8032

## 7.3 เนื้อหา ทฤษฎีที่เกี่ยวข้อง

- รีจีสเตอร์ที่สำคัญ

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **TMOD** | **G(T1)** | **C/T** | **M1** | **M0** | **G(T0)** | **C/T** | **M1** | **M0** |
| **TCON** | **TF1** | **TR1** | **TF0** | **TR0** | **IE1** | **IT1** | **IE0** | **IT0** |
| **SCON** | **SM0** | **SM1** | **SM2** | **REN** | **TB8** | **RB8** | **TI** | **RI** |

- เริ่มต้นใช้งาน Serial Communication
  - Data Frame → SCON
  - Baud Rate → TMOD,TCON

### Using Timer1, Mode2

http://www.intel.com/design/mcs51/applnots/270490.htm

Serial Mode1,3 และใช้ Timer 1Mode2 → $\text{BaudRate} = \frac{2^{\text{SMOD1}} \times \text{Oscillater Frequency}}{32 \times 12 \times [256 - (\text{th1})]}$

9600 @18.432MHz → SMOD1=0, TH1=251(FBH)

9600 @11.0592MHz → SMOD1=0, TH1=253(FDH)

```
MOV   SCON,#050H ; Serial Port Mode 3
ANL   PCON,#7FH  ; SMOD 1=2
MOV   TMOD,#20H  ; Timer1 Mode2
MOV   TH1,#0FDH  ; Reload value for desired baud
SETB  TR1        ; Turn on Timer1
```

Using Timer2

$$\text{Buad rate} = \frac{\text{Oscillater Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

9600 @18.432MHz → RCAP2H,RCAP2L = 65476 = FFC4H

9600 @11.0592MHz → RCAP2H,RCAP2L = 65500 = FFDCH

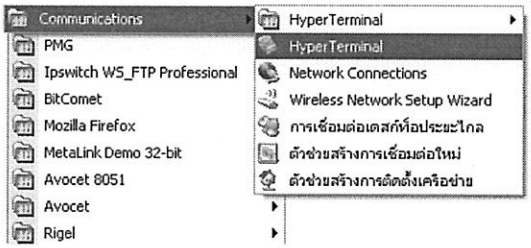
```
MOV   SCON,#01010000B  ; Mode1 Serial Port
MOV   RCAP2H,#0FFH     ; Reload values for desired
MOV   RCAP2L,#0DCH     ; baud rate 9600@11.0592
MOV   T2CON,#00110100B ; Timer 2 as baud rate generator
                       ; and turn on Timer 2
```

## 7.4 การทดลอง

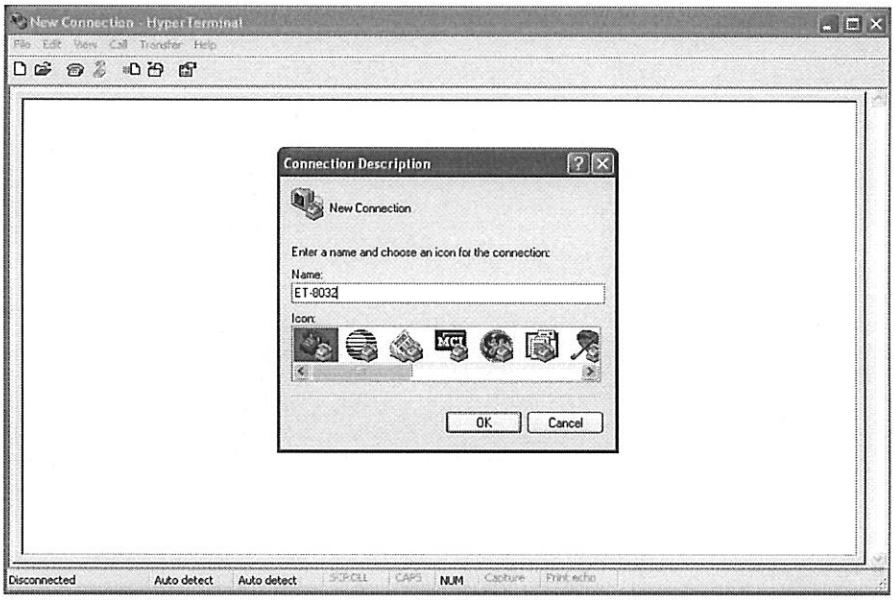
1. การเรียกใช้โปรแกรม Hyper Terminal

   1.1 เรียกใช้โปรแกรมจาก Windows Desktop ดังนี้ (ต้องปิดโปรแกรม PCPLUS ก่อน)

