ภาคผนวก ข
หัวข้อการทดลองในรายวิชาไมโครโพรเซสเซอร์

การทดลองที่ 1

Microcontroller Basics

คำถามก่อนการทดลอง

1) ในการโหลดโปรแกรมนั้น โปรแกรมจะถูกโหลดไปเก็บในหน่วยความจำที่เรียกว่า RAM ตำแหน่งเริ่มต้นของ RAM อยู่ที่ตำแหน่งใด (หมายถึงจะต้องโหลดโปรแกรมไปไว้ในตำแหน่งใด)
2) คำสั่งที่ใช้สั่งให้บอร์ด ANT-32 เตรียมรับโปรแกรมที่จะโหลดจากเครื่องคอมพิวเตอร์พีซีคือคำสั่งอะไร
3) คำสั่งที่ใช้ในการสั่งให้โปรแกรม ProComm Plus ส่งข้อมูลจากเครื่องคอมพิวเตอร์พีซีไปยังบอร์ด ANT-32 คือคำสั่งอะไร
4) คำสั่งที่จะทำให้โปรแกรม Run ตามที่ต้องการคือคำสั่งอะไร

บทนำ

ชิพไมโครคอนโทรลเลอร์ที่ถูกนำมาประยุกต์ใช้งานกับระบบควบคุมอัตโนมัติในระดับ 8 บิตที่โดดเด่นมากตัวหนึ่งคือ ชิพไมโครคอนโทรลเลอร์ตระกูล MCS-51 ของ Intel ซึ่งประกอบไปด้วย CPU เบอร์ต่าง ๆ ได้แก่ 8031, 8051, 8032, 8052, 8751, 8752 และ 8052 AHBASIC

ANT-32 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ถูกนำไปใช้งานในลักษณะ Embedded Controller กล่าวคือ เป็นบอร์ดที่ถูกออกแบบมาเพื่องานควบคุมโดยเฉพาะ โดยถูกติดตั้งอยู่ในเครื่องจักรกล เครื่องใช้ไฟฟ้า รวมทั้งระบบอัตโนมัติต่าง ๆ บอร์ดนี้สามารถใช้กับซีพียูเบอร์ต่าง ๆ ดังที่กล่าวมาแล้วได้ทั้งสิ้น ANT-32 ได้ถูกออกแบบและมีการพัฒนามาอย่างต่อเนื่อง โดยจะประกอบไปด้วยวงจรในส่วนของ Watchdog Timer, Battery Backup และ Power Fail Detector ด้วยชิพ MAX691 และวงจร Real-Time Clock ด้วยชิพ DS1202

การใช้งานบอร์ด ANT-32 ผู้ใช้จำเป็นต้องเขียนโปรแกรมควบคุม เรียกว่า มอนิเตอร์โปรแกรมขึ้นมาโดยเฉพาะ เพื่อทำให้งานที่ต้องการพัฒนาสำเร็จได้ ในขั้นตอนการพัฒนานี่เองที่เป็นจุดเด่นของ ANT-32 โดยมีโปรแกรมให้เลือก 2 ลักษณะด้วยกัน คือ REM31 หรือ Paulmon และ BASIC32 (ในการทดลองนี้จะใช้โปรแกรม Paulmon version 2) หลักการของโปรแกรมทั้งสองคือ ให้ผู้ใช้นำ EPROM ที่บรรจุโปรแกรมนี้ไปติดตั้งลงบนบอร์ด ANT-32 ที่ตำแหน่งหน่วยความจำ U2 (EPROM) แล้วทำการต่อสายสัญญาณพอร์ตอนุกรมระหว่างบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี จากนั้นที่เครื่องคอมพิวเตอร์พีซีให้ใช้โปรแกรมสำหรับการสื่อสารข้อมูลอนุกรม ซึ่งในการทดลองนี้จะใช้โปรแกรม ProComm Plus

REM31 (8031 Remote Monitor) ใช้พัฒนาโปรแกรมด้วยภาษาแอสเซมบลีด้วย REM31 ผู้ใช้จะมีชุด คำสั่งในการพัฒนาโปรแกรมถึง 19 คำสั่ง ลักษณะคำสั่งจะคล้ายคลึงกับคำสั่ง DEBUG ของ DOS ทำให้ผู้ที่คุ้นเคยแล้วจะใช้งานได้ง่ายขึ้น REM31 ใช้กับ CPU ได้ทั้งเบอร์ 8031 และ 8032
(*หมายเหตุ* สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับการใช้งานบอร์ด ANT-32 นั้น สามารถดูได้จากคู่มือ ANT-32)

Paulmon เป็น monitor program ที่แจกให้ใช้งานได้ฟรี การดูการใช้งานอย่างคร่าว ๆ ของโปรแกรมให้กด ? ขณะที่ Run โปรแกรมอยู่ การเริ่มเข้าสู่โปรแกรม ให้กด Enter ขณะที่อยู่ในหน้าต่างของโปรแกรม ProComm Plus โปรแกรมมอนิเตอร์ จะมีส่วนของ Auto Baud Rate Detection โดยอัตโนมัติ
(*อ้างอิง* http://www.pjrc.com/tech/8051/paulmon2.html)

ส่วนการใช้งานโปรแกรม ProComm Plus (PCPlus) นั้นสามารถดูได้จาก Help โดยการกดปุ่ม Alt-Z และเมื่อต้องการออกจากโปรแกรม ทำได้โดยการกดปุ่ม Alt-X

จุดประสงค์

1. ศึกษาและเรียนรู้การใช้งานบอร์ด ANT-32
2. เรียนรู้การใช้งานโปรแกรม monitor ซึ่งในการทดลองนี้จะใช้โปรแกรม monitor ที่ชื่อ Paulmon
3. เรียนรู้การใช้งานโปรแกรม PCPlus
4. เรียนรู้วิธีการโหลดโปรแกรมลงไปบนบอร์ดและทดสอบโปรแกรม

อุปกรณ์

1. บอร์ด ANT-32 พร้อมคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Monitor ชื่อ Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซี 1 เครื่องพร้อมซอฟต์แวร์ PCPlus พร้อมวิธีการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51

วิธีการทดลอง

1. ตรวจสอบอุปกรณ์ว่ามีครบทุกชิ้นหรือไม่ และอยู่ในสภาพพร้อมใช้งานหรือไม่
2. ทำการตั้งค่า Baud Rate ในการรับส่งข้อมูลของโปรแกรม PCPlus ให้มีค่าเท่ากับ 9,600 บิตต่อวินาที เพื่อให้สอดคล้องกับความถี่ในการรับส่งข้อมูลของบอร์ด ANT-32
3. โหลดโปรแกรมที่เตรียมไว้ให้ซึ่งเป็น Hex File จากเครื่องคอมพิวเตอร์พีซีไปไว้ในบอร์ด
4. Run โปรแกรมและสังเกตผลที่ได้พร้อมบันทึกผลการทดลอง
5. ใช้โปรแกรม Monitor เพื่อดูข้อมูลที่เก็บอยู่ใน RAM และทดลองแก้ไขข้อมูลที่เก็บใน RAM พร้อมบันทึกผลการทดลอง เช่นแก้ไขข้อมูลใน RAM ตั้งแต่ตำแหน่ง 9000H ถึงตำแหน่ง 9005H เป็นค่า 55H
6. สรุปผลและวิจารณ์ผลการทดลองพร้อมทั้งข้อเสนอแนะ

การทดลองที่ 2

Microprocessors Arithmetic

คำถามก่อนการทดลอง

1) หน้าที่ของ Assembler คืออะไร
2) หน้าที่ของ Compiler คืออะไร
3) เขียนโปรแกรมง่าย ๆ ในการบวกค่า A3H กับ 49H แล้วนำผลลัพธ์ที่ได้ไปเก็บไว้ในรีจีสเตอร์ A โดยใช้ภาษาแอสเซมบลี
4) บอกหลักการคร่าว ๆ ในการบวกเลข 16 บิต โดยใช้ภาษาแอสเซมบลี

บทนำ

Microprocessors Arithmetic หรืออาจเรียกว่า กลุ่มคำสั่งทางคณิตศาสตร์ไม่ว่าจะเป็นการบวก ลบ คูณ และหาร ในไมโครคอนโทรลเลอร์ MCS-51 การกระทำทางคณิตศาสตร์ต้องกระทำกับรีจีสเตอร์ A (Accumulator) เป็นหลัก และผลลัพธ์ที่ได้จากการคำนวณจะถูกเก็บไว้ใน Accumulator เสมอ

กลุ่มคำสั่งทางคณิตศาสตร์ในไมโครคอนโทรลเลอร์ MCS-51 จัดแบ่งได้เป็น 4 กลุ่ม ดังนี้

1. กลุ่มคำสั่งการบวก สามารถแบ่งออกเป็นอีก 2 ลักษณะคือ คำสั่งการบวกแบบไม่คิดตัวทด และคำสั่งการบวกแบบคิดตัวทด แฟลกที่มีการเปลี่ยนแปลงคือ แฟลกทด (C), แฟลกทดเสริม (AC) และแฟลก-เกิน (OV)

ถ้าหากผลของการบวกมีค่าเกิน 255 หรือ 0FFH นั่นคือ เกิดการทดจากบิต 7 ของ Accumulator แฟลกทดจะเซ็ต (ลอจิก “1”) และถ้าหากเกิดบิตทดจากบิต 3 มายังบิต 4 แฟลกทดเสริมจะเซ็ต ถ้าหากผลของการบวกเกิน 127 หรือเกิดการทดจากบิต 6 มายังบิต 7 แฟลกเกินจะเซ็ต

2. กลุ่มคำสั่งการลบ คำสั่งการลบในไมโครคอนโทรลเลอร์ MCS-51 จะมีเฉพาะการลบแบบคิดตัวยืม (Full Subtractor) แฟลกที่มีการเปลี่ยนแปลงคือ แฟลกทด (C) แฟลกทดเสริม (AC) และแฟลกเกิน (OV)

ถ้าหากผลของการลบแล้วเกิดการยืมจากบิต 7 ของ Accumulator แฟลกทดจะเซ็ต และถ้าหากเกิดการยืมจากบิต 3 มายังบิต 4 แฟลกทดเสริมจะเซ็ต ถ้าหากผลของการลบแล้วเกิดการยืมจากบิต 7 มายังบิต 6 แฟลกเกินจะเซ็ต

3. กลุ่มคำสั่งการคูณและหาร ในไมโครคอนโทรลเลอร์ MCS-51 มีคำสั่งการคูณและหารข้อมูล โดยต้องกระทำผ่าน Accumulator กับรีจีสเตอร์ B เท่านั้น จะเป็นการคูณและหารแบบไม่คิดเครื่องหมายแฟลกที่มีการเปลี่ยนแปลง คือ แฟลกเกิน (OV) ซึ่งจะเปลี่ยนแปลงเมื่อกระทำคำสั่งการคูณเท่านั้น สำหรับแฟลกทดทั้งหมดจะเป็น “0” ตลอดการกระทำคำสั่งการคูณและหาร

เมื่อกระทำคำสั่งการคูณ ผลลัพธ์ที่ได้จะเก็บไว้ใน Accumulator และ B โดย Accumulator ใช้เก็บผลลัพธ์ในไบต์ล่างหรือข้อมูลบิต 7-0 ในขณะที่รีจีสเตอร์ B ใช้เก็บผลลัพธ์ในไบต์บนหรือข้อมูลบิต 15 ถึง 8 ในกรณีที่ผลการคูณมีค่ามากกว่า 255 หรือ FFH แฟลกเกินจะเซ็ต เพื่อแจ้งให้ทราบว่า ผลลัพธ์มีค่ามากกว่า 8 บิต จะต้องไปอ่านผลลัพธ์ที่เหลือจากรีจีสเตอร์ B มารวมกับ Accumulator

ในกรณีที่กระทำคำสั่งการหารต้องกำหนดให้ Accumulator เป็นตัวตั้ง โดยที่ตัวหารคือข้อมูลที่อยู่ในรีจีสเตอร์ B ผลหารจะเก็บไว้ใน Accumulator ถ้าหากหารไม่ลงตัว คือ มีเศษเกิดขึ้น ค่าของเศษนั้นจะถูกเก็บไว้ในรีจีสเตอร์ B

