เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ

AUTOMATIC PCB DRILLER

โดย

นายพีรวุฒิ คันธวัฒน์

นายภาณุมาศ ศิริเฑียรทอง

นายสุรชัย คัมภีร์รัตนอาภา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ
AUTOMATIC PCB DRILLER

โดย

นายพีรวุฒิ คันธวัฒน์ รหัสประจำตัว 40010525
นายภาณุมาศ ศิริเทียรทอง รหัสประจำตัว 40010559
นายสุรชัย คัมภีร์รัตนอาภา รหัสประจำตัว 40010903

อาจารย์ที่ปรึกษา

รศ. อิทธิชัย อรุณศรีแสงไชย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ (Automatic PCB Driller)

ผู้จัดทำ

1. นายพีรวุฒิ คันธวัฒน์ 40010525
2. นายภาณุมาศ ศิริเตียรทอง 40010559
3. นายสุรชัย คัมภีร์รัตนอาภา 40010903

..... อาจารย์ที่ปรึกษา

(รศ. อิทธิชัย อรุณศรีแสงไชย)

เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ

Automatic PCB Driller

นายพีรวุฒิ คันธวัฒน์ รหัสประจำตัว 40010525

นายภาณุมาศ ศิริเตียรทอง รหัสประจำตัว 40010559

นายสุรชัย คัมภีร์รัตนอาภา รหัสประจำตัว 40010903

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะได้รับการสอบได้

ลงชื่อ.....(อาจารย์ที่ปรึกษา)

(รศ. อิทธิชัย อรุณศรีแสงไชย)

วันที่ 4 / มี.ค. / 54

เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ

พีรวุฒิ คันธวัฒน์
ภาณุมาศ ศิริเฑียรทอง
สุรชัย คัมภีร์รัตนอาภา

อาจารย์ที่ปรึกษา
รศ.อิทธิชัย อรุณศรีแสงไชย
ปีการศึกษา 2543

บทคัดย่อ

โปรเจคนี้นำเสนอ เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ โดยจะประกอบไปด้วย ส่วนของเครื่องกล , ระบบวงจร ,และส่วนของการควบคุมโดยโปรแกรมคอมพิวเตอร์ ซอร์ฟแวร์ถูกพัฒนาขึ้นเพื่อที่จะติดต่อกันระหว่าง ส่วนของ ระบบทางกล คอมพิวเตอร์และไมโครคอมพิวเตอร์ ความสามารถของเครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ มีขอบเขตในการเจาะ 17ซม. X 33 ซม. และความละเอียดของการเคลื่อนสเต็ป 0.049 มิลลิเมตร และความเร็วของสว่าน คือ 23,000 รอบต่อนาที

Automatic PCB Driller

Peerawut Kantawat

Panumas Siritianthong

Surachai Khumpeeratanaarpa

Advisor

Assoc.Prof.Itthichai Arungsrisangchai

Academic year 2000

Abstract

This project presents an Automatic Printed Circuit Board Driller machine . The machine consists of mechanical part , electronic part and interface part . Software was also developed to communicate between the machine and PC . The machine could drill the PCB within 17x33cm^2 area with the precision of 0.049mm . The speed of the driller is 23000 rounds/minutes.

สารบัญ

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

ในปัจจุบัน การทำแผ่นวงจรพิมพ์ ขั้นตอนที่ค่อนข้างยากและเสียเวลา คือ ขั้นตอนการเจาะแผ่นวงจรพิมพ์ ซึ่งถ้าต้องเจาะแผ่นวงจรพิมพ์ทีละมากๆแล้วนั้น จะเสียเวลาเป็นอย่างมาก ซึ่งในอุตสาหกรรมทำแผ่นวงจรพิมพ์นั้น จำเป็นต้องมีเครื่องเจาะแผ่นวงจรพิมพ์อัตโนมัติไว้ใช้งานเพื่อเพิ่มความสะดวกในการเจาะแผ่น PCB มากขึ้น ดังนั้นในโครงงานนี้จึงสร้างเครื่องเจาะแผ่น PCB ขึ้นมา โดยสามารถเชื่อมต่อกับคอมพิวเตอร์ และสามารถเจาะรูตามไฟล์ PCB ของโปรแกรม Protel ได้ เพราะโปรแกรมProtel เป็นที่นิยมในการออกแบบลายวงจรพิมพ์

1.2 ลักษณะของโครงงาน

เครื่องเจาะแผ่นปริ้นท์อัตโนมัติ ทำงานโดยวงจรอิเล็กทรอนิกส์ควบคุมการทำงานของกลไกเครื่องกล โดยกลไกทางกลนั้น จะใช้สเต็ปปิ้งมอเตอร์เป็นตัวเคลื่อนตำแหน่งตามแกน X-Y-Z และมีไมโครคอนโทรลเลอร์ควบคุมการทำงานของวงจรอิเลกทรอนิกส์อีกทีหนึ่ง

การที่จะให้เครื่องเจาะแผ่น PCB สามารถเจาะรูตามไฟล์ PCB ของโปรแกรม Protel ได้นั้น ต้องมีการเชื่อมโยงระหว่างไมโครคอนโทรลเลอร์กับ คอมพิวเตอร์ โดยผ่านพอร์ทอนุกรมของคอมพิวเตอร์ ในที่นี้จะใช้ MAX232 เพื่อแปลงแรงดันตามมาตราฐาน RS-232 เป็นระดับแรงดันทีทีแอล ส่วนซอร์ฟแวร์ในการสั่งการทำงานของเครื่องจะใช้โปรแกรม วิชวลเบสิก เป็นตัวสั่งการ

1.3 ขั้นตอนและวิธีการดำเนินงาน

1) ศึกษาและออกแบบโครงสร้างทางกล

2) ศึกษาวิธีการควบคุมสเต็ปปิ้งมอเตอร์ และออกแบบวงจรสเต็ปปิ้งมอเตอร์

3) ออกแบบวงจรควบคุมการทำงานของวงจรขับสเต็ปปิ้งมอเตอร์และทดสอบการทำงานของโครงสร้างทางกล

4) ออกแบบวงจรที่ใช้เชื่อมต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์

5) ศึกษาโปรแกรม Protel และเขียนซอร์ฟแวร์ที่ใช้ควบคุมการทำงานของเครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ

6) นำฮาร์ดแวร์และซอร์ฟแวร์มาประกอบกัน

7) ทดสอบการทำงานของระบบและทำการปรับปรุง

8) สรุปผลการทำงานและเสนอแนวทางแก้ไข

1.4 ประโยชน์ที่คาดว่าจะได้รับ

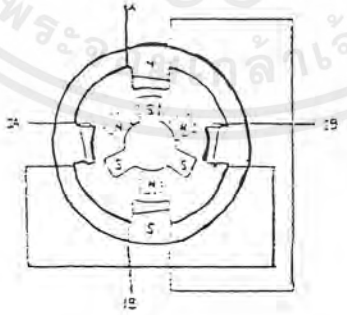
หลังจากทำโครงงานแล้ว สามารถนำเครื่องเจาะแผ่นปริ้นท์อัตโนมัติ ไปใช้งานได้ และสามารถใช้ข้อมูลจากโปรแกรม Protel มาใช้กับเครื่องเจาะได้ เพื่อเพิ่มความสะดวกในการทำลายวงจรพิมพ์

บทที่ 2
ทฤษฎีพื้นฐานและการทำงานของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์เป็นเครื่องจักรกลไฟฟ้าประเภทหนึ่งเช่นเดียวกับ ดีซีมอเตอร์ หรือ เอซีมอเตอร์ แต่สเต็ปปิ้งมอเตอร์มีคุณสมบัติพิเศษบางอย่างที่แตกต่างจากมอเตอร์ชนิดทฤษฎีพื้นฐานและหลักการทำงานของสเต็ปปิ้งมอเตอร์อื่นๆ กล่าวคือ มีการหมุนเป็นแบบสเต็ปปิ้งตามจังหวะของสัญญาณไฟฟ้าที่ป้อน ลักษณะเช่นนี้เองจึงทำให้เราสามารถควบคุมจำนวนสเต็ปปิ้งมอเตอร์ของการหมุนได้ด้วยการควบคุมการป้อนสัญญาณไฟฟ้า หรือ สามารถควบคุมการหมุนแบบลูปเปิด (OPEN LOOP) ได้นั่นเอง และถ้านำสเต็ปปิ้งมอเตอร์ ไปต่อใช้งานกับเครื่องมือบางประเภทเช่น พล็อตเตอร์ เครื่องเจาะอัตโนมัติ ก็จะทำให้การควบคุมตำแหน่งเป็นไปตามที่ต้องการได้อย่างแม่นยำ

2.1 หลักการทำงานของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ สามารถแบ่งโครงสร้างทางกายภาพออกได้เป็น 2 ส่วน คือ สเตเตอร์ (STATOR) และ โรเตอร์ (ROTOR) ตัวสเตเตอร์เป็นส่วนที่อยู่กับที่ ประกอบด้วยขดลวดทองแดงซึ่งพันอยู่รอบแกนเหล็ก เพื่อสร้างสนามแม่เหล็กเมื่อมีการจ่ายกระแสผ่านขดลวด ส่วนโรเตอร์เป็นส่วนที่เคลื่อนที่ มีลักษณะเป็นแท่งเหล็กทรงกลม และที่ผิวรอบนอกมีลักษณะเป็นซีกฟันซึ่งทำจากแม่เหล็กถาวร ดังรูปที่ 2.1

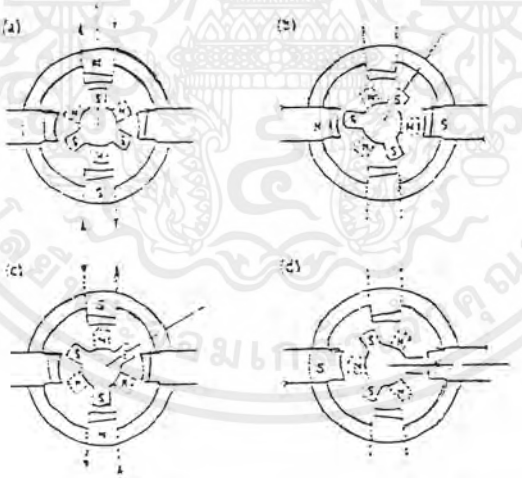


รูปที่ 2.1 แสดงโครงสร้างของไฮบริดจ์สเต็ปปิ้งมอเตอร์ที่มีจำนวนสเต็ปต่อรอบเท่ากับ12

เมื่อยังไม่มีการจ่ายกระแสให้กับขดลวดของมอเตอร์ซีกฟันอันใดอันหนึ่งของโรเตอร์จะอยู่ในตำแหน่งที่ตรงกับซีกฟันอันใดอันหนึ่งของสเตเตอร์ ทั้งนี้เป็นเพราะแม่เหล็กถาวรที่ตัวของ

โรเตอร์พยายามที่จะทำให้ค่าความต้านทานทางแม่เหล็กไฟฟ้า (RELUCTANCE) มีค่าน้อยที่สุด ซึ่ง ณ ที่จุดที่ซี่ฟันของโรเตอร์และสเตเตอร์ตรงกันนั้นมีค่าความต้านทานทางแม่เหล็กไฟฟ้าน้อยที่สุด ทำให้เกิดเส้นแรงแม่เหล็กไฟฟ้ามากที่สุด และจากรูปที่ 2.1 เส้นแรงแม่เหล็กไฟฟ้าจะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ขึ้นมา 2 คู่ ทั้งที่ตัวสเตเตอร์และตัวโรเตอร์ดังรูป ค่าทอร์ค (TORQUE) ที่ทำให้ตัวโรเตอร์สามารถยึดอยู่ในตำแหน่งดังกล่าวนี้เรียกว่า ดีเท็นท์ ทอร์ค (DETENT TORQUE) (หมายความว่า การที่จะทำให้มอเตอร์เคลื่อนที่ในขณะที่ไม่ได้จ่ายกระแสให้กับขดลวดของมอเตอร์จะต้องออกแรงมากกว่าค่าของดีเท็นท์ทอร์ค จึงจะทำให้โรเตอร์เคลื่อนที่ได้) รูปที่ 2.1 นั้น มี 12 ตำแหน่งที่สามารถเกิดดีเท็นท์ทอร์คได้

เมื่อจ่ายกระแสให้กับขดลวดที่อยู่ในสเตเตอร์คู่ใดคู่หนึ่ง ดังรูปที่ 2.2 a จะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ที่ซีกฟันของตัวสเตเตอร์ ซึ่งจะดึงดูดซีกฟันของตัวโรเตอร์ที่มีขั้วแม่เหล็กที่มีศักย์ต่างกันที่อยู่ใกล้ที่สุดเข้าไว้ ตำแหน่งนี้เรียกว่า สเตเบิ้ลโพสซิชั่น (STABLE POSITION) ของโรเตอร์ จะมีจำนวนตำแหน่งเท่ากับจำนวนซีกฟันของโรเตอร์ และแรงที่จะทำให้โรเตอร์เปลี่ยนตำแหน่งไปจากตำแหน่งสเตเบิ้ล โพสซิชั่นนี้เรียกว่า โฮสดิ้ง ทอร์ค (HOLDING TORQUE)



รูปที่ 2.2 แสดงขั้นตอนการทำงานของสเต็ปปิ้งมอเตอร์แบบ เต็มสเต็ปหนึ่งเฟส

เมื่อสับเปลี่ยนการจ่ายกระแสให้แก่ขดลวด จากขดหนึ่งไปยังอีกขดหนึ่ง เนื่องจากขดลวดวางอยู่ในตำแหน่งที่ต่างกัน 90 องศา ก็จะทำให้ตัวสเตเตอร์ดึงดูดซีกฟันของตัวโรเตอร์อีกซีกหนึ่งที่ใกล้ที่สุดเข้าไว้ ซึ่งจะทำให้ตัวโรเตอร์เคลื่อนที่ไป 1 สเต็ปหรือ 30 องศา ดังรูปที่ 2.2b จากนั้นก็เปลี่ยนไปจ่ายกระแสให้กับขดลวดชุดแรกโดยในคราวนี้เปลี่ยนทิศทางการไหลของกระแสให้ตรง

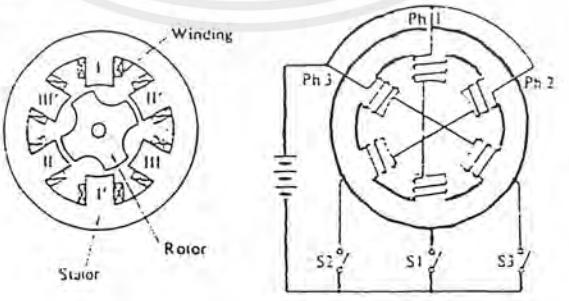
กันข้ามกับครั้งแรก ซึ่งจะทำให้ตัวโรเตอร์เคลื่อนที่ไปอีก 1 สเต็ป (เคลื่อนที่ไป 30 องศา) ดังรูปที่ 2.2c หลังจากนั้นก็ไปจ่ายกระแสให้กับขดลวดชุดที่ 2 โดยกลับทิศทางของกระแสที่ป้อนให้อีกเช่นกันทำให้โรเตอร์หมุนไป 90 องศา ดังรูปที่ 2.2d และ ถ้าหากเราป้อนกระแสให้กับมอเตอร์เหมือนที่เราป้อนในครั้งแรกแล้ว ซีกฟันซีกถัดไปของตัวโรเตอร์จะอยู่ในตำแหน่งที่เหมือนกับใน รูปที่ 2.2a อีกครั้งหนึ่ง ถ้าหากเราต้องการเคลื่อนที่หนึ่งรอบ เราต้องทำการกระตุ้นให้มอเตอร์เคลื่อนที่ไปจนครบ 12 สเต็ป และถ้าต้องการให้โรเตอร์หมุนไปอีกทิศทางหนึ่ง ก็จะทำการสลับลำดับในการจ่ายกระแส จากรูปที่ 2.2a,2.2d,2.2c,2.2b ตามลำดับ

2.2 ชนิดของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ สามารถแบ่งออกได้หลายชนิดตามลักษณะโครงสร้างและการใช้งานดังต่อไปนี้

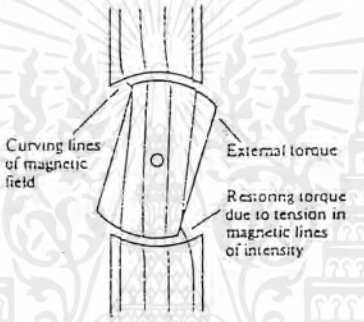
2.2.1 สเต็ปปิ้งมอเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ (VARIABLE RELUCTANCE STEPPING MOTOR)

สเต็ปปิ้งมอเตอร์ชนิดนี้สามารถปรับค่ารีลัคแตนซ์ได้ ซึ่งรูปที่ 2.3 แสดงภาพตัดขวางของสเต็ปปิ้งมอเตอร์แบบ 3 เฟส โดยที่สเตเตอร์มีฟันทั้งหมด 6 ซี่ ซี่ที่อยู่ตรงข้ามกันหรือทำมุม 180 องศาซึ่งกันและกันจะเป็นเฟสเดียวกัน ขดลวดที่พันอยู่ที่ฟันของสเตเตอร์ในแต่ละเฟสจะต่ออนุกรมหรือขนานก็ได้ จากรูปที่ 2.3 เป็นการต่อแบบอนุกรม ส่วนโรเตอร์นั้นมีฟัน 4 ซี่ ทั้งโรเตอร์และสเตเตอร์ทำมาจากโลหะซิลิกอน ซึ่งมีสภาพซึมซับทางแม่เหล็กสูงและยอมให้สนามแม่เหล็กจำนวนมากไหลผ่านได้ ฟันของสเตเตอร์ในเฟสเดียวกันจะมีขั้วต่างกันโดยซี่ I,II,III เป็นขั้วเหนือและซี่ I',II',III' เป็นขั้วใต้หลังจากถูกกระตุ้น

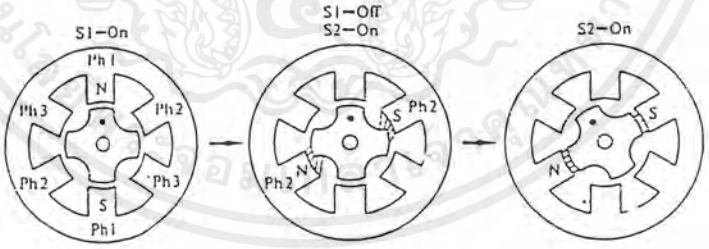


รูปที่ 2.3 แสดงภาพตัดขวางของสเต็ปปิ้งมอเตอร์แบบ 3 เฟส

รูปที่ 2.4 แสดงตำแหน่งสมดุลย์ เมื่อเฟสใดเฟสหนึ่งของสเต็ปปิ้งมอเตอร์ถูกกระตุ้น



รูปที่ 2.5 แสดงแรงภายนอกที่มีผลต่อเส้นแรงแม่เหล็ก



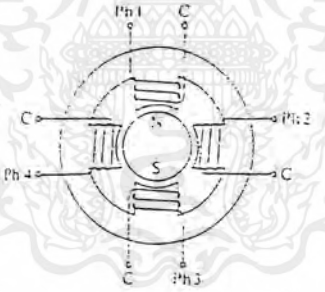
รูปที่ 2.6 แสดงขั้นตอนการเคลื่อนที่ของโรเตอร์เมื่อสเต็ปปิ้งมอเตอร์ถูกกระตุ้น

กระแสที่ไหลในแต่ละเฟสถูกควบคุมโดยสวิทช์ ปิด/เปิด ถ้าเฟส 1 ถูกกระตุ้นจะมีกระแสไหลและเกิดฟลักซ์แม่เหล็กดังแสดงในรูปที่ 2.4 แกนโรเตอร์จะอยู่ในตำแหน่งเดียวกับซี่ I และ I' ทำให้ทั้งโรเตอร์และสเตเตอร์อยู่ในแนวเดียวกันกรณีนี้จะทำให้ค่า รีลัคแตนซ์ มีค่าน้อยที่สุดซึ่งเป็นตำแหน่งที่สมดุลย์ถ้าโรเตอร์ถูกกระทำจากแรงภายนอกจะทำให้เปลี่ยนตำแหน่ง ดังรูปที่ 2.5 แรงบิดกระทำกับโรเตอร์ในทิศตามเข็มนาฬิกาทำให้ตำแหน่งเปลี่ยนไป มีผลทำให้เส้นแรงแม่เหล็กเคลื่อน

ที่จากซี่ของโรเตอร์และสเตเตอร์ เมื่อโรเตอร์และสเตเตอร์ไม่ได้อยู่ในแนวเดียวกันแล้วค่ารีลัคแตนซ์จะมีค่ามาก จากนั้นสเต็ปปิ้งมอเตอร์จะทำให้มีค่ารีลัคแตนซ์น้อยที่สุด พอเฟส II ถูกกระตุ้น ดังรูปที่ 2.6 โรเตอร์ถูกแรงภายนอกกระทำให้เคลื่อนไป 30 องศา ในทิศทวนเข็มนาฬิกา จากนั้นก็จะย้ายจากมุมที่เกิดการกระตุ้นกลับไปยังตำแหน่งที่ค่ารีลัคแตนซ์น้อยที่สุด การย้ายจากมุมที่เกิดการกระตุ้นแต่ละครั้งให้กลับไปยังตำแหน่งเดิมเรียกว่าสเต็ป

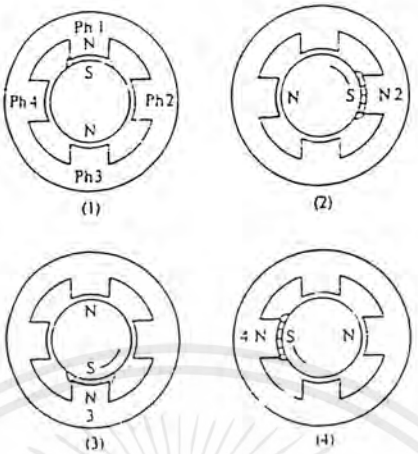
2.2.2 สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร

สเต็ปปิ้งมอเตอร์ชนิดนี้ใช้แม่เหล็กถาวรเป็นโรเตอร์ และมีซี่ฟันของสเตเตอร์ล้อมรอบ ซี่ฟันของสเตเตอร์ถูกพันด้วยขดลวดสำหรับสร้างสนามแม่เหล็ก เมื่อต้องการให้สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรมีขนาดมุมของสเต็ปเล็กลงจะต้องเพิ่มขั้วแม่เหล็กของโรเตอร์ และจำนวนซี่ฟันของสเตเตอร์ แต่ก็มีขีดจำกัดในการเพิ่มจำนวนขั้วแม่เหล็กของโรเตอร์เนื่องจากการสร้างแม่เหล็กถาวรสร้างโดยมีขั้วแม่เหล็กหลายขั้วทำได้ยาก



รูปที่ 2.7 แสดงโครงสร้างของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร

ตัวอย่างการทำงานของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร สมมติว่าสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร ขนาด 4 เฟส มีโรเตอร์เป็นแม่เหล็กถาวรทรงกระบอกและสเตเตอร์ มี 4 ซี่ฟันซึ่งรอบๆพันด้วยขดลวด มีรูปแแบบพื้นฐานของการทำงาน คือ เมื่อสร้างสัญญาณกระตุ้นตามลำดับเฟส โรเตอร์จะหมุนไปตามทิศทางของการกระตุ้น ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 แสดงการทำงานของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรขนาด 4 เฟส

ข้อเสียของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร คือ มีขนาดมุมสเต็ปใหญ่ทำให้ความละเอียดของสเต็ปต่อรอบน้อยเนื่องจากโครงสร้างของโรเตอร์เป็นแม่เหล็กถาวร การสร้างแม่เหล็กถาวรให้มีขั้วหลายขั้วทำได้ยากทำให้ไม่สามารถสร้างสเต็ปขนาดเล็กได้ สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรส่วนใหญ่จะมีโครงสร้างขนาดเล็ก ทำให้ค่าทอร์คที่ได้ต่อหน่วยต่อปริมาตรต่ำ ถ้าต้องการปรับปรุงประสิทธิภาพในเรื่องของทอร์ค แม่เหล็กถาวรที่ใช้ต้องทำจากสารแม่เหล็กที่มีสภาพความเป็นแม่เหล็กสูง

2.2.3 สเต็ปปิ้งมอเตอร์แบบไฮบริดจ์

สเต็ปปิ้งมอเตอร์ชนิดนี้มีแกนโรเตอร์เป็นแม่เหล็กถาวร โดยมีการทำงานร่วมกันของมอเตอร์แบบแม่เหล็กถาวรและมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ได้ ไฮบริดจ์สเต็ปปิ้งมอเตอร์นี้มีโครงสร้างของสเตเตอร์คล้ายกับโครงสร้างของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ แต่ต่างกันที่การต่อขดลวดโดยที่แต่ละเฟสของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์จะมีขดลวด 2 ขด แต่ละขดมีขั้วต่างกัน แต่ไฮบริดจ์สเต็ปปิ้งมอเตอร์ขดลวดทั้งหมด จะพันอยู่ที่ขั้วเดียวกันเรียกว่า ไปโพลาร์ (BIPOLAR) ซึ่งในการกระตุ้นแต่ละครั้งจะให้ขั้วที่แตกต่างกัน