   Start --> Program --> Accessories --> Communications --> HyperTerminal

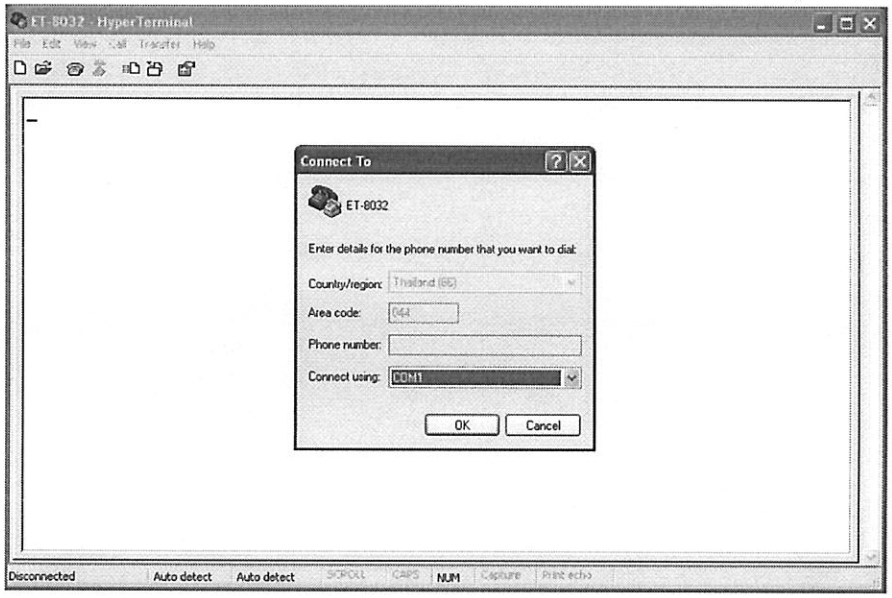



   1.2 จะปรากฏหน้าต่างดังต่อไปนี้

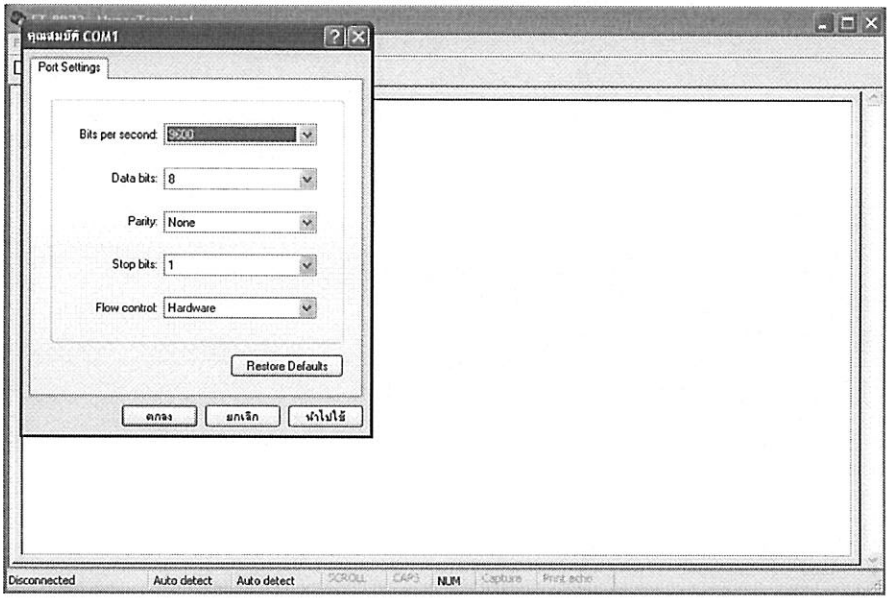


1.3 ให้ตั้งชื่ออุปกรณ์ที่ต้องการเชื่อมต่อกับพอร์ตอนุกรม เช่น ET-8032 และเลือกไอคอนรูปโทรศัพท์ทางด้านซ้ายมือสุด

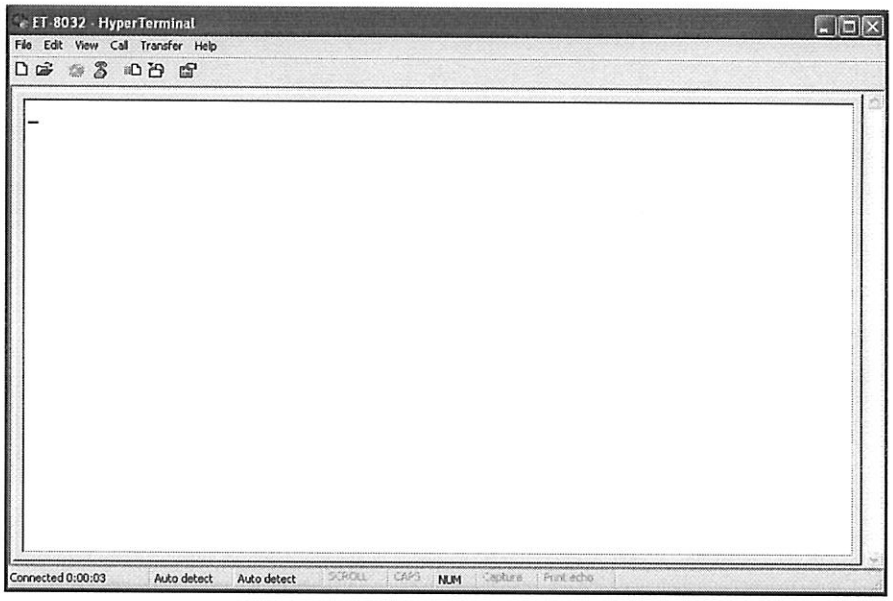
1.4 กด OK จากนั้นเลือกประเทศ รหัสโทรศัพท์ท้องถิ่น และพอร์ตการเชื่อมต่อให้ถูกต้อง




1.5 กำหนดอัตราบอด (baud rate) และจำนวนบิตการส่งข้อมูลให้ถูกต้อง



1.6 กด OK จะได้หน้าต่างของโปรแกรมไฮเปอร์เทอร์มินอลที่พร้อมใช้งาน




2. ศึกษาการเขียนโปรแกรมเพื่อส่งค่าผ่าน Serial Port เพื่อแสดงผลที่หน้าจอโปรแกรม Hyper Terminal จากโปรแกรม "Lab7Srl1.Asm" และทดสอบการทำงานของโปรแกรม

   Q21. จากโปรแกรม "Lab7Srl1.Asm" ให้ศึกษาความสำคัญของโปรแกรมแต่ละบรรทัด

   Q22. จากโปรแกรม "Lab7Srl1.Asm" ปรับปรุงโปรแกรมให้บอร์ดทำหน้าที่ส่งข้อมูลไปยังเครื่องพีซี ด้วย Baud Rate = 9600 bps ให้โปรแกรมวนลูปส่งรหัสแอสกี้ออกไปยังเครื่องพีซีทีละ 1 ตัวทุก ๆ 0.5 วินาที(โดยประมาณ) ➔ " 0,1,2,3,4,5,6,7,8,9,ขึ้นบรรทัดใหม่ และชิดซ้ายบรรทัด"

   Q23. จากโปรแกรม "Lab7Srl1.Asm" ปรับปรุงโปรแกรมให้วนลูปส่ง " รหัสนักศึกษา, ชื่อ-นามสกุล, ขึ้นบรรทัดใหม่-ชิดซ้าย, หน่วงเวลา 500mSec" วนไปเรื่อยๆ

3. ศึกษาการเขียนโปรแกรมเพื่อรับค่าจาก Serial Port ผ่านโปรแกรม Hyper Terminal จากโปรแกรม "Lab7Srl2.Asm" และทดสอบการทำงานของโปรแกรม โดยผลการทดสอบเมื่อกดค่า A หรือ B ที่ PC Keyboard ไมโครคอนโทร์ลเลอร์จะแสดงค่าที่ Display E060H