4. กลุ่มคำสั่งเพิ่มและลดค่า เป็นกลุ่มคำสั่งที่ใช้ในการเพิ่ม (Increment) และลด (Decrement) ค่าของข้อมูล ค่าของรีจีสเตอร์ และค่าแอดเดรสของหน่วยความจำ การทำงานของกลุ่มคำสั่งเปรียบเทียบได้กับการบวกหรือลบค่าด้วยข้อมูล 01H เมื่อกระทำคำสั่งในกลุ่มนี้ จะไม่ทำให้สถานะของแฟลกมีการเปลี่ยนแปลง

จุดประสงค์

1. เพื่อศึกษาเครื่องมือที่ใช้ในการแปลงภาษาแอสเซมบลีให้เป็นภาษาเครื่อง
2. เพื่อศึกษาเครื่องมือที่ใช้ในการแปลงภาษาซีให้เป็นภาษาเครื่อง
3. เพื่อศึกษาการบวกลบเลขแบบ 8 บิตของไมโครโพรเซสเซอร์
4. เพื่อศึกษาการบวกลบเลขแบบ 16 บิตของไมโครโพรเซสเซอร์

อุปกรณ์

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ PCPlus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51

วิธีการทดลอง

1. ทำการเขียนโปรแกรมด้วยภาษาแอสเซมบลีในการบวกตัวเลข 8 บิต ค่า C3H โดยให้เก็บที่ 9000H และค่า 45H เก็บที่ 9001H โดยผลลัพธ์เก็บไว้ที่ 9002H พร้อมทั้งทำการทดสอบโปรแกรมโดยใช้โปรแกรม Monitor
2. ทำการเขียนโปรแกรมด้วยภาษาแอสเซมบลีในการบวกตัวเลข 16 บิต ค่า 2451H โดยให้เก็บที่ 9000H กับที่ 9001H และค่า 6A8EH ให้เก็บที่ 9002H กับที่ 9003H โดยผลลัพธ์เก็บไว้ที่ 9004H และที่ 9005H พร้อมทั้งทำการทดสอบโปรแกรม
3. ทำการเขียนโปรแกรมภาษาซีในการบวกตัวเลข 8 บิต พร้อมทั้งทำการทดสอบโปรแกรม โดยใช้โปรแกรม Monitor
4. ทำการเขียนโปรแกรมภาษาซีในการบวกตัวเลข 16 บิต พร้อมทั้งทำการทดสอบโปรแกรม โดยใช้โปรแกรม Monitor

ข้อเสนอแนะ

สำหรับการทดลองในข้อที่ 3 และข้อที่ 4 ในส่วนของการทดสอบโปรแกรม นักศึกษาสามารถทำการตรวจสอบตำแหน่งของตัวแปรที่ใช้ในโปรแกรม จาก Map File ว่าอยู่ที่ตำแหน่งใดในหน่วยความจำ

การทดลองที่ 3

Loop and Conditional Program

คำถามก่อนการทดลอง

1) จงอธิบายความหมายของคำสั่งต่อไปนี้มาโดยสังเขป
   - JC - CJNE
   - JNB - DJNZ
2) เขียนโปรแกรมการบวกเลขจากหน่วยความจำ 9000H ถึง 901FH แล้วเก็บผลลัพธ์ที่ 9020H, 9021H โดยใช้ภาษาแอสเซมบลี

จุดประสงค์

1. ให้นักศึกษาเข้าใจและสามารถเขียนโปรแกรมที่มีเงื่อนไขโดยภาษาแอสเซมบลีและภาษาซีได้
2. ให้นักศึกษาเข้าใจและสามารถเขียนโปรแกรมที่มีการวนรอบโดยภาษาแอสเซมบลีและภาษาซีได้

อุปกรณ์

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ PCPlus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51
6. Flow Chart ของโปรแกรมตัวอย่าง

วิธีการทดลอง

ให้นักศึกษาทำความเข้าใจโปรแกรมเงื่อนไขและวนรอบดังต่อไปนี้

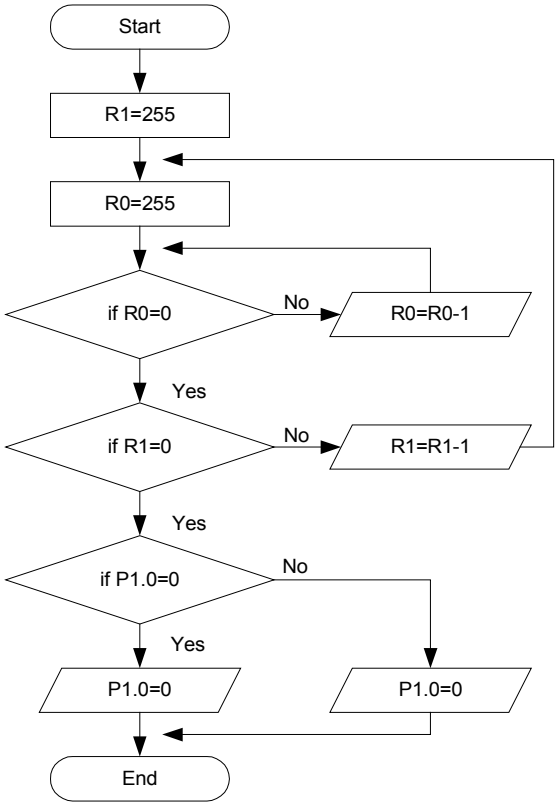
โปรแกรมเงื่อนไขในภาษาแอสเซมบลี

```
          ORG   8000H              ;ตำแหน่งเริ่มต้นที่ 8000H สำหรับบอร์ด ANT-32
MAIN:     MOV   DPTR,#9000H        ;ชี้ไปยังข้อมูลที่ตำแหน่ง 9000H
NOT_EQU:  MOVX  A,@DPTR            ;ย้ายข้อมูลจากตำแหน่งที่ DPTR ชี้อยู่ไปไว้ที่รีจีสเตอร์ A
          INC   DPTR               ;เพิ่มค่าให้ DPTR ไปอีกหนึ่ง
          MOVX  @DPTR,A            ;ย้ายข้อมูลจากตำแหน่งที่ DPTR ชี้อยู่ไปไว้ที่รีจีสเตอร์ A
          CJNE  A,#7FH,NOT_EQU     ;เปรียบเทียบค่าในรีจีสเตอร์ A กับ 7FH หากไม่เท่ากันจะ
                                   ;กระโดดไปยัง NOT_EQU (if A <> 7FH then …)
          JMP   0000H              ;Boot Paulmon2
          END                      ;จบการทำงาน
```

โปรแกรมวนรอบในภาษาแอสเซมบลี

```
          ORG     8000H           ;ตำแหน่งเริ่มต้นที่  8000H สำหรับบอร์ด ANT-32
          CLR     P1.0            ;กำหนดให้ P1.0 มีค่าเป็น 0
MAIN:     MOV     R1,#0FFH        ;กำหนดให้รีจีสเตอร์ R1 มีค่า 255
DELAY:    MOV     R0,#0FFH        ;กำหนดให้รีจีสเตอร์ R1 มีค่า 255
WAIT:     DJNZ    R0,WAIT         ;ทำการลดค่ารีจีสเตอร์ R0 หากไม่เท่ากับ 0 จะกระโดดไป
                                  ;ที่ WAIT คือ กระโดดอยู่กับที่
          DJNZ    R1,DELAY        ;ทำการลดค่ารีจีสเตอร์ R0 หากไม่เท่ากับ 0 จะกระโดดไป
                                  ;ที่ DELAY แล้วจะกำหนดค่ารีจีสเตอร์ R0 ใหม่
          JB      P1.0,CLR_BIT    ;จะใช้ตรวจสอบว่า P1.0 เท่ากับ 1 หรือไม่
                                  ;หากเป็น 1 จะกระโดดไปที่ CLR_BIT
          SETB    P1.0            ;สั่งให้ P1.0 เท่ากับ 1
          JMP     MAIN            ;กระโดดไป MAIN
CLR_BIT:  CLR     P1.0            ;สั่งให้ P1.0 เท่ากับ 0
          JMP     MAIN            ;กระโดดไป MAIN
          END                     ;จบการทำงาน
```

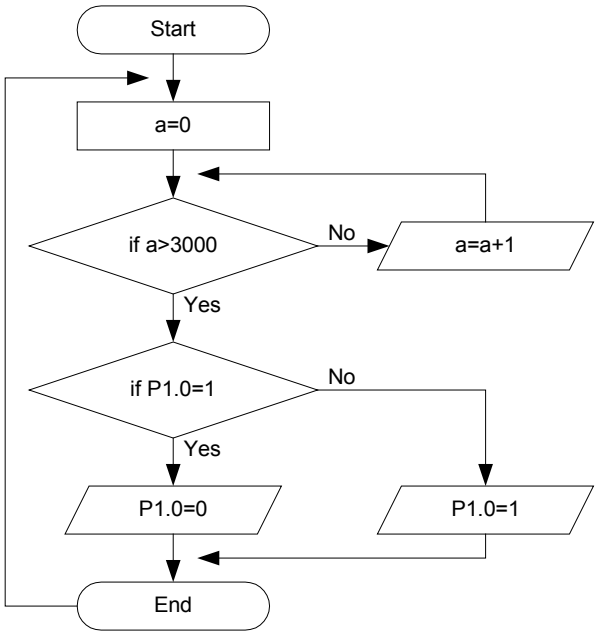
Flow Chart การทำงาน



โปรแกรมวนรอบและเงื่อนไขในภาษาซี

```
#include<8051.h>                    /*เป็นไฟล์ที่เก็บข้อมูลเกี่ยวกับรีจีสเตอร์ของ 8051*/
main(){                             /*เริ่มโปรแกรม*/
      int a;                        /*กำหนดตัวแปร 16 บิตเพื่อใช้วนรอบ */
      P1_BITS.B0=0;                 /*กำหนด P1.0 เท่ากับ 0*/
      for(;;){                      /*กำหนดวงรอบไม่สิ้นสุด*/
            for(a=0;a<3000;a++);    /*กำหนดวงรอบโดยใช้ a เป็นตัววนตั้งค่าไว้ตั้งแต่*/
                                    /* 0 ถึง 3000*/
            if(P1_BITS.B0==0)       /*ตรวจสอบว่า P1.0 เท่ากับ 0 หรือไม่*/
                  P1_BITS.B0=1;     /*กำหนด P1.0 เท่ากับ 1*/
            else                    /*ถ้าไม่เป็นไปตามเงื่อนไขข้างต้น*/
                  P1_BITS.B0=0;     /*กำหนด P1.0 เท่ากับ 0*/
      }
}
```

Flow Chart การทำงาน



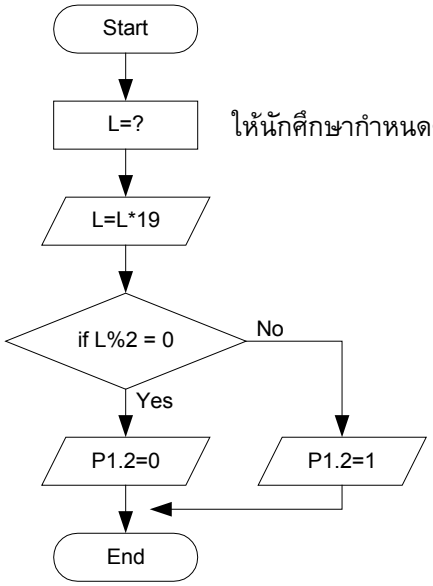
การทดลองที่ 4

Program Control

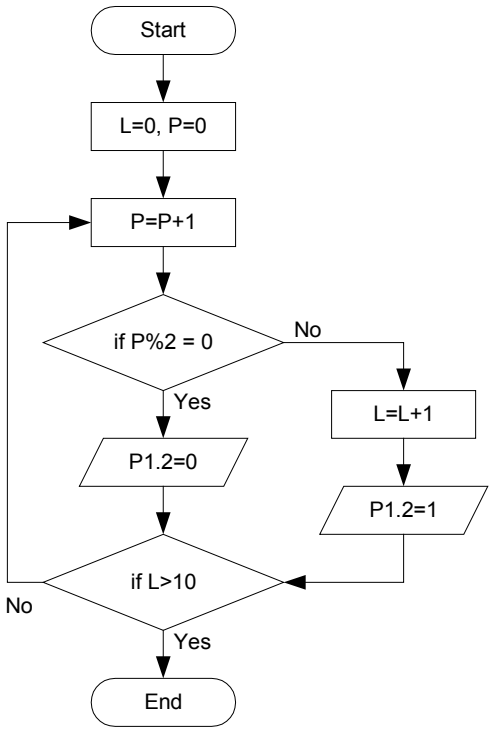
คำถามก่อนการทดลอง

เขียนโปรแกรมภาษาแอสเซมบลีและภาษาซีจาก Flow Chart ที่กำหนดให้ โดยจะนำไปควบคุม Port1 ของไมโครคอนโทรลเลอร์ตระกูล 8051 (เครื่องหมาย % แทนการ Mod)