หลักการทำงานของไฮบริดจ์สเต็ปปิ้งมอเตอร์ที่แตกต่างจากสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์คือแรงบิดที่เกิดจากสนามแม่เหล็กจะไม่ขึ้นอยู่กับกระแสที่ไหลผ่านขดลวดเพียงอย่างเดียวแต่ขึ้นอยู่กับโครงสร้างของซี่ฟันด้วย ซึ่งซี่ฟันถูกออกแบบเพื่อให้ได้โครงสร้างขนาดเล็ก และใช้แม่เหล็กถาวรเป็นแกนกลางเพื่อลดผลของการเกิดออสซิลเลททางกล

ข้อดีของไฮบริดจ์สเต็ปปิ้งมอเตอร์ คือ มีขนาดสเต็ปขนาดเล็ก มีความละเอียดของสเต็ปต่อรอบสูง มีค่าทอร์คสูงกว่าสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ แต่สเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ มีแรงเฉื่อยทางแมคคานิคส์น้อยกว่าไฮบริดจ์สเต็ปปิ้งมอเตอร์

นอกจากสเต็ปปิ้งมอเตอร์ทั้ง 3 ชนิดที่กล่าวมาแล้วยังมีสเต็ปปิ้งมอเตอร์ชนิดอื่นๆ ที่ไม่ได้กล่าวถึงอีกเช่น ลิเนียร์สเต็ปปิ้งมอเตอร์ ซึ่งเป็นมอเตอร์ที่ได้รับการออกแบบให้มีการเคลื่อนที่แบบเป็นเชิงเส้น อิเล็คโตรดไฮดรอลิคสเต็ปปิ้งมอเตอร์ ซึ่งเป็นสเต็ปปิ้งมอเตอร์กำลังสูงที่ใช้ในงานอุตสาหกรรม เป็นต้น

2.3 วงจรขับ (DRIVER CIRCUITS)

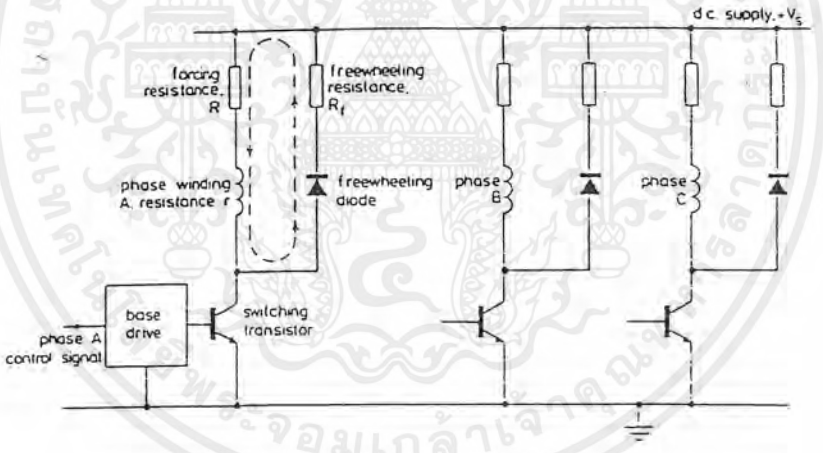
สัญญาณควบคุมที่ใช้สำหรับควบคุมการทำงานของสเต็ปปิ้งมอเตอร์มักจะเป็นสัญญาณที่สร้างจากวงจรดิจิตอล เช่น จากไมโครคอนโทรลเลอร์ ซึ่งเป็นอุปกรณ์จำพวก TTL แรงดันที่ใช้มีค่าเท่ากับ 5 โวลต์ และสามารถจ่ายกระแสที่สูงกว่านั้น ดังนั้นจึงจำเป็นที่จะต้องมีวงจรขับ เพื่อทำหน้าที่จ่ายแรงดันและกระแสที่เพียงพอให้กับตัวสเต็ปปิ้งมอเตอร์ โดยทั่วๆไป วงจรขับมักจะสร้างจากไบโพลาร์ทรานซิสเตอร์ที่นำมาต่อใช้งานเป็นสวิตซ์ ลักษณะของวงจรขับขึ้นอยู่กับชนิดของสเต็ปปิ้งมอเตอร์ที่ใช้ เช่น ถ้าใช้แบบสเต็ปปิ้งมอเตอร์วาริเอเบิลรีลัคแตนซ์ ซึ่งจะมีอย่างน้อย 3 เฟส วงจรขับที่ใช้จะทำงานในลักษณะของสวิตซ์
ปิด/เปิด ให้กระแสไหลผ่านไปยังขดลวดในทิศทางเดียว เราเรียกวงจรขับแบบนี้ว่า ยูนิโพลาร์ การจ่ายกระแสเพียงทิศทางเดียว แต่ถ้าใช้สเต็ปปิ้งมอเตอร์แบบไฮบริดจ์หรือแบบแม่เหล็กถาวร ซึ่งมักจะมีเพียงสองเฟส จะต้องใช้วงจรขับที่สามารถจ่ายกระแสได้สองทิศทาง เรียกวงจรขับประเภทนี้ว่าไบโพลาร์ ซึ่งประกอบด้วย ทรานซิสเตอร์หลายตัวต่อเป็นวงจรแบบบริดจ์

2.3.1 วงจรขับแบบยูนิโพลาร์ (UNIPOLAR DRIVE CIRCUIT)

วงจรขับพื้นฐานที่เหมาะสำหรับสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ดังที่แสดงไว้ในรูปที่ 2.9 ขดลวดแต่ละเฟสจะถูกกระตุ้นโดยวงจรขับแต่ละชุด ซึ่งวงจรขับแต่ละชุดก็จะได้รับสัญญาณควบคุมจากไมโครคอนโทรลเลอร์

กระแสจะไหลผ่านขดลวดแต่ละเฟสเมื่อสวิตซ์ซึ่งทรานซิสเตอร์อยู่ในสภาวะอิ่มตัว เนื่องจากกระแสที่ไบอัสทางด้านเบส ในสภาวะเช่นนี้แรงดันดีซีจากแหล่งจ่ายไฟจะไหลผ่านตัวต้านทานผ่านขดลวดของสเต็ปปิ้งมอเตอร์และไหลผ่านทรานซิสเตอร์ เนื่องจากแรงดันตกคร่อมทรานซิสเตอร์ในสภาวะอิ่มตัวมีค่าน้อย (ประมาณ 0.1 โวลต์) แรงดันจากแหล่งจ่ายไฟทั้งหมดจะทำให้เกิดกระแสไหลผ่านขดลวดโดยมีความสัมพันธ์ กับผลรวมของความต้านของขดลวด (r) และความต้านทานตัวต้านทาน (R) ดังสมการ 2.1

$$Vs = I (r+R) \qquad (2.1)$$



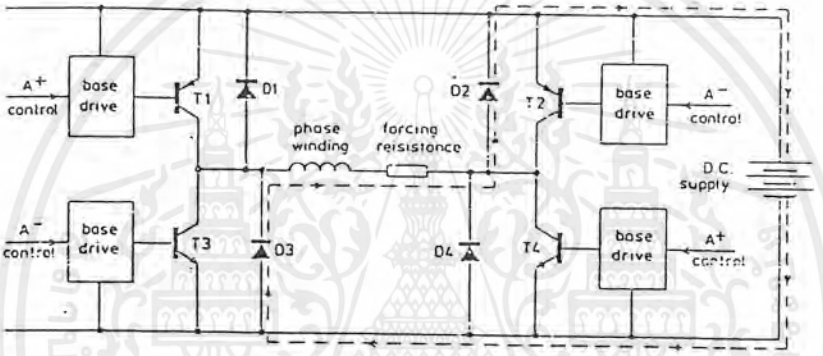
รูปที่ 2.9 วงจรขับแบบยูนิโพลาร์

ตามปกติขดลวดของสเต็ปปิ้งมอเตอร์จะแสดงคุณสมบัติของตัวเหนี่ยวนำ (L) ซึ่งทำให้การตอบสนองต่อกระแสที่ไหลผ่านตัวมันช้า ส่งผลต่อการทำงานของมอเตอร์ที่ความเร็วสูง การใส่ตัวต้านทานอนุกรมเข้าไว้กับขดลวดจะช่วยแก้ปัญหานี้ได้

ความเหนี่ยวนำของขดลวดยังทำให้กระแสไม่หยุดไหลทันทีทันใดที่ทรานซิสเตอร์อยู่ในสถานปิด ทำให้เกิดแรงดันเหนี่ยวนำตกคร่อมคอลเล็คเตอร์และอีมิตเตอร์ ซึ่งเป็นสาเหตุที่ทำให้วงจรขับเสียหาย ปัญหานี้แก้ไขได้โดยการต่อ ฟรีวิลลิ่งไดโอด (FREEWHEELING DIODE) และฟรีวิลลิ่งรีซิสเตอร์ (FREEWHEELING RESISTANCE) เพื่อเป็นทางผ่านของกระแสแทน

2.3.2 วงจรขับแบบไบโพลาร์ (BIPOLAR DRIVE CIRCUIT)

วงจรขับชนิดนี้จะต่อทรานซิสเตอร์เป็นแบบบริดจ์ วงจรบริดจ์หนึ่งชุดจะขับมอเตอร์ได้หนึ่งเฟส เหมาะที่ใช้กับสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร หรือแบบไฮบริดจ์ ดังแสดงในรูปที่ 2.10



รูปที่ 2.10 วงจรขับแบบไบโพลาร์

ทรานซิสเตอร์จะผลัดกันทำงานทีละคู่ตามทิศทางของกระแสที่ต้องการ สำหรับการกระตุ้นขดลวดในทิศบวก ทรานซิสเตอร์ T1 และ T4 จะทำงาน ทำให้เกิดทางเดินของกระแสจากแหล่งจ่ายไฟ ไหลผ่านทรานซิสเตอร์ T1 จากนั้นไหผ่านขดลวดของมอเตอร์ผ่านตัวต้านทานแล้วจึงไหลเข้าทรานซิสเตอร์ T4 กลับเข้าสู่แหล่งจ่ายไฟ ในทางกลับกันในกรณีการจ่ายกระแสในทิศลบ (ทิศตรงกันข้าม) ทรานซิสเตอร์ T2 และ T3 จะทำงาน เพื่อให้กระแสไหลผ่านขดลวดในทิศตรงกันข้าม

จากรูปจะเห็นว่ามีไดโอดสี่ตัว ต่อขนานกับทรานซิสเตอรแต่ละตัว จุดประสงค์เพื่อ ให้เกิดเส้นทางการไหลของกระแสฟรีวิลลิ่ง (FREEWHEELING CURRENT) บริเวณไดโอด D2 และ D3 จะเป็นทางผ่านของกระแสหลังจากที่ทรานซิสเตอร์ T1 และ T4 หยุดทำงาน ส่วน D1 และ D4 จะมีกระแสไหลผ่านขณะที่ทรานซิสเตอร์ T2 และ T3 หยุดทำงาน กระแสฟรีวิลลิ่งในวงจรขับแบบไบโพลาร์จะสิ้นสุดเร็วกว่าแบบยูนิโพลาร์ เพราะฉะนั้นจึงไม่จำเป็นต้องต่อฟรีวิลลิ่งรีซิสแตนซ์เหมือนยูนิโพลาร์

2.4 การขับสเต็ปปิ้งมอเตอร์

การขับสเต็ปปิ้งมอเตอร์ สามารถทำได้ 2 วิธี ตามลักษณะสเต็ปการหมุนของมอเตอร์ได้ 2 วิธี คือ

2.4.1 การขับสเต็ปปิ้งมอเตอร์แบบฟูลสเต็ป

เป็นการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ โดยจ่ายไฟไปยังสเต็ปปิ้งมอเตอร์ให้ครบ 4 เฟส ซึ่งแบ่งออกเป็น 2 รูปแบบ คือ

- แบบฟูลสเต็ป 1 เฟส เป็นการกระตุ้นรูปแบบที่ง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่ง ไล่เรียงถัดกันไปให้ครบ 4 เฟส แล้วทำการกระตุ้นวนไปเรื่อยๆ ถ้าต้องการให้ทิศทางการหมุนสวนกัน ให้ทำลำดับการกระตุ้น เรียงกลับกันจากเฟส 4 ไปยัง เฟส 1 ขั้นตอนการทำงานแสดงดังตารางที่ 2 - 1

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

ตารางที่ 2 - 1 การขับสเต็ปปิ้งมอเตอร์แบบฟูลสเต็ป 1 เฟส

- แบบฟูลสเต็ป 2 เฟส คล้ายกับการกระตุ้นแบบ 1 เฟส แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายไฟให้ขดลวด 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไปจนครบ 4 จังหวะ แล้วกระตุ้นวนไปเรื่อยๆ ขั้นตอนการทำงานดังแสดงในตารางที่ 2 - 2

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

ตารางที่ 2 - 2 การขับสเต็ปปิ้งมอเตอร์แบบฟูลเสต็ป 2 เฟส

2.3.2 การขับสเต็ปปิ้งมอเตอร์แบบแบ่งละเอียด

โดยทั่วไปการขับสเต็ปปิ้งมอเตอร์จะใช้วงจรขับแบบโวลท์เตจคงที่ ซึ่งสามารถขับสเต็ปปิ้งมอเตอร์ได้แบบเต็มสเต็ปและแบบครึ่งสเต็ปเท่านั้น ไม่สามารถควบคุมการขับสเต็ปที่ละเอียดได้ ถ้าจะควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบแบ่งละเอียด ทำได้โดยอาศัยหลักการควบคุมการจ่ายกระแสของแต่ละเฟสให้เป็นสัดส่วนต่อกัน โดยให้กระแสที่ไหลในขดลวด A และ B ของสเต็ปปิ้งมอเตอร์มีค่าลดหลั่นกัน ค่าของกระแสที่ลดหลั่นกันนี้จะเป็นลำดับแทรกกันอยู่ของค่ากระแสในการหมุนแบบเต็มสเต็ป ถ้าค่าของระดับกระแสถูกแบ่งเป็นระดับที่มากขึ้นความละเอียดของการหมุนของสเต็ปปิ้งมอเตอร์ก็จะมีค่ามากขึ้น ข้อดีของการขับสเต็ปปิ้งมอเตอร์แบบแบ่งละเอียด คือ สามารถควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ให้มีความละเอียดสูงได้และเหมาะกับงานที่ต้องการความละเอียดสูง แต่การขับด้วยวิธีนี้ทำได้ยากกว่าแบบเต็มสเต็ป

บทที่ 3
พื้นฐานของไมโครคอนโทรลเลอร์

3.1 แนวความคิดพื้นฐาน

ไมโครคอนโทรลเลอร์อาจจะเป็นคำที่ไม่ค่อยคุ้นเคยมากนัก เมื่อเปรียบเทียบกับคำว่าไมโครโปรเซสเซอร์ ซึ่งมีการใช้งานแพร่หลายมากกว่า เมื่อมีการนำไปใช้งานจะต้องมีไอซี ประกอบภายนอกเพิ่มเติมเข้าไปให้ระบบที่สมบูรณ์ เช่น หน่วยความจำ และพอร์ตควบคุม เป็นต้น สำหรับไมโครคอนโทรลเลอร์ก็มีหลักการพื้นฐานเช่นเดียวกัน เพียงแต่องค์ประกอบการทำงานเหล่านั้นหลายส่วนได้รับการออกแบบให้บรรจุอยู่ภายในไอซีเพียงตัวเดียวเท่านั้น การนำไปใช้งานก็ค่อนข้างจะสะดวกเพียงการต่อคริสตอลเพื่อเป็นฐานเวลาและแหล่งจ่ายไฟให้เท่านั้น

EMBEDDED CONTROLLERS

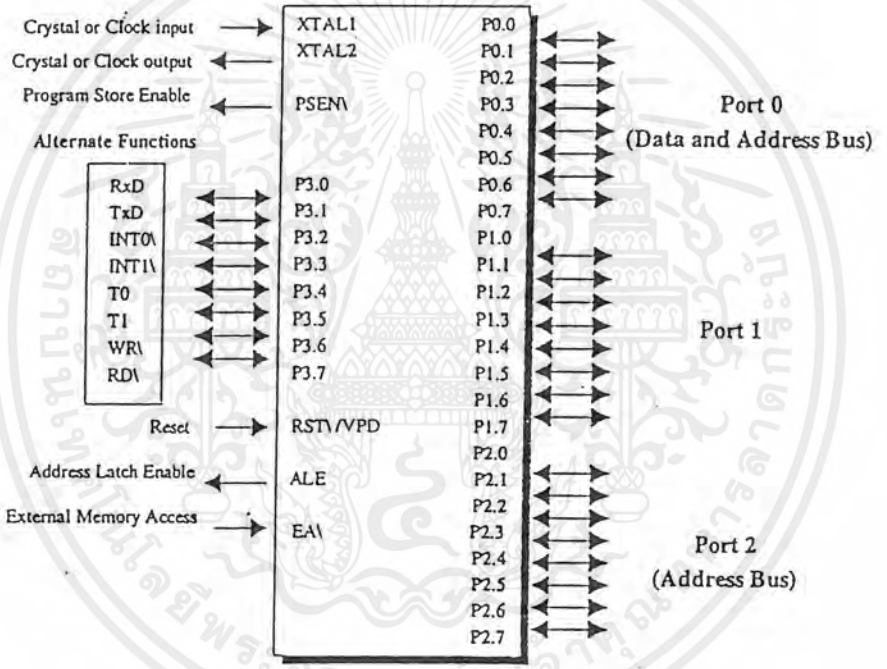
Feature	8051AH	8031AH	8751H	80C51BH	80C31BH	87C51	8052AH	8032AH	8752	8044H
Program Memory(Bytes)	4K	-	4K	4K	-	4K	8K	-	8K	4K
RAM Memory(Bytes)	128	128	128	128	128	128	256	256	256	192
Program Memory Expansion (Off Chip) (Bytes)	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K
Data Memory Expansion (Off Chip) (Bytes)	64K	64K	64K	64K	64K	64K	64K	64K	64K	64K
Max. Clock Frequency (MHz.)	12	12	12	16	16	16	16	12	12	12
Typical Instruction Time (uS.)	1	1	1	0.75	0.75	0.75	1	1	1	1
16-Bit Timer/Counter	2	2	2	2	2	2	3	3	3	2
Serial Communications	Synchronous Mode, Asynchronous Mode, 8 or 10 Bit Programmable									HDLC/ SDLC
No. of I/O Lines	32	16	32	32	16	32	32	16	32	32
Interrupt Source (Two Priority Levels)	5	5	5	5	5	5	6	6	6	5
Power Requirements (I_{CC} Max. mA.)	125	250	24	24	20	175	175	175	200	

ตารางที่ 3.1 แสดงไมโครคอนโทรลเลอร์ตระกูล MCS-51 ของบริษัทอินเทล

ลักษณะงานที่เหมาะสมกับการนำไมโครคอนโทรลเลอร์ไปใช้งาน มักจะเป็นงานประยุกต์ที่เกี่ยวข้องกับการควบคุม เช่น ระบบควบคุมภายในเครื่องใช้ไฟฟ้า เป็นต้น ซึ่งงานควบคุมเหล่านี้มักจะไม่มีการคิดคำนวณที่ซับซ้อนมากนัก และต้องการพื้นที่ของแผงวงจรควบคุมที่จำกัด

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ประกอบด้วย ไมโครคอนโทรลเลอร์หลายรุ่น ซึ่งมีสถาปัตยกรรมพื้นฐานที่เหมือนกัน เพียงแต่มีขนาดหรือจำนวนของหน่วยทำงานภายในที่แตกต่างกันออกไป

8051 มีวงจรออสซิลเลเตอร์อยู่ภายใน สำหรับการสร้างพัลส์ของสัญญาณนาฬิกา ซึ่งจะนำไปเป็นฐานเวลา หรือการกำหนดจังหวะการทำงานของหน่วยการทำงานทั้งหมดให้สอดคล้องกัน (Synchronization) โดยปกติแล้วก็มักจะทำได้โดยการใช้คริสตอลเชื่อมต่อเข้ากับขาสัญญาณ XTAL1 และ XTAL2 พร้อมกับตัวเก็บประจุ



รูปที่ 3.1 การกำหนดหน้าที่ขาสัญญาณของไอซี 8051

รูปความถี่ของสัญญาณนาฬิกาจะเรียกว่าพัลส์ (Pulse) และคาบของสัญญาณนาฬิกานี้เรียกว่า คาบเวลาออสซิลเลเตอร์ (Oscillator peroid) ค่าหนึ่งแมชชีนไซเคิล จะใช้เวลา 12 คาบเวลาออสซิลเลเตอร์ การคำนวณหาว่าเวลาที่ใช้ในการทำคำสั่งใดจนเสร็จสิ้น จะต้องดูว่าคำสั่งนั้นใช้จำนวนแมชชีนไซเคิลเป็นเท่าไรในการประมวลผล เวลาที่ใช้จะคำนวณตามสูตรสมการ 3.1

$$T = (C*12) / F \qquad (3.1)$$

โดย C เป็นค่าจำนวนแมชชีนไซเคิลของคำสั่ง

F เป็นค่าความถี่ของคริสตอลที่ใช้กับ 8051

3.2 หน่วยความจำโปรแกรม

ในระบบของไมโครคอนโทรลเลอร์ 8051 จำเป็นต้องมีหน่วยความจำสำหรับบรรจุคำสั่งหรือ โปรแกรมที่ผู้ใช้พัฒนาขึ้น ที่เรียกว่า หน่วยความจำโปรแกรม (Program Memory) โดยอาจจะประกอบอยู่ภายในตัวไอซีของ 8051 เอง หรือเป็นไอซีหน่วยความจำอีพรอมหรือรอม แยกออกต่างหากได้ ในกรณีหลังนี้จำเป็นจะต้องมีการใช้พอร์ตอินพุท/เอาท์พุท ทำหน้าที่เป็นบัสแแอดเดรสและบัสข้อมูล ส่วน 8031 นั้นไม่มีหน่วยความจำโปรแกรมอยู่ภายในตัวไอซีเลย ดังนั้นในการนำไปใช้งานจึงจำเป็นต้องอาศัยหน่วยความจำโปรแกรมภายนอกเสมอ

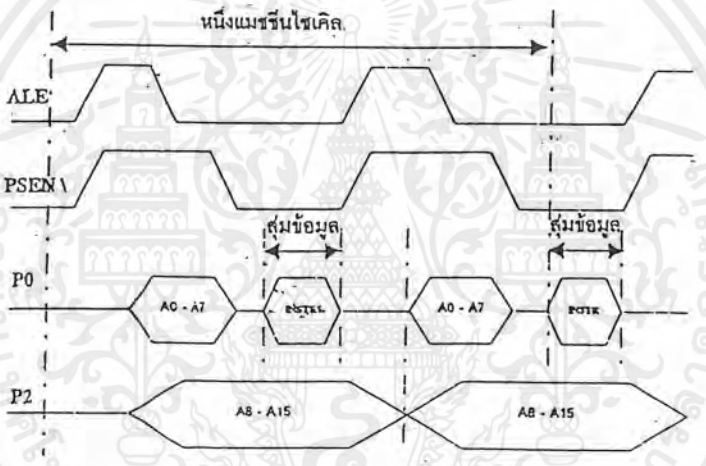
สำหรับการเชื่อมต่อหน่วยความจำแบบอีพรอมนั้น เนื่องจากระบบบัสแอดเดรสและบัสข้อมูลเป็นลักษณะแบบใช้การมัลติเพล็กซ์จากพอร์ตเดียวกัน กล่าวคือ ในระยะเวลาเริ่มต้นเส้นสัญญาณเหล่านี้ของพอร์ตจะใช้ในการส่งค่าแอดเดรสของตำแหน่งที่ต้องการติดต่อด้วย ในช่วงเวลาต่อมาจึงจะเปลี่ยนไปเป็นสภาวะอิมพีแดนซ์สูง เพื่อใช้งานในฐานะของบัสข้อมูล แต่เนื่องจากว่า อีพรอมที่ใช้งานกับทั่วไปนั้นไม่ใช้การมัลติเพล็กซ์ มีขาสัญญาณบัสแอดเดรสและบัสข้อมูลแยกออกจากกันโดยชัดเจน ดังนั้นการต่ออีพรอมเพื่อทำหน้าที่เป็นหน่วยความจำโปรแกรมจึงจำเป็นต้องมีวงจรประเภทค้างข้อมูล (Latch) ประกอบเพิ่มเติมขึ้น เพื่อทำการค้างค่าของแอดเดรสที่ส่งออกมาจาก 8051 ในช่วงเวลาแรกให้กับขาสัญญาณแอดเดรสของอีพรอมต่อไป

จากตารางที่ 3.2 แสดงให้เห็นถึงสัญญาณต่างๆของ 8051 ซึ่งนำมาใช้ในการติดต่อกับหน่วยความจำภายนอก