   Q31. จากโปรแกรม "Lab7Srl2.Asm" ให้ศึกษาความสำคัญของโปรแกรมแต่ละบรรทัด

   Q32. จากโปรแกรม "Lab7Srl2.Asm" เดิมมีการรับคีย์เฉพาะ A หรือ Bให้ปรับปรุงเพื่อแสดงผลเฉพาะกรณีที่มีการกดคีย์จาก 0-9 และคีย์อื่นๆ ให้แสดง – (ขีดกลาง)

4. การสร้างโปรแกรมย่อยเพื่อเรียกใช้
   - พิมพ์โค้ดโปรแกรมตามกรอบ "System.Sub" แล้วทำการบันทึกที่โฟล์เดอร์ "C:\Ett" ชื่อไฟล์ "System.Sub"
   - โปรแกรมหลักสามารถเรียกใช้ทุกชุดคำสั่งย่อยที่มีใน System.Sub ได้ทุกชุดแต่ต้องมีการประกาศบรรทัด $Include "System.Sub" ที่โปรแกรมหลักก่อนตามตัวอย่าง "Lab7Srl3.Asm" เป็นต้น
   - จากโปรแกรม "Lab7Srl3.Asm" และทดสอบการทำงานของโปรแกรม

   Q41. จากโปรแกรม "Lab7Srl3.Asm" ให้ศึกษาความสำคัญของโปรแกรมแต่ละบรรทัด

   Q42. จากโปรแกรม "Lab7Srl3.Asm" ปรับปรุงโปรแกรมให้การนับของ Counter เป็นการนับเลขแบบBCD

## 7.5 วิเคราะห์และอภิปรายผลการทดลอง

1. จงอธิบายหลักการสื่อสารผ่านพอร์ตอนุกรมระหว่าง ET-8032 และ PC
2. จงอธิบายหน้าที่ ความสำคัญและการใช้งานรีจิสเตอร์ TMOD SCON TH1 และ SBUF
3. จงอธิบายหน้าที่ และความสำคัญของ RI และ TI ที่ใช้ในการทดลอง
4. จงอธิบายการทำงานของโปรแกรมที่ดำเนินการทดลองและตอบคำถามของท้ายการทดลองข้อ 2.Q23
5. จงอธิบายการทำงานของโปรแกรมที่ดำเนินการทดลองและตอบคำถามของท้ายการทดลองข้อ 3.Q32

## 7.6 คำถามท้ายการทดลอง

1. จงแสดงการคำนวณค่า ของรีจีสเตอร์ RCAP2HRCAP2L เพื่อตั้งค่าอัตราบอดตามตารางสำหรับไมโครคอนโทรลเลอร์ 8051 ที่มี oscillator frequency ต่างๆ พร้อมคำนวณ Error

- จากสมการ $$\text{Buad rate} = \frac{\text{Oscillater Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

ถ้า Baud = 9600, 12.000MHz ได้ RCAP2HRCAP2L = 65496.94

เลือก RCAP2HRCAP2L = 65497

คำนวณกลับได้ Baud rate = 6915.385

Error = $\frac{15.385}{1900} \times 100\%$

| Item | อัตราบอด | Crystal (MHz) | Timer2 / RCAP2H,RCAP2L | Error |
|---|---|---|---|---|
| 1 | 9,600 | 11.0592 | 65500 | 0 |
| 2 | 9,600 | 18.432 | | |
| 3 | 9,600 | 12.000 | 65497 | 0.16% |
| 4 | 19,200 | 11.0592 | | |
| 5 | 19,200 | 18.432 | | |
| 6 | 19,200 | 12.000 | | |

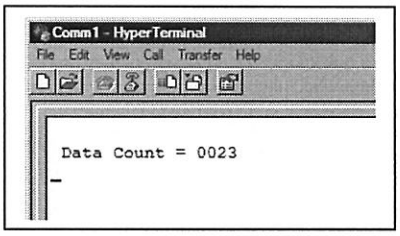
2. จงเขียนโปรแกรมเพื่อรับอินพุตจากสวิตช์ที่ต่อกับพอร์ต P1 จำนวนทั้งสิ้น 8 ตัว โดยกำหนดให้ P1.1 – P1.7 ทำหน้าที่กำหนดจำนวนบรรทัดของข้อมูลทั้งหมดที่จะส่งไปยังโปรแกรมไฮเปอร์เทอร์มินอล และ P1.0 เป็นสวิตช์ควบคุมให้ส่งข้อมูลได้ ดังนี้

| พอร์ต | การทำงาน |
|---|---|
| P1.0 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบแบบไม่รู้จบ |
| P1.1 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบจำนวน 1 บรรทัด |
| P1.2 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบจำนวน 2 บรรทัด |
| P1.3 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบจำนวน 3 บรรทัด |
| P1.4 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบจำนวน 4 บรรทัด |
| P1.5 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบจำนวน 5 บรรทัด |
| P1.6 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบจำนวน 6 บรรทัด |
| P1.7 = 0 | ให้ส่งตัวอักษร A ถึง Z แล้วขึ้นบรรทัดใหม่ วนรอบจำนวน 7 บรรทัด |

3. ให้เขียนโปรแกรมสำหรับนับจำนวนชิ้นงานที่ไหลผ่านสายพานลำเลียง**ข้อสอบเก่า (แนวข้อสอบ)**

Step 1: ทดสอบ Serial Port โดยให้ส่งข้อมูลไปยังเครื่องพีซี ด้วย Baud Rate = 9600 bps ให้โปรแกรมวนลูปส่งรหัสแอสกี้ออกไปยังเครื่องพีซีทีละ 1 ตัวทุก ๆ 0.5 วินาที ตามลำดับดังนี้ → " 0,1,2,3,4,5,6,7,8,9,ขึ้นบรรทัดใหม่ และชิดซ้ายบรรทัด "

Step 2: ทดสอบการนับโดยให้โปรแกรมควบคุม Timer 0 ให้ทำงานเป็น Counter mode 1 ใช้นับเลข HEX จากสัญญาณภายนอก จากนั้นนำค่า TH0TL0 ส่งออกผ่าน Serial Port ไปยังเครื่องพีซีเพื่อแสดงผล