1)



2)

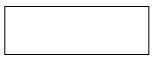


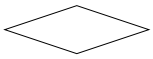
บทนำ

ลักษณะของการทำงานของโปรแกรมทั่วไปจะถูกเขียนขึ้นโดยใช้ Flow Chart เป็นตัวอ้างอิง เพื่อให้สามารถทำความเข้าใจการทำงานของโปรแกรมได้ง่ายขึ้น และ สามารถแก้ไขได้สะดวกหากต้องการจะพัฒนาโปรแกรม และลักษณะของ Flow Chart จะเป็นแผนภาพที่เป็นลำดับจากบนลงล้างโดยจะมีสัญลักษณ์ดังนี้


หมายถึง จุดเริ่มต้นและจุดสิ้นสุด (Start/Stop)


หมายถึง การรับข้อมูลและการแสดงผล (อินพุต/เอาท์พุต)

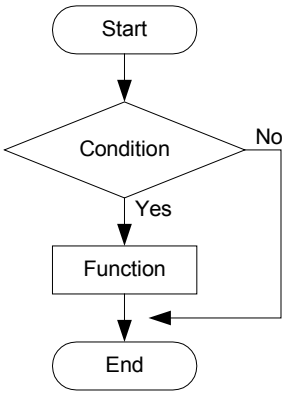

หมายถึง การคำนวณและการประมวลผล (Process)


หมายถึง การตรวจสอบเงื่อนไขและการเปรียบเทียบ (Compare)


หมายถึง จุดต่อ (Connection)


หมายถึง ทิศทางการทำงาน (Direction)

ตัวอย่าง Flow Chart ที่เป็นลักษณะการทำงานของฟังก์ชันเงื่อนไขทั่ว ๆ ไป



จุดประสงค์

1. ศึกษาและสามารถทำความเข้าใจการทำงานของ Flow Chart
2. ศึกษาการเขียนขั้นตอนการทำงานของ Flow Chart ได้
3. ให้นักศึกษาสามารถเขียนโปรแกรมที่ใช้ควบคุมไมโครคอนโทรลเลอร์ในภาษาแอสเซมบลีและภาษาซีจาก Flow Chart ได้

อุปกรณ์

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องพีซีคอมพิวเตอร์
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ PCPlus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51

วิธีการทดลอง

1. เขียนโปรแกรมภาษาแอสเซมบลีและภาษาซีจาก Flow Chart
2. แอสเซมเบลอร์หรือคอมไพล์โปรแกรมที่เขียนขึ้นให้อยู่ในรูปแบบ Intel Hex Format (Hex File) เพื่อที่จะสามารถบันทึกลงในไมโครคอนโทรลเลอร์
3. โหลดโปรแกรมลงในบอร์ด ANT-32
4. Run โปรแกรมที่นักศึกษาเขียนและบันทึกผลการทดลอง

การทดลองที่ 5

System Clocks and Buses

คำถามก่อนการทดลอง

1) สัญญาณใดที่ใช้ในการระบุความแตกต่างระหว่างสัญญาณข้อมูลและสัญญาณแอดเดรสของพอร์ต P0
2) สัญญาณใดที่ใช้บอกถึงการอ่านข้อมูลจากหน่วยความจำ
3) สัญญาณใดที่ใช้บอกถึงการเขียนข้อมูลลงสู่หน่วยความจำ
4) ให้สเกตช์ Timing Diagram ของโปรแกรมต่อไปนี้

```
        MOV         A,#7FH
        MOV         DPTR,#9000H
L1:     MOVX        @DPTR,A
        JMP         L1
```

บทนำ

การทดลองที่ผ่านมาได้แนะนำให้นักศึกษาเรียนรู้สถาปัตยกรรมของบอร์ดไมโครคอนโทรลเลอร์ การคำนวณทางคณิตศาสตร์อย่างง่าย รวมทั้งการทำงานของโปรแกรมจาก Flow Chart เป็นต้น ส่วนที่สำคัญอีกส่วนหนึ่งที่จะนำเสนอในการทดลองนี้คือการศึกษาสัญญาณต่าง ๆ ของไมโครคอนโทรลเลอร์ รวมถึง Timing Diagram ของสัญญาณเหล่านั้น

การทำงานของไมโครคอนโทรลเลอร์หรือไมโครโปรเซสเซอร์โดยทั่วไปจำเป็นต้องพึ่งพาสัญญาณนาฬิกาเป็นตัวกำหนดจังหวะการทำงานภายในทั้งหมด สำหรับไมโครคอนโทรลเลอร์ตระกูลนี้สัญญาณนาฬิกาที่ใช้ในบอร์ดนี้คือ 11.0592 MHz วงจรกำเนิดสัญญาณนาฬิกาภายในชิพ (Internal Clock Generator) จะเป็นตัวกำหนดลำดับของสถานะที่ทำให้เกิดแมชชีนไซเกิลของไมโครคอนโทรลเลอร์

ในหนึ่ง Machine Cycle ประกอบด้วย 6 State คือ State 1 ถึง State 6 แต่ละ State ใช้เวลา 2 Oscillator Period ดังนั้นใน 1 Machine Cycle จะใช้เวลา 12 Oscillator Period นั่นคือหากใช้ความถี่ Oscillator 12 MHz จะได้ว่าใน 1 Machine Cycle ใช้เวลา 1 มิลลิวินาที

จุดประสงค์

1. ศึกษาสัญญาณต่าง ๆ ที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล 8051 อย่างเช่นสัญญาณ ALE, $\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$ เป็นต้น
2. เรียนรู้และสามารถเขียน Timing Diagram ที่สัมพันธ์กับการอ่านและเขียนข้อมูลลงในหน่วยความจำ

อุปกรณ์

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ดกับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ PCPlus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51
6. ออสซิลโลสโคป
7. กระดาษกราฟ เพื่อบันทึกผลการทดลอง 4-5 แผ่น

วิธีการทดลอง

1. ทำการโหลดโปรแกรมต่อไปนี้ลงในหน่วยความจำ

```
      MOV   8000H
      MOV   A,#5AH
      MOV   DPTR,#9000H
LBL1: SETB  P1.0
      MOVX  @DPTR,A
      MOVX  A,@DPTR
      CLR   P1.0
      JMP   LBL1
      END
```

2. จากนั้นทำการ Run โปรแกรมและทำการวัดสัญญาณที่ขา ALE, $\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$ และ P1.0 โดยใช้ออสซิลโลสโคป
3. บันทึกสัญญาณที่ได้ลงในกระดาษกราฟ โดยให้บันทึกค่า Volt/Div และ Time/Div ลงไปด้วย
4. สรุปผลและวิจารณ์ผลการทดลองที่ได้

การทดลองที่ 6

I/O Subsystem

คำถามก่อนการทดลอง

1) เขียนวงจรที่ใช้เพื่อไปขับลำโพง พร้อมอธิบายการทำงานคร่าว ๆ
2) จะเขียนโปรแกรมอย่างไรให้เกิดความถี่แต่ละความถี่
3) ในการสร้างสัญญาณ Square Wave ต้องทำการเซ็ตโหมดของ Timer อย่างไร
4) ถ้าต้องการให้เสียงออกมาเป็น Do ต้องทำการเซ็ตค่าเริ่มต้นของ Timer ให้มีค่าเท่าไร

ตารางแสดงความถี่ของตัวโน้ตแต่ละตัว

Melody	Tone Frequency (Hz)
Do	131
Ra	147
Me	165
Fa	175
So	196
La	220
Ti	247

บทนำ

ถ้าหากคอมพิวเตอร์กำลังทำงานโปรแกรมหลักอยู่ เมื่อมีการ Interrupt เข้ามาคอมพิวเตอร์จะละทิ้งโปรแกรมหลัก แต่จะกระโดดไปทำโปรแกรมตอบสนองการ Interrupt (Interrupt Service Routine) โดยตำแหน่งที่จะกระโดดไปเรียกว่า Interrupt vectors เมื่อทำโปรแกรมตอบสนองการ Interrupt เสร็จแล้วจะกระโดดมาทำงานยังตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการ interrupt จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่า Program Counter (PC) ซึ่งจะชี้ตำแหน่งที่จะอ่านค่าคำสั่งถัดมา ลงหน่วยความจำสแตค ซึ่งอยู่ที่หน่วยความจำที่ถูกชี้โดยรีจีสเตอร์ Stack Pointer (SP) เมื่อโปรแกรมตอบสนองการ Interrupt เสร็จแล้วจะคืนค่าในหน่วยความจำ สแตคให้ PC ตามเดิม ค่า Interrupt Vector ของ MCS-51 แสดงดังตาราง

ตารางแสดง Interrupt Vector ของ Interrupt ต่าง ๆ

Interrupt	Interrupt Vector
System Reset	0000H
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial Port	0023H
Timer 2	002BH

รีจีสเตอร์ที่เกี่ยวข้องกับการทำงานของ Timer/Counter 0 และ 1 เช่น

TCON (Timer/Counter Control Register)

เป็นรีจีสเตอร์ควบคุมการทำงานของ Timer/Counter มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง 88H ในพื้นที่ของรีจีสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
TF1	TR1	TF0	TR0	ID1	IT1	IE0	IT0

- *TF1 (Timer 1 Overflow Flag):* เซ็ตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของรีจีสเตอร์ Timer 1 เกิดนับเกินหรือเกิด Overflow การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นกันโดยบิตนี้จะถูกเคลียร์เมื่อมีการ Interrupt เกิดขึ้น
- *TR1 (Timer 1 Rrun Control Bit):* ใช้ในการเปิดปิดการทำงานของ Timer 1 (Enable/Disable) ทำการเซ็ตหรือเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ Timer 1 ทำงานต้องเซ็ตบิตนี้ให้เป็น “1”
- *TF0 (Timer 0 Overflow Flag):* เซ็ตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของรีจีสเตอร์ Timer 0 เกิดการนับเกินหรือเกิด Overflow การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นกัน โดยบิตนี้จะถูกเคลียร์เมื่อมีการ Interrupt เกิดขึ้น
- *TR0 (Timer 0 Rrun Control Bit):* ใช้ในการเปิดปิดการทำงานของ Timer 0 (Enable/Disable) ทำการเซ็ตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ Timer 0 ทำงานต้องเซ็ตบิตนี้ให้เป็น “1”
- *IE1 (External Interrupt 1 Edge Flag):* บิตนี้จะใช้กระบวนการ Interrupt สามารถเซ็ตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณ Interrupt จากภายนอกที่ขา Input Interrupt 1 (INT1) ได้ และจะถูกเคลียร์เมื่อมีการ Interrupt Service เกิดขึ้น
- *IT1 (Interrupt 1 Type Control Bit):* บิตนี้จะใช้ในกระบวนการ Interrupt โดยใช้ในการเลือกลักษณะของสัญญาณ Interrupt จากภายนอกที่ต้องการตอบสนองสำหรับขา Input Interrupt 1 (INT1) การเซ็ตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์
- *IE0 (External Interrupt 0 Edge Flag):* บิตนี้จะใช้ในกระบวนการ Interrupt สามารถเซ็ตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณ Interrupt จากภายนอกที่ขา Input Interrupt 0 (INT0) ได้ และจะถูกเคลียร์เมื่อมีการ Interrupt Service เกิดขึ้น
- *IT0 (Interrupt 0 Type Control Bit):* บิตนี้จะใช้ในกระบวนการ Interrupt โดยใช้ในการเลือกลักษณะของสัญญาณ Interrupt จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขา Input Interrupt 0 (INT0) การเซ็ตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