สัญญาณ	คำจำกัดความ	ขาสัญญาณ	หน้าที่
EA	External Access	31	เลือกประเภทหน่วยความจำในหรือนอก
ALE	Address Enable	30	สัญญาณเอาท์พุทสำหรับการแลตช์ข้อมูลแอดเดรสจากบัส
P2.0-P2.3	Port 2	21-28	เป็นข้อมูลแอดเดรสไบต์สูงของหน่วยความจำ
P0.0-P0.7	Port 0	39 -32	มัลติเพล็กสัญญาณของบัสแอดเดรสและบัสข้อมูล
PSEN	Program Store Enable	29	สัญญาณระบุการอ่านให้กับหน่วยความจำอีพรอม

ตารางที่ 3.2 สัญญาณที่ใช้ระหว่างการติดต่อเพื่ออ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

สัญญาณ EA (External Access) ใช้ในการกำหนดเลือกว่า จะอ่านข้อมูลมาจากหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไอซีไมโครคอนโทรลเลอร์เอง ซึ่งหากเป็นระดับลอจิกต่ำ จะอ่านข้อมูลมาจากหน่วยความจำโปรแกรมภายนอก จากแผนภาพเวลาการติดต่อกับหน่วยความจำโปรแกรมภายนอกของ 8051 จะเห็นว่าภายในช่วงเวลาของแมชชีนไซเคิลหนึ่งๆ นั้น พอร์ท 0 จะเป็นค่าของแอดเดรสไบต์ต่ำ (A0-A7) และเวลาต่อมาจึงจะเป็นบัสข้อมูล การส่งค่าของแอดเดรสไบต์ต่ำนี้จะอยู่ในราวช่วงเวลาขอบขาลงสัญญาณ ALE และจะยังคงอยู่จนเมื่อสัญญาณ PSEN เปลี่ยนไปเป็นระดับสัญญาณลอจิกต่ำ ดังนั้นการออกแบบวงจรจึงมักใช้สัญญาณ ALE นี้ในการทำให้ไอซีแลตซ์ภายนอกค้างระดับสัญญาณแอดเดรสเหล่านี้ไว้ ส่วนสัญญาณ PSEN จะใช้ในการเลือกให้อีพรอมทำงานและอ่านค่าข้อมูลกลับมา

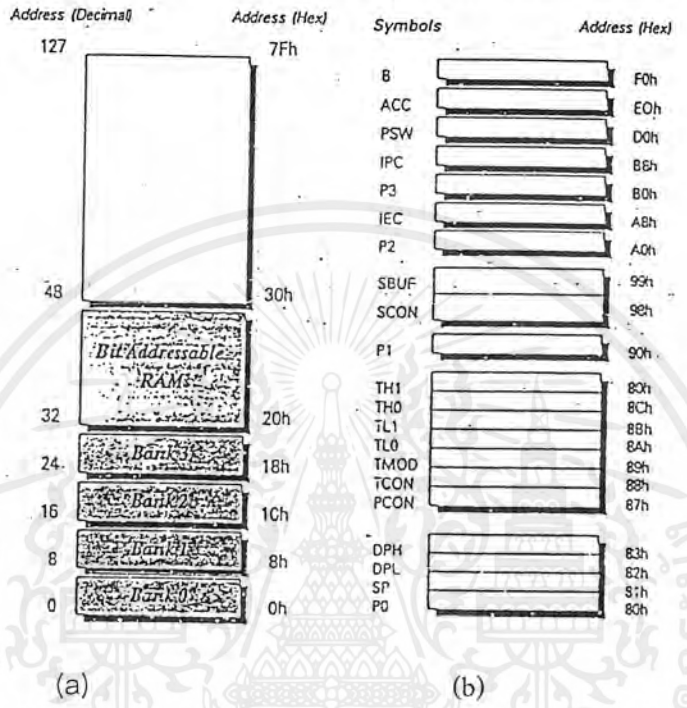


รูปที่ 3.2 แผนภาพสัญญาณเวลาแสดงการติดต่อกับหน่วยความจำโปรแกรมภายนอก

3.3 หน่วยความจำข้อมูล

หน่วยความจำข้อมูลมีหน้าที่สำหรับเก็บข้อมูล หรือตัวแปรที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว โดยพื้นฐานแล้ว หน่วยความจำข้อมูลจัดเป็นหน่วยความจำแรมแบบสแตติก ดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบ ก็จะมีผลทำให้ข้อมูลที่จัดเก็บไปไว้ภายในหน่วยความจำนี้สูญหายไป พื้นที่ของหน่วยความจำข้อมูลนี้สามารถมีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทออกเป็นสองลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน (Internal Data Memory) และหน่วยความจำข้อมูลภายนอก (External Data Memory) ซึ่งเป็นการใช้ไอซีหน่วยความจำแรมมาเพิ่มเติมเข้าไปในวงจร

หน่วยความจำข้อมูลภายในมีจำนวนทั้งหมด 256 ไบต์ โดยจำแนกออกได้เป็นสองลักษณะ คือ พื้นที่เฉพาะสำหรับตัวประมวลผลกลางใช้งานเท่านั้น และพื้นที่ใช้งานทั่วไป จากรูปที่ 3.3 แสดงให้เห็นถึงการจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในของ 8051 ดังนี้



รูปที่ 3.3 การจัดพื้นที่หน่วยความจำข้อมูลภายใน

(a)ช่วงตั้งแต่แอดเดรส 00-7FH

(b)ช่วงแอดเดรส 80-FFH ซึ่งกำหนดให้เป็นพื้นที่ของริจิสเตอร์ใช้งานพิเศษ

3.4 พอร์ทอินพุท/เอาท์พุท

พอร์ต มีความหมายถึง แอดเดรสหนึ่งที่ได้รับการกำหนดไว้เพื่อการโอนย้ายข้อมูลระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอก การกำหนดประเภทของการติดต่อขึ้นอยู่กับทิศทางการไหลของข้อมูลเมื่อพิจารณาจากไมโครคอนโทรลเลอร์เป็นหลัก ดังนั้นการนำข้อมูลจากวงจรภายนอกจึงเรียกว่า การอินพุท และในกรณีตรงกันข้ามเพื่อส่งออกข้อมูลก็จะเรียกว่าการเอาท์พุท

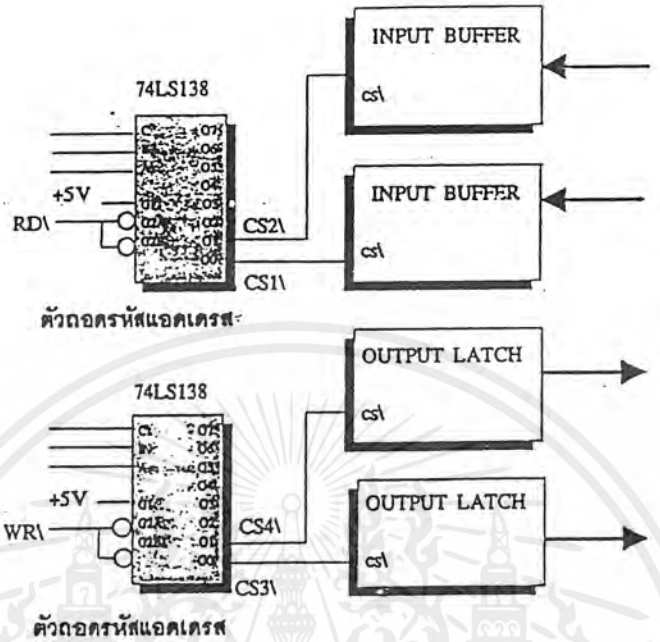
วิธีการส่งข้อมูลภายในพอร์ทจะสามารถแยกประเภทของพอร์ทออกได้เป็นสองลักษณะคือ พอร์ตแบบขนาน (Parallel port) ซึ่งทำการส่งจำนวนบิทข้อมูลทั้งหมดออกมาหรือนำเข้าไปพร้อมกันในคราวเดียว และ พอร์ตแบบอนุกรม (Serial port) ซึ่งทำการโอนย้ายข้อมูลคราวละบิตๆ จนครบจำนวน

ดังได้ทราบมาแล้วว่า เมื่อมีการเพิ่มเติมหน่วยความจำพายนอก จะต้องนำพอร์ต 0 และพอร์ต 2 ทั้งหมดไปใช้งาน ดังนั้นหากมีความต้องการที่จะให้มีจำนวนของพอร์ตเท่าเดิม ก็จำเป็นจะต้องนำไอซีพื้นฐานอื่นๆ เพิ่มเติมเข้าไปในการออกแบบเพื่อทำงานเป็นพอร์ตให้กับระบบแทน เช่น ไอซีบัฟเฟอร์สามสถานะ (Tri-state buffers) สำหรับพอร์ตอินพุท หรือไอซีแลตซ์ สำหรับพอร์ตเอาท์พุท เป็นต้น สำหรับสัญญาณที่ควบคุมหรือเลือกให้ไอซีเพิ่มเติมเหล่านี้ทำงานสามารถใช้เทคนิควิธีได้หลายลักษณะ เช่น อาจเป็นการใช้เส้นแอดเดรสเดี่ยวเพื่อกำหนดสภาวะลอจิกให้ถูกต้อง หรืออาจจะใช้การถอดรหัสแอดเดรสก็ได้ ในกรณีที่เป็นการถอดรหัสแอดเดรสโดยใช้ไอซีถอดรหัส (เช่น 74LS138) ก็มักจะใช้เส้นสัญญาณ WR ควบคุมการทำงานของตัวถอดรหัส

การเพิ่มเติมพอร์ตอินพุท โดยการนำไอซี TTL 74LS244 มาใช้งานเพื่อทำหน้าที่เป็น บัฟเฟอร์สำหรับข้อมูลนำเข้ามายังระบบ โดยขาสัญญาณ OE1 และ OE2 เชื่อมต่อเข้ากับขาสัญญาณควบคุม RD ของ 8051

การเพิ่มเติมพอร์ตเอาท์พุท สามารถกระทำได้ด้วยวิธีการที่คล้ายคลึงกันกับการเพิ่มเติมพอร์ตอินพุท โดยการเพิ่มไอซี 74LS138 ซึ่งทำหน้าที่เป็นตัวแลตซ์ หรือค้างค่าข้อมูลที่ต้องการส่งออกจากบัสของระบบ โดยนำขาสัญญาณ CLK มาเชื่อมต่อกับสัญญาณ WR ของ 8051

ในกรณีที่ต้องการขยายพอร์ตจำนวนมากกว่าหนึ่งพอร์ตขึ้นไป ก็สามารถทำได้ในลักษณะที่คล้ายคลึงกัน สำหรับการสร้างสัญญาณตามแผนภาพในรูปที่ 3.4 เป็นอีกลักษณะหนึ่งซึ่งใช้วิธีการแบบถอดรหัสแอดเดรส (Address Decoder) เพื่อถอดรหัสแอดเดรสสำหรับสร้างเป็นสัญญาณเลือกอุปกรณ์ โดยนำบัสแอดเดรสมาต่อเข้ายังอินพุทของไอซี 74LS138 ซึ่งเอาท์พุทที่ได้ก็จะเป็นสัญญาณเลือกอุปกรณ์ให้กับบัฟเฟอร์แต่ละตัว การเชื่อมต่อสัญญาณ RD\ หรือ WR\ ในลักษณะเช่นนี้เพื่อควบคุมให้ตัวถอดรหัส แอดเดรสทำการสร้างสัญญาณเอาท์พุท เมื่อต้องการจะติดต่อกับพอร์ตเท่านั้น



รูป 3.4 แผนภาพวงจรบางส่วน แสดงการเพิ่มจำนวนพอร์ต โดยใช้วิธี การถอดรหัสแอดเดรส

3.5 เครื่องมือช่วยการพัฒนาระบบ

การใช้งานไมโครคอนโทรลเลอร์ให้ได้ผลอย่างมีประสิทธิภาพนั่น นอกเหนือไปจากจะต้องมีความเข้าใจในวงจรหรือหน่วยทำงานของระบบเพื่อประโยชน์ในด้านของการออกแบบแล้ว เรื่องของการพัฒนาซอฟต์แวร์หรือที่เรียกว่า โปรแกรม นั้นก็มีความสำคัญมากเช่นกัน ลำดับขั้นตอนที่จะต้องใช้ในการพัฒนาโปรแกรมสำหรับระบบไมโครคอนโทรลเลอร์ 8051 สามารถสรุปเป็นลำดับขึ้นตอนการทำงานได้ดังนี้

ขั้นตอนที่ 1 การออกแบบขั้นต้น

ขั้นตอนแรกนี้จะเป็นการศึกษาหรือทำความเข้าใจในระบบที่เราต้องการออกแบบว่าควรจะต้องสร้างหรือนำวงจรอย่างใดอย่างหนึ่งมาประกอบเข้ากับระบบ 8051 ขั้นตอนนี้มักจะเขียนออกมาเป็นแผนภาพร่างของวงจร และแผนภาพแบบบล็อกเท่านั้น

ขั้นตอนที่2 ตรวจสอบการออกแบบขั้นต้น

การตรวจสอบการออกแบบในขั้นต้นนั้น สามารถกระทำได้หลายลักษณะ เช่น อาจจะเป็นการตรวจสอบลอจิกการทำงานของแผนภาพวงจรบนแผ่นกระดาษ หรืออาจจะลงมือสร้างวงจรต้นแบบ (Prototype) เฉพาะในส่วนของวงจรอินเตอร์เฟสที่จะต้องสร้างขึ้นเพิ่มเติมจากระบบไมโครคอนโทรลเลอร์ปกติ โดยใช้แผ่นวงจรทดสอบอเนกประสงค์ (Protoboard)

ขั้นตอนที่3 การพิจารณาทางด้านฮาร์ดแวร์และซอฟต์แวร์

มีจุดมุ่งหมายเพื่อที่จะหาข้อตัดสินใจ ว่าวงจรที่ทดสอบขั้นต้นนั้นนำไปใช้ในวงจรจริงหรือจะปรับเปลี่ยนไปใช้วงจรรวมอื่นๆ มาทำหน้าที่แทน การพิจารณานี้มักจะมีองค์ประกอบอื่นที่นำมาใช้ประกอบการตัดสินใจ เช่น จำนวนของอุปกรณ์ที่ใช้งาน ความซับซ้อนของวงจร หรือ งบประมาณที่ใช้ เป็นต้น การพิจารณานี้มีผลต่อการดำเนินการด้านซอฟต์แวร์ด้วย

ขั้นตอนที่4 การทดสอบและพัฒนาทางด้านซอฟต์แวร์และฮาร์ดแวร์

เป็นการดำเนินการสร้าง ทดสอบ แก้ไข และพัฒนาตามการออกแบบที่ดำเนินการในขั้นตอนที่ผ่านมาข้างต้น ซึ่งอาจจะต้องกระทำควบคู่ไปพร้อมกันทั้งสองด้าน อย่างไรก็ตาม โดยทั่วไปแล้วงานด้านการพัฒนาซอฟต์แวร์ก็มักจะใช้เวลานานกว่าด้านฮาร์ดแวร์มาก

ขั้นตอนที่ 5 ปรับปรุงการออกแบบ

เป็นการดำเนินการเพื่อปรับเปลี่ยนการออกแบบที่ได้ดำเนินการมาอีกครั้งหนึ่งจากขั้นตอนที่สาม ซึ่งในบางครั้งอาจจะมีความจำเป็น ที่จะต้องปรับปรุงประสิทธิภาหรือเพิ่มสมรรถนะของการทำงาน

ขั้นตอนที่6 การทดสอบระบบโดยรวม

การทดสอบในขั้นตอนนี้เป็นการรวมวงจรและโปรแกรมส่วนต่างๆ ประกอบเข้าด้วยกัน ซึ่งเดิมอาจจะแยกการพัฒนาเป็นส่วนๆ การดำเนินการขั้นตอนนี้มักจะมีความซับซ้อนและใช้เวลาในการทดสอบมากที่สุด ซึ่งหากว่าผ่านการทดลองนี้ไปได้ ก็เป็นการเสร็จสิ้นกระบวนการออกแบบ

ขั้นตอนที่7 การประเมินผลสมรรถนะการทำงาน

โดยมากแล้วขั้นตอนสุดท้ายนี้ก็มักจะใช้กับระบบที่นำไปผลิตจำหน่าย ซึ่งนอกเหนือไปจากทำงานได้ตามความต้องการแล้ว ยังต้องการประสิทธิภาพในการทำงานอีกด้วย ในกรณีที่มีการออกแบบที่ดีแล้ว การปรับปรุงประสิทธิภาพของระบบในขั้นตอนนี้ก็ควรจะเป็นเฉพาะทางด้านซอฟต์แวร์

บทที่ 4

การสื่อสารระหว่างอุปกรณ์ภายนอกกับคอมพิวเตอร์ผ่านพอร์ทอนุกรม

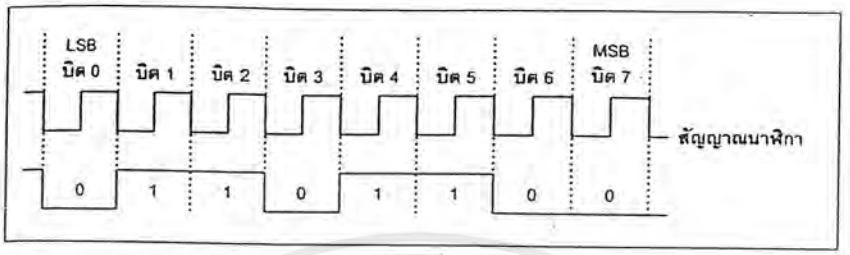
การที่จะเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงอื่น ๆ หรือคอมพิวเตอร์ด้วยกัน มีอยู่ 2 วิธี นั่นคือการรับส่งข้อมูลแบบขนานและการรับส่งข้อมูลแบบอนุกรม การรับส่งข้อมูลแบบขนานจะเป็นการรับหรือส่งข้อมูลคราวละ 4 หรือ 8 บิตในเวลาเดียวกัน ซึ่งจะทำให้การรับและส่งข้อมูลทำได้ที่ความเร็วสุง ซึ่งก็หมายความว่าจำนวนของสายที่ใช้ในการส่งจะต้องมีมากเท่ากับจำนวนบิตของข้อมูลที่จะส่งด้วย นอกจากนี้ยังจะต้องรวมถึงสายที่ใช้สำหรับการควบคุมและการตรวจสอบการรับส่งข้อมูลด้วย ซึ่งอาจจะต้องใช้สายมากเป็น 2 เท่าของจำนวนบิตข้อมูลที่จะส่งก็ได้ ซึ่งก็เป็นปัญหาในเรื่องราคาของสายที่ใช้ในการเชื่อมต่อแบบขนานมักจะมีราคาแพง

ในขณะที่การรับส่งข้อมูลแบบอนุกรมเป็นการรับส่งข้อมูลครั้งละ1 บิต แต่ก็สามารถรับส่งข้อมูลได้คราวละหลาย ๆบิตได้ หากแต่จะต้องมีการตกลงกันระหว่างตัวสงและตัวรับว่า จะรับส่งข้อมูลคราวละกี่บิต ตัวรับจะต้องรอข้อมูลมาให้ครบทุกบิตเสียก่อนจึงทำการประมวลผลส่งผลให้การสื่อสารข้อมูลอนุกรมอาจมีความเร็วต่ำกว่าแบบขนาน ในด้านจำนวนสายสัญญาฯการรับส่งข้อมูลแบบอนุกรมจะใช้จำนวนสายที่น้อยกว่ามาก อย่างน้อยที่สุดใช้ เพียง 2-3 เส้นเท่านั้นแต่อัตราเร็วในการรับส่งข้อมูลอาจต่ำกว่าแบบขนาน อย่างไรก็ตามการรับส่งข้อมูลแบบอนุกรมสามารถใช้สายสัญญาณที่มีความยาวมากกว่าแบบขนาน อย่างไรก็ตามการรับส่งข้อมูลแบบอนุกรมสามารถใช้สายสัญญาณที่มีความยาวมากกว่าแบบขนาน ทำให้ระยะทางในการสื่อสารข้อมูลแบบอนุกรมสามารถทำได้มากกว่า

4.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมนั้นจุแบ่งออกได้เป็น 2 แบบคือการสื่อสารอนุกรมแบบซิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อ

กันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้น คือ สัญญาณนาฬิกา ข้อมูล และกราวด์



รูปที่ 4.1 แสดงให้เห็นถึงไทมิ่งไดอะแกรมของการส่งข้อมูลแบบซิงโครนัส

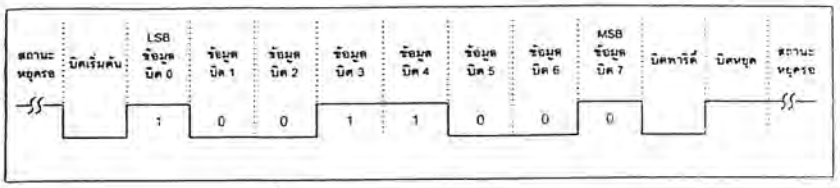
4.2 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยเหมือนกับการรับส่งข้อมูลแลลซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูล หรือ บอดเรต มีหน่วยเป็น บิตต่อวินาที

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น ซึ่งจะมีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมจะมีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ จะมีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้าย จะมีขนาด 1,1.5 หรือ 2 บิต

รูปที่ 4.2 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส ซึ่งเมื่อไม่มีข้อมูลที่จะส่ง DATA จะมีสถานะ ลอจิก 1 ซึ่งจะเรียกสถานะนี้ว่าสถานะหยุดรอ การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก 0 ด้วยช่วงระยะเวลา 1 บิต ซึ่งจะเรียกบิตนี้ว่าบิตเริ่มต้นจากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด ก่อนซึ่งข้อมูลในไบต์ที่จะส่งอาจจะมีจำนวนบิต 5 ,6,7 หรือ 8 บิตก็ได้ จากนั้นจะตามด้วยบิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้าย ซึ่งจะให้ขาดาต้ามีสถานะลอจิก 1 อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต , 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว



รูปที่ 4.2 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรมแบบอะซิงโครนัส

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและส่งข้อมูลแบบอะซิงโครนัสเรียกว่า UART อัตราความเร็วการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสคือค่า บอดเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับและส่งข้อมูล บอดเรตมาตรฐานที่ใช้สำหรับพอร์ตอนุกรม RS-232 ได้แก่ 110,150,300,600,1200,2400,4800,9600 และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีรของคอมพิวเตอร์ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มอาจจะสามรถกำหนดค่าบอดเตรได้สูงถึง 115200 บิตต่อวินาที เน่องจากบอดเรตคือจำนวนบิตของข้อมูลที่สามรถถ่ายทอดได้ภายใน 1 วินาที ยกตัวอย่าง ข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1บิต และบิตปิดท้าย 1 บิตความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้บอดเตรในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาริตี้ความเร็วในการรับส่งข้อมูลจะเหลือเป็น 872 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd) , แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบพาริตี้เป็นการตรวจสอบจำนวนรวมของบิตที่เป็นลอจิก "1" ภายในขอมูลที่ส่งไป 1 ไบต์ว่ามีจำนวนรวมเป็นเลขคู่หรือเลขคี่โดยต้องรวมบิตพาริตี้เข้าไปด้วย ยกตัวอย่างข้อมูลที่จะทำการส่งมีขนาด 8 บิตและมีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสอง จะเห็นว่าข้อมูลในไบต์นี้จะมีจำนวนลอจิก "1 " จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดเลขพาริตี้เป็นคู่ค่าในบิตพาริตี้จะมีค่าเป็น "0 " แต่ถ้าพาริตี้เป็นคี่ค่าที่บิตพาริตี้จะมีค่าเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้มีจำนวนบิตที่เป็นลอจิก "1" มีจำนวนรวมกันเป็นเลขคี่ ในตารางที่ 4.1 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	0	1
11111111	1	0

ตารางที่ 4.1 แสดงบิตพาริตี้ของข้อมูล

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลบของ UART ซึ่งทางภาครับจะต้องทำการกำหนดคุณสมบัติการตรวจสอบพาริตี้ให้ตรงกันว่าจะตรวจสอบพาริตี่คู่หรือพาริตี้คี่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดบิตพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาว่าเป็นตัวเลขคี่ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ นับเป็นการตรวจสอบข้อผืดพลาดที่เกิดขึ้นในการถ่ายทอดข้อมูลที่ง่ายที่สุด แต่จะเชื่อถือได้เมื่อการส่งข้อมูลมีบิตผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งทางภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ UART เบอร์ 8250 UART ชิปเหล่านี้มีระดับแรงดันเป็นแบบทีทีแอล(0 และ +5 V) แต่เพื่อให้มีแรงดันเป็นไปตามมาตราฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะไกลมากขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงแรงดันที่สูงขึ้น โดยลอจิก "0" มีระดับแรงดัน +3V ถึง +12V ในขณะที่ลอจิก "1" มีระดับแรงดัน –3V จนถึง –12V

4.3 มาตราฐานพอร์ทอนุกรมแบบ RS-232

มาตราฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตราฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตราฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจาก

โมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดที่อยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า *สมาคมอุตสาหกรรมอิเลกทรอนิกส์* (Electronic Industries Association : EIA) ได้วางมาตราฐานที่มีชื่อว่า EIA RS-232 มาตราฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3 ถึง –12 V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12V แสดงว่าเป็นช่องว่าง (Space)

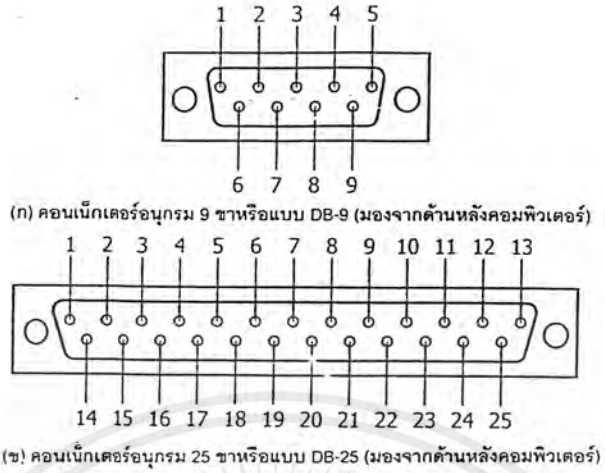
มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (DataCircuit Terminating : DCE) ไว้ว่าอุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้มูลระหว่างอุปกรณ์ทั้งจะกระทำผ่านมาตราฐาน RS-232

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ทอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็น DCE

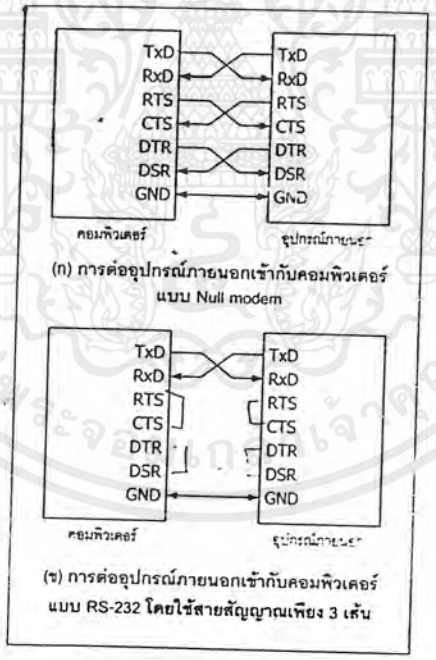
สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลที่ความยาวสายสูงสุด 20 เมตร

4.4คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตราฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือDB - 9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9. เนื่องจากขาอื่นๆที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขา ในรูปที่ 4.3



รูปที่ 4.3 การจัดขาของคอนเน็กเตอร์พอร์ทอนุกรมตามมาตราฐาน RS-232 ทั้งแบบ DB9 และแบบ DB-25



รูปที่ 4.4 การต่ออุปกรณ์ภายนอกกับพอร์ทอนุกรมของคอมพิวเตอร์ในลักษณะต่างๆ

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกดังแสดงในรูปที่ 4.4 ลูกศรในรูปแสดงถึงทิศทางของข้อมูล ในรูปที่ 4.4(ก) เป็นการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็คเต็มรูปแบบ ส่วนในรูปที่ 4.4(ข) เป็นการ

เชื่อมต่อแบบ Null modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นหนึ่งสำหรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่หน้าที่การทำงานในแต่ละขาพอร์ตอนุกรม RS-232 มีดังนี้

- **Data Carier Detect : DCD** หรืออาจเรียกรวมว่า Carier Detect : CD ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์การสื่อสารข้อมูลเช่น โมเด็ม สำหรับการใช้งานปกติขานี้จะไม่ได้ถูกใช้งานมากนัก

- **Receive Data : RD** หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์ บัฟเฟอร์

- **Data terminal Ready : DTR** เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์ เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วย โดยขา DRT นี้จะต้องเชื่อมต่อกับขา DSRของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้ในการเชื่อมต่อเป็นแบบ Null modem ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

- **Serial Ground : GND** ขากราวด์ของระบบ

- **Data Set Ready : DSR** ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

- **Request To Send : RTS** เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยที่รับสัญญาณ RST ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null modem 3 สาย จะต้องเชื่อมต่อ ขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

- **Clear To Send : CTS** ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่

- **Ring Indicator : RI** ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

4.5 UART

UART มาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั้นเองสำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

หน้าที่หลักของ UART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัส แล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังแจ้งข้อมูลอื่น ๆ ให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล รูปแบบการส่งข้อมูล ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูลเป็นต้น

ภายใน UART จะมีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้มีขนาด 16 บิตดังนั้นจึงสามารถกำหนดตัวหารอยู่ในช่วง 1 – 65535 UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ และ ฟูลดูแพล็กซ์ โดยการส่งแบบฮาล์ฟดูเพล็กซ์เป็นการส่งแบบทิศทางเดียว ส่วนการส่งแบบฟูลดูเพล็กซ์นั้นสามารถรับและส่งข้อมูลได้ในคราวเดียวกัน

4.5.1 ชนิดของ UART

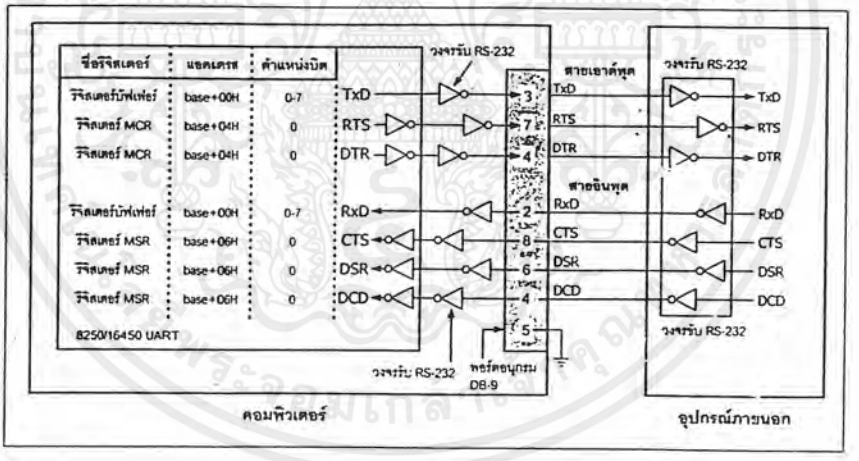
ในเครื่องคอมพิวเตอร์ทั่วไปมี UART ที่ใช้งานกันอยู่ 2 เบอร์คือ 8250 ซึ่งเป็น UART มาตรฐานที่มีใช้กันมายาวนาน UART เบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตำแหน่งเดียวกัน ทำให้การรับและส่งข้อมูลถูกจำกับความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่ UART เบอร์นี้ก็ถือว่าเป็นต้นแบบของ UART ที่ใช้ในคอมพิวเตอร์โดยคอมพิวเตอร์ทุก ๆ รุ่นจะต้องสนับสนุนการทำงานตามรูปแบบของ UART เบอร์นี้

UART อีกเบอร์หนึ่งคือ 16450 มีความสามารถรับส่งข้อมูลที่ได้ความเร็ว 115200บิตต่อวินาที และเพิ่มรีจิสเตอร์สำหรับพักข้อมูลสำหรับ UART นอกจากนั้นยังเพิ่มส่วนของชิฟต์รีจิสเตอร์แบบ FIFO ขนาด 16 ไบต์เข้าไป ทำให้สามารถสนับสนุนความเร็วในการรับส่งข้อมูลที่ 256 กิโลบิตต่อวินาทีได้ โดยคอมพิวเตอร์ในปัจจุบันใช้ UART เบอร์นี้หรือใหม่กว่า เช่น เบอร์ TL 16C750 ซึ่งมีรีจิสเตอร์แบบ FIFO ขนาด 64 ไบต์ ทำงานได้ที่ระดับแรงดัน +5 V และ+3V มีโหมดประหยัดพลังงาน สามารถรับส่งข้อมูลได้ที่ความเร็ว 1 เมกะบิตต่อวินาทีเมื่อใช้สัญญาณนาฬิกา 16 MHz

อย่างไรก็ตาม ความเร็วในการส่งข้อมูลที่มากมายของ UART เบอร์ใหม่ ๆ ก็ไม่ได้ช่วยให้การรับส่งข้อมูลของคอมพิวเตอร์เร็วขึ้น เนื่องจากว่าคอมพิวเตอร์ยังใช้ความถี่ของสัญญาณนาฬิกาในการแปลงข้อมูลเพียง 1.8432 MHz เท่านั้น

4.6 ลักษณะสัญญาณอินพุตและเอาต์พุตของพอร์ต RS-232

สัญญาณเอาต์พุตที่ใช้ควบคุม และสัญญาณแสดงสถานะอินพุต ของพอร์ตอนุกรม RS-232 จะถูกส่งกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและรับจไม่ถูกส่งออกมาจาก UART จึงต้องส่งเข้าสู่วงจรขับเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรขับที่ใช้ทั้งภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะ ดังแสดงเป็นบล็อกไดอะแกรมใน



รูปที่ 4.5 ไดอะแกรมโครงสร้างทางฮาร์ดแวร์ของพอร์ทอนุกรม

4.8 แอดเดรสของพอร์ตอนุกรม

แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแหน่งดังนี้คือ

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหมด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรม ไบออสจะทำแอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับพอร์ตอนุกรม COM1 จะเก็บไว้ที่แอดเดรส 0000:0400H และ 0000:0400H ส่วนตำแหน่งอื่น ๆ มีรายละเอียดดังนี้

COM 2 = 0000:0402H – 0000:0403H

COM 3 = 0000:0404H – 0000:0405H

COM 4 = 0000:0406H – 0000:0407H

นอกจากนี้ที่หน่วยความจำแอดเดส 0000:0411H ยังใช้สำหรับแสดงจำนวนของพอร์ตอนุกรมที่มีใช้อยู่ในคอมพิวเตอร์อีกด้วย โดยมีรายละเอียดดังแสดงในตารางที่ 4.2

บิต 3	บิต 2	บิต 1	จำนวนพอร์ท
0	0	0	ไม่มีพอร์ทอนุกรม
0	0	1	มีพอร์ทอนุกรม 1 พอร์ท
0	1	0	มีพอร์ทอนุกรม 2 พอร์ท
0	1	1	มีพอร์ทอนุกรม 3 พอร์ท
1	0	0	มีพอร์ทอนุกรม 4 พอร์ท

ตารางที่4.2 แสดงข้อมูลในแอดเดรส 0000: 0411H ที่ใช้แจ้งจำนวนพอร์ทอนุกรม

4.9 การกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรม

ก่อนการใช้งานพอร์ตอนุกรมนั้นจะต้องมีการกำหนดค่าเริ่มต้นให้กับตัวมันก่อน ซึ่งก็คือการกำหนดจำนวนบิตข้อมูลที่ต้องการส่ง,จำนวนบิตปิดท้าย, ชนิดของพาริตี้ที่ใช้ และบอดเรต

การกำหนดสามารถทำได้หลายวิธี วิธีแรกเป็นการกำหนดจากดอสพร้อมพ์ โดยใช้คำสั่ง MODE :ซึ่งมีวิธีการใช้งานดังนี้

MODE COMm : baud=b , parity=p , data=d, stop=s, retry=r

หรือ MODE COMm :b , p , d , s , r

4.10 การรับและส่งข้อมูลแบบอนุกรมจากเครื่องคอมพิวเตอร์

มีหลากหลายวิธีในการรับและส่งข้อมูลแบบอนุกรมผ่านพอร์ตอนุกรม RS-232 เช่น ใช้คำสั่งพิมพ์ออกทางเครื่องพิมพ์ เรียกอินเตอร์รัปต์ของไบออสหรือของดอส การเขียนหรืออ่านไปยังแอดเอรสของพอร์ตโดยตรง วิธีสุดท้ายเป็นวิธีที่มีความยืดหยุ่นในการใช้งานที่สุด ยกตัวอย่าง ถ้าต้องการส่งข้อมูลไปยังพอร์ตอนุกรม COM1 สามารถเขียนข้อมูลโดยตรงไปที่รีจีสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล โดยใช้คำสั่ง QBASIC ง่าย ๆ ดังนี้

OUT &H3F8 , X

ค่า X ในที่นี้หมายถึงข้อมูลที่ต้องการส่ง มีขนาด 8 บิต

สำหรับการอ่านข้อมูลจากพอร์ตอนุกรม จะเป็นการอ่านข้อมูลมาจากรีจีสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (แอดเดรส 3F8H เช่นเดียวกัน) ซึ่งสามารถเขียนโปรแกรมง่าย ๆ ได้ดังนี้

Y = INP(&H3F8)

ค่า Y ในที่นี้คือค่าที่อ่านได้จากรีจีสเตอร์บัฟเฟอร์สำหรับรับข้อมูล โดยมีขนาด 8 บิต

สำหรับการเขียนโปรแกรมด้วย TURBO PASCAL ก็สามารถใช้คำสั่ง

PORT[$3F8] = X

สำหรับการเขียนข้อมูลไปยังพอร์ตอนุกรมและ

Y = PORT[$3F8]

สำหรับการอ่านข้อมูลจากพอร์ตอนุกรม

แต่เมื่อใช้คำสั่งนี้ในขณะที่โปรแกรมทำงานผ่านระบบปฏิบัติการวินโดวส์ จะไม่สามารถใช้งานได้เนื่องจากระบบปฏิบัติการวินโดว์ ได้เข้าฝังตัวพอร์ตอนุกรมเข้าเป็นส่วยหนึ่งของระบบปฏิบัติการแล้ว ดังนั้นการเรียกใช้งานจึงจำเป็นต้องเรียกผ่านเครื่องมือที่ติดต่อผ่านระบบปฏิบัติการ เช่นการใช้คอนโทรล MSCOMM321.OCX ของโปรแกรม Visual BASIC

คอนโทรล MSComm

MSComm จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier Detect (DCD) หรือขา Request To Send (RTS) เหตุการณ์ Oncomm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่สองเป็นการคอยตรวจสองค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในฟังก์ชันต่าง ๆ ไปเรียบร้อยแล้วซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSComm 1 ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบคุมพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel

ถึงแม้ว่า คอนโทรล MSComm จะมีคุณสมบัติ (property) มากมาย แต่สามารถทำความเข้าใจได้ไม่ยาก

Properties

เหมื่อนการคอนโทรลอื่น ๆ MSComm มี associated properties การเปลี่ยนตำแหน่งพอร์ต การส่งผ่านข้อมูล การใช้ สัญญาณ handshaking การคอนโทรลที่ชี้ properties เฉพาะเอกสารของ Visual Basic รวมถึง อักขระและรายละเอียดอื่น ๆ ที่จำเป็นในการใช้ Properties

Configuring

CommID ส่งค่ากลับไปเก็บในอุปกรณ์ indentifie

CommPort กำหนดค่า และ ส่งกลับไปยัง port number

InBufferSize กำหนดค่า และ ส่งกลับไปยัง receive buffer's size

InputLen กำหนดค่าและส่งกลับไปยัง number of characters Input เพื่ออ่านข้อมูล

InputMode กำหนดค่าและส่งแต่ละชนิดของข้อมูลกลับโดย Input และการยอมรับโดย Output

NullDiscard ให้ความจำกัดความ โดย Null characters จะถูกส่งจาก พอร์ตที่มีการรับ buffer และ drop

OutBufferSize กำหนดค่าและส่งกลับ the transmit buffer's size

ParityReplace กำหนดค่าและส่งกลับ ตัวอักษรที่ซึ่งแทนที่ ตัวอักษรที่ไม่ถูกต้องบน parity error

PortOpen กำหนดค่าและส่งกลับ สถานะของพอร์ต

Rthreshold กำหนดค่าและส่งกลับ ตัวเลขของ ตัวอักษร ที่จะส่งก่อน การ trigger ของ comEvReceive

Settings กำหนดค่าและส่งกลับ bit rate, parity, และ ตัวเลขของ ข้อมูล และ บิตหยุด

SThreshold กำหนดค่าและส่งกลับ จำนวนที่น้อยที่สุดของตัวอักษรใน transmit buffer ก่อน การ trigger ของ comEvSend

Transferring Data

CommEvent	ส่งค่าคืนกลับevent และ error
InbufferCount	ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับ
Input	อ่านและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ
OutputBufferCount	คืนค่าจำนวนของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่ง และสามารถใช้คำสั่งนี้เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย
Output	ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

Handshaking

Handshaking	กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์
Break	กำหนดค่าและ เคลียร์ค่า สัญญาณ break
CDHolding	คืนค่าที่สถานะของ CD
CTSHolding	คืนค่าที่สถานะของ CTS
DSRHolding	คืนค่าที่สถานะของ DSR
DTREnable	กำหนดค่า และ เคลียร์ DTR
RTSEnable	กำหนดค่า และ เคลียร์ RTS

Identification

Index	กำหนดค่าและคืนกลับ ตัวเลยที่ซึ่ง เจาะจงการคอลโทรลใน collection
Name	กำหนดการคอลโทรล
Object	คืนค่า control และ การตั้งค่าของการ control
Parent	คืนค่าจากฟอร์ม Object หรือ การ collection
Tag	กำหนดค่าและคืนค่า expression

4.10.1 **การส่ง** Text และ Binary

Visual Basic สามารถอนุญาติให้ส่งชุดของ ข้อมูล Text และ Binary ซึ่งจะแปลงข้อมูลเป็นแบบ ASCII ในการส่ง

Text Mode

สามารถเซตการทำงาน โหมด Text โดยการเซตที่ property ของ MSComm ส่วนของ ComInputModeTextสำหรับการส่งและการรับอักขระมารตรฐาน ANSI ตัวอย่างดังนี้

```
Dim SampleText as String
SampleText = 'ABC'
```

```
MSComm1.Output = SampleText
```

สำหรับการรับข้อมูลจากภายนอก ตัวอย่างดังนี้

```
Dim SampleText as String
SampleText = MSComm.Input
```

Binary Mode

สามารถเซตการทำงานในโหมด Binary โดยการเซตที่ property ส่วนของ ComnputmodBinaryโดยการเซตอาร์เรย์ ที่มีสมาชิก 2 ตัวชนิดไบต์ ตามตัวอย่างดังนี้

```
Dim ByteToSend(0 to 1) As Byte
Dim Buffer As Variant
BytesToSend(0) = &H4A
BytesToSend(1) = &H23
Buffer = BytesToSend()
MSComm.Output = Buffer
```

ในกรณีรับข้อมูล ตามตัวอย่างดังนี้

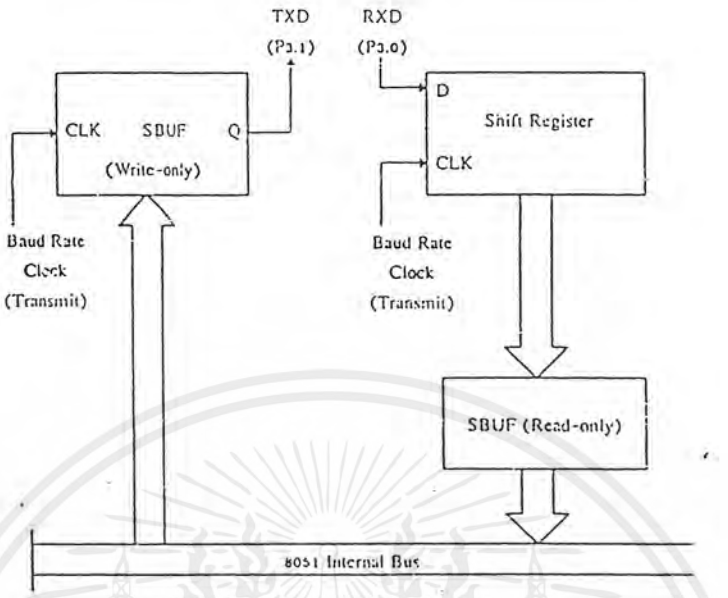
```
Dim BytesRecieved() As Byte
Dim Buffer As Variant
Buffer = MSComm1.Input
BytesReceived() = Buffer
```

4.11 MCS – 51 กับการรับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลแบบอนุกรมกับไมโครคอนโทรลเลอร์ MCS–51 นั้นภายในชิพ MCS–51 จะมี UART อยู่ในตัว ซึ่งเป็นข้อดีของไมโครคอนโทรลเลอร์ ถ้าเป็นไมโครโปรเซสเซอร์ เช่น เบอร์ Z – 80 ถ้าต้องการรับส่งข้อมูลแบบอนุกรมจะต้องนำชิพ UART มาประกอบด้วย

พอร์ทอนุกรมของ MCS – 51 จะใช้ขา TXD และ RXD ในการรับส่งข้อมูล โดยขาทั้งสองจะอยู่ในพอร์ท 3 คือ P3.1 หรือขา 11 เป็น TXD และ P3.0 หรือขา 10 เป็น RXD พอร์ทอนุกรมของ MCS – 51 สามารถทำงานแบบ Full Duplex ได้ คือสามารถส่งและรับข้อมูลในเวลาเดียวกันได้ โดยในการรับและส่งข้อมูลจะมีบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้

รีจีสเตอร์ที่สำคัญในการรับส่งข้อมูลคือ SBUF และ SCON ซึ่งเป็นรีจีสเตอร์ที่อยู่ใน Special Function Register โดยรีจีสเตอร์ Serial Port Buffer (SBUF) จะอยู่ในตำแหน่ง 99H ถ้าเขียนข้อมูลไปที่ตำแหน่งนี้ จะเป็นการส่งข้อมูลออกทางพอร์ทอนุกรม และถ้าอ่านข้อมูลจากตำแหน่งนี้จะเป็นการรับข้อมูลจากพอร์ทอนุกรม โดยใน SBUF จะประกอบด้วยบัฟเฟอร์ 2 ตัว สำหรับส่งและรับข้อมูล ดังรูปที่4.6



รูปที่4.6 การรับส่งข้อมูลระหว่างรีจิสเตอร์กับบัสภายใน

สำหรับ Serial Port Control Register (SCON) ซึ่งอยู่ที่ตำแหน่ง 98H จะเป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ รีจีสเตอร์นี้จะทำหน้าที่ควบคุมและบอกสถานะต่าง ๆ ของการรับส่งข้อมูลแบบอนุกรม

สำหรับความเร็วของการส่งข้อมูล (Baud Rate) สามารถหาได้จากการหารสัญญาณนาฬิกาที่ใช้กับ MCS – 51

4.11.1 Serial Port Control Register

MCS – 51 มีโหมดการทำงานของพอร์ทอนุกรมหลายโหมด ซึ่งสามารถโปรมแกรมโหมดการทำงานได้โดยเขียนข้อมูลไปยังรีจีสเตอร์ SCON ความหมายของแต่ละบิตแสดงดังตารางที่ 4.3 และ 4.4

ตารางที่ 4.3 บิตต่าง ๆ ของรีจีสเตอร์ SCON

บิต	ชื่อ	ตำแหน่ง	ความหมาย
SCON.7	SM0	9FH	บิตเลือกโหมดการทำงานบิต 0
SCON.6	SM1	9EH	บิตเลือกโหมดการทำงานบิต 1

SCON.5	SM2	9DH	**บิตเลือกโหมดการทำงานบิต 2**
SCON.4	REN	9CH	**บิตแฟลกกำหนดยอมให้มีการรับข้อมูล**
SCON.3	TB8	9BH	**ค่าของบิต 9 สำหรับส่งข้อมูลในโหมด 2 และ 3** สามารถ Set และ Clear ได้โดยSoftware
SCON.2	RB8	9AH	**ค่าของบิต 9 เมื่อรับข้อมูลเข้ามา**
SCON.1	TI	99H	บิตแฟลกแสดงการอินเตอร์รัพท์ภายหลังการส่งข้อมูลออกไปโดยจะ Set เมื่อส่งข้อมูลออกไปหมดแล้วและสามารถ Clear ได้โดย Software
SCON.0	RI	98H	บิตแฟลกแสดงการอินเตอร์รัพท์ภายหลังการรับข้อมูลเข้ามาและสามารถ Clear ได้โดย Software

ตารางที่ 4.4 แสดงโหมดต่าง ๆ ของการรับส่งแบบอนุกรม

SM0	**SM1**	**MODE**	**ความหมาย**	**BAUD RATE**
0	**0**	**0**	**Shift Register**	**เปลี่ยนแปลงไม่ได้ (Oscillator Frequency /12)**
0	**1**	**1**	**8- bit UART**	**สามารถเปลี่ยนแปลงได้โดยกำหนดจาก Timer**
1	**0**	**2**	**9- bit UART**	**เปลี่ยนแปลงไม่ได้ (Oscillator Frequency /12 or 64)**
1	**1**	**3**	**9- bit UART**	**สามารถเปลี่ยนแปลงได้โดยกำหนดจาก Timer**

ก่อนที่จะใช้พอร์ทอนุกรมจะต้องโปรแกรมให้กับ SCON เสียก่อนเพื่อกำหนดโหมดการทำงานและลักษณะต่าง ๆ เช่น

MOV SCON, #01010010 B

เป็นการกำหนดให้พอร์ทอนุกรมทำงานในโหมด 1 และอีนาเบิลให้มีการรับข้อมูล พร้อมกับกำหนดให้ TI เป็น 1