## 7.7 โปรแกรมทดสอบ

- **Lab7Srl1.ASM**

```
            ORG     8000H           ; Lab7Srl1.Sub
            CLR     EA              ; Disable All
            CALL    Delay           ; Start Delay
            MOV     TMOD,#00100001B ; T1M2 T0M1
            MOV     SCON,#01010000B ; Serial Mode1
            MOV     TH1,#0FDH       ; 9600 FDH@11.0592MHz
            SETB    TR1             ; Start Timer1

Main_Loop:  MOV     A,#'A'
            MOV     SBUF,A
            JNB     TI,$
            CLR     TI
            CALL    Delay
            JMP     Main_Loop

Delay:      CLR     A
            MOV     B,A
Dly00:      DJNZ    Acc,Dly00
            DJNZ    B,Dly00
            RET

            END
```

- **Lab7Srl2.ASM**

```
            ORG     8000H           ; Lab7Srl2.Sub
            CLR     EA              ; Disable All
            MOV     DPTR,#0E060H
            MOV     TMOD,#00100001B ; T1M2 T0M1
            MOV     SCON,#01010000B ; Serial Mode1
            MOV     TH1,#0FDH       ; 9600 FDH@11.0592MHz
            SETB    TR1             ; Start Timer1

Main_Loop:  CLR     RI              ; Ready for New Data
            JNB     RI,$            ; Wait Until Data In
            MOV     A,SBUF          ; Get Data In

_IS_ASCII_A: CJNE   A,#'A',_IS_ASCII_B
            MOV     A,#77H
            MOVX    @DPTR,A
            JMP     Main_Loop

_IS_ASCII_B: CJNE   A,#'B',_IS_ASCII_Etc
            MOV     A,#7CH
            MOVX    @DPTR,A
            JMP     Main_Loop

_IS_ASCII_Etc: NOP
            JMP     Main_Loop

            END
```

- **System.Sub**

```
;============ Initial Serial Port ============
Initial_Serial:   MOV   TMOD,#00100001B   ; T1M2 T0M1
                  MOV   SCON,#01010000B   ; Serial Mode1
                  MOV   TH1,#0FDH         ; 9600 FDH@11.0592MHz
                  MOV   A,#00H            ; FBH@18.432MHz
                  MOV   PCON,A            ; SMOD = 0
                  CLR   EA                ; Disable All
                  SETB  TR1               ; Start Timer1
                  RET

;========== Send Data to Serial Port =========
SEND_2HEX:        PUSH  Acc
                  SWAP  A
                  CALL  SEND_1HEX
                  POP   Acc
SEND_1HEX:        PUSH  DPH
                  PUSH  DPL
                  MOV   DPTR,#HEXASC_TAB
                  ANL   A,#0FH
                  MOVC  A,@A+DPTR
                  POP   DPL
                  POP   DPH
SEND_ASCII:       PUSH  IE
                  CLR   ES
                  CLR   TI
                  MOV   SBUF,A
                  JNB   TI,$
                  CLR   TI
                  POP   IE
                  RET
HEXASC_TAB:       DB    '0123456789ABCDEF'

;========= Send Line Feed ====================
Send_LineFeed:    MOV   A,#0DH
                  CALL  Send_ASCII
                  MOV   A,#0AH
                  CALL  Send_ASCII
                  RET

;======== Send Line Feed ======================
Send_ClrScr:      MOV   A,#0CH
                  CALL  Send_ASCII
                  RET

;== Send Data Table to Serial Start@DPTR to '00H' =
Send_Table:       CLR   A
                  MOVC  A,@A+DPTR
                  JZ    _X_Send_Table
                  CALL  SEND_ASCII
                  INC   DPTR
                  JMP   Send_Table
_X_Send_Table:    RET
```

```
;========== Send 1-Byte@Acc to SERIAL ==========
TX_BYTE:        PUSH    IE
                CLR     ES
                CLR     TI
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                POP     IE
                RET

;====== Receive Data From SERIAL to Acc ========
RX_BYTE:        PUSH    IE
                CLR     ES
                JNB     RI,$                    ; Wait data
                CLR     RI
                MOV     A,SBUF
                POP     IE
                RET

;#################################################
;#************************************************
;#     All Delay for XTAL 18.432MHz 12Clk/MC
;#************************************************
DELAY_4SEC:     PUSH    07H                     ; Delay 80*50MilliSec
                MOV     R7,#80                  ;  = 4000MilliSec
                JMP     _DELAY_50M

DELAY_2SEC:     PUSH    07H                     ; Delay 40*50MilliSec
                MOV     R7,#40                  ;  = 2000MilliSec
                JMP     _DELAY_50M

DELAY_1SEC:     PUSH    07H                     ; Delay 20*50MilliSec
                MOV     R7,#20                  ;  = 1000MilliSec
                JMP     _DELAY_50M

DELAY_500M:     PUSH    07H                     ; Delay 10*50MilliSec
                MOV     R7,#10                  ;  = 500MilliSec
                JMP     _DELAY_50M

DELAY_250M:     PUSH    07H                     ; Delay 5*50 MilliSec
                MOV     R7,#5                   ;  = 250 MilliSec
                JMP     _DELAY_50M

DELAY_100M:     PUSH    07H                     ; Delay 2*50 MilliSec
                MOV     R7,#2                   ;  = 100 MilliSec
                JMP     _DELAY_50M

DELAY_50M:      PUSH    07H                     ; Delay 1*50 MilliSec
                MOV     R7,#1                   ;  = 50 MilliSec
                JMP     _DELAY_50M

_DELAY_50M:     PUSH    Acc
                PUSH    B
_DELAY_050M:    MOV     B,#200                  ; Delay 50 MilliSec
```

```
_DELAY_250U:  MOV   A,#191          ; 1MC ,250 uSec
              DJNZ  Acc,$           ; 2MC ,(192-1)*0.651*2 = 250 uSec
              DJNZ  B,_DELAY_250U   ; 250*200=50mS
              DJNZ  R7,_DELAY_050M  ; 50*5 =250mS
              POP   B
              POP   Acc
              POP   07H
              RET
```