  “0” เลือกขอบขาลงของสัญญาณ (Falling Edge)

  “1” เลือกระดับลอจิกต่ำ (Low Level Triggered)

TMOD (Timer/Counter Mode Control Register)

เป็นรีจีสเตอร์เลือกโหมดการทำงานของ Timer/Counter มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง 89H ในพื้นที่ของรีจีสเตอร์ SFR ไม่สามารถเข้าถึงได้ในระดับบิต แบ่งการทำงานออกเป็น 2 ส่วนคือ 4 บิตล่างใช้ในการเลือกโหมดการทำงานของ Timer 0 และ 4 บิตบน ใช้ในการเลือกโหมดการทำงานของ Timer 1 ดังนั้นในการอธิบายการทำงานจะขออธิบายเพียงส่วนเดียวดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
GATE	$\overline{C/T}$	M1	M0	GATE	$\overline{C/T}$	M1	M0
Timer 1				Timer 0			

- *GATE:* ใช้เลือกลักษณะการควบคุมการทำงานของ Timer/Counter
  - “0” Timer/Counter จะทำงานเมื่อบิต TRx ในรีจีสเตอร์ TCON เป็น “1” เรียกการควบคุมแบบนี้ว่า การควบคุมทางซอฟต์แวร์
  - “1” Timer/Counter จะทำงานเมื่อบิต TRx ในรีจีสเตอร์ TCON เป็น “1” และสถานะลอจิกที่ขา Input Interrupt INT0 และ INT1 เป็น “1” เรียกการควบคุมแบบนี้ว่า การควบคุมทางฮาร์ดแวร์
- *C/T (Timer or Counter Selector):* ใช้เลือกลักษณะการทำงานของ Timer/Counter
  - “0” เลือกให้ทำงานเป็น Timer โดยใช้สัญญาณอินพุตจากสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์
  - “1” เลือกให้ทำงานเป็น Counter โดยรับสัญญาณอินพุตจากภายนอกที่เข้ามาทางขา T0 หรือ T1
- *M1 M0 (Mode Selector Bit):* ใช้เลือกโหมดการทำงานของ Timer/Counter
  - “00” เลือกให้ทำงานในโหมด Timer/Counter 13 บิต
  - “01” เลือกให้ทำงานในโหมด Timer/Counter 16 บิต
  - “10” เลือกให้ทำงานในโหมด Timer/Counter ขนาด 8 บิตแบบตั้งค่าอัตโนมัติ
  - “11” สำหรับ Timer 0 เลือกให้ทำงานในโหมด Timer/Counter แยกส่วน โดยแยกออกเป็น Timer/Counter 8 บิต 2 ตัว รีจีสเตอร์ TL0 จะได้รับการควบคุมการเปิดปิดจากบิต TR0 ในรีจีสเตอร์ TCON และรีจีสเตอร์ TH0 ซึ่งเป็น Timer/Counter 8 บิตอีกตัวหนึ่ง จะได้รับการควบคุมจากบิต TR1 ในรีจีสเตอร์ TCON

จุดประสงค์

1. เพื่อศึกษาวิธีที่ทำให้เกิดเสียงที่ลำโพง
2. เพื่อศึกษาวิธีการใช้ Timer เพื่อกำเนิดเสียงที่ความถี่ต่างๆ

อุปกรณ์

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ PCPlus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51
6. วงจรที่ทำหน้าที่ขับลำโพงสำหรับสัญญาณดิจิตอล

วิธีการทดลอง

ทำการเขียนโปรแกรมภาษาแอสเซมบลีหรือภาษาซี โดยใช้ Timer ในการสร้างเสียงดนตรี

<u>คำแนะนำ</u>

ตัวโน้ต (Melody) แต่ละตัวแตกต่างกันที่ความถี่

การทดลองที่ 7

Serial Communication

คำถามก่อนการทดลอง

1) จะต้องกำหนดค่า TH1 เท่าใดบ้าง เมื่อต้องการคำนวณหาค่า Baud Rate ดังต่อไปนี้
   - 9,600 บิตต่อวินาที - 3,600 บิตต่อวินาที
   - 4,800 บิตต่อวินาที - 1,200 บิตต่อวินาที

   พร้อมทั้งแสดงการคำนวณทุกข้อ (ใช้ Timer 1 Mode 2, Crystal 11.0592MHz)
2) ค่าที่จะกำหนดให้ SCON ในการส่งแต่ละ Mode ทั้ง 4 Mode ระบุเป็นเลขฐาน 16 และฐาน 2
3) จากการทดลองที่ 5 I/O Sub Systems จงเขียนโปรแกรมภาษาซี ให้รับค่าจาก PCPlus (กำหนด Baud Rate = 9,600 บิตต่อวินาที) จากการกด Keyboard แล้วทำการสร้างเสียงดนตรีดังนี้
   - Key '1' → Do - Key '5' → So
   - Key '2' → Ra - Key '6' → La
   - Key '3' → Me - Key '7' → Ti
   - Key '4' → Fa

   พร้อมทั้งแสดงค่า Key ที่รับได้บนหน้าจอ

บทนำ

1. PCON : Power Control Register (NOT Bit Addressable)

Bit	7	6	5	4	3	2	1	0
Description	SMOD	-	-	–	GF1	GF0	PD	IDL

PCON.7	SMOD	Double baud rate bit. If SMOD = 1, baud rate is doubled when used in mode 1, 2 or 3.
PCON.6	-	Not used in standard 8051.
PCON.5	-	Not used in standard 8051.
PCON.4	-	Not used in standard 8051.
PCON.3	GF1	General purpose bit.
PCON.2	GF0	General purpose bit.
PCON.1	PD	Power down bit. If set, the oscillator is stopped. This mode can be cancelled by an RESET or Interrupt.
PCON.0	IDL	IDLE bit. If set the activity CPU is stopped. This mode can be cancelled by an RESET or Interrupt.

2. Setting the Serial Port Mode

ใน 8051 จะมีส่วนที่เรียกว่า UART ช่วยในการอ่านและเขียนผ่านพอร์ตอนุกรม แต่จะไม่มีการใช้สัญญาณนาฬิกาเข้ามาช่วย ซึ่งประกอบด้วย Start Bits, Stop Bits และ Parity Bits ซึ่งการที่เราจะกำหนด Baud Rate เราสามารถที่จะกำหนดผ่าน SFR register เพราะ 8051 จะเขียนค่าการรับและส่งลงที่บิตของรีจีสเตอร์ SFR เราจะมาทำความรู้จักรูปแบบในการกำหนดค่าต่าง ๆ ที่เกี่ยวข้องกับการใช้งานพอร์ตอนุกรมผ่านบิตของ SFR “Serial Control” (SCON) โดยมีรายละเอียดดังนี้

Bit	Name	Bit Addres	Explanation of Function
7	SM0	9FH	Serial port mode bit 0.
6	SM1	9EH	Serial port mode bit 1.
5	SM2	9DH	Mutliprocessor Communications Enable (explained later).
4	REN	9CH	Receiver Enable. This bit must be set in order to receive characters.
3	TB8	9BH	Transmit bit 8. The 9th bit to transmit in mode 2 and 3.
2	RB8	9AH	Receive bit 8. The 9th bit received in mode 2 and 3.
1	TI	99H	Transmit Flag. Set when a byte has been completely transmitted.
0	RI	98H	Receive Flag. Set when a byte has been completely received.

การกำหนดพอร์ตอนุกรมแต่ละโหมดมีดังนี้

SM0	SM1	Serial Mode	Explanation	Baud Rate
0	0	0	8-bit Shift Register	Oscillator / 12
0	1	1	8-bit UART	Set by Timer 1 (*)
1	0	2	9-bit UART	Oscillator / 32 (*)
1	1	3	9-bit UART	Set by Timer 1 (*)

หมายเหตุ (*) ถ้าบิต PCON.7 (SMOD) =1 จะทำให้ Baud Rate มีค่าเพิ่มขึ้นเป็น 2 เท่า ใน Mode 1,2,3 จะกล่าวถึงรายละเอียดของ "Serial Control" (SCON) แต่ละบิต

- *SM0 และ SM1:* ทั้ง 4 Mode ใน Mode 0 และ 2 Baud Rate จะขึ้นกับความถี่ Oscillator ที่ใช้ และ ใน Mode 1 และ 3 จะขึ้นกับการ Over Flow ของ Timer 1
- *SM2:* บิตนี้ค่อนข้างจะยุ่งยาก เพราะจะทำหน้าที่เป็น Flag สำหรับ Multiprocessor Communication คือ ใน Mode 2 และ 3 จะมีค่า 1 และ RI บิตจะไม่ทำงานเมื่อรับข้อมูลบิต 9 เป็น 0 ส่วน Mode 1 หาก SM2 เป็น 1 บิต RI จะไม่ทำงานหาก Stop Bit ไม่ผิดพลาด และ Mode 0จะกำหนดเป็น 0
- *REN:* Receiver Enable จะถูกกำหนดไว้ที่ 1 เสมอ มีไว้เพื่อควบคุมการรับข้อมูล
- *TB8:* ใช้ใน Mode 2 และ 3 หากมีการส่งแบบ 9 บิตจะมีค่า 1 แต่ถ้าเป็น 8 บิตจะมีค่า 0
- *RB8:* ใน Mode 2 และ 3 จะเป็นบิตที่ 9 ส่วน Mode 1 ถ้า SM2 =0 จะเป็น Stop Bit และ Mode 0 จะไม่ใช้งาน
- *TI:* Transmit Interrupt จะทำงานที่ Mode 0 ในการส่งออกทางพอร์ตอนุกรมเท่านั้น โดยจะเป็น Flag ก่อนถึง Stop Bit ส่วน Mode อื่น ๆ จะเป็น 0
- *RI:* Receive Interrupt จะทำงานแบบเดียวกับ บิต TI แต่จะไว้สำหรับการอ่านค่าจากพอร์ตอนุกรม

3. Setting the Serial Port Baud Rate

การที่จะตั้งค่า Baud Rate นั้นแยกได้เป็น 2 กลุ่ม คือ

- ***พิจารณาจาก Crystal*** ใน Mode 0 จะหารจากค่า Crystal ที่ใช้ด้วย 12 คือ หากใช้ Crystal 11.059 MHz หารด้วย 12 ได้ 921,583 บิตต่อวินาที แต่ใน Mode 2 จะหารด้วย 64 ซึ่งที่ Crystal 11.059 MHz จะได้เพียง 172,797 บิตต่อวินาที
- ***พิจารณาจาก Over Flow*** ใน Mode 1 และ 3 จะพิจารณาจาก Over Flow ของ Timer 1 ซึ่งจะใช้ Timer Mode 8-Bit Auto-Reload หรือ Timer Mode 2 ตัวอย่างที่จะคำนวณให้เห็น จะกำหนดให้ PCON.7 = “0” เพื่อไม่ให้ได้ baud rate ออกมาสองเท่า โดยสิ่งที่เราใช้กำหนดค่าบิต TH1 ดังนี้

  TH1 = 256 - ((Crystal / 384) / Baud Rate)

  ถ้า PCON.7 = “1” จะได้ Baud Rate เป็น 2 เท่า ซึ่งจะคำนวณได้ดังนี้

  TH1 = 256 - ((Crystal / 192) / Baud Rate)

ตัวอย่างการคำนวณ

ถ้าใช้ Crystal 11.059 Mhz และ ต้องการ Baud Rate ที่ 19,200 บิตต่อวินาที จะคำนวณดังนี้

TH1 = 256 - ((Crystal / 384) / Baud Rate)