ในการส่งข้อมูลทุกโหมดสามารถทำได้โดยเขียนข้อมูลไปยัง SBUF เมื่อข้อมูลถูกส่งไปแล้วบิต TI จะถูกเซตเป็น " 1 " ในการส่งข้อมูล จะต้องคอยตรวจสอบบิต TI เพราะว่าถ้า TI ยังไม่เป็น " 1 " แสดงว่าข้อมูลยังส่งไปไม่หมด ถ้าหากมีการเขียนข้อมูลไบต์ต่อไปไปยัง SBUF จะทำให้เกิดข้อผิดพลาดขึ้น สำหรับในการรับข้อมูลบิต REN จะต้องเซตให้เป็น " 1 " ยกเว้นโหมด 0 เพื่ออนุญาตให้รับข้อมูลได้ เมื่อข้อมูลรับเข้ามาเรียบร้อยแล้ว บิต RI จะถูกเซตเป็น " 1 "

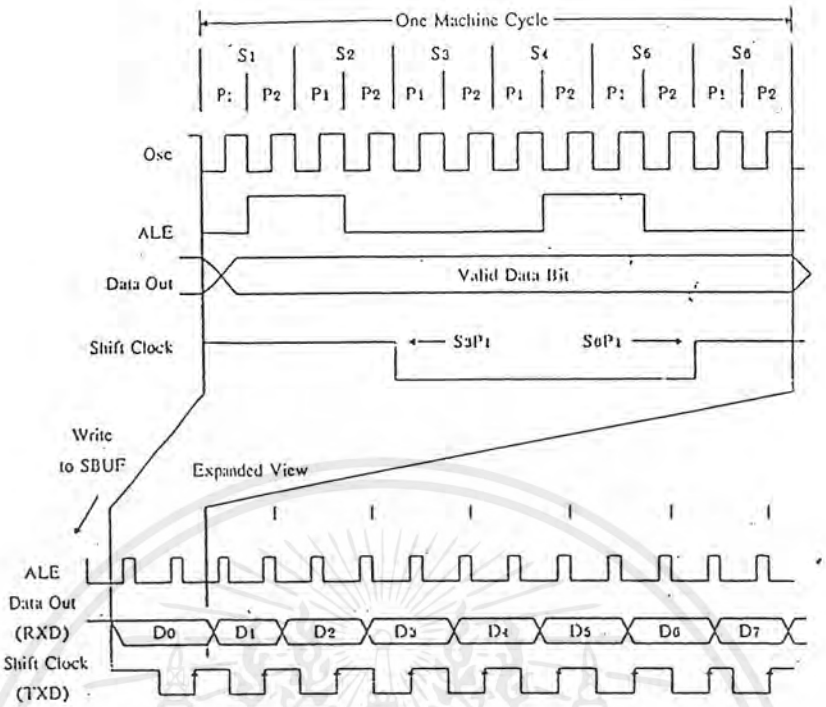
4.11.2 Mode Of Operation

ใน MCS – 51 การสื่อสารทางพอร์ทอนุกรมจะมีอยู่ 4 ประเภท หรือ 4 โหมด ซึ่งจะกำหนดได้ที่บิต SM0 และ SM1 ใน SCON โดยจะมี 3 โหมด เป็นการสื่อสารแบบ Asynchronous โดยลักษณะของข้อมูลที่ส่งจะมีบิตเริ่มต้น (Start Bit) และบิตจบ (Stop Bit) คล้ายกับการสื่อสารแบบ RS – 232 ในระบบคอมพิวเตอร์ อีกโหมดหนึ่งจะเป็นการใช้พอร์ทอนุกรมในลักษณะชีพดีรีจีสเตอร์

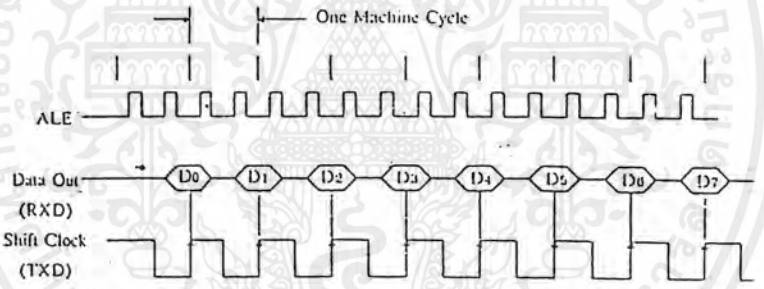
4.11.3 8 – Bit Shift Register (Mode 0)

การทำงานในโหมดนี้จะใช้ขา RXD ในการรับส่งข้อมูลโดยต่อกับ Shift Register ภายนอก ส่วนขา TXD จะเป็น Out Put Clock เพื่อกระตุ้นรีจีสเตอร์ภายนอกให้เลื่อนบิตถ้ามีการส่งข้อมูลหรือรับข้อมูล 8 บิต จะเริ่มที่บิตต่ำสุดก่อน โดยมีค่า Baud Rate เท่ากับ 1/12 ของความถี่ที่ใช้บนชิพ

ในการส่งข้อมูลจะทำโดย เขียนข้อมูลไปที่รีจีสเตอร์ SBUF ข้อมูลจะถูกส่งออกมาทางขา RXD (P3.0) โดยจะสอดคล้องกับสัญญาณที่ออกมาทางขา TXD ซึ่งสัญญาณของขา TXD จะถูกส่งออกมาทุก ๆ Machine Cycle โดยจะเป็นลอจิก " 0 " ใน S3P1 และจะกลับเป็นลอจิก " 1 " ใน S6P1 ซึ่งแสดงได้ดังรูปที่ 4.7



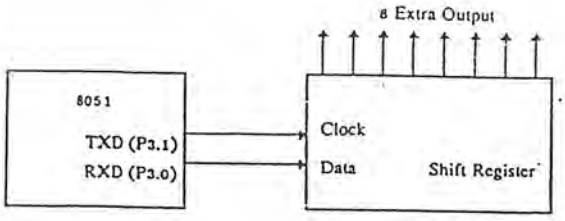
รูปที่ 4.7 ไดอะแกรมเวลาการส่งข้อมูล



รูปที่ 4.8 ไดอะแกรมเวลาการรับข้อมูล

สำหรับการรับข้อมูลจะรับได้เมื่อเซตขา Receiver Enable (REN) เป็น " 1 " และเคลียร์ขา Receiver Interrupt Bit (RI) เป็น " 0 " ข้อมูลจะเข้าสู่ MCS – 51 เมื่อ Clock Shift ถูกส่งออกไปทาง TXD ที่ขอบขาขึ้นของ Clock Shift บิตต่ำจะถูกส่งเข้ามาก่อนดังรูปที่ 4.8

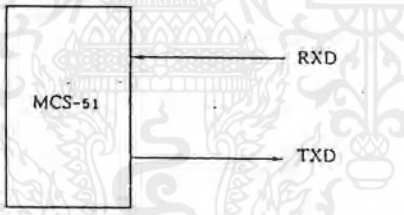
ในการประยุกต์ใช้งานโหมดนี้จะต้องมีไอซิพรีจีสเตอร์มาต่อภายนอก เช่น ถ้าหากต้องการส่งข้อมูลออกมาทางพอร์ทอนุกรม อาจต่อวงจรได้ดังรูป 4.9 โดยใช้ ไอซี Serial – To – Parallel Shift Register โดยส่งข้อมูลออกมาทาง RXD และใช้ TXD เป็น Clock



รูปที่ 4.9 การส่งข้อมูลออกโดยใช้ชิพตรีจีสเตอร์

4.11.4 8 – Bit UART With Variable Baud Rate (Mode 1)

ในโหมดนี้จะเป็นการรับส่งข้อมูลแบบ 10 บิตซึ่งประกอบด้วยบิตเริ่มต้น (เป็น "0") ข้อมูล 8 บิตและบิตจบ (เป็น " 1 ") นอกจากนี้ยังสามารถกำหนดค่า Baud Rate ได้โดยค่า Baud Rate นี้จะแปรตามตัวจับเวลาตัวที่ 1 ในโหมดนี้ จะส่งข้อมูลเข้าทาง TXD และรับข้อมูลเข้าทางRXD ถ้าเป็นการรับข้อมูลเข้าตัว Stop Bit จะเข้ามายังบิต RB8 ใน SCON



รูปที่ 4.10 การรับส่งข้อมูลในโหมด 1

ค่า Baud Rate ที่ใช้ในการรับส่งข้อมูลจะกำหนดโดย Timer 1 หลังจากโปรแกรมไปใน Timer 1 แล้วสามารถเลือกค่า Baud Rate ได้อีกสองค่าคือ ค่าจาก Timer 1 Overflow หาร 32 กับค่าจาก Timer 1 Overflow หาร 16

การส่งข้อมูลทำได้โดยการเขียนข้อมูล 8 บิตไปที่ SBUF โดยบิตที่ 9 (Stop Bit) ให้เขียนลงใน TB8 ใน SCON จากนั้นข้อมูลจะถูกส่งออกมาทางขา TXD โดยส่ง Start Bit ออกมาก่อนตามด้วยข้อมูล 8 บิต และจบด้วย Stop Bit เมื่อข้อมูลถูกส่งออกไปหมดแล้วบิต Interrupt Flag (TI)จะเป็น "1 " ดังนั้นในการเขียนข้อมูลใหม่ลงไปจะต้องตรวจสอบบิตนี้

ในการรับข้อมูล จะเริ่มจากเมื่อมีการเปลี่ยนแปลงลอจิกจาก 1 เป็น 0 ทางขา RXD หมายความว่าเริ่มรับบิตเริ่มต้น จากนั้นข้อมูลอีก 8 บิตจะถูกเก็บลงใน SBUF และ Stop Bit จะถูกเก็บในบิต RB8 ของรีจีสเตอร์ SCON เมื่อข้อมูลเข้ามาครบแล้วบิต Interrupt Flag (RI) จะถูกเซต ดังนั้นในการอ่านข้อมูลจะอ่านได้เมื่อบิต RI ถูกเซตแล้ว เมื่ออ่านข้อมูลไปแล้วจะต้องเคลียร์บิตนี้

4.11.4 9 – Bit UART With Fixed Baud Rate (Mode 2)

การทำงานในโหมดนี้ไม่สามารถกำหนดค่า Baud Rate ได้ ซึ่งค่า Baud Rate จะมีสองค่าคือ 1 / 64 และ 1 / 32 ของสัญญาณนาฬิกาบนชิพ การรับส่งข้อมูลจะเป็นชุดข้อมูล 9 บิต บิตเริ่มต้น บิตหยุด รวมเป็น 11 บิต โดยข้อมูล 9 บิตจะเป็นจำนวนข้อมูล 8 บิต และบิตที่โปรแกรมได้อีก 1 บิต โดยบิตนี้จะเป็นบิตที่ 9 ซึ่งจะใช้เป็น Parity บิต ในการส่งข้อมูลจะต้องเขียนไปที่ บิต TB8 ในรีจีสเตอร์ SCON สำหรับการรับข้อมูลบิตที่ 9 จะถูกเก็บในบิต RB8

4.11.5 9 – Bit UART With Variable Baud Rate (Mode 3)

การทำงานในโหมดนี้จะคล้ายกับโหมด 2 แต่สามารถกำหนดค่า Baud Rateได้โดยการโปรแกรมไปที่ Timer 1 หลังจากโปรแกรมแล้วยังสามารถเลือกได้อีก 2 ค่าคือ ความถี่การ Overflow ของ Timer 1 หารด้วย 16 และหารด้วย 32

4.11.6 การกำหนดค่าเริ่มต้นให้รีจีสเตอร์ในการรับส่งข้อมูล

การรับข้อมูล ถ้าจะให้ MCS –51 รับข้อมูลทางพอร์ทอนุกรมจะต้องโปรแกรมไปที่บิต Receiver Enable (REN) ในรีจีสเตอร์ SCON ให้เป็นลอจิก " 1 " ซึ่งอาจทำได้สองวิธีดังนี้

SETB REN

เป็นการเซตบิต REN ให้เป็น " 1 " หรืออาจทำโดยใช้คำสั่ง

MOV SCON , #xxx1xxxxB

ซึ่งเป็นการย้ายข้อมูลที่ทำให้บิต REN เป็น 1 สำหรับค่า x หมายความว่าเป็นอะไรก็ได้ขึ้นกับการใช้งานในโหมดต่าง ๆ

ข้อมูลแบบ 9 บิต ในการรับส่งข้อมูลที่มีบิตข้อมูลแบบ 9 บิต ได้แก่ การใช้งานในโหมด 2 และโหมด 3 การส่งข้อมูลบิตที่ 9 จะถูกเขียนในบิต TB8 โดยการเขียนโปรแกรม สำหรับการรับข้อมูลเมื่อข้อมูลเข้าถึงบิตที่ 9 จะถูกเขียนลงในบิต RB8

การเพิ่มบิต Parity การส่งข้อมูลแบบ 9 บิต สามารถใช้บิตที่ 9 เป็นบิต Parity ได้ซึ่งบิต Parity จะอยู่ใน Program Status Word (PSW) โดยจะถูกเซตหรือเคลียร์ทุก ๆ แมชชีนไซเคิลที่เกี่ยว

ข้องกับ Accumulator เช่น ถ้าจะส่งข้อมูลแบบ 8 บิต ตามด้วยบิต Even Parity เป็นบิตที่ 9 สามารถเขียนโปรแกรมได้ดังนี้

```
MOV   C , P          ; อ่านค่าบิต P มาเก็บใน C
MOV   TB8 , C        ; นำค่าบิต Parity เขียนลงใน TB8
MOV   SBUF , A       ; ส่งข้อมูลไปทางพอร์ทอนุกรม
```

ถ้าเป็นแบบ Odd Parity ให้แก้ไขข้อมูลที่อ่านได้จากบิต Parity เสียก่อนที่จะส่งออกไป ซึ่งเขียนโปรแกรมได้ดังนี้

```
MOV   C , P          ; อ่านค่าบิต Parity มาเก็บใน C
CPL   C              ; กลับค่าให้เป็น Odd Parity
MOV   TB8 , C        ; เขียนค่าลงใน TB8
MOV   SBUF , A       ; ส่งข้อมูลออกไปทางพอร์ทอนุกรม
```

การส่งข้อมูลแบบมี Parity บิตด้วยไม่ใช่จะส่งได้แบบ 9 บิต หรือโหมด 2 และ 3 เท่านั้น ในโหมด 1 ซึ่งส่งข้อมูลแบบ 8 บิตก็สามารถทำได้ อย่างเช่นการส่งรหัส ASCII จะใช้บิตข้อมูล 7 บิต สำหรับบิตที่เหลืออีกหนึ่งบิตจะเป็นบิต Parity รวมเป็น 8 บิต ซึ่งสามารถเขียนโปรแกรมได้ดังนี้

```
CLR   ACC.7          ; เคลียร์ค่าบิต 7 เพื่อใช้เป็น Parity บิต
MOV   C , P          ; นำบิต Parity มาเก็บใน C
MOV   ACC.7 , C      ; เขียนค่าบิต Parity ลงในรีจีสเตอร์ A
MOV   SBUF , A       ; ส่งข้อมูลออกไปทางพอร์ทอนุกรม
```

แฟลกอินเทอร์รัพท์เมื่อมีการรับส่งข้อมูลเสร็จสิ้นจะมีผลต่อแฟลกอินเทอร์รัพท์ (RI และ TI) ในรีจีสเตอร์ SCON ซึ่งบิตเหล่านี้จะถูกเซตโดย Hardware แต่ต้องเคลียร์ด้วย Software

บิต RI ถ้าถูกเซตหมายความว่าบัฟเฟอร์ที่ใช้รับข้อมูลเต็มให้อ่านไปได้แล้ว และบิตนี้สามารถใช้อินเทอร์รัพท์ MCS – 51 แต่ถ้าเขียนโปรแกรมจะใช้วิธีตรวจเช็กบิตนี้ ถ้าเป็น " 1 " หมายความว่าให้อ่านข้อมูลมาเก็บใน รีจีสเตอร์ A ได้ แต่ก่อนอ่านจะต้องเคลียร์ RI เสียก่อนเพื่อจะได้รับข้อมูลถัดไปได้ ซึ่งเขียนโปรแกรมได้ดังนี้

```
WAIT : JNB   RI , WAIT   ; ถ้าบิตนี้ไม่เป็น " 1 " จะทำงานอยู่ที่เดิม
       CLR   RI          ; เคลียร์ RI
       MOV   A           ; อ่านค่ามาเก็บใน A
```

บิต TI เมื่อส่งข้อมูลออกไปแล้วบิตนี้จะถูกเซตเป็นการบอกว่าบัฟเฟอร์ส่งข้อมูลว่างแล้วให้ส่งข้อมูลใหม่เข้าไปได้ ซึ่งสามารถใช้บิตนี้อินเทอร์รัพท์ MCS – 51 ได้เช่นกัน แต่ถ้าเขียนโปรแกรมคอยตรวจเช็คอาจเขียนได้ดังนี้

```
WAIT : JNB    TI , WAIT      ; ตรวจบิต TI ว่าเป็น " 1 " หรือยัง
       CLR    TI             ; เคลียร์ TI
       MOV    SBUF , A       ; เขียนข้อมูลลงไป
```

4.11.7 อัตราการส่งข้อมูลของพอร์ทอนุกรม

จากการศึกษาการรับส่งข้อมูลในโหมดต่าง ๆ พบว่าในโหมด 0 และ โหมด 2 ไม่สามารถกำหนด Baud Rate เองได้ โดยในโหมด 0 ค่า Baud Rate จะมีค่าเท่ากับความถี่ของ Oscillator หารด้วย 12 ในโหมด 1 จะมีสองค่าคือ ความถี่ Oscillator หารด้วย 32 และหารด้วย 64 สองค่านี้เรียกว่า SMOD 0 และ SMOD 1 ซึ่งสามารถกำหนดได้ในรีจีสเตอร์ PCON บิตที่ 7 ในรีจีสเตอร์ PCON นี้ ไม่สามารถเข้าถึงข้อมูลระดับบิตได้ การเขียนข้อมูลลงไปทีละบิตจะต้องใช้วิธีที่เรียกว่า " Read – Modify – Write " คืออ่านค่าขึ้นมาแก้ไขแล้วเขียนลงไปใหม่ ตัวอย่างเช่น

```
       MOV    A , PCON       ; อ่านค่าจาก PCON มาเก็บในรีจีสเตอร์ A
       SETB   ACC.7          ; เซตบิต 7 ( SMOD )
       MOV    PCON , A       ; เขียนค่าลงไปใหม่ใน PCON
```

สำหรับโหมด 1 และโหมด 3 สามารถกำหนดค่า Baud Rate ได้โดยการโปรแกรมลงใน Timer 1 ในการโปรแกรมแต่ละครั้งจะมี SMOD สองค่าเช่นกัน ค่า Baud Rate ของโหมดต่าง ๆ แสดงได้ดังรูปที่ 4.11

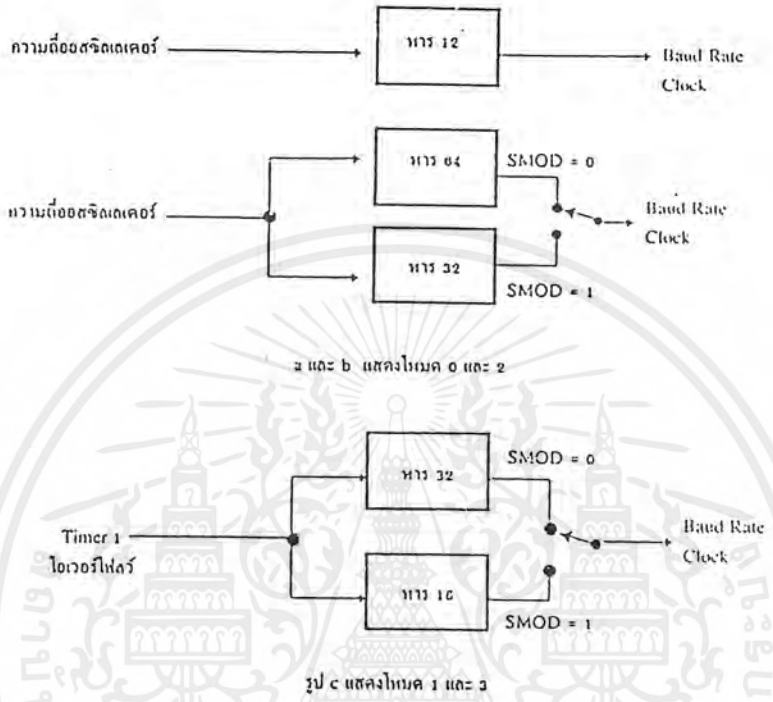
การใช้ Timer 1 กำหนด Baud Rate Clock

การกำหนดค่าลงใน Timer 1 ทำได้โดยการโปรแกรมไปที่ TMOD ให้ทำงานแบบ 8 – Bit Auto Reload Mode (โหมด 2) โดยเขียนค่าไปที่ TH1 ซึ่งโปรแกรมที่รีจีสเตอร์ TMOD ได้ดังนี้

```
MOD    TMOD , #0010xxxxB
```

ค่า x หมายความว่าเป็นอะไรก็ได้เพราะบิตเหล่านี้ใช้ใน Timer 0

ถ้าต้องการ Baud Rate ต่ำ ๆ สามารถใช้ 16 – Bit Mode ได้ โดยโปรแกรมเป็น TMOD = 0001xxxxB ค่า Baud Rate ที่ส่งออกมาจะมีค่าเท่ากับ ความถี่ของ Timer 1 เกิด Overflow หารด้วย 32 (หรือหารด้วย 16 ถ้าเป็น SMOD = 1)



รูปที่ 4.11 แสดงการกำหนด Baud Rate ในโหมดต่าง ๆ

รูปแบบทั่วไปของการหาค่า Baud Rate ในโหมด 1 และ 3 สามารถหาได้ดังนี้

Baud Rate = Timer 1 Overflow Rate / 32

ถ้าต้องการค่า Baud Rate เท่ากับ 1,200 สามารถคำนวณค่าความถี่ Overflow ของ Timer 1 ได้ดังนี้

1,200 = Timer 1 Overflow Rate / 32

จะได้ Timer 1 Overflow Rate เท่ากับ 38.4 kHz

ถ้าระบบ MCS – 51 ใช้ความถี่สัญญาณนาฬิกาจาก Crystal เท่ากับ 12 MHz ตัว Timer 1 จะได้รับ Clock เท่ากับ 1 MHz หรือ 1,000 kHz ถ้าเราต้องการ Timer 1 Overflow เท่า

กับ 38.4 kHzดังนั้นค่าอัตรา Overflow มีค่าเท่ากับ 1,000 / 38.4 = 26.04 Clock โดยค่า Overflow จะเกิดเมื่อเกิดการเปลี่ยนจาก FFH เป็น 00H ดังนั้นจะต้องให้ Timer 1 นับไป 26 Count ดังนั้นค่าที่จะให้รีจีสเตอร์ TH1 มีค่าเท่ากับ –26 ซึ่งใช้เป็นค่า Reload ดังนั้นเขียนคำสั่งได้ดังนี้

MOV TH1 , # -26

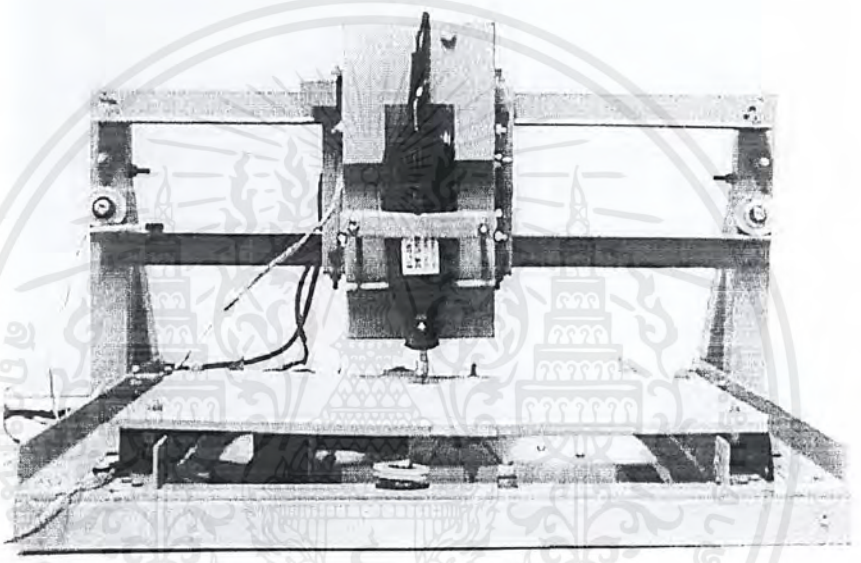
ตัวโปรแกรมแอสแซมเบอร์ทั่วไปจะแปลงค่า -26 เป็น 0E6H เอง จากที่ผ่านมาจะเห็นว่าความถี่ Baud Rate จะมีความสัมพันธ์กับค่าสัญญาณนาฬิกาที่ใช้จาก Crystal ในตารางที่ 4.3 จะเป็นค่าที่ต้องกำหนดใน Timer 1 เมื่อต้องการค่า Baud Rate ต่าง ๆ