- **Lab7Srl3.ASM**

```
Counter       EQU     40H             ; Lab7Srl1.Sub
PortO_LEDGreen EQU    P3.2
              ORG     8000H
              CLR     EA
              CALL    Delay_250M
              CALL    Initial_Serial
              CALL    Send_ClrScr
              MOV     Counter,#00
Main_Loop:    MOV     DPTR,#Header_Fram
              CALL    Send_Table
              MOV     A,Counter
              CALL    Send_2HEX
              CALL    Send_LineFeed
              CPL     PortO_LEDGreen
              INC     Counter
              CALL    Delay_500M
              JMP     Main_Loop
              $INCLUDE "System.Sub"
Header_Fram:  DB      'Data Counter = ',00
              END
```

## 7.8 ตารางบันทึกผล

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 4. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 5. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 6. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |

# การทดลองที่ 8

# การบริการอินเตอร์รัปต์จากภายนอก ET-8032

## 8.1 วัตถุประสงค์

1. เพื่อศึกษาการให้บริการอินเตอร์รัปต์ (interrupt services) ของ ET-8032 ได้

## 8.2 อุปกรณ์ทดลอง

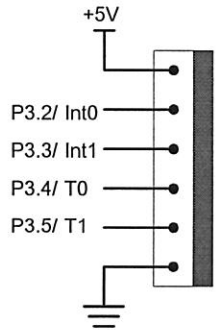
1. ET-8032 V2.0 MCS-51 single board microcontroller
2. CPDL Logic Board
3. เครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับเชื่อมต่อกับ ET-8032

## 8.3 เนื้อหา ทฤษฎีที่เกี่ยวข้อง

**1. Interrupts Vectors**

| Interrupt Source | Single Chip 8052 | ET-8032 V2 Board | Priority |
|---|---|---|---|
| Power On Reset | 0000H | - | 0 |
| External Interrupt 0 | 0003H | 9F3FH | 1 |
| Timer 0 Interrupt | 000BH | 9F42H | 2 |
| External Interrupt 1 | 0013H | 9F45H | 3 |
| Timer 1 Interrupt | 001BH | 9F48H | 4 |
| Serial Interrupt | 0023H | 9F4BH | 5 |
| Timer 2 Interrupt | 002BH | 9F4EH | 6 |

**2. การใช้พอร์ต 3 ในฟังก์ชันอินเตอร์รัปท์**



**3. การตั้งค่า Register ที่เกี่ยวข้องเพื่อใช้งาน Interrupts**

***IE: Interrupt Enable Register*** มีตำแหน่งหน่วยความจำภายในที่ A8H

| EA | - | ET2 | ES | ET1 | Ex1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|

EA: เป็นบิตควบคุมการ Disable Interrupt ทั้งหมด โดยถ้าเป็น 0 จะหมายความว่าจะเป็นการกำหนดไม่ให้เกิดการ Interrupt ทั้งหมด แต่ถ้าเป็น 1 หมายความว่าการ Enable/Disable จะขึ้นอยู่กับบิตควบคุมของ Interrupt source แต่ละแหล่ง

ET2 : เป็นควบคุมการสั่งให้เกิดการ Enable Timer2 Interrupt โดยถ้าถูกกำหนดเป็น 1 จะหมายความว่าสามารถเกิดการ Interrupt ได้เมื่อ Timer2 เกิดการ Overflow

ES : เป็นควบคุมการสั่งให้เกิดการ Enable Serial Port โดยถ้าถูกกำหนดเป็น 1 จะหมายความว่าสามารถเกิดการ Interrupt ได้เมื่อมีข้อถูกรับส่งผ่าน Serial Port

ET1 : เป็นควบคุมการสั่งให้เกิดการ Enable Timer1 Interrupt โดยถ้าถูกกำหนดเป็น 1 จะหมายความว่าสามารถเกิดการ Interrupt ได้เมื่อ Timer2 เกิดการ Overflow

EX1 : เป็นควบคุมการสั่งให้เกิดการ Enable External Interrupt โดยถ้าถูกกำหนดเป็น 1 จะหมายความว่าสามารถเกิดการ Interrupt ได้เมื่อระดับสัญญาณที่ขา INT1 มีระดับเป็น 0 หรือเกิดการเปลี่ยนแปลงจาก 1 เป็น 0 ขึ้นการกับการกำหนดค่าที่บิต ***IT1*** ในรีจิสเตอร์ TCON ด้วย

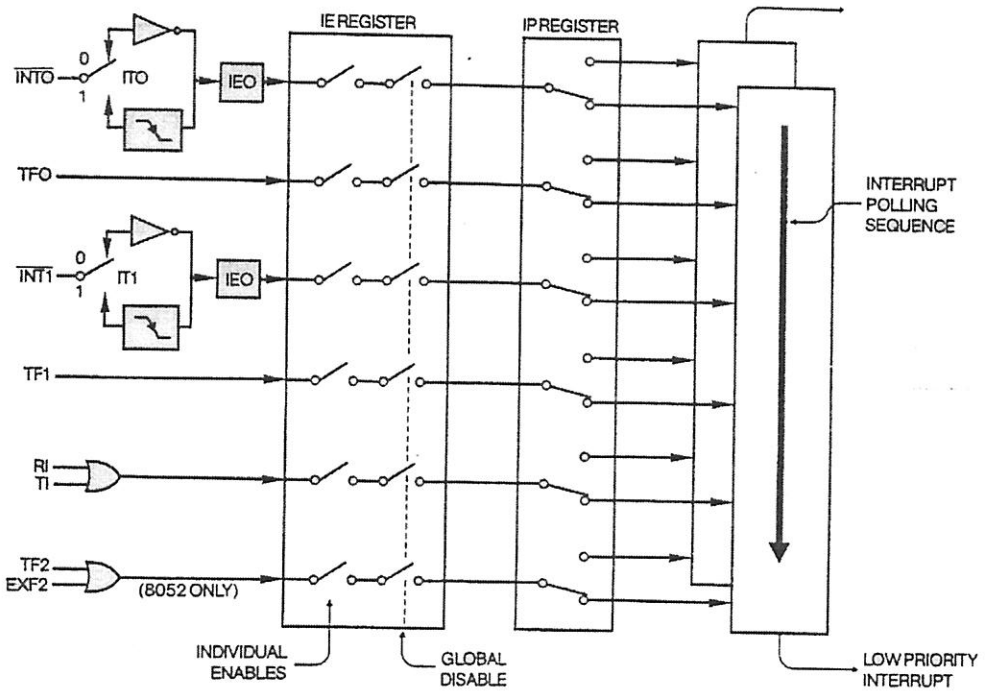
ET0 : เป็นควบคุมการสั่งให้เกิดการ Enable Timer0 Interrupt โดยถ้าถูกกำหนดเป็น 1 จะหมายความว่าสามารถเกิดการ Interrupt ได้เมื่อ Timer0 เกิดการ Overflow