TH1 = 256 - ((11059200 / 384) / 19200)

TH1 = 256 - ((28,799) / 19200)

TH1 = 256 - 1.5 = 254.5

จะพบว่าเกิดปัญหาที่จุดทศนิยม เพราะจะกำหนดไม่ได้ในความเป็นจริง ซึ่งจะสามารถกำหนดได้ที่ 2 แต่จะได้ Baud Rate 14,400 บิตต่อวินาที และที่ 255 จะได้ Baud Rate 28,800 บิตต่อวินาที จึงจำเป็นที่จะใช้ PCON.7 (SMOD) =1 จะคำนวณดังนี้

TH1 = 256 - ((Crystal / 192) / Baud Rate)

TH1 = 256 - ((11059000 / 192) / 19200)

TH1 = 256 - ((57699) / 19200)

TH1 = 256 - 3 = 253

ซึ่งจะมีค่าใกล้เคียงกับค่า Baud Rate ที่ต้องการมากกว่า

สามารถที่จะนำขั้นตอนข้างต้นมาเขียนใหม่เป็นลำดับได้ดังนี้

1) กำหนดเลือกพอร์ตอนุกรม Mode 1 หรือ 3
2) กำหนดให้ Timer 1 เป็นแบบ Mode 2 (8-Bit Auto-Reload)
3) กำหนดค่า TH1 = 253 เพื่อให้ได้ค่า Baud Rate 19,200 บิตต่อวินาที
4) กำหนด PCON.7 (SMOD) ให้ได้ Baud Rateเป็น 2 เท่า

4. Writing to the Serial Port

เมื่อเราสามารถที่จะควบคุมให้พอร์ตอนุกรมทำการรับส่งข้อมูลได้ตาม Baud Rate ที่ต้องการแล้ว จะขอกล่าวถึงการเขียนข้อมูลต่าง ๆ ลงในพอร์ตอนุกรมโดยสามารถที่จะเขียนผ่านรีจีสเตอร์ SBUF ที่แอดเดรส 99H จะแสดงการใส่ "A" ไปในพอร์ตอนุกรม

```
MOV SBUF,#'A'
```

แต่ก่อนที่จะทำการส่งจะต้องทำการควบคุมบิตในการส่ง โดยบิต TI จะมีค่าเป็น 1 เมื่อทำการส่งเสร็จสิ้นจึงต้องตรวจสอบว่า การส่งนั้นเสร็จสินตามตัวอย่าง Source Code นี้

```
CLR TI                  ;เพื่อให้แน่ใจ ก่อนส่ง
MOV SBUF,#'A'           ;ส่งตัว 'A' ไปทางพอร์ตอนุกรม
JNB TI,$                ;การวนอยู่กลับที่ จนกว่า TI เป็น 1
```

เครื่องหมาย $ หมายถึง "Jump if the TI bit is not set to $" คือ รอจนกว่าตัวแปรหรือค่าในรีจีสเตอร์ที่กำหนดจะเป็น 1

5. Reading the Serial Port

การที่จะรับก็คล้ายคลึงกันโดยจะใช้รีจีสเตอร์ SBUF (99H) แต่จะตรวจสอบการรับจากบิต RI ใน SCON และสามารถดูตัวอย่างการรับได้ ซึ่งจะทำการรับไปใส่ใน Accumulator

```
JNB RI,$                ;รอข้อมูลที่เข้ามาจนครบ
MOV A,SBUF              ;อ่านข้อมูลที่เข้ามาใส่ใน Accumulator
```

ตัวอย่างโปรแกรมการใช้บอร์ด ANT-32 ติดต่อกับกับเครื่อง PC ผ่านทางพอร์ตอนุกรม

```
#include "8051.h"
#include "stdio.h"

void main(void){
      unsigned char a;

      SCON = 0x50;    /* SCON: mode 1, 8-bit UART, enable rcvr */
      TMOD |= 0x20;   /* TMOD: timer 1, mode 2, 8-bit reload */
      TH1 = 0xf3;     /* TH1: reload value for 2500 baud */
      TR1 = 1;        /* TR1: timer 1 run */
      TI = 1;         /* TI: set TI to send first char of UART */
      while(1){
           if(kbhit()){
              a = getch();
              putch(a);
           }
      }
}
```

จุดประสงค์

1. เพื่อศึกษาการใช้บอร์ด ANT-32 ติดต่อกับเครื่องคอมพิวเตอร์พีซีผ่านทางพอร์ตอนุกรม
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมการใช้บอร์ด ANT-32 ติดต่อกับเครื่องคอมพิวเตอร์พีซีผ่านทางพอร์ตอนุกรม

อุปกรณ์

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ PCPlus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51

วิธีการทดลอง

1. ให้นักศึกษาทำความเข้าใจโปรแกรมการใช้บอร์ด ANT-32 ติดต่อกับกับเครื่องคอมพิวเตอร์พีซีผ่านทางพอร์ตอนุกรมจากตัวอย่าง
2. ให้นักศึกษาเขียนโปรแกรมรับค่าจากคีย์บอร์ดเพื่อทำให้เกิดเสียงตัวโน้ตเสียงต่าง ๆ

การทดลองที่ 8

Motor Control

คำถามก่อนการทดลอง

1) Bit Pattern ที่จะสั่งให้ Stepping Motor หมุนไปทางซ้าย(ทวนเข็มนาฬิกา)
2) Bit Pattern ที่จะสั่งให้ Stepping Motor หมุนไปทางขวา(ตามเข็มนาฬิกา)
3) หลักการที่จะใช้ปรับความเร็วของ Stepping Motor
4) สัญญาณจากพอร์ต I/O (ชิพ 8255) จะไปขับให้ขดลวดหมุนได้อย่างไร (ต้องใช้อุปกรณ์ใดบ้าง และต้องผ่านอุปกรณ์ใดบ้าง)

บทนำ

ในการทดลองนี้จะใช้ไมโครคอนโทรลเลอร์สำหรับควบคุมการหมุนและความเร็วของมอเตอร์ มอเตอร์ที่ใช้ในการทดลองนี้คือ Stepping Motor โดยจะใช้บอร์ด EX-STEPM เป็นบอร์ดขับ Stepping Motor รายละเอียดของบอร์ดนี้แนบมาพร้อมกับเอกสารการทดลองนี้ Stepping Motor เป็นมอเตอร์ที่ไม่มี Commutators การควบคุมการทำ Commutation สามารถทำได้โดยใช้คอนโทรลเลอร์ควบคุมจากภายนอก

Stepping Motor สามารถนำมาใช้งานกับระบบควบคุมแบบ Open-Loop อย่างง่ายได้ เพราะ Stepping Motor มักจะถูกนำมาใช้งานกับระบบที่ทำงานที่ความเร่งต่ำและมีโหลดค่อนข้างคงที่

ตัวอย่างของ Stepping Motor ที่ถูกนำมาใช้งานใน Floppy Disk Drive ของเครื่องคอมพิวเตอร์พีซี เช่น

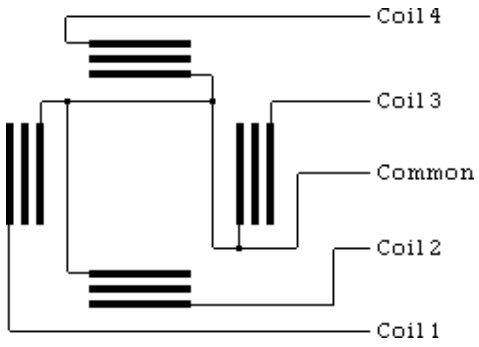
- Tandon themselves and Japan Servo Motors - Type KP4M4-001
- Minebea Co., Ltd - Type 17PS-C007-04
- Shinano Kenshi Co., Ltd - Type STH-42G100

ซึ่งส่วนใหญ่จะมีคุณสมบัติเหมือนกัน ได้แก่ +12 VDC, 4-Phase, Unipolar, 3.6° Per Step โดยปกติแล้วมอเตอร์จะมีการระบุอัตราการเปลี่ยน Step ต่อวินาทีสูงสุดที่มอเตอร์สามารถทำได้มาให้ด้วย

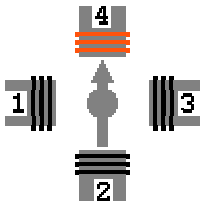
Stepping Motor จะมีขดลวดมากกว่า 2 ขดอยู่ภายใน อาจเป็น 4, 5, 6 หรือมากกว่านั้น Stepping Motor ที่มีสายไฟ 5 เส้นส่วนใหญ่จะเป็นมอเตอร์แบบ Unipolar แบบ 4 เฟส ส่วน Stepping Motor ที่มีสายไฟ 6 เส้นจะเป็นมอเตอร์แบบ Unipolar แบบ 4 เฟสเช่นกัน แต่สายไฟ 2 เส้นในจำนวนนั้นเป็นสายไฟ Common Power ส่วนมอเตอร์ที่มีสายไฟ 4 เส้นส่วนใหญ่จะเป็นมอเตอร์แบบ Bipolar

ส่วนวิธีการจำแนกสายไฟแต่ละเส้นสามารถทำได้ดังนี้

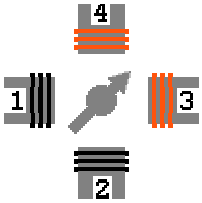
1) แยกสาย Common Power ออกมาโดยการใช้โอห์มมิเตอร์วัดความต้านทานระหว่างคู่สายจะพบว่าสาย Common Power จะเป็นเส้นเดียวที่มีความต้านทานเป็นครึ่งเดียวระหว่างสาย Common Power กับสายเส้นอื่น ๆ



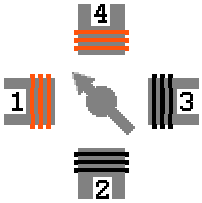
2) การระบุลำดับของขดลวด โดยป้อนแรงดันเข้าที่เส้น Common Power จากนั้นเลือกสายไฟที่เหลือมา 1 เส้นมาต่อกับกราวน์ จากนั้นก็เลือกสายไฟเส้นที่เหลือที่ละเส้นมาต่อกับกราวน์แล้วสังเกตผล เช่นเลือกสายไฟเส้นหนึ่งมาต่อกับกราวน์และกำหนดเป็นขดลวดที่ 4



3) จากนั้นต่อกราวน์สายไฟที่เหลืออีก 3 เส้นทีละเส้น โดยที่การต่อกราวน์กับสายไฟเส้นหนึ่งควรจะทำให้โรเตอร์เกิดการหมุนตามเข็มเล็กน้อยและกำหนดให้เป็นขดลวดที่ 3



4) การกราวน์สายไฟอีกเส้นควรจะทำให้โรเตอร์เกิดการหมุนในทิศทวนเข็มนาฬิกาเล็กน้อย กำหนดให้เป็นขดลวดที่ 1



5) การต่อกราวน์กับสายไฟอีกเส้นที่เหลือควรจะไม่ทำให้เกิดการเปลี่ยนแปลงใด ๆ ขึ้นมา

6) การกำหนดหมายเลขให้กับขดลวดเป็นการกำหนดขึ้นเอง ส่วนสำคัญอยู่ที่การเรียงลำดับของหมายเลขดังกล่าว

หลักการของ Stepping motor มีดังนี้คือ จากรูป Fig 2.2 จะพบว่าโรเตอร์เป็นแท่งแม่เหล็กที่หมุนรอบตัวเอง มีลูปของขดลวดอยู่ 2 ลูป แต่ละลูปจะทำให้เกิดสนามแม่เหล็กไฟฟ้าของตัวเองและแต่ละปลายจะมีขั้วที่แตกต่างกัน