ตารางที่ 4.5แสดงความถี่สัญญาณนาฬิกาที่ใช้กำหนด Baud Rate ค่าต่าง ๆ

ค่า Baud Rate	Crystal	SMOD โหมด	ค่า TH1	ค่า Baud Rate ที่ได้	ผิดพลาด
9600	12.000	1	-7 (F9H)	8923	7%
2400	12.000	0	-13 (F3H)	2404	0.16%
1200	12.000	0	-16 (E6H)	1202	0.16%
19200	11.059	1	-3 (FDH)	19200	0
9600	11.059	0	-3 (FDH)	9600	0
2400	11.059	0	-12 (F4H)	2400	0
1200	11.059	0	-24 (E8H)	1200	0

บทที่ 5
การทดลองและการออกแบบ

5.1 การออกแบบระบบทางกล



รูปที่ 5.1 เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ

โครงสร้างเครื่องกล

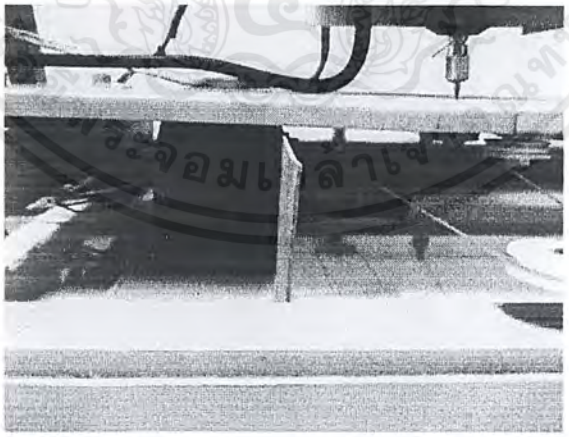
การพันสลิง สลิงเป็นตัวส่งกำลังที่สำคัญของเครื่อง เจาะ การออกแบบจะแบ่งแรงออกจากมอเตอร์ในสองทิศทางที่สมดุลย์กันซึ่งเป็นข้อดี คือ

1. จะทำให้อายุการใช้งานของแบริ่งของมอเตอร์ยืนยาวมาก การพันสลิงจะมีการทดแรงตามวิชารอกซึ่งจะได้แรงเพิ่มขึ้นอีกเท่าตัว
2. การพันตามลักษณะนี้จะมีผลทำให้มีระยะทางสั้นลงเนื่องจากการทดแรงเราจะชดเชยด้วยการเพิ่มขนาดของพู่เลย์ให้มีเส้นรอบวงมากขึ้นเพื่อชดเชยกับระยะทางที่เสียไป
3. จะทำให้อายุการใช้งานของลวดสลิงเพิ่มขึ้นเพราะลวดสลิงเมื่อขดตัวในวงล้อที่มีเส้นรอบวงน้อยในค่า ๆ หนึ่งจะทำให้ขาดง่าย

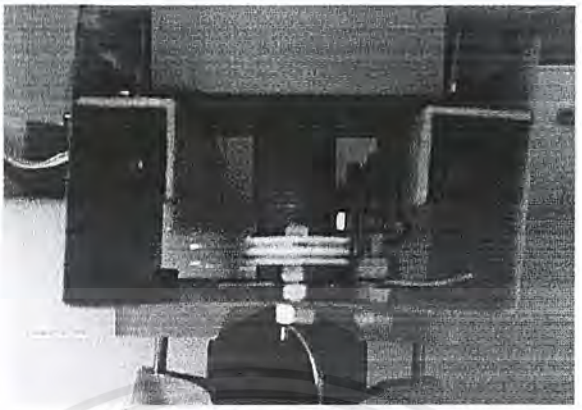
ขั้นตอนการพันสลิง

การพันสลิงเข้ากับแกนต่าง ๆ เริ่มพันสลิงที่แกน Y ก่อนโดยยึดปลายสลิงที่ด้านหน้ากับจุดยึดสลิงให้เหลือปลายประมาณ 5 เซนติเมตร เผื่อไว้เพื่อจับยึดได้ตอนปรับแต่ง เลื่อนแท่ง Y มาด้านหน้าใช้คีมล็อกไม่ให้แท่น Y เคลื่อนที่จากนั้นทำสลิงไปโค้งเข้ากับลูกล้อพันที่พูเล่ย์ 6 รอบ สอดปลายสลิงเข้ารูกลาย และโค้งรอบพูเล่ย์ให้ได้ทิศทางตรงกันข้ามและนำปลายไปพันกับที่ยึดสลิงติด ตั่งแกน X กับ แกน Z คล้ายกันเพียงแต่ พันสลิงที่แกน X 9 รอบ และแกน Z 4 รอบ

รูปที่ 5.2 การพันสลิงแนวแกน Y



รูปที่ 5.3 การพันสลิงแนวแกน X



รูปที่ 5.4 การพันสลิงแนวแกน Z

5.2 การออกแบบส่วนวงจร

5.2.1 การออกแบบวงจรไมโครคอนโทรลเลอร์

วงจรไมโครคอนโทรลเลอร์ประกอบด้วยไมโครคอนโทรลเลอร์เบอร์ 89C52 ซึ่งมีหน่วยความจำรอมภายในขนาดความจุขนาด 8 กิโลไบต์ และใช้คริสตรอลความถี่ 11.059 เมกกะเฮิรตซ์ เป็นตัวสร้างสัญญาณออสซิลเลทให้แก่ไมโครคอนโทรลเลอร์ ที่ขา P3.1 และขา P3.2 ซึ่งเป็นพอร์ทอนุกรม ใช้ส่งและรับข้อมูลตามลำดับจะต่ออยู่กับไอซีเบอร์ MAX232 ซึ่งเป็นไอซีที่แปลงสัญญารจากคอมพิวเตอร์ตามมาตราฐาน RS-232 ให้เป็นสัญญาณในระดับบ TTL ที่ขา P1.0 , P1.1 และ P1.2 จะต่ออยู่กับวงจรออปโต้เซ็นเซอร์ของแกน X –Y - Z ตามลำดับ ที่ขา P1.3 ถึง P1.6 ใช้สร้างสัญญาณเพื่อขับสเต็ปปิ้งมอเตอร์แกน Z (แกนเจาะ) และต่ออยู่กับไอซีแลตซ์เบอร์ 74HC573 ก่อนที่จะส่งไปยังวงจรขับสเต็ปปิ้งมอเตอร์แกน Z

ที่ขา P0.0 ถึง P0.4 และ P2.0 ถึง P2.4 ใช้สร้างสัญญาณควบคุมวงจรขับสเต็ปปิ้งมอเตอร์แกนX และ แกนY ซึ่งวงจรขับในแกนX และแกน Y จะใช้ ไอซีเบอร์ LMD18245 เนื่องจากวงจรขับของทั้ง 2 แกนนี้ใช้สัญญาณควบคุมจากไมโครคอนโทรลเลอร์ที่ขาเดียวกันแต่แยกการทำงานคนละเวลากัน จึงต้องมีการควบคุมว่าจะให้สัญญาณควบคุมส่งไปยังวงจรขับของแกนไหนโดยใช้ไอซีแลตซ์เบอร์ 74HC573 ต่ออยู่และสัญญาณ Enable ได้มาจากไมโครคอนโทรลเลอร์ที่ขา P3.5 สำหรับแกน X และ ขาP3.6 สำหรับแกน Y วงจรส่วนไมโครคอนโทรลเลอร์แสดงในรูปที่ 5.5

5.2.2 วงจรเซ็นเซอร์

เนื่องจากการทำงานของเครื่องเจาะแผ่นปริ้นท์อัตโนมัติ จะต้องรู้ตำแหน่งเริ่มต้นการทำงานของตัวเองเสียก่อนซึ่งเมื่อเรื่อมทำงาน แกนทั้ง 3 จะต้องเคลื่อนไปยังตำแหน่งเริ่มต้น เมื่อถึงตำแหน่งเริ่มต้นจะมีเซ็นเซอร์เป็นตัวตรวจจับ เพื่อส่งค่าให้กับตัวประมวลผลกลาง เซ็นเซอร์ที่ใช้จะใช้ออปโต้ที่มีตัวรับส่งในตัวเดียวกันวางในลักษณะตรงกันข้ามกัน โดยมีร่องผ่ากลางเพื่อให้วัตถุที่

ต้องการตรวจจับผ่านเข้ามาบังแสง แล้วเกิดการตรวจจับขึ้น ในขณะปรกติจะมีลอจิกออกมาเป็น 0 เมื่อมีวัตถุเคลื่อนที่มาบังแสงจะให้เอาท์พุทออกมาเป็นลอจิก 1 ตัวประมวลผลกลางจะรับรู้แล้วหยุดมอเตอร์แกนนั้นเสีย ซึ่งตำแหน่งที่หยุดคือตำแหน่งเริ่มต้น เซ็นเซอร์ออปโต้ที่ใช้คือเบอร์ H21A1 ดังวงจรในรูปที่ 5.5

5.2.3 วงจรขับสเต็ปปิ้งมอเตอร์แกน X และ แกน Y

ในการเคลื่อนตำแหน่งของแกน X และแกน Y เพื่อเจาะแผ่ปริ๊นท์นั้นจำเป็นต้องใช้การเคลื่อนที่อย่างละเอียดเพื่อให้การเจาะมีความแม่นยำ ดังนั้นวงจรขับสเต็ปปิ้งมอเตอร์ที่ใช้จึงควรเป็นแบบไมโครสเต็ป ซึ่งในปริญญานิพนธ์นี้จะใช้ ไอซีเบอร์ LMD18245 ซึ่งมีวงจรภายในประกอบด้วย DMOS ต่อ แบบฟูลบริดจ์ และยังมีวงจรแลงสัญญาณดิจิตอลเป็นอนาล็อกแบบ 4 บิต ภายในหน้าที่ของวงจรแปลงสัญญาณจากดิจิตอลเป็นอนาล็อกคือ ในสัญญาณ DIRECTION 1 ครั้ง จะสามารถแบ่งกระแสให้แก่ขดลวดได้เท่ากับ $2^4 = 16$ ระดับดังนั้นใน 1 สเต็ปจะมีระดับแรงดัน 8 ระดับ ต่อ 1 สเต็ป ถ้ามอเตอร์ที่ใช้หมุนที่ฟูลสเต็ป 1.8 องศา ต่อสเต็ป (200 สเต็ป ต่อ รอบ) เมื่อใช้ LMD 18245 จะสามารถขับได้จำนวนสเต็ปต่อรอบเท่ากับ 1600 สเต็ปต่อรอบซึ่งละเอียดขึ้น 8 เท่าจากเดิม ลายละเอียดของสัญญาณควบคุมดูได้จากเอกสารแสดงคุณลักษณะในภาคผนวก

ที่ขา 13 ของ LMD 18245 จะต่ออยู่กับ R_{SENSE} เพื่อกำหนดค่ากระแสที่ไหลในขดลวดของสเต็ปปิ้งมอเตอร์ในแต่ละเฟส ซึ่งตามเอกสารแสดงคุณสมบัติของไอซีซึ่งแสดงในภาคผนวก สามารถคำนวณค่า R_{SENSE} ได้จาก

$$R_{SENSE} = V_{REF} * 400 / I_{OUT} \qquad (5.1)$$

ถ้าใช้สเต็ปปิ้งมอเตอร์ 24 โวลท์ กระแส 0.3 แอมป์และ $V_{REF} = 5$ โวลท์ ดังนั้นจะได้ $R_{SENSE} = 66.66$ กิโลโอห์ม เลือกใช้ค่า 68กิโลโอห์ม วงจรขับสเต็ปปิ้งมอเตอร์แกน X และ Y แสดงดังรูปที่ 5.6

5.2.4 วงจรขับสเต็ปปิ้งมอเตอร์แกน Z (แกนเจาะ)

วงจรขับสเต็ปปิ้งมอเตอร์แกน Z (แกนเจาะ) จะใช้การขับแบบ ฟูลสเต็ป ซึ่งจะใช้ทรานซิสเตอร์ต่อแบบดาร์ลิงตันกันการเคลื่อนที่ของสเต็ปปิ้งมอเตอร์แกน Z นั้น ต้องยกน้ำหนัก

ของส่วนเจาะรวมทั้งน้ำหนักของสว่านด้วยทำให้เกิดแรงบิดสูง ดังนั้นมอตอร์จะดึงกระแสสูงจึงต่อทรานซิสเตอร์แบบดาร์ลิงตันกันเพื่อกำลังงานที่จะเกิดขึ้นที่ตัวทรานซิสเตอร์กำลัง ลักษณะของวงจรขับในแกน Z แสดงในรูปที่ 5.6

5.2.5 วงจรเปิดปิดสว่าน

ในการทำงานของเครื่องเจาะ ขณะที่ยังไม่มีการส่งข้อมูลหรือสั่งให้เครื่องทำการเจาะตัวสว่านก็จะปิด แต่เมื่อเครื่องเจาะแผ่นปริ๊นท์อัตโนมัติได้รับข้อมูลที่จะเจาะ ตัวสว่านก็จะทำงานจนกระทั่งเสร็จการเจาะและสว่านจะทำงานเมื่อได้รับข้อมูลชุดใหม่มา ดังนั้นจึงต้องมีวงจรที่ทำหน้าที่ควบคุมการเปิด และ ปิด สว่าน วงจรเปิดปิดสว่านประกอบด้วยไตรแอกเบอร์ BTA04-600 ซึ่งสามารถทนกระแสได้ 4 แอมป์ ทนแรงดันได้ถึง 600 โวลท์ ต่อกับ ออปโต้ไอโซเลเตอร์เบอร์ MOC3041 ซึ่งสัญญาณควบคุมต่ออยู่กับขา P3.2 ของไมโครคอนโทรลเลอร์ถ้าต้องการให้สว่านเปิดไมโครคอนโทรเลอร์ต้องส่งลอจิก 0 มาแต่ถ้าต้องการให้สว่านปิด ต้องส่งลอจิก 1 มาจากไมโครคอนโทรลเลอร์ วงจรเปิดปิดสว่านแสดงดังรูปที่ 5.7

5.2.6 วงจรไฟเลี้ยง

ไฟเลี้ยงของเครื่องเจาะแผ่นปริ๊นท์อัตโนมัติประกอบด้วย 3 ส่วน คือ ส่วนวงจรไฟเลี้ยง 5 โวลท์ ใช้จ่ายให้แก่ ไมโครคอนโทรลเลอร์ , ไอซีแลตซ์ , MAX232 และใช้เป็นระดับแรงดันอ้างอิงในไอซี LMD18245 สร้างโดยใช้ไอซีเรกกูเลเตอร์เบอร์ 7805 และจำกัดกระแสไว้ที่ 2 แอมป์ โดยใช้ทรานซิสเตอร์ประเภท PNP

วงจรไฟเลี้ยง 12 โวลท์ ใช้จ่ายให้แก่วงจรขับสเต็ปปิ้งมอเตอร์ในแกน Z สร้างโดยใช้ไอซีเรกกูเลเตอร์เบอร์ 7812 จำกัดกระแสที่ 2 แอมป์เช่นเดียวกับวงจรไฟเลี้ยง 5 โวลท์ และมีทรานซิสเตอร์ช่วยขยายกระแสด้วย เนื่องจาก สเต็ปปิ้งมอเตอร์แกน Z กินกระแสสูง

วงจรไฟเลี้ยง 34 โวลท์ใช้จ่ายให้แก่ ไอซีขับสเต็ปปิ้งมอเตอร์เบอร์ LMD18245 สร้างโดยนำไฟกระแสสลับ 24 โวลท์ จากหม้อแปลงมาเรียงกระแสแล้วกรองกระแสให้เรียบ จะได้ค่าแรงดันไฟตรงออกมา 34 โวลท์เมื่อจ่ายให้แก่ไอซี ตัวไอซีจะรักษาระดับแรงดันเอาท์พุทให้ออกมาเป็น 24 โวลท์ แก่สเต็ปปิ้งมอเตอร์แกน X และแกน Y รูปที่ 5.7 แสดง วงจรไฟเลี้ยงทั้ง 3 ส่วน

5.3 การออกแบบโปรแกรมควบคุม

ก่อนที่จะเขียนโปรแกรมรับส่งข้อมูลของไฟล์จากโปรแกรม Protel จากคอมพิวเตอร์นั้นต้องมีความเข้าใจลักษณะข้อมูลจากโปรแกรม Protel ก่อน ซึ่งเมื่อเราเขียนลายวงจรเสร็จแล้วและ

ต้องการข้อมูลของรูอุปกรณ์ต่างๆ สามารถทำได้โดย ใช้โปรแกรม Protel เลือกที่ฟังก์ชั่น NC DRILL โปรแกรมจะทำการบันทึกข้อมูลของรูในรูปแบบของไฟล์ประเภท TEXT ซึ่งมีตัวอย่างดังนี้

M48
T1F00S00
T2F00S00
%
T01
X1176Y084
X1186
X1196
X1206
X1216
X1226
X1126Y09
T02
X1126Y087
Y088
Y089
X1156
Y088
Y087
Y09
M30

ไฟล์ที่ได้จะขึ้นต้นด้วย M48 และลงท้ายด้วย M30 T01 , T02 , T03 จะแยกข้อมูลรูออกตามขนาด และแต่ละบรรทัดจะบอก โคออดิเนตของรูต่างๆ โดยจะบอกตำแหน่ง X แล้วตามด้วย ค่าโคออดิเนตเป็นตัวเลข 5 หลัก ถ้าตัวเลขมีไม่ถึง 5 หลักให้เติม 0 จนครบ 5 หลัก ค่านั้น คือค่าของตำแหน่งโคออดิเนตที่มีค่าเป็นหน่วย mil ในบางบรรทัดจะมี แค่ X หรือ Y อย่างเดียวนั่นหมายถึงค่าอีกค่าที่เหลือจะเป็นตามบรรทัดข้างบน

เมื่อทราบรายละเอียดข้อมูลจากไฟล์ที่ต้องการนำมาเจาะแล้ว ก็นำมาเขียนโปรแกรมซึ่งจะติดต่อจากคอมพิวเตอร์ไปยังเครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ ซึ่งในปริญญานิพรธ์นี้จะใช้โปรแกรมภาษาวิชวลเบสิกในการเขียนโปรแกรมควบคุมการทำงานซึ่งมีรายละเอียดแบ่งเป็นส่วนต่างๆดังนี้

- **ส่วนเจาะอัตโนมัติ**

โปรแกรมส่วนนี้ จะเป็นการรับไฟล์ข้อมูล Text file ที่ได้จากการทำ NC DRILL ในโปรแกรม Protel มาอ่านค่าของรู โดยโปรแกรมที่เขียนโดยวิชวลเบสิก จะให้ผู้ใช้ เลือกไฟล์ที่เป็น Text file ที่จะนำมาเจาะ ถ้าไม่ใช่ ไฟล์ที่ต้องการเจาะจะให้เลือกใหม่หรือจะยกเลิกการทำงานก็ได้ แต่ถ้าเป็นๆไฟล์ที่ต้องการเจาะโปรแกรมจะทำงานต่อไป โดยจะเก็บค่าตำแหน่งโคออดิเนต X และ Y ของรูทุกรูของแต่ละขนาดไว้ในอาร์เลย์ หลังจากนั้นก็จะทำการเรียกอาร์เลย์ที่ทำการเก็บตำแหน่งต่างๆของรูขนาดเล็กสุดมาคำนวณว่าในแต่ละโคออดิเนต นั้น จะต้องส่งข้อมูลของแกน X ว่าจะต้องหมุนไปทางซ้ายหรือขวา และหมุนไปกี่สเต็ป แล้วตามด้วยข้อมูลของแกน Y ที่มีลักษณะเดียวกัน ข้อมูลที่ส่งไปว่าเป็นซ้ายหรือขวาจะส่งไปเป็น ASCII ส่วน ข้อมูลจำนวนสเต็ป จะส่งไปเป็นค่าไบนารี่

เมื่อเมื่อข้อมูลตำแหน่งโคออดิเนตของรูทุกรู ที่มีขนาดเล็กที่สุด ถูกส่งไปครบหมดแล้ว เครื่องเจาะแผ่นปริ๊นท์จะทำการเจาะโดยอัตโนมัติ และรอให้เครื่องเจาะครบทุกรู หลังจากนั้น ก็จะทำการส่งข้อมูลรูขนาดถัดไปแล้วเครื่องเจาะแผ่นปริ๊นท์จะทำการเจาะ ทำอย่างนี้เรื่อยๆจนครบทุกขนาดของรู ซึ่งโปรแกรมการเจาะอัตโนมัติสามารถเขียนได้ โฟล์วชาร์ท ในรูปที่ 5.8

- **ส่วนรับโคออดิเนต 10 ค่าจากคีย์บอร์ด**

เป็นโปรแกรมที่สร้างขึ้นเพื่อ ทดสอบความแม่นยำของการเคลื่อนที่ของสเต็ปปิ้งมอเตอร์ทั้งแกน X และแกน Y เพราะสามารถกำหนดตำแหน่งโคออดิเนตที่จะสั่งให้เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ ไปเจาะได้ จากคีย์บอร์ดได้ เมื่อกำหนดค่าครบ 10 ค่าแล้วทำการสั่งให้เครื่องทำการเจาะข้อมูลของรูเจาะก็จะถูกส่งไปยังไมโครคอนโทรลเลอร์ หลังจากนั้นเครื่องก็จะเจาะรูตั้งแต่รูแรกที่คีย์เข้าไปจนครบถึงตำแหน่งสุดท้ายที่รับมาจากคีย์บอร์ด

- **โปรแกรมส่วนรับค่าส่วนตั้งค่า offset**

เนื่องจากการวางแผ่นลายวงจรพิมพ์ ที่ต้องการจะเจาะถ้าวางตำแหน่งที่หัวสว่านอยู่ที่ตำแหน่งเริ่มต้น ข้อมูลจุด โคออดิเนต ที่ส่งจากคอมพิวเตอร์ ที่ส่งมายังตัวประมวลผลของเครื่องเจาะจะเป็นข้อมูลที่ตรงกับตำแหน่งรูที่ต้องการจะเจาะ แต่ถ้าวางแผ่นลายวงจรที่ตำแหน่งอื่นที่ไม่ใช่ตำแหน่งเริ่มต้นของหัวสว่าน เมื่อส่งข้อมูลมายังส่วนประมวลผลของเครื่องเจาะ เครื่องเจาะก็จะเจาะรูได้ไม่ตรงกับตำแหน่งแผ่นลายวงจรที่วางเอาไว้ ดังนั้นจึงสร้างส่วนโปรแกรมที่รับค่าตำแหน่ง

ที่วางแผ่นลายวงจรพิมพ์โดยเรียกค่านี้ว่า offset ซึ่งโปรแกรมส่วนนี้จะรับค่า offset ของแกน X และของแกน Y แล้วนำค่า 2 ค่านี้ไปบวกรวมกับข้อมูลที่จากไฟล์ที่จะนำมาเจาะแผ่นลายวงจร เพื่อให้เครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ เคลื่อนที่ไปเจาะรูได้ตรงกับตำแหน่งที่วางแผ่นวงจรไว้