EX0 : เป็นควบคุมการสั่งให้เกิดการ Enable External Interrupt โดยถ้าถูกกำหนดเป็น 1 จะหมายความว่าสามารถเกิดการ Interrupt ได้เมื่อระดับสัญญาณที่ขา INT0 มีระดับเป็น 0 หรือเกิดการเปลี่ยนแปลงจาก 1 เป็น 0 ขึ้นการกับการกำหนดค่าที่บิต ***IT0*** ในรีจิสเตอร์ TCON ด้วย

***IP : Interrupt Priority*** มีตำแหน่งหน่วยความจำภายในคือ 0B8H

| - | - | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

ในการตอบสนอง Interrupt ถ้าหากเราต้องการใช้สัญญาณ Interrupt จากแหล่งกำเนิดหลายแหล่ง เราต้องมีการกำหนดระดับความสำคัญ (priority) ของสัญญาณ Interruptสำหรับรีจิสเตอร์ที่ใช้ในการกำหนดความสัญคือ IP แต่ถ้าหากเราไม่ได้กำหนดความสำคัญของ Interrupt ในตัวของ MCS-51 จะเรียงลำดับความสำคัญตามค่าต่อไปนี้ จากความสำคัญสูงสุดไปยังความสำคัญน้อยที่สุดดังนี้ IE0, TF0, IE1, TF1 และ RI+TI




รูปแสดงการทำงานของระบบการขัดจังหวะใน MCS-51

## 8.4 การทดลอง

1. ศึกษาการทำงานของโปรแกรม "Lab81Ex1.Asm" ที่ให้มาซึ่งเป็นโปรแกรมหลักวนลูป เพื่อนำค่าในตัวแปร Data_Count ไปแสดงผลที่ 7_Segment หากเมื่อไรมีสัญญาณอินเตอร์รัพท์ที่ขา INT1 จากภายนอกโปรแกรมย่อยบริการอินเตอร์รัพท์ที่เก็บไว้จะทำการเพิ่มค่าในตัวแปร Data_Count จากนั้นทำการ Delay เพื่อแก้ Bounce Clear IE1 และกลับสู่โปรแกรมหลัก

เมื่อ Run โปรแกรมทดสอบแล้วทำการ ป้อนลอจิก 0 โดยการแตะที่ขา Int1

**Q11:** จากโปรแกรมนี้จะเกิดอะไรขึ้นถ้าไม่มีบรรทัด CALL Debounce และ CLR IE1 ใน EX_Routine

**Q12:** ปรับปรุงจากโปรแกรมนี้ ให้สามารถทำงานนับขึ้นเมื่อป้อนสัญญาณที่ ขา Int0

**Q13:** ปรับปรุงจากโปรแกรมนี้ ให้ใช้ Int0=นับขึ้น และ Int1=นับลง ระหว่าง 0 – F

2. ศึกษาการทำงานของโปรแกรม "Lab81Wrk.Asm" ที่ให้มา ซึ่งเป็นโปรแกรมหลักวนลูป เพื่อนำค่าในตัวแปร Data มาตรวจสอบ

หากบิตสุดท้ายเป็น 0 ➔ จะทำงาน JOB1: ให้ 7_Segment ติดค้างไว้

หากบิตสุดท้ายเป็น 1 ➔ จะทำงาน JOB2: ให้ 7_Segment กระพริบ

และเมื่อมีสัญญาณอินเตอร์รัพท์ที่ขา INT1 จากภายนอกโปรแกรมย่อยบริการอินเตอร์รัพท์จะทำงานดังนี้ กลับค่า Data, หน่วงเวลากำจัด Bounce, Clear ป้องกัน Interrupt ซ้ำ และกลับ Main Program

**Q21:** ปรับปรุงจากโปรแกรมนี้ ให้เป็นแสดงผลที่ E060 แสดงเลข 8 แบบไฟวิ่งวนขวาและแสดงเลข 8 แบบไฟวิ่งวนซ้ายสลับกันเมื่อมีการอินเตอร์รัพท์

3. ศึกษาการทำงานของโปรแกรม "Lab83Tmr.Asm" ที่ให้มาซึ่งเป็นโปรแกรมสร้างความถี่ 1234 Hz โดยการใช้ Timer Interrupts

**Q31:** ปรับปรุงจากโปรแกรมนี้ ให้สร้างความถี่ 1000 + รหัสนักศึกษา 3 ตัวสุดท้ายหน่วยเป็น Hz ด้วยการใช้เทคนิค Timer Interrupts

4. ศึกษาการทำงานของโปรแกรม "Lab84Srl.Asm" ที่ให้มาซึ่งเป็นโปรแกรมรับค่าการกดจากคีย์บอร์ดพีซี ผ่านโปรแกรม HyperTerminal แล้วนำค่าคีย์ที่ได้แสดงที่ 7_Segment Display 6 หลัก ตำแหน่งที่ 2,3 การรับค่าจากคีย์บอร์ดพีซีโดยการใช้เทคนิค Serial Interrupts

**Q41:** ปรับปรุงจากโปรแกรมนี้ ให้แสดงค่าเริ่มต้นเท่ากับ 80H ถ้ามีการกดเครื่องหมาย + จากคีย์บอร์ดพีซีให้ทำการเพิ่มค่า 1 ค่า และถ้ามีการกดเครื่องหมาย - จากคีย์บอร์ดพีซีให้ทำการลดค่า 1 ค่า

**Q42:** ปรับปรุงจากโปรแกรมนี้รับค่าจากคีย์บอร์ดพีซี โดยถ้ากดคีย์ "P" ครั้งที่ 1 ให้ 7_Segment E060H กระพริบตัว A และกดคีย์ "P" อีกครั้ง ให้ 7_Segment E060H ติด A ค้างไว้ ทำซ้ำไปเรื่อยๆ (เหมือนกับให้กระพริบ A แล้วกดคีย์ Pause "P" เพื่อค้าง A ไว้แล้วกด Pause "P" เพื่อทำงานต่อไป)