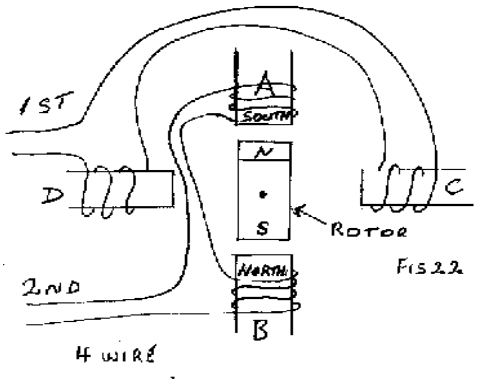


Fig 2.2

ถ้าหากเราป้อนแรงดันให้กับขดลวดที่ 2 ทำให้ขั้ว A กลายเป็นขั้วใต้ และขั้ว B เป็นขั้วเหนือซึ่งขั้วที่จะเกิดขึ้นนั้นขึ้นอยู่กับวิธีการพันขดลวด โรเตอร์จะวางตัวดังรูปตราบเท่าที่ยังป้อนแรงดันให้กับขดลวด การจะทำให้โรเตอร์หมุนคือเปลี่ยนเป็นป้อนแรงดันให้กับขดลวดที่ 1 แทนจะทำให้ขั้ว A และขั้ว B ไม่มีการดึงดูดของสนามแม่เหล็กอีก แต่ขั้ว C และ D จะเกิดการดึงดูดของสนามแม่เหล็กแทน ดังรูป Fig 2.3 ซึ่งขั้ว C จะกลายเป็นขั้วใต้ส่วนขั้ว D กลายเป็นขั้วเหนือ นั่นคือแท่งแม่เหล็กของโรเตอร์จะเปลี่ยนตำแหน่งและหมุนไปในทิศตามเข็มนาฬิกาเป็นมุม 90 องศา

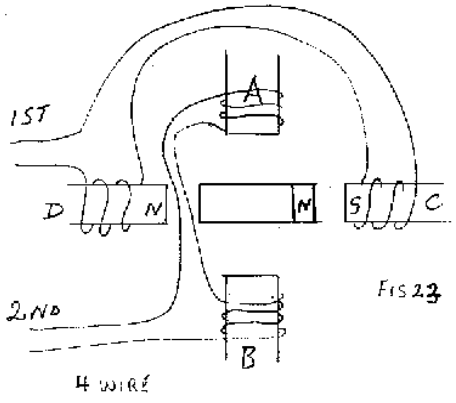


Fig 2.3

ในการจะทำให้โรเตอร์หมุนในทิศตามเข็มนาฬิกาต่อไปนั้นคือการสลับไปป้อนแรงดันให้ขดลวดที่ 2 แทนแต่คราวนี้กลับทิศของแรงดันเพื่อทำให้ขั้ว A กลายเป็นขั้วเหนือและขั้ว B กลายเป็นขั้วใต้ซึ่งจะทำให้ขั้วแม่เหล็กหมุนไปอีก 90 องศา ดังแสดงในรูป Fig 2.4

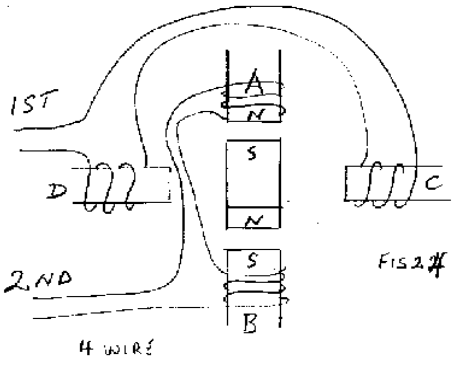


Fig 2.4

จากนั้นสลับไปป้อนแรงดันให้กับขดลวดที่ 1 อีกครั้งโดยสลับขั้วแรงดันดังรูป Fig 2.5

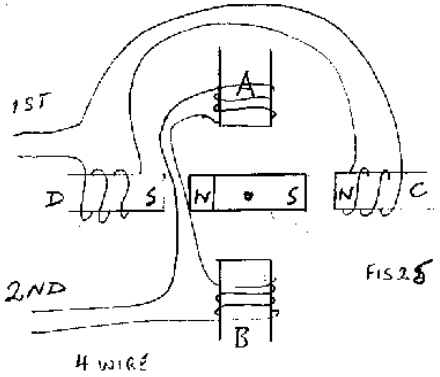


Fig 2.5

ซึ่งในขณะนี้ขั้ว C กลายเป็นขั้วเหนือและขั้ว D กลายเป็นขั้วใต้ การทำให้โรเตอร์หมุนต่อไปเพื่อกลับมายังตำแหน่งเริ่มต้นนั้นต้องสลับแรงดันไปป้อนให้กับขดลวดที่ 2 ในทิศทางที่ทำให้ขั้ว A กลายเป็นขั้วใต้และขั้ว B กลายเป็นขั้วเหนือ การทำเช่นนี้ซ้ำ ๆ กันไปจะทำให้ Stepping Motor หมุนในทิศตามเข็มนาฬิกา

หลักการทำงานของ Stepping Motor ที่กล่าวไปนั้นเป็นแบบ Full Step มีการหมุน 4 Step ต่อรอบ

8255 Programmable Peripheral Interface (PPI) เป็นชิพพอร์ตแบบขนานที่เป็นที่นิยมใช้งานกันมากมาย สำหรับ ANT-32 ใช้พอร์ต 8255 จำนวน 2 ตัวทำหน้าที่เป็นพอร์ตอินพุต/เอาท์พุตถึง 24 x 2 = 48 บิต โดยแบ่งเป็น User Port 1 และ 2 มีตำแหน่งแอดเดรสดังนี้

User Port 1 (U10) แอดเดรส F800H + 8255 Offset Addr = Actual Addr

Port A	ตำแหน่งแอดเดรส F800H + 00H = F800H
Port B	ตำแหน่งแอดเดรส F800H + 01H = F801H
Port C	ตำแหน่งแอดเดรส F800H + 02H = F802H
Mode Port	ตำแหน่งแอดเดรส F800H + 03H = F803H

User Port 1 (U11) แอดเดรส FC00H + 8255 Offset Addr = Actual Addr

Port A	ตำแหน่งแอดเดรส FC00H + 00H = FC00H
Port B	ตำแหน่งแอดเดรส FC00H + 01H = FC01H
Port C	ตำแหน่งแอดเดรส FC00H + 02H = FC02H
Mode Port	ตำแหน่งแอดเดรส FC00H + 03H = FC03H

ก่อนที่จะใช้งานพอร์ต I/O ผู้ใช้ต้องทำการกำหนดโหมดการทำงาน(configuration) ของพอร์ต A, B และ C ให้เป็นอินพุตหรือเอาท์พุต โดยทำการเขียนค่า Control Code ไปที่ Mode Port นี้สามารถเขียนได้เท่านั้นไม่สามารถอ่านได้

จุดประสงค์

1. เรียนรู้วิธีการควบคุม Stepping Motor
2. เรียนรู้วิธีการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051 กับ Stepping Motor

อุปกรณ์การทดลอง

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ ProComm Plus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51
6. Stepping Motor และบอร์ด Stepping Motor Controller
7. สายแพ

วิธีการทดลอง

1. ต่อวงจร Stepping Motor เข้ากับบอร์ดไมโครคอนโทรลเลอร์และบอร์ดควบคุม
2. ทดสอบ Run โปรแกรมแอสเซมบลีควบคุมมอเตอร์ตามที่แนบมาข้างล่างนี้ Run โปรแกรมและบันทึกผล
3. เขียนโปรแกรมด้วยภาษาซีเพื่อควบคุมให้ Stepping Motor หมุนในทิศทวนเข็มนาฬิกา และทิศตามเข็มนาฬิกา
4. เขียนโปรแกรมด้วยภาษาซีเพื่อให้ปรับความเร็วของ Stepping Motor

โปรแกรมแอสเซมบลีควบคุม Stepping Motor

```
      ORG   8000H
; ********* VARIABLE SET ***********
; *********** USER PORT1 ***********
UP1A  EQU   0F800H            ;PORT A
UP1P  EQU   0F803H            ;MODE PORT
; ***********  TTLIO MAIN **********
TTLIO:      MOV   A,#80H      ;SET CONTROL CODE
      MOV   DPTR,#UP1P        ;USER PORT1
      MOVX  @DPTR,A           ;PA, PB, PC=OUTPUT
      MOV   B,#8              ;LOOP=8
TTLIO1: MOV R2,#HIGH 2000     ;R2R3 DELAY
      MOV   R3,#LOW 2000
      MOV   R6,#01H           ;DEFAULT BIT PATTERN1
TTLIO2:     MOV   A,R6
      MOV   DPTR,#UP1A
      MOVX  @DPTR,A
      RL    A                 ;ROTATE LEFT
      MOV   R6,A
      LCALL DELAY             ;DELAY
      DJNZ  B,TTLIO2
      SJMP  TTLIO1            ;LOOP AGAIN
```

```
; ***************** DELAY SUB ******************
DELAY:      MOV   A,R2
      MOV   R0,A
      MOV   A,R3
      MOV   R1,A
DELAY1:XCH  A,R1              ;R0R1 = R0R1 – 1
      JNZ   DELAY2
      DEC   R0
DELAY2:DEC  A
      XCH   A,R1
      MOV   A,R0
      ORL   A,R1
      JNZ   DELAY1
      RET
      END
```

เอกสารอ้างอิง

1. คู่มือการใช้งานบอร์ด ANT-32 version 3.0 EMBEDDED CONTROL BOARD ของบริษัทศิลารีเสิร์ช จำกัด
2. Control of Stepping Motors ของ Douglas W. Jones แห่ง University of Iowa Department of Computer Science
3. Basic Stepper Motor Concept จาก http://home.mira.net/~tonymerc/steptheo/steptheo.htm
4. Other Step Motor จาก http://www.doc.ic.ac.uk/~ih/doc/stepper

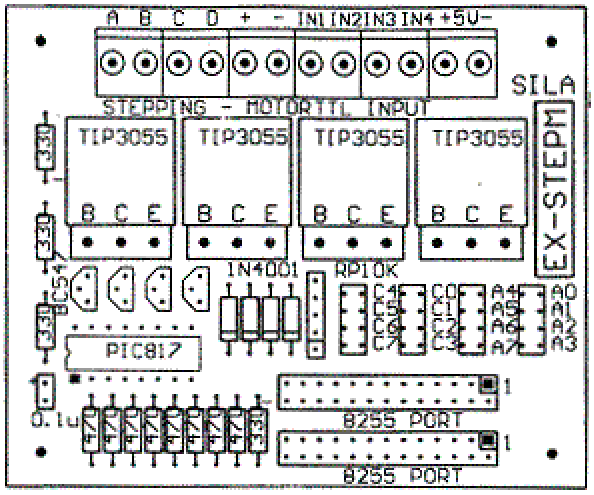
เอกสารของบอร์ดขับ Stepping Motor EX-STEPM

SILA EX-STEPM

4 Phase Step-Motor Driver Board

EX-STEPM เป็นบอร์ดขับสเต็ปเปอร์มอเตอร์สามารถใช้งานกับมอเตอร์ที่ต้องการกระแสสูงสุดถึง 5แอมป์ ภายในบอร์ดมีขั้วต่อสำหรับขับสเต็ปเปอร์มอเตอร์ทั้ง 4 จุด< 4 PHASE >และขั้วไฟบวกลบของสเต็ปเปอร์มอเตอร์ นอกจากนี้ยังมีขั้วอินพุทอีก 4 จุดพร้อมทั้งไฟบวกลบของชุดอินพุท สำหรับชุดจ่ายไฟของสเต็ปเปอร์มอเตอร์กับชุดจ่ายไฟของสวิทช์อินพุทจะแยกอิสระจากกัน จึงช่วยตัดปัญหาในเรื่องสัญญาณรบกวนและปัญหาอื่น ๆที่จะเข้ามารบกวนระบบควบคุมของบอร์ดไมโคร

การเลือกพอร์ทที่จะใช้งานของบอร์ด EX-STEPM สามารถเลือกได้โดยการปรับ JUMPER SELECT PORT ตามรูปแสดงตำแหน่งต่าง ๆ บนบอร์ด