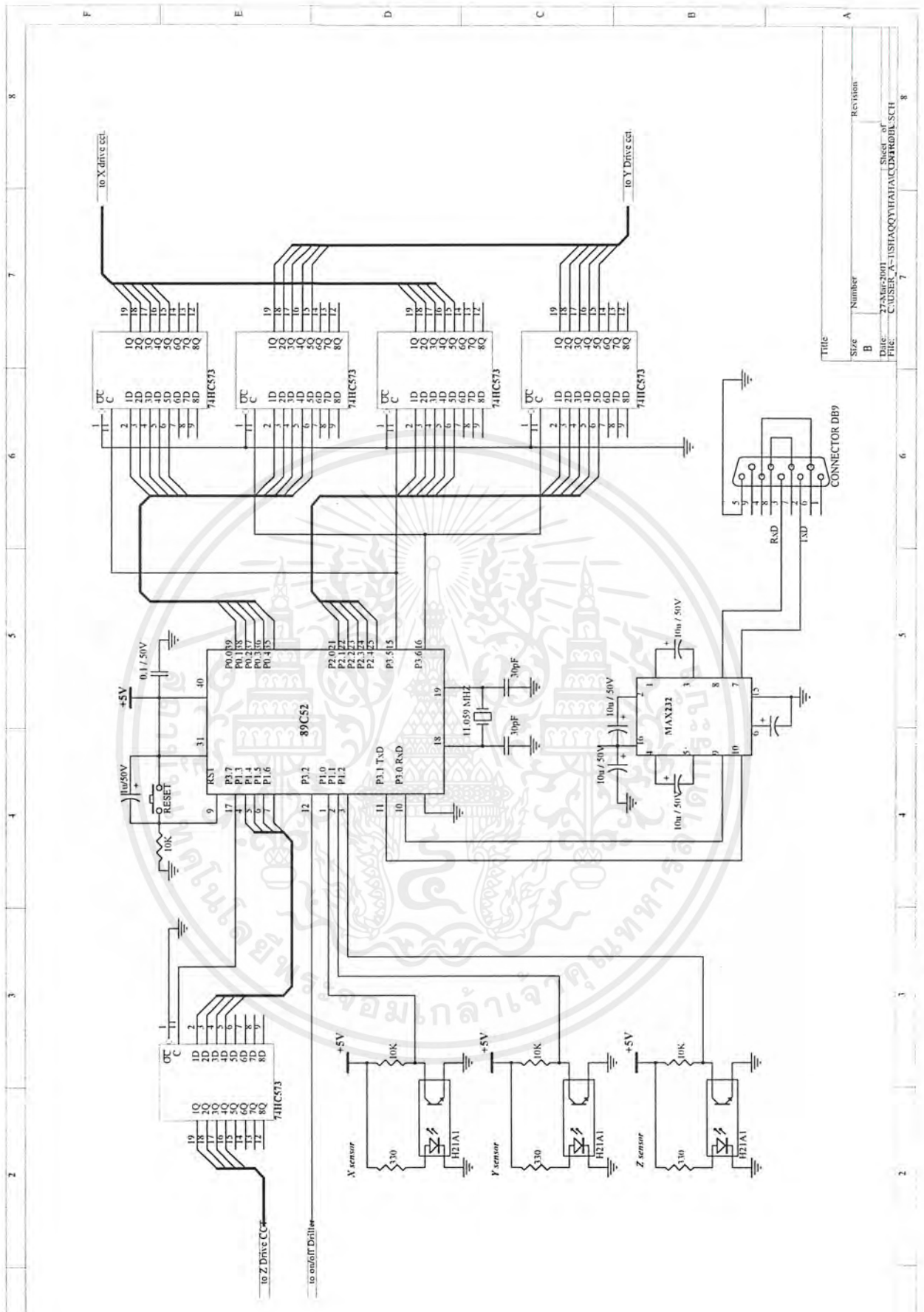
- **โปรแกรมส่วนควบคุมวงจร**

ส่วนโปรแกรมเพื่อควบการทำงานของวงจรต่าง ๆ จะใช้ไมโครคอนโทรลเลอร์ 8252 รับข้อมูลจากพอร์ทอนุกรม RS232

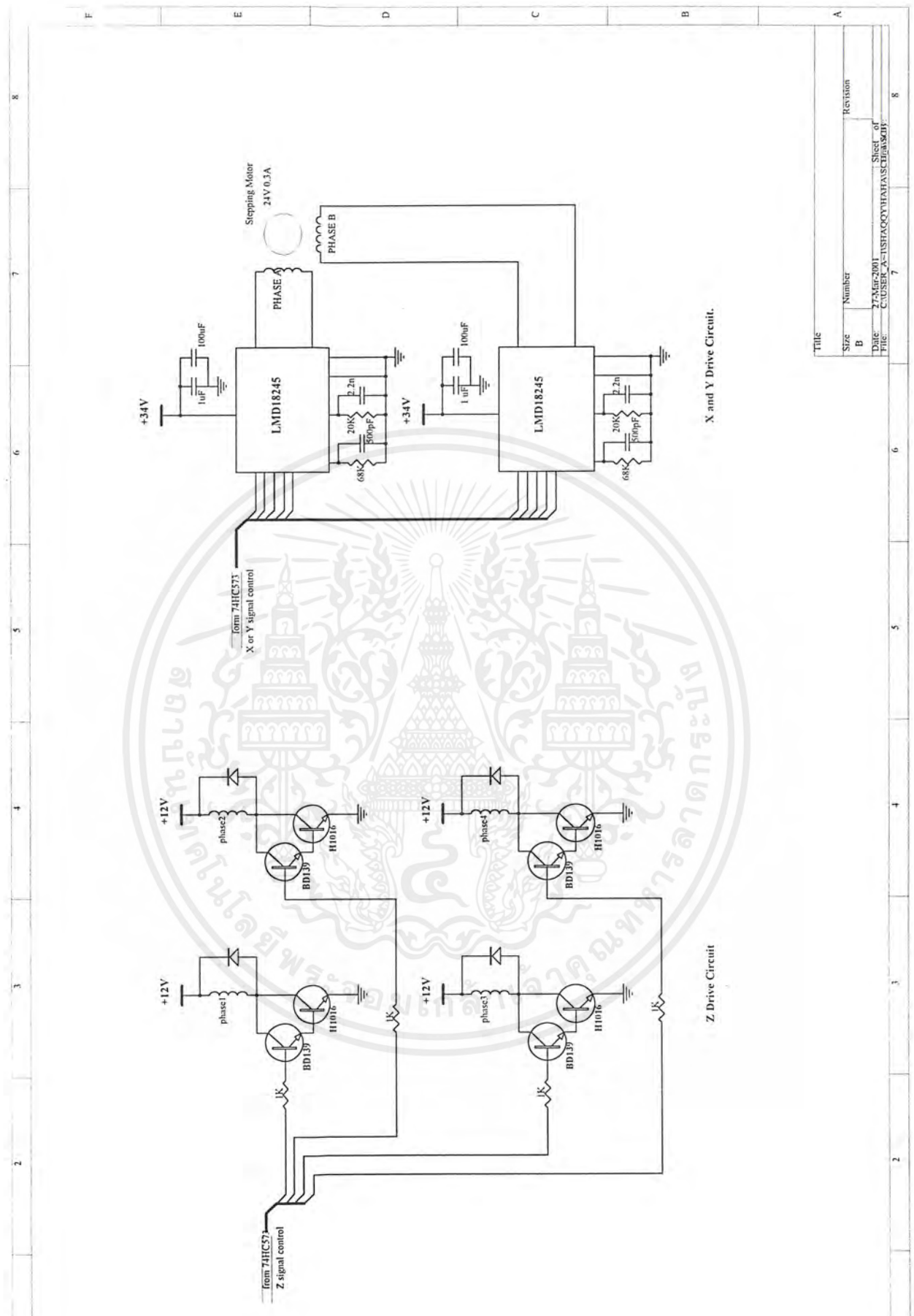
ในส่วนการเขียนโปรแกรมจะรับค่าจากพอร์ทอนุกรมเพื่อนำมาเรียงในแรมในของไมโครคอนโทรลเลอร์โดยจะรับค่ามาเก็บเรียงขึ้นไปเรื่อย ๆ จนกระทั่งตรวจพบค่าที่กำหนดให้เป็น End Of File (EOF) ซึ่งกำหนดเป็น ค่า 0FEH แล้วจึงเข้าสู่ส่วนการทำงาน MAIN

หลังจากนั้นจะมีข้อมูลที่เป็นตัวบ่งชี้ว่าแกน X หมุนไปทางลบหรือทางบวก และตามด้วยค่าไบนารี่สองไบต์เป็นจำนวน สเต็ปที่แกน X จะหมุนหลังจากนั้นจะตามด้วยข้อมูลที่บอกว่าแกนY จะหมุนไปดทางบวกหรือทางลบและจะตามด้วยค่าไบนารี่สองไบต์ เป็นตัวบอกจำนวนสเต็ปที่ต้องการให้มอเตอร์แนวแกน Y หมุนเป็นเช่นนี้สลับไปเรื่อย ๆ จานตรวจพบค่าที่บ่งบอกว่าเป็นข้อมูลบไบต์สุดท้าย (End Of File) โดยที่ว่าAddress อ้างอิงตัวแรกคือที่ตำแหน่งหน่วยความจำ INDEX

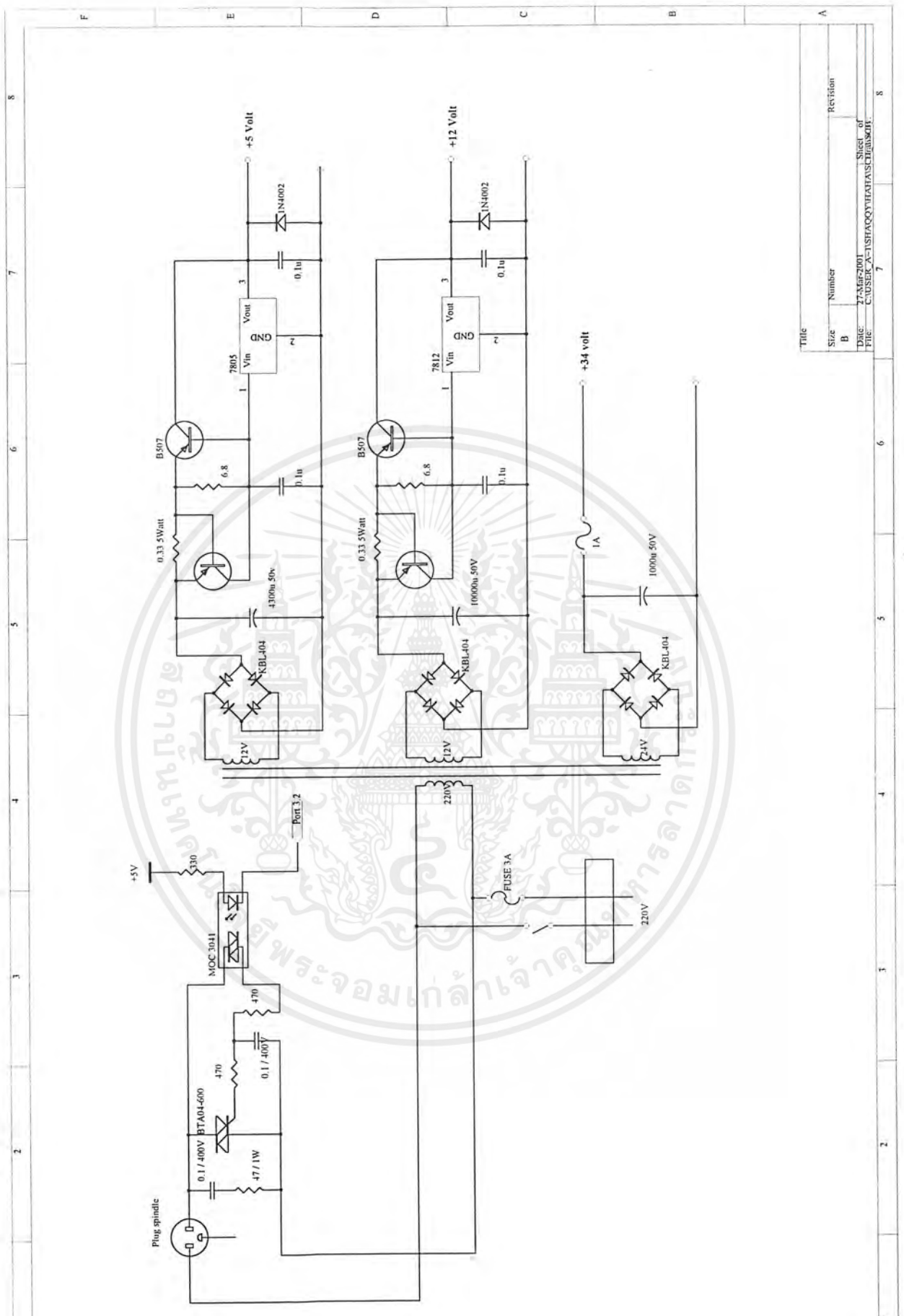
โดยที่ต้องการนำข้อมูลจากหน่วยความจำข้อมูลส่วนนี้มาสร้างกรณีเพื่อให้มอเตอร์แกน X และแกน Y หมุนได้ตามทิศทางและจำนวนสเต็ปได้อย่างถูกต้องโดยมีโฟลชาต์ทเป็นส่วน ๆ ดังแสดงไว้ ซึ่งการทำงานของโปรแกรมแสดงตามโฟลว์ชาร์ทในรูปที่ 5.9



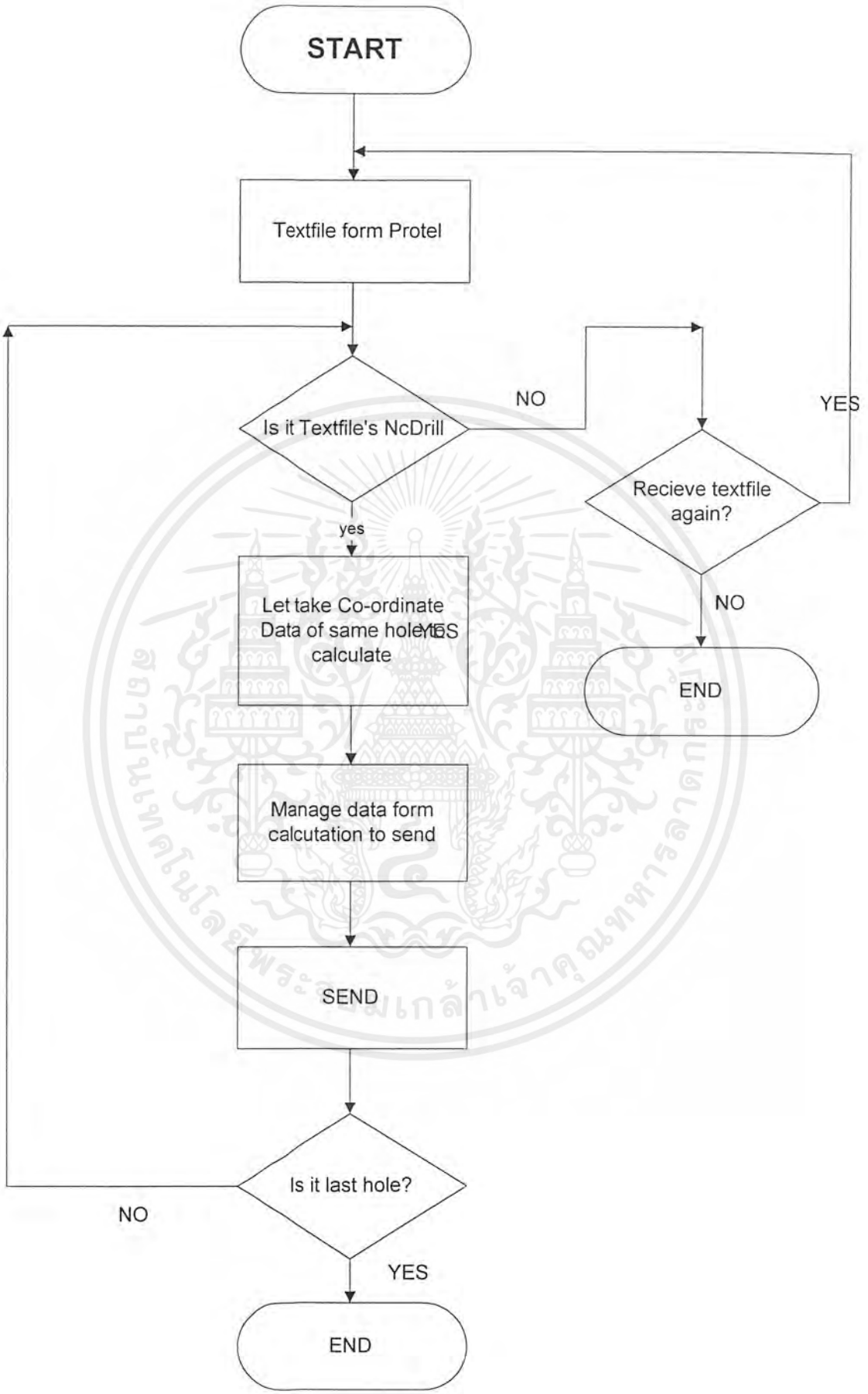
รูปที่ 5.5 วงจรไมโครคอนโทรลเลอร์และวงจรควบคุม



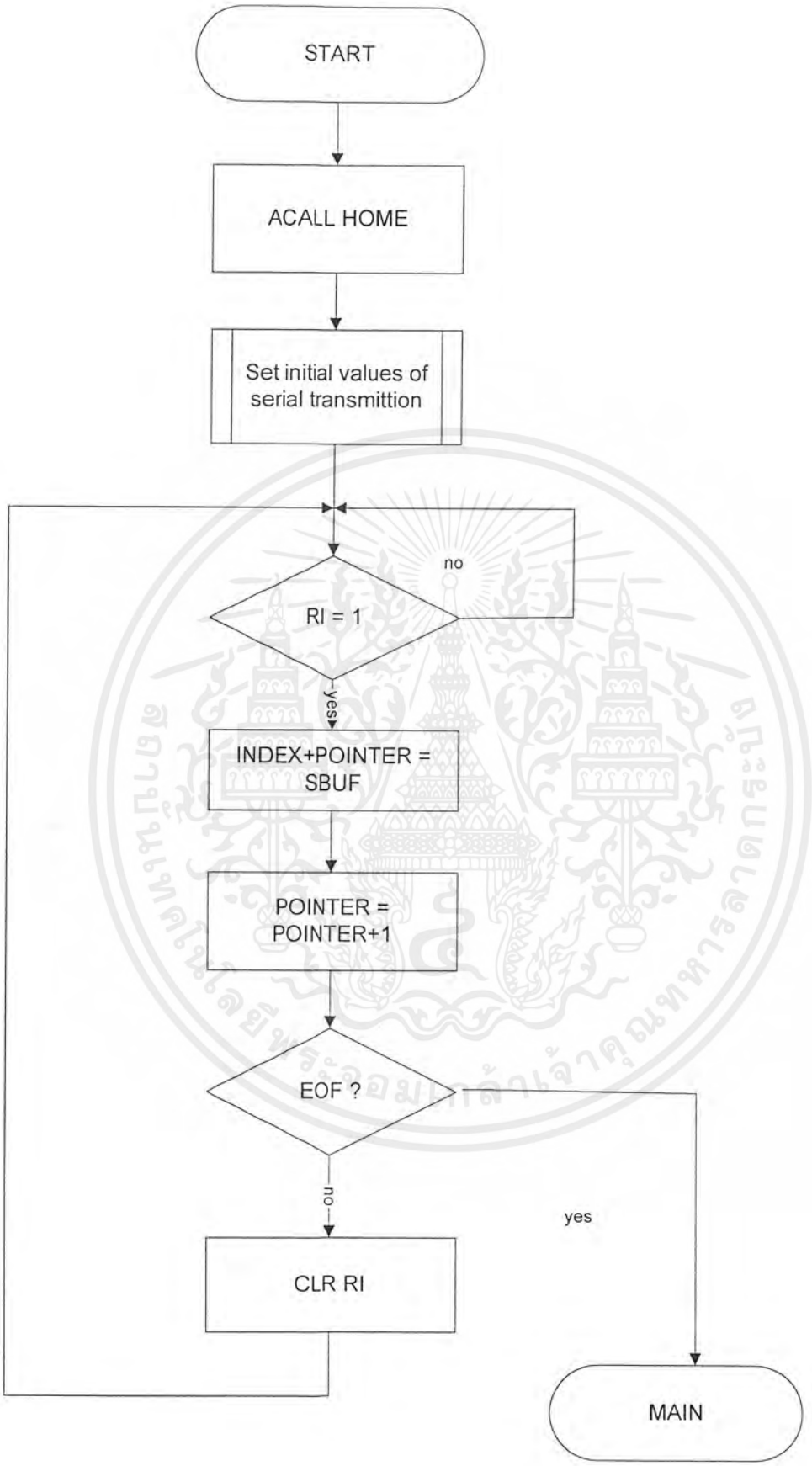
รูปที่ 5.6 วงจรขับสเต็ปมอเตอร์แกน X แกน Y และแกน Z



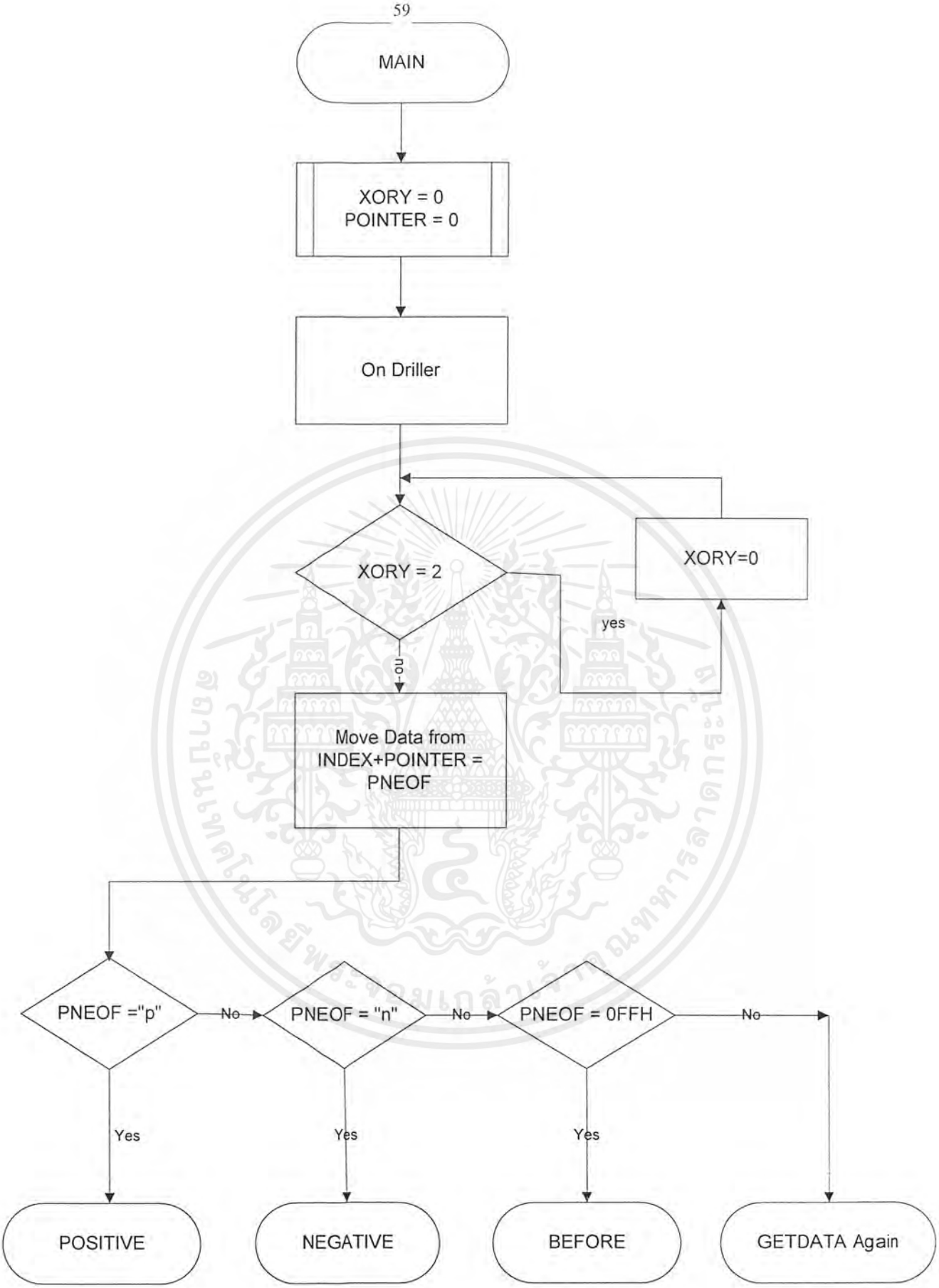
รูปที่5.7 วงจรปิดสวิตช์สว่านและวงจรไฟเลี้ยง



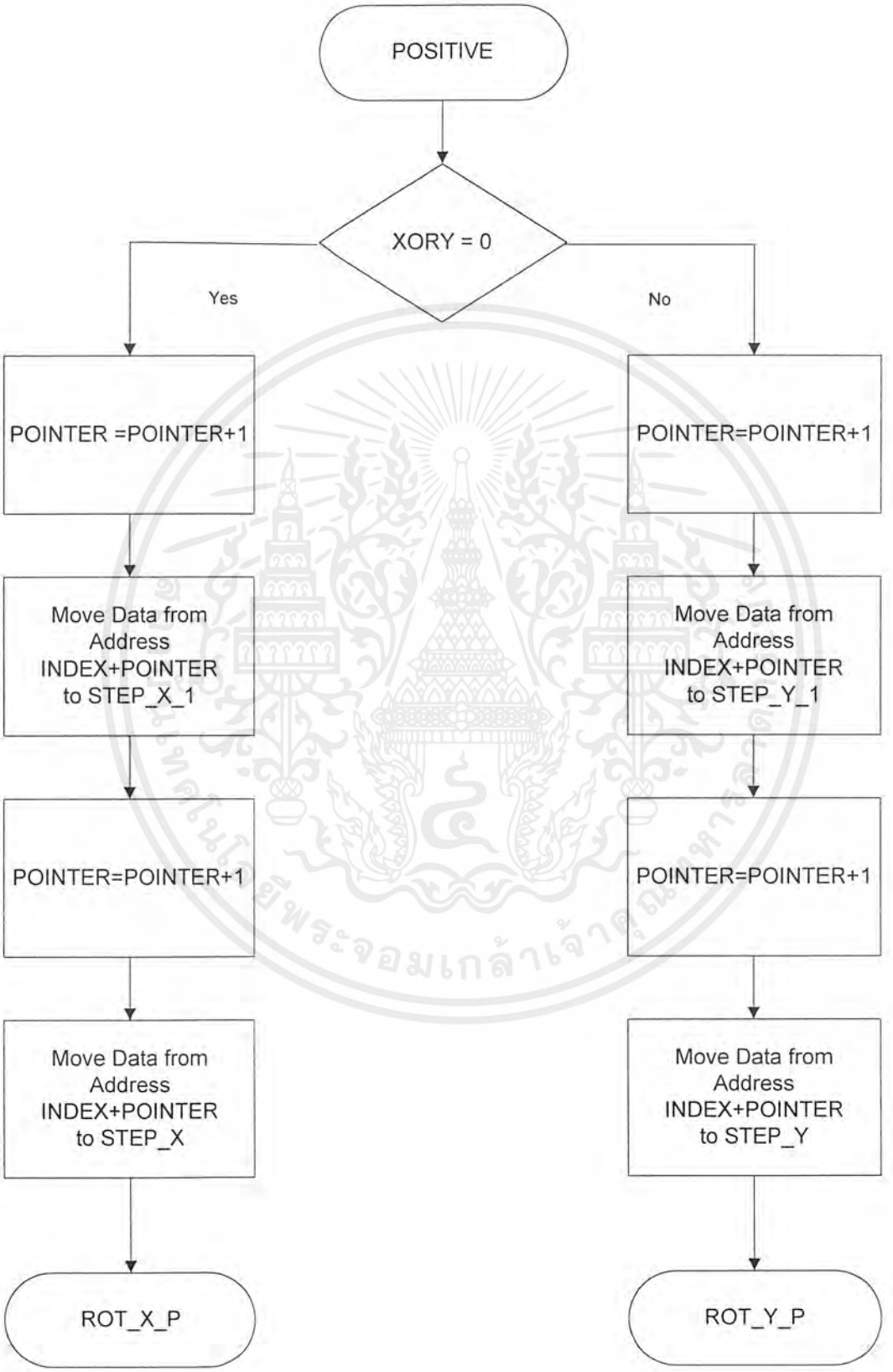
รูปที่ 5.8 ไดอะแกรมแสดงการทำงานของโปรแกรมควบคุมทางคอมพิวเตอร์