## 8.5 วิเคราะห์และอภิปรายผลการทดลอง

1. จงอธิบายหลักการให้บริการอินเตอร์รัปต์ของบอร์ด ET-8032
2. จงอธิบายความแตกต่างของอินเตอร์รัปต์ชนิดต่างๆ และให้อธิบาย interrupt priority ด้วย
    - อินเตอร์รัปต์จากภายนอก (external interrupt)
    - อินเตอร์ตัวจับเวลา (timer interrupt)
    - อินเตอร์รัปต์ด้วยการสื่อสารอนุกรม (serial interrupt)
3. จงอธิบายหน้าที่ ความสำคัญและการใช้งานรีจิสเตอร์ IE ที่ใช้ในการทดลอง
4. อินเตอร์รัปต์เวกเตอร์ (interrupt vectors) มีหน้าที่อะไร และสำคัญอย่างไร
5. จงอธิบายการทำงานของโปรแกรมที่ดำเนินการทดลองและตอบคำถามของท้ายการทดลองข้อ 1.Q13
6. จงอธิบายการทำงานของโปรแกรมที่ดำเนินการทดลองและตอบคำถามของท้ายการทดลองข้อ 2.Q21

## 8.6 คำถามท้ายการทดลอง

1. จงเขียนออกแบบการวงจรรับสัญญาณจากเซ็นเซอร์สำหรับตรวจนับจำนวนคนเข้า-ออกห้องสมุด จากนั้นเขียนโปรแกรมเพื่อนับจำนวนคนเข้า-ออกห้องสมุด แสดงผลจำนวนคนเข้าห้องสมุด จำนวนคนออกจากห้องสมุด และจำนวนคนที่เหลือในห้องสมุดที่พอร์ตอนุกรม
2. จงเขียนออกแบบการวงจรรับสัญญาณจากเซ็นเซอร์สำหรับวัดรอบการหมุนของล้อรถ จากนั้นเขียนโปรแกรมเพื่อวัดความเร็วแสดงผลความเร็วที่เป็นรอบต่อนาทีที่พอร์ตอนุกรม

## 8.7 ข้อสอบเก่า (แนวข้อสอบ)

1a. กำหนดให้ ค่าการนับมีค่าเท่ากับ 34H เขียนโปรแกรมหลักนำค่านี้แสดงผลที่ 7_Segment

1b. ให้เขียนโปรแกรมบริการอินเตอร์รัพท์เพิ่มเติมโดยใช้ External Interrupt0 ให้ทำงานดังนี้

1. ลดค่าการนับลง 1 ค่า
2. Delay ประมาณ 750 มิลลิวินาที
3. เคลียร์ค่าบิทรีจิสเตอร์ IE0
4. กลับสู่โปรแกรมหลัก

2a. ให้เขียนโปรแกรมหลักวนลูปไม่รู้จบ ให้ Register B ควบคุมลักษณะการส่งค่าออกที่ Serial Port คือ

- ถ้า RegB.0 = 0 ให้แสดงอักษรภาษาอังกฤษตัวใหญ่ A-Z ขึ้นบรรทัดใหม่ชิดซ้าย แล้ววนกลับมาเริ่มที่ A ใหม่ ให้เว้นระยะระหว่างตัวอักษรประมาณ 0.5 วินาที
- ถ้า RegB.0 = 1 ให้แสดงอักษรภาษาอังกฤษตัวเล็ก a-z ขึ้นบรรทัดใหม่ชิดซ้าย แล้วกลับมาเริ่มที่ a ใหม่ ให้เว้นระยะระหว่างตัวอักษรประมาณ 0.5 วินาที

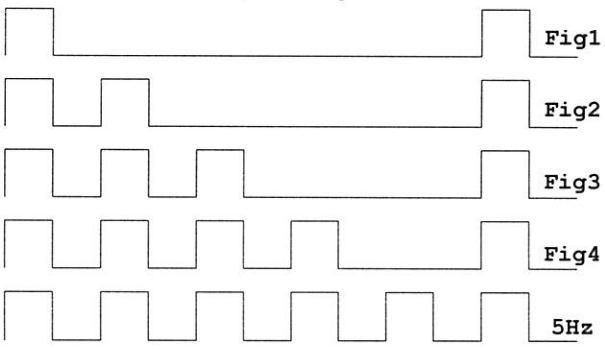
2b. หากเมื่อไรมีสัญญาณอินเตอร์รัพท์ที่ขา INT1 จากภายนอกให้เขียนโปรแกรมย่อยบริการอินเตอร์รัพท์ทำงานดังนี้

1. กลับค่าใน Register B
2. Delay ประมาณ 500 มิลลิวินาที
3. เคลียร์ค่าบิทรีจิสเตอร์ IE1
4. กลับสู่โปรแกรมหลัก

3a. ให้นักศึกษาเขียนโปรแกรมไฟกระพริบโดยใช้ LED P3.5 ให้แสดงรูปแบบล่างสุดความถี่ 5 Hz

3b. ปรับแก้โปรแกรมให้สามารถควบการกระพริบจาก PC ผ่านโปรแกรม Hyper Terminal ดังนี้

- หากกดคีย์ 1 ให้แสดงรูปแบบที่ 1
- หากกดคีย์ 2 ให้แสดงรูปแบบที่ 2
- หากกดคีย์ 3 ให้แสดงรูปแบบที่ 3
- หากกดคีย์ 4 ให้แสดงรูปแบบที่ 4
- หากกดคีย์อื่นๆ นอกจาก 1-4 ให้แสดงรูปแบบเดิม



## 8.8 โปรแกรมทดสอบ

- **Lab81Ex1.Asm**

```
Data_Count      EQU     40H

                ORG     8000H
                JMP     START

                ORG     9F45H
                JMP     EX1_ROUTINE

                ORG     8100H
START:          SETB    IT1             ; Adge Trigger
                MOV     IE,#10000100B
                MOV     Data_Count,#0
Main_Loop:      MOV     A,Data_Count
                CALL    Show_7Segment
                JMP     Main_Loop
;=============================================
EX1_ROUTINE:    PUSH    B
                PUSH    Acc
                INC     Data_Count
                CALL    Debounce
                CALL    Debounce
                CLR     IE1             ; Clear Interrupt External_0
                POP     Acc
                POP     B
                RETI                    ; Return from Interrupt
;=============================================
Debounce:       CLR     A
                MOV     B,A
_Dly00:         DJNZ    Acc,_Dly00
                DJNZ    B,_Dly00
                RET
;=============================================
Show_7Segment:  ANL     A,#0FH          ; Only 0-F
                MOV     DPTR,#CODE_TABLE
                MOVC    A,@A+DPTR
                MOV     DPTR,#0E060H
                MOVX    @DPTR,A
                RET
CODE_TABLE:     DB      3FH,06H,5BH,4FH
                DB      66H,6DH,7DH,07H
                DB      7FH,6FH,77H,7CH
                DB      79H,5EH,79H,71H

                END
```