สเต็ปเปอร์มอเตอร์ โดยทั่วไปมีอยู่ 3 ชนิด คือ

- VARIABLE RELUCTANCE < VR >
- PERMANENT MAGNET < PM >
- HYBRID STEPPER MOTOR < HSM >

VARIABLE RELUCTANCE < VR > จะเป็นสเต็ปเปอร์มอเตอร์ที่ถูกกล่าวถึงและนำไปใช้งานมากที่สุด โดยเฉพาะแบบ 4 เฟส โรเตอร์และสเตเตอร์จะทำมาจากเหล็กผสมซิลิคอน เพลาของมอเตอร์จะหมุนไปเป็นค่ามุมคงที่หรือเรียกว่า มุมสเต็ป (θ_s) มักจะมีมุมที่ต่างกัน 0.72 , 0.9 , 1.8, 2.0, 3.6, 7.5, 15 สำหรับสเต็ปเปอร์มอเตอร์ชนิดนี้ส่วนใหญ่จะใช้วงจรขับแบบ UNIPOLAR ซึ่งลักษณะการทำงานของวงจร จะต้องป้อนค่า VOLTAGE เข้าไปในแต่ละขั้วตามลักษณะของ PULSE ที่ใช้ขับชุด DRIVER ค่า VOLTAGE ที่จ่ายให้ มอเตอร์จะใช้อ้างอิงกับกราวด์เสมอ การหาจำนวน STEP ต่อการหมุน 1 รอบของเพลา

$$S = 360^{\circ}/\theta_s \qquad \text{หน่วยเป็น Step}$$

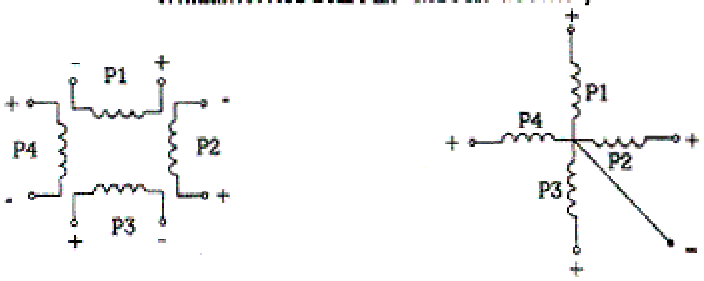
โดยที่ $\theta_s = mN_r$: m = จำนวนเฟส

: N_r = จำนวนฟันของโรเตอร์

PERMANENT MAGNET < PM > จะเป็นสเต็ปเปอร์มอเตอร์แบบที่ตัวโรเตอร์ทำด้วยแม่เหล็กถาวร สเต็ปเปอร์มอเตอร์แบบนี้ จะใช้ลักษณะการหมุนแบบ BIPOLAR ซึ่งลักษณะการทำงานจะต้องจ่ายไฟบวกสลับเข้าแต่ละเฟสของมอเตอร์โดยตรง

HYBRID STEPPER MOTOR < HSM > จะเป็นแบบผสมระหว่าง < PM > และ < VR >จำนวนซี่ฟันของโรเตอร์และสเตเตอร์จะไม่เท่ากัน สเต็ปเปอร์มอเตอร์แบบนี้จะมีลักษณะการทำงานที่ซับซ้อน จึงเป็นแบบที่ไม่ค่อยนิยมใช้งานเท่าใดนัก

ภาพแสดงวงจรขับ STEPPER MOTOR แบบต่าง ๆ



BIPOLAR UNIPOLAR

โดยทั่วไปวงจรที่ใช้ขับสเต็ปเปอร์มอเตอร์จะนิยมใช้ทั้ง 2 แบบนี้ คือ แบบ BIPOLAR หรือ CHOPPERจะใช้กับสเต็ปเปอร์มอเตอร์ที่เป็นชนิดPM และแบบUNIPOLAR จะใช้กับชนิด VR ส่วนการขับของสเต็ปเปอร์มอเตอร์ชนิด HSM จะมีวงจรการขับที่แตกต่างออกไป เนื่องมาจากคุณสมบัติของมอเตอร์ชนิดนี้สามารถทำงานตามกระแสได้ทั้งสองทิศทาง(สลับขั้วไฟบวก-ลบที่จ่ายให้กับมอเตอร์) ส่วนบอร์ด EX-STEPM จะมีวงจรขับเป็นแบบ UNIPOLAR

วิธีการกระตุ้นเฟส

SINGLE PHASE EXITATION เป็นการกระตุ้นแบบเฟสเดี่ยว ตามจังหวะของสัญญาณพัลส์ที่ป้อนเข้าชุดขับสเต็ปเปอร์มอเตอร์ < EX-STEPM BOARD >

PHASE	PULSE							
A	1	0	0	0	1	0	0	0
B	0	1	0	0	0	1	0	0
C	0	0	1	0	0	0	1	0
D	0	0	0	1	0	0	0	1

TWO PHASE EXITATION การกระตุ้นเป็นแบบทีละสองเฟสคู่พร้อมกัน

PHASE	PULSE							
A	1	0	0	1	1	0	0	1
B	1	1	0	0	1	1	0	0
C	0	1	1	0	0	1	1	0
D	0	0	1	1	0	0	1	1

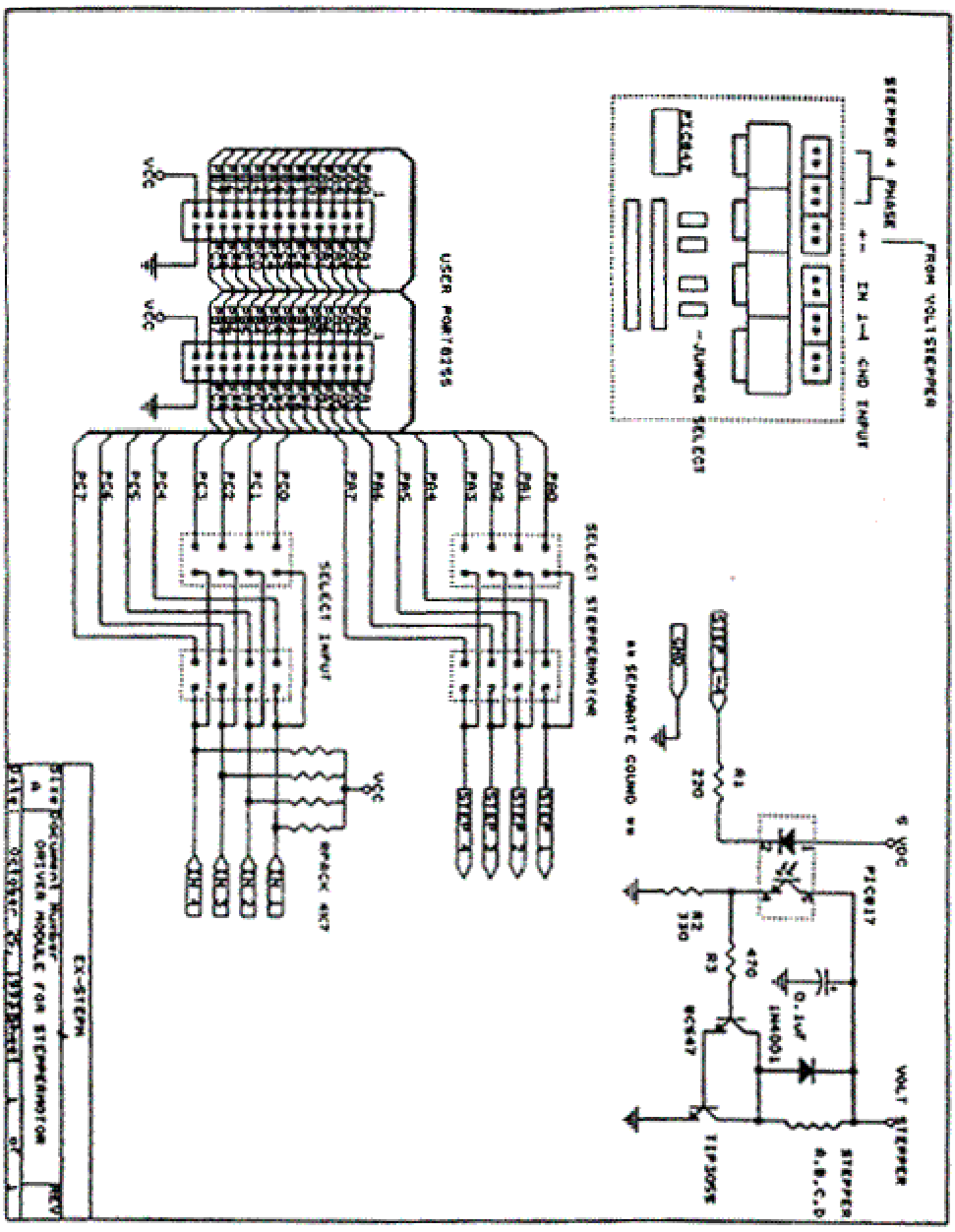
HALF STEP EXITATION แบบกึ่งสเต็ป จะเป็นการรวมรูปแบบการหมุนของทั้งสองแบบไว้ใน แบบเดียวกัน

PHASE	PULSE							
A	1	1	0	0	0	0	0	1
B	0	1	1	1	0	0	0	0
C	0	0	0	1	1	1	0	0
D	0	0	0	0	0	1	1	1

วิธีการกระตุ้นเฟสทั้ง3แบบนี้จะมีคุณสมบัติที่แตกต่างกัน ดังนั้นการเลือกวิธีการกระตุ้นจึงจำเป็นที่จะต้องพิจารณาเพื่อให้เหมาะสมกับงานนั้น ๆ การกระตุ้นแบบ SINGLE PHASE จะเป็นแบบที่มีความเที่ยงตรงของตำแหน่งงานน้อยและจะมีแรงบิด (TORQUE) น้อย ส่วนการกระตุ้นแบบ TWO PHASE มีความเที่ยงตรงของตำแหน่งน้อยแต่มีแรงบิดสูงกว่า และการกระตุ้นแบบ HALF STEP เป็นแบบที่มีความเที่ยงตรงของตำแหน่งมากกว่าและจะมีแรงบิดสูงมากเช่นกัน

โปรแกรมตัวอย่าง XEXSTEPM.ASM

```
;FILENAME       XEXSTEPM.ASM
;DESCRIPTION    DEMO PROGRAM
;HARDWARE       JAZZ-31 + EX-STEPM BOARD

 USER   EQU     0FC00H
 UBEEP  EQU     00A5H       ;UBEEP SUB

; ############ COMMENT ############
;THIS EXAMPLE PROGRAM ROTATE TYPE SINGLE-STEP
;DEFAULT SETTING POINT :-
;PORT SELECT
;       INPUT  = PC0 - 3
;       OUTPUT = PA0 - 3

        ORG     8000H

        MOV     DPTR,#USER+3
        MOV     A,#89H    ;A=I B=I C=I
        MOVX    @DPTR,A

MAIN:   MOV     DPTR,#USER+2
        LCALL   KEYP
        CJNE    A,#0EH,$+6
        LCALL   DRVL
        CJNE    A,#0DH,$+6
        LCALL   DRVR
        CJNE    A,#0BH,$+6
        LCALL   DRVL
        CJNE    A,#07H,$+6
        LCALL   DRVR
        LJMP    MAIN

DRVL:   LCALL   UBEEP
DRVL1:  MOV     DPTR,#USER
        MOV     R0,#4
        MOV     R1,#1
DRVL2:  MOV     A,R1
        MOVX    @DPTR,A
        RL      A
        MOV     R1,A
        LCALL   DELAY
        DJNZ    R0,DRVL2
        MOV     DPTR,#USER+2
        MOVX    A,@DPTR
        ANL     A,#0FH
        CJNE    A,#0FH,DRVL3
        LJMP    DRVL1
DRVL3:  RET
DRVR:   LCALL   UBEEP
DRVR1:  MOV     DPTR,#USER
        MOV     R0,#4
        MOV     R1,#8
DRVR2:  MOV     A,R1
        MOVX    @DPTR,A
        RR      A
        MOV     R1,A
        LCALL   DELAY
        DJNZ    R0,DRVR2
        MOV     DPTR,#USER+2
        MOVX    A,@DPTR
        ANL     A,#0FH
        CJNE    A,#0FH,DRVR3
        LJMP    DRVR1
DRVR3:  RET

KEYP:   MOVX    A,@DPTR
        ANL     A,#0FH
        CJNE    A,#0FH,KEYP1
        SJMP    KEYP

KEYP1:  MOV     R1,#0
        DJNZ    R1,$
        MOVX    A,@DPTR
        ANL     A,#0FH
        CJNE    A,#0FH,KEYP2
        SJMP    KEYP
KEYP2:  RET

; ############ DELAY SUB ############
DELAY:  MOV     R7,#0
        MOV     R6,#0
        DJNZ    R6,$
        DJNZ    R7,$-4
        RET
        END
```