- **Lab82Wrk.Asm**

```
Data        EQU         20H
            ORG         8000H
            JMP         Start
            ORG         9F45H
            JMP         Ext_Int1
            ORG         8100H
Start:      MOV         IE,#10000100B     ; Enable All and EX1
            SETB        IT1               ; 1=Adge Triger
            MOV         Data,#0           ; Init Value
            MOV         DPTR,#0E060H
Main_Loop:  MOV         A,Data
            JNB         Acc.0,JOB_1
            JB          Acc.0,JOB_2
            JMP         Main_Loop
;-----------------------------------------
Job_1:      MOV         A,#01111111B
            MOVX        @DPTR,A
            JMP         Main_Loop
;-----------------------------------------
Job_2:      MOV         A,#01111111B
            MOVX        @DPTR,A
            CALL        Delay
            MOV         A,#00000000B
            MOVX        @DPTR,A
            CALL        Delay
            JMP         Main_Loop
;=========================================
Delay:      PUSH        B
            PUSH        Acc
            MOV         B,#00H
            MOV         A,#00H
_Delay00:   DJNZ        Acc,_Delay00
            DJNZ        B,_Delay00
            POP         Acc
            POP         B
            RET
;=========================================
Ext_Int1:   PUSH        Acc
            MOV         A,Data
            CPL         A
            MOV         Data,A
            CALL        Delay
            CLR         IE1               ; Clear Interrupt External_1
            POP         Acc
            RETI                          ; Return from Interrupt
            END
```

- **Lab83Tim.Asm**

```
Data            EQU     40H

                ORG     8000H           ; Lab6_Timer.Asm
                JMP     Start

                ORG     9F42H
                JMP     Timer0_Intr

                ORG     8100H
Start:          MOV     TMOD,#00010001B ; Timer0,1 Mode1 16Bit
                SETB    EA
                SETB    ET0
                SETB    TR0

Main_Loop:      MOV     A,Data
                MOV     P1,A            ; Out P1
                MOV     DPTR,#0E000H    ; Out Sound
                MOVX    @DPTR,A
                JMP     Main_Loop

Timer0_Intr:    CLR     TR0             ; 1MC
                PUSH    DPH             ; 2MC
                PUSH    DPL             ; 2MC
                MOV     DPTR,#-374+21   ; 2MC
                MOV     TH0,DPH         ; 2MC
                MOV     TL0,DPL         ; 2MC
                SETB    TR0             ; 1MC
                MOV     A,Data          ; 1MC
                CPL     A               ; 1MC
                MOV     Data,A          ; 1MC
                POP     DPL             ; 2MC
                POP     DPH             ; 2MC
                RETI                    ; 2MC

                END
```

สร้างความถี่ 1234 Hz

Period Time Delay = 100000/1234 = 810.37 MicroSec

Duty 50% Delay Step =810.37/2 =405.185 MicroSec

1 MC = 12/11.0592 MicroSec➔ =405.185*11.0592/12 = 373.4 MC

- **Lab84Srl.Asm**

```
Rx_Data         EQU     40H

                ORG     8000H
                JMP     8100H

                ORG     9F4BH
                JMP     Serial_Intr

                ORG     8100H
                CALL    Delay
                MOV     SCON,#50H           ; Mode1 Serial Port
                MOV     RCAP2H,#0FFH        ; Reload values for desired
                MOV     RCAP2L,#0DCH        ; Baud rate 9600@11.0592MHz
                MOV     T2CON,#34H          ; Turn on Timer 2
                MOV     IE,#10010000B       ; Enable All and Serial
                MOV     A,#0CH              ; Clear Screen
                MOV     SBUF,A              ; Send ClearScreen
                JNB     TI,$
                MOV     Rx_Data,#'A'        ; Init Value
Main_Loop:      MOV     A,Rx_Data
                CALL    SegShow_2HEX
                JMP     Main_Loop

;=== Show 2HEX @Acc to 7_Segment Display ===
SegShow_2HEX:   PUSH    Acc
                SWAP    A
                MOV     B,#2                ; Digit 2
                CALL    SegShow_1HEX
                POP     Acc
                MOV     B,#3                ; Digit 3
SegShow_1HEX:   XCH     A,B
                MOV     DPTR,#0E002H        ; Digit Show
                MOVX    @DPTR,A
                XCH     A,B
                ANL     A,#0FH
                MOV     DPTR,#Code_Segment
                MOVC    A,@A+DPTR
                MOV     DPTR,#0E001H        ; Position Show
                MOVX    @DPTR,A
                CLR     A
                DJNZ    Acc,$
                RET
Code_Segment:   DB      3FH,06H,5BH,4FH
                DB      66H,6DH,7DH,07H
                DB      7FH,6FH,77H,7CH
                DB      79H,5EH,79H,71H
```

```
;=========================================================================
Delay:          CLR     A
                MOV     B,A
_Dly00:         DJNZ    Acc,_Dly00
                DJNZ    B,_Dly00
                RET

;=========================================================================
;  Serial Interrupt Service Program
;=========================================================================
Serial_Intr:    CLR     ES
                JB      TI,_X_Srl_Intr
                MOV     A,SBUF              ; Get Data
                MOV     SBUF,A              ; Echo Data
                MOV     Rx_Data,A
_X_Srl_Intr:    CLR     TI
                CLR     RI
                SETB    ES
                RETI

                END
```

## 8.9 ตารางบันทึกผล

| | | |
|---|---|---|
| 1. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 2. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 3. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 4. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 5. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |
| 6. เขียนโปรแกรมข้อ:<br>เรื่อง: | เวลา: | อาจารย์: |