บริษัท สิลาวิเสิร์ช จำกัด 1108/41 ถนนสุขุมวิท แขวงพระโขนง เขตคลองเตย กรุงเทพฯ 10110
TEL. 712-2850-2 FAX. 381-1447 www.sillaresearch.com sale@sillaresearch.com

การทดลองที่ 9

Liquid Crystal Display (LCD)

คำถามก่อนการทดลอง

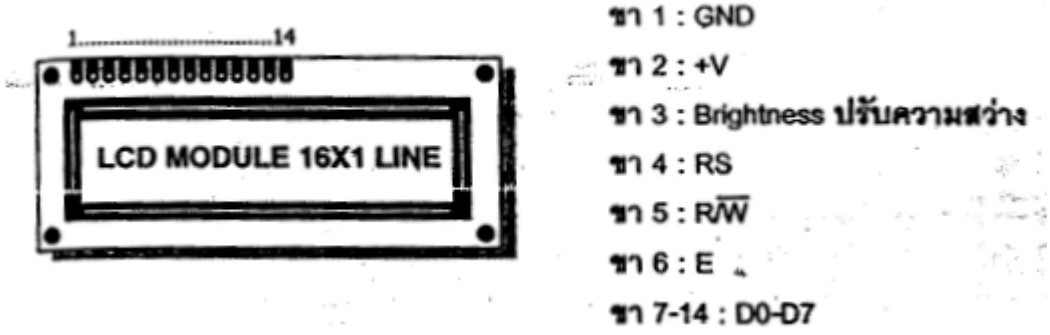
1) โมดูล LCD ต่อกับบอร์ด ANT-32 ได้อย่างไร
2) ทำการ Initialize โมดูล LCD ได้อย่างไร
3) จะส่งตัวอักษร (Character) ไปแสดงบนหน้าจอ LCD ได้อย่างไร
4) จะส่งข้อความ (String) ไปแสดงบนหน้าจอ LCD ได้อย่างไร

บทนำ

ในโมดูล LCD จะมีส่วนประกอบหลัก 3 ส่วนดังนี้

1. *ตัวแสดงผล* (Display) ภายในเป็นผลึกเหลวสามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD
2. *ตัวควบคุม* (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิพควบคุมโดยเฉพาะชิพที่นิยมใช้ คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟฟิก
3. *ตัวขับ* (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิพที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่เบอร์ HD44100H และ MSM5259 เป็นต้น

สำหรับโมดูล LCD ที่ยกมาใช้ในการเรียนรู้ในการทดลอง เป็นขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตรฐาน มีผู้ผลิตหลายราย และมีการระบุเบอร์แตกต่างกันออกไปตามผู้ผลิต อาทิ LM020L ของฮิตาชิ, DMC-16117A ของคอปเทร็กซ์ (Optrex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือ เบอร์เดียวกันนั่นคือเบอร์ HD44750 ของฮิตาชิ



รูปที่ 1 แสดงรูปร่างและการจัดขาโมดูล LCD แบบอักขระ

โมดูล LCD ขนาด 16x1 มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังรูปที่ 1 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

- V_{SS} (ขา 1) : ต่อกราวน์
- V_{DD} (ขา 2) : ต่อไฟเลี้ยง +5 VDC
- V_O (ขา 3) : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล
- RS (ขา 4) : เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่า เป็นคำสั่งสำหรับรีจีสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจีสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล
- $\overline{R/W}$ (ขา 5) : เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล
- E (ขา 6) : เป็นขา Enable ให้ LCD ทำงาน
- D0 ถึง D7 (ขา 7 ถึง 14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอก มีขนาด 8 บิต

ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายในโมดูล LCD แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เสร็จเรียบร้อยก่อน จากนั้นจึงจะรับข้อมูลหรือดำเนินการต่อไป

จุดประสงค์

1. เรียนรู้วิธีการเชื่อมต่อโมดูล LCD เข้ากับบอร์ดไมโครคอนโทรลเลอร์ตระกูล 8051
2. เรียนรู้วิธีการเขียนโปรแกรมแสดงผลบนหน้าจอ LCD

อุปกรณ์

1. บอร์ด ANT-32 และคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซีพร้อมซอฟต์แวร์ PCPlus พร้อมคู่มือการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51
6. โมดูล LCD พร้อมคู่มือการใช้งาน

วิธีการทดลอง

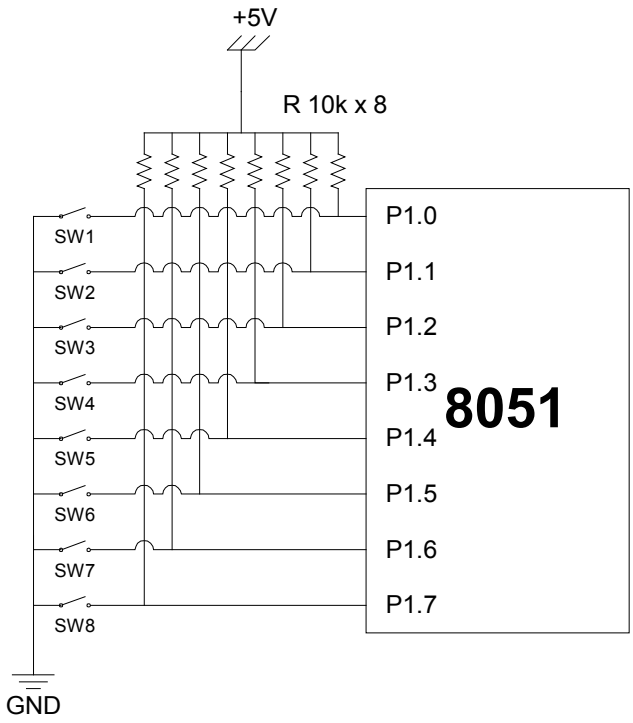
1. ทำการต่อโมดูล LCD เข้าบอร์ด ANT-32
2. เขียนโปรแกรมด้วยภาษาซีเพื่อทำการ Initialize โมดูล LCD
3. เขียนโปรแกรมด้วยภาษาซีเพื่อส่งตัวอักษร (Character) ไปแสดงบนหน้าจอ LCD
4. เขียนโปรแกรมด้วยภาษาซีเพื่อส่งข้อความ (String) ไปแสดงบนหน้าจอ LCD

การทดลองที่ 10

Keypad

คำถามก่อนการทดลอง

1) จงเขียนโปรแกรมตรวจสอบการกดคีย์จำนวน 8 คีย์ ซึ่งต่ออยู่ที่พอร์ต P1.0 ถึง P1.7 แล้วส่งค่าออกทางพอร์ตอนุกรม



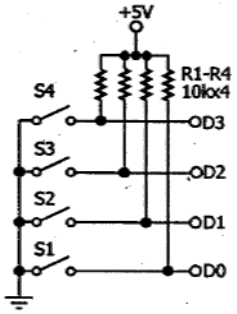
2) จงเขียนผังวงจรของคีย์แบบเมตริกขนาด 8x4 โดยใช้ Port1 และ Port3 ของไมโครคอนโทรลเลอร์

บทนำ

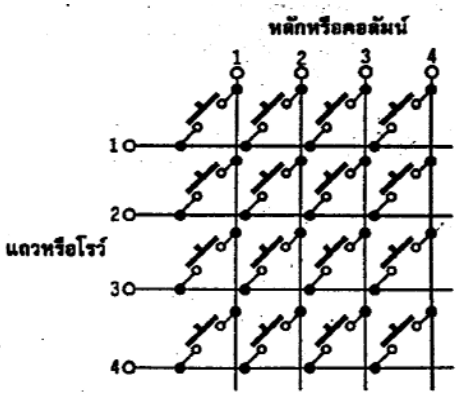
การอ่านค่าหรือรับค่าการกดสวิตช์เป็นอีกงานหนึ่งที่ไมโครคอนโทรลเลอร์ต้องสามารถรองรับและเชื่อมต่อใช้งานร่วมด้วยได้วงจรของสวิตช์มีด้วยกัน 2 ลักษณะใหญ่ๆ คือ ต่อเข้ากับไฟเลี้ยงหรือกราวน์โดยตรง เมื่อสวิตช์ตัวใดต่อวงจรสามารถอ่านค่าได้โดยตรง ดังในภาพที่ 1 วงจรในลักษณะนี้ไม่มีความซับซ้อนสามารถอ่านค่าของสวิตช์ได้ง่ายและรวดเร็ว แต่มีข้อเสียคือ ถ้าหากจำนวนของสวิตช์มีมากๆจำนวนของสายข้อมูลก็จะมีมากตาม ทำให้ระบบหรือวงจรโดยรวมมีขนาดใหญ่และสิ้นเปลือง

วงจรของสวิตช์อีกลักษณะหนึ่งคือ การต่อวงจรแบบเมตริกซ์ (Matrix Switch) ดังในภาพที่ 2 สวิตช์จะถูกต่อกันในแนวแกนตั้งและแกนนอน จะเรียกแนวตั้งว่า หลักหรือคอลัมน์ (Column) ในขณะที่แนวนอนจะเรียกว่า แถวหรือโรว์ (Row) ดังนั้นค่าของสวิตช์จะต้องประกอบด้วย ตำแหน่งในแนวหลักและแถว กระบวนการที่จะทำให้ได้มาซึ่งค่าของสวิตช์มีขั้นตอนซับซ้อนพอสมควร แต่วงจรของสวิตช์แบบนี้มีข้อดีคือ สามารถรองรับการเพิ่มของสวิตช์ได้อย่างสะดวก เพียงเพิ่มเติมจำนวนสวิตช์และแก้ไขซอฟต์แวร์อีกเล็กน้อยเท่านั้น

ทำให้วงจรสวิตช์แบบเมตริกซ์เป็นที่นิยมใช้มากในระบบควบคุมอัตโนมัติหรือกึ่งอัตโนมัติที่มีจำนวนสวิตช์มากกว่า 8 ตัว ในการใช้งานทั่วไปจะเรียกสวิตช์แบบเมตริกซ์นี้ว่า คีย์แพด (Keypad)



ภาพที่ 1 วงจรของสวิตช์แบบต่อเข้ากับไฟเลี้ยงและกราวน์



ภาพที่ 2 วงจรของสวิตช์แบบเมตริกซ์หรือคีย์แพด

จุดประสงค์

1. เรียนรู้วิธีการเชื่อมต่อ Keypad เข้ากับบอร์ดไมโครคอนโทรลเลอร์ 8051
2. เรียนรู้วิธีการเขียนโปรแกรมเพื่อใช้งาน Keypad

อุปกรณ์

1. บอร์ด ANT-32 พร้อมคู่มือการใช้งาน
2. สายเชื่อมต่อบอร์ด ANT-32 กับเครื่องคอมพิวเตอร์พีซี
3. โปรแกรม Paulmon พร้อมคู่มือ
4. เครื่องคอมพิวเตอร์พีซี 1 เครื่องพร้อมซอฟต์แวร์ PCPlus พร้อมวิธีการใช้งานฉบับย่อ
5. ชุดโปรแกรม MCS-51
6. Keypad ขนาด 4x3 คีย์ พร้อมคู่มือการใช้งาน

วิธีการทดลอง

เขียนโปรแกรมเพื่อรับค่าที่ส่งจาก Keypad ให้แสดงยังหน้าจอคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม

คำแนะนำ

ทดลองรับค่าจากพอร์ตทีละพอร์ตก่อน โดยตรวจสอบว่ามีการกดสวิตช์ที่พอร์ตใดแล้วจึงแก้ไขโปรแกรมให้สามารถรับค่าได้ทั้ง 12 คีย์