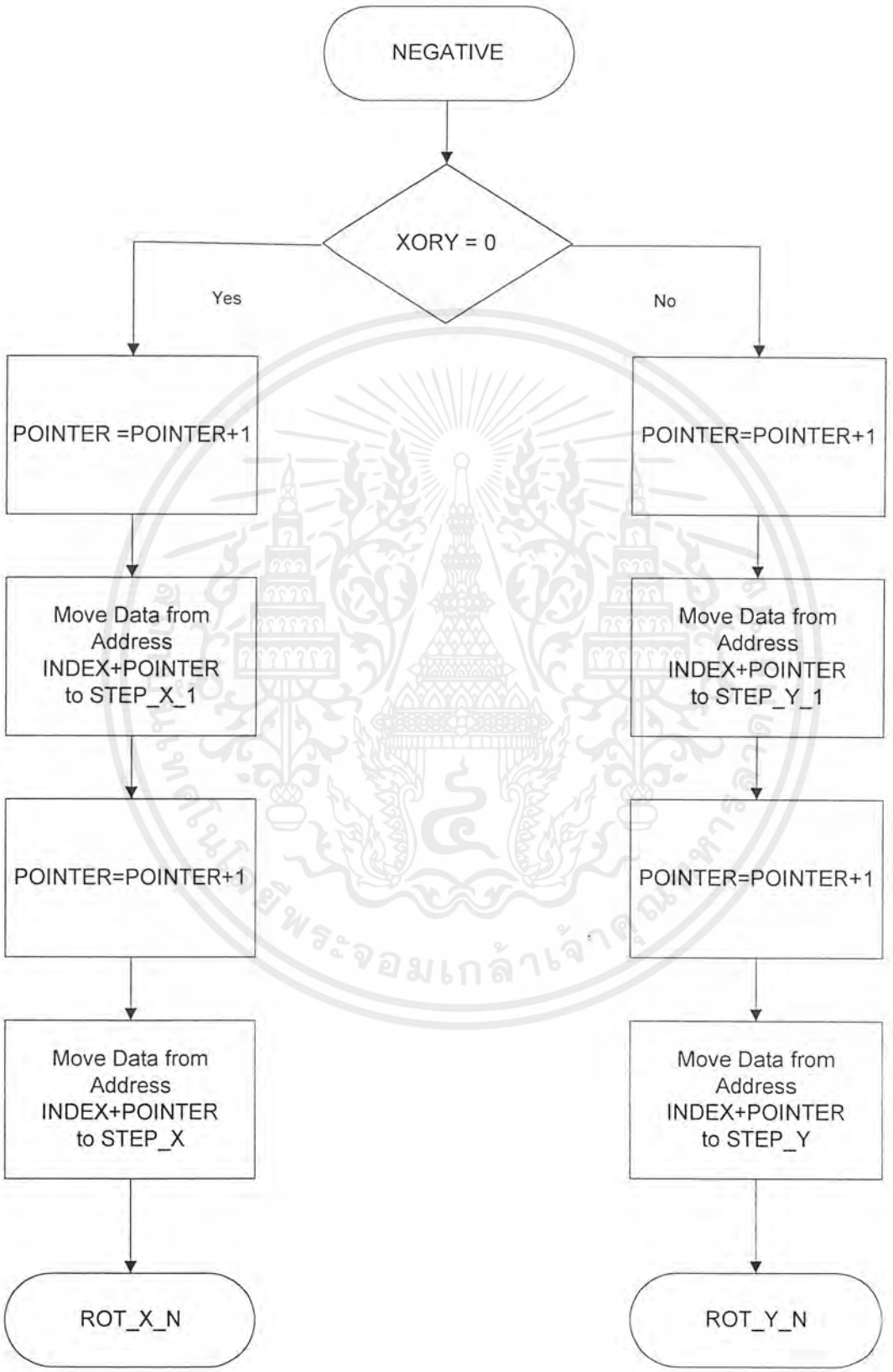
รูปที่ 5.9 ไดอะแกรมการทำงานเริ่มต้นใน MCS 51



รูปที่ 5.10 ไดอะแกรมการทำงานของลาเบล Main ใน MCS51



รูปที่ 5.11 ไดอะแกรมการทำงานของลาเบล POSITIVE ใน MCS 51



รูปที่ 5.11 ไดอะแกรมการทำงานของลาเบล NEGATIVE ใน MCS51

บทที่ 6
การทดลองและผลการทดลอง

6.1 การทดลอง

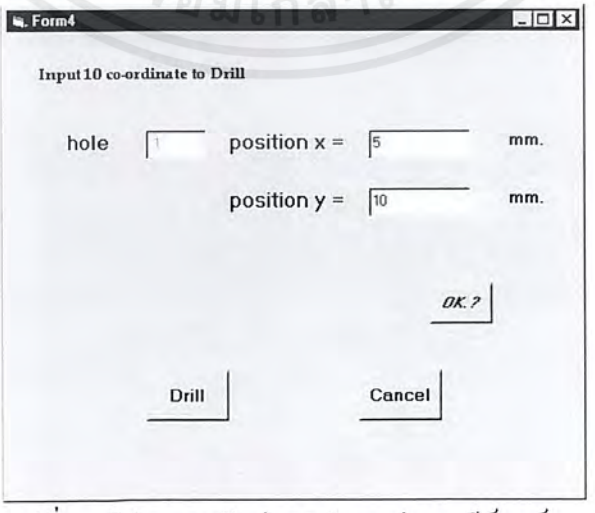
ในขั้นตอนการทดลองหลังจากที่ประกอบวงจรและกลไกเสร็จเรียบร้อยแล้วสามารถแบ่งการทดลองเป็น 2 ส่วนคือ

6.1.1 การทดลองป้อนค่าจาก คีย์บอร์ด 10 ค่า

ในขั้นตอนแรกของการปรับแต่งควรเลื่อนมอเตอร์ทั้งสามแกน ออกมาจากเซ็นเซอร์ ทั้ง 3 โดยในตอนเริ่มต้นการทำงานมอเตอร์ทั้ง 3 จะหมุนเคลื่อนที่เข้าหา เซ็นเซอร์เพื่อเป็นการหาเฟสที่ต้องการกระตุ้นและเป็นการเซตจุดเริ่มต้นการทำงานด้วย

หลังจากนั้นเปิดโปรแกรมควบคุมโดยเข้าโปรแกรม PCB.EXE ดังรูปที่ 6.1

รูปที่ 6.1 โปรแกรมควบคุมการทำงานของเครื่องเจาะแผ่นปริ๊นท์อัตโนมัติ



รูปที่ 6.2 โปรแกรมรับค่า 10 ตำแหน่งทางคีย์บอร์ด

แล้วเลือกส่วนโปรแกรม Manual ดังรูปที่ 6.2 แล้วเซตค่าตำแหน่งที่ต้องการเจาะหน่วยเป็นมิลลิเมตร 10 ค่า โดย 5 ค่าแรกให้เปลี่ยนแปลงแต่แกน X และ 5 ค่าสุดท้ายให้เปลี่ยนแปลงแต่แกน Y หลังจาก เซตครบแล้วก็ทำการส่งข้อมูลออกไปที่เครื่องเจาะเพื่อให้เครื่องเจาะทำงานแล้วทำการสังเกต ตำแหน่งที่เครื่องเลื่อนไปเจาะกับค่าที่ตั้งไว้ในโปรแกรม

6.1.2 การทดลองป้อน Text file จาก protel ในโปรแกรมเจาะอัตโนมัติ

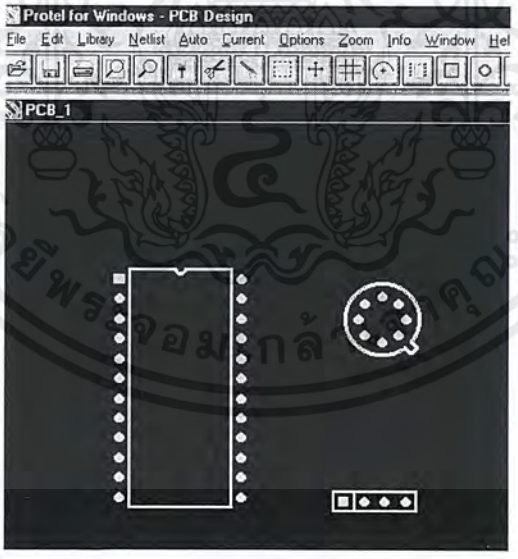
เริ่มต้นจากการวาง component ในโปรแกรม Protel ทั้งหมด 3 ตัว ดังนี้

SIP4 ขนาด รู 28 mil

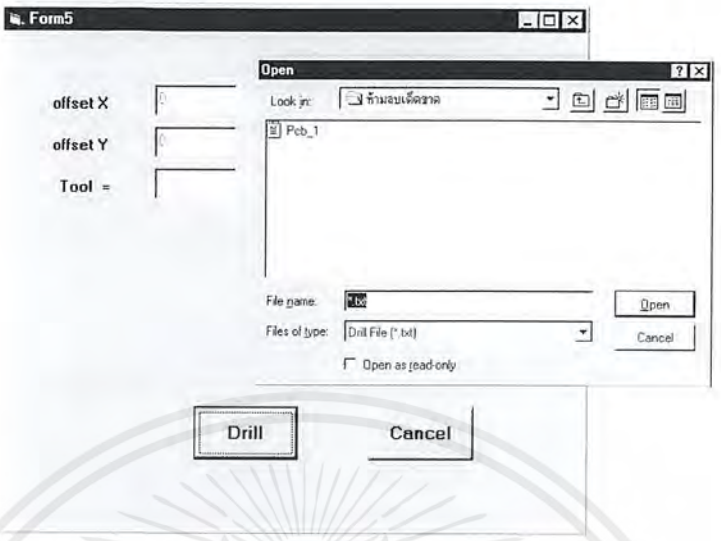
CAN8ขนาด รู 38 mil

DIP24 ขนาด รู 32 mil

ซึ่งวางอุปกรณ์ดังรูปที่6.3 แล้วทำการแปลงเป็น Textfile โดยใช้คำสั่ง NCDrill หลังจากนั้นก็เปิดโปรแกรมควบคุมโดยเลือกส่วนอัตโนมัติดังรูปที่ 6.4แล้วทำการเจาะ โดยการ เปิด Text file นั้นเครื่องก็จะทำการเจาะจนครบทุกรูและสังเกตผลที่ได้และนำมาเปรียบเทียบกับ file ในโปรแกรม Protel



รูปที่6.3 component ในโปรแกรม Protel ที่ใช้ทดสอบ



รูปที่ 6.4 โปรแกรมควบคุมการทำงานส่วนเจาะอัตโนมัติ

6.2 ผลการทดลอง

6.2.1 ผลการทดลองจากการป้อนค่าจากคีย์บอร์ด 10 ค่า

หลังจากทดลองแล้วสังเกตพบว่า รูที่เจาะ 5 รูแรกเนื่องจากให้เปลี่ยนแปลงแต่ แกน X ฉะนั้นลักษณะรู ต้องเรียงกันไปขนานกับแกน X แต่ปรากฏว่า เอียงจากแนวแกน X เป็นมุม 4 องศา แต่ค่าระยะห่างระหว่างรูที่วัดได้ตรงกับค่าที่ตั้งไว้ ส่วนรูที่เจาะ 5 รูสุดท้ายเนื่องจากเปลี่ยนแปลงแต่แกน Y รูที่เครื่องเจาะได้ ก็เกิดการเอียงทำมุมเป็นมุม 4 องศาจากแนวแกน Y แต่เมื่อวัดระยะระหว่างรูพบว่าถูกต้องตามที่กำหนดไว้ในการทดลอง

6.2.2 ผลการทดลองป้อน textfile จาก protel ในโปรแกรมเจาะอัตโนมัติ

พบว่าเครื่องจะเริ่มเจาะที่ SIP4 ก่อนเพราะเป็นขนาดรูที่เล็กที่สุดตามด้วย CAN 8 และ DIP24 ตามลำดับ ซึ่งจะเริ่มเจาะจากรูที่เล็กที่สุดไปหารูที่ใหญ่ที่สุดเมื่อเจาะเรียบร้อยแล้วทำการเปรียบเทียบตำแหน่งรูที่เจาะเกิดการคลาดเคลื่อนเล็กน้อยแต่เมื่อวัดระยะระหว่างรูพบว่าถูกต้อง

บทที่ 7
ผลวิจารณ์และสรุปผล

ภาคกลไก

หลังจากออกแบบแล้วทำการประกอบนำมาทดลองพบว่าส่วนการเคลื่อนที่ของ แกนX และแกน Y เคลื่อนที่ในแนวที่ไม่ตั้งฉากกัน กับระบบที่ออกแบบไว้เมื่อนำมาเจาะแผ่นปริ๊นท์ ตำแหน่งที่เจาะได้ไม่ตรงกับตำแหน่งที่ต้องการเจาะ โดยคลาดเคลื่อนเล็กน้อยซึ่งถ้าแก้ปัญหาส่วนระบบทางกลนี้จะทำให้การเจาะมีความแม่นยำขึ้นมาก

ปัญหาอีกประการหนึ่งคือส่วนทางกลนี้มีน้ำหนักมากเนื่องจากวัสดุที่ใช้เป็นเหล็กซึ่งราคาถูกกว่า แนวทางแก้ไขคือลองเปลี่ยนเป็นวัสดุชนิดอื่นเช่นอลูมิเนียมที่มีน้ำหนักลดลง

ภาควงจรควบคุม

วงจรในการขับมอเตอร์แกน X , Y เนื่องจากการขับแบบไมโครสเต็ปมอเตอร์ 1 ตัวต้องใช้สายสัญญาณ 10 เส้น มอเตอร์ 2 ตัวจำเป็นต้องใช้ สัญญาณ 20 เส้น แต่เมื่อพิจารณาว่าสามารถขับมอเตอร์ได้ทีละตัว จึงใช้บัฟเฟอร์ 74HC573 มาเป็นตัวสวิทช์ มอเตอร์ให้ทำงานทีละตัว ซึ่งมีข้อดีคือราคาถูกกว่าวงจรขยายพอร์ท 8255 มาก แต่มีข้อเสียคือ ต้องใช้บัฟเฟอร์4 ตัวทำให้วงจรรวมมีขนาดใหญ่ขึ้น

วงจรขับไฟเลี้ยง 12 โวลท์ ให้มอเตอร์แกน Z เนื่องการเคลื่อนที่แกน Z กินโหลดมากทำให้วงจรมีความร้อนสูง เมื่อทำงานได้ระยะหนึ่งต้องพักเสียก่อน ซึ่งสามารถแก้ปัญหาโดยการออกแบบวงจรให้ไอซีเรกกูเรเตอร์ให้สามารถกินกระแสได้มากขึ้น

ปัญหาอีกประการคือวงจรทั้งหมดต่อแยกกันจึงทำให้มีขนาดใหญ่เทอะทะ สามารถแก้ปัญหาโดยการนำวงจรทั้งหมดมาออกแบบลายวงจรรวมกันจะทำให้ขนาดเล็กลงมาก ในส่วนการทำงานของวงจรอื่นทำงานค่อนข้างราบรื่น

ภาคโปรแกรมควบคุม

สามารถรับข้อมูล Textfile จากโปรแกรม Protel แล้วทำการเจาะรูได้ตามโปรแกรมที่เขียนไว้ สามารถแยกรูเป็นแต่ละขนาดได้แต่ไม่สามารถบ่งบอกได้ว่ามีขนาดเท่าไร ซึ่งในอนาคตอาจพัฒนาให้สามารถบอกขนาดได้และสามารถปรับความเร็วของการเจาะได้

ภาคผนวก

โปรแกรมส่วนควบคุมเจาะไฟล์จากโปรแกรม Protel ทางคอมพิวเตอร์

general

```
Dim filecheck As Integer
Dim datavar As String
Dim checkstr As String
Dim recstr, recstr2 As String
Dim t() As Long
Dim maxhole(1 To 4) As Integer
Dim checkxy As String
Dim numarray(1 To 4, 1 To 2) As Integer
Dim i, i2, i3, i4, i5, tcheck As Integer
Dim max, coun1, coun2, longcheck As Integer
Dim REF_X, REF_Y, j, k, dx, dy As Long
Dim buffer As Variant
Dim a ,b,o,g As String
Dim s As Integer
Dim c, h As Long

Private Sub Command1_Click()
Dim pulse_x   As Long
Dim pulse_y As Long
Dim d As String
Dim e As String
Dim byteTosend(0 To 1) As Byte

error:
CommonDialog1.ShowOpen
If (CommonDialog1.FileName = "") Or (CommonDialog1.FileName = "*.txt") Then
MsgBox ("Please choose file again")
GoTo error:
End If
filecheck = FreeFile
Open CommonDialog1.FileName For Input As #filecheck
Do While Not datavar <> "" 'find m48
```

```
  Line Input #filecheck, datavar
Loop
If datavar <> "M48" Then
MsgBox ("This file is not for this program.")
If MsgBox("Do you want to continue.", 68, "Information") = 6 Then
Close #filecheck
GoTo error:
Else

Close #filecheck
Form1.Show
Unload Form5
GoTo no:
End If
End If

i = 0
Do While Not datavar = "%" 'How many T
Line Input #filecheck, datavar
If datavar <> "" Then
If datavar <> "%" Then
i = i + 1
End If
End If 'Now i have T num
Loop
Text3.Text = Str(i)

Do While Not EOF(filecheck) 'gen array
Line Input #filecheck, datavar
i2 = InStr(1, datavar, "T")
If i2 >= 1 Then
tcheck = tcheck + 1

   i5 = 0
```

```
End If

  checkxy = Left$(datavar, 1)
  If checkxy = "X" Or checkxy = "Y" Then
    i5 = i5 + 1
  End If
  maxhole(tcheck) = i5
Loop
  Close #filecheck
  'Find max array


ReDim t(1 To i, 1 To 2, 1 To 500) As Long 'gen array finnis
'Add to array
tcheck = 0
coun1 = 1
coun2 = 1

filecheck = FreeFile       ' bring value to array
Open CommonDialog1.FileName For Input As #filecheck
Do While Not EOF(filecheck)
Line Input #filecheck, datavar
i2 = InStr(1, datavar, "T0")   ' check tool
If i2 >= 1 Then
tcheck = tcheck + 1          ' count tool
coun2 = 1
End If
i3 = InStr(1, datavar, "X")
i4 = InStr(1, datavar, "Y")
If i3 > 0 And i4 > 0 Then
  longcheck = Len(datavar)
 recstr = Mid$(datavar, i3 + 1, i4 - i3 - 1)
  If Len(recstr) < 5 Then
    Do While Not Len(recstr) = 5
```

```
    recstr = recstr + "0"
    Loop
   End If
  recstr2 = Right$(datavar, longcheck - i4)
   If Len(recstr2) < 5 Then
  Do While Not Len(recstr2) = 5
  recstr2 = recstr2 + "0"
  Loop
  End If
 t(tcheck, 1, coun2) = Val(recstr)
 t(tcheck, 2, coun2) = Val(recstr2)
       'Debug.Print t(tcheck, 1, coun2)
       'Debug.Print t(tcheck, 2, coun2)
 coun2 = coun2 + 1
GoTo pass:
End If
If i3 > 0 Or i4 > 0 Then
 If i3 > 0 Then
 recstr = Mid$(datavar, 2, Len(datavar) - 1)
 If Len(recstr) < 5 Then
 Do While Not Len(recstr) = 5
 recstr = recstr + "0"
 Loop
 End If
 t(tcheck, 1, coun2) = Val(recstr)
 t(tcheck, 2, coun2) = t(tcheck, 2, coun2 - 1)
coun2 = coun2 + 1
 End If
 If i4 > 0 Then
 recstr2 = Mid$(datavar, 2, Len(datavar) - 1)
 If Len(recstr2) < 5 Then
 Do While Not Len(recstr2) = 5
 recstr2 = recstr2 + "0"
 Loop
```

```
  End If
  t(tcheck, 1, coun2) = t(tcheck, 1, coun2 - 1)
  t(tcheck, 2, coun2) = Val(recstr2)
coun2 = coun2 + 1
  End If
End If
pass:
Loop
Close #filecheck

   REF_X = t(1, 1, 1)  ' reference point of x
   REF_Y = t(1, 2, 1)  ' reference point of y


For j = 1 To i                ' change tool
   For k = 1 To maxhole(j)    ' change hole
      If (j = 1) And (k = 1) Then
      GoTo Shaq:
      End If

      If (j > 1) And (k = 1) Then
         dx = t(j, 1, 1) - REF_X + (40 * Form3.x)
         dy = t(j, 2, 1) - REF_Y + (40 * Form3.y)

      End If
      If (j = 1) And (k > 2) Then
         dx = t(j, 1, k) - t(j, 1, k - 1)
         dy = t(j, 2, k) - t(j, 2, k - 1)

      End If      ' end of finding dx & dy

      If (j = 1) And (k = 2) Then

    dx = t(j, 1, 2) - REF_X + (40 * Form3.x)
```

```
dy = t(j, 2, 2) - REF_Y + (40 * Form3.y)

End If

If (j > 1) And (k > 1) Then
  dx = t(j, 1, k) - t(j, 1, k - 1)
  dy = t(j, 2, k) - t(j, 2, k - 1)

End If

pulse_x = Abs(dx) * 0.2546 * 2
pulse_y = Abs(dy) * 0.2546 * 2  ' amount of pulse
 If dx >= 0 Then     ' start to find + or -
    buffer = "P"
    MSComm1.Output = buffer
  Else

    buffer = "N"
  MSComm1.Output = buffer
  End If

a = bin(pulse_x)          ' start to send y data

  If Len(a) <= 8 Then
    byteTosend(0) = &H0
   byteTosend(1) = dec(a)
   buffer = byteTosend()
    MSComm1.Output = buffer
  End If
  If Len(a) > 8 Then
    d = Right(a, 8)
    byteTosend(1) = dec(d)
    c = Left(a, Len(a) - 8)
    byteTosend(0) = dec(c)
```

```
        buffer = byteTosend()
        MSComm1.Output = buffer
    End If      ' end of send x data
    If dy >= 0 Then     ' send + or - of y axis
      buffer = "P"
      MSComm1.Output = buffer
      Else
     buffer = "N"
      MSComm1.Output = buffer
      End If          ' end of send
   b = bin(pulse_y)
    If Len(b) <= 8 Then
        byteTosend(0) = &H0
        byteTosend(1) = dec(b)
        buffer = byteTosend
        MSComm1.Output = buffe
      End If
      If Len(b) > 8 Then
        d = Right(b, 8)
        byteTosend(1) = dec(d)
        e = Left(b, Len(b) - 8)
        byteTosend(0) = dec(e)
        buffer = byteTosend()
        MSComm1.Output = buffer
      End If
   If  k = 30 Then
    buffer = Chr(&HFF)
    MSComm1.Output = buffer
MsgBox ("waiting for machine stop and then continue")
    End If
  If k = maxhole(j) Then
  buffer = Chr(&HFE)
    MSComm1.Output = buffer
```

```
    MsgBox ("drilling the " + Str(j) + " tool  waiting!! and then change tool")
   End If
  If (j = i) And (k = maxhole(i)) Then
    buffer = Chr(&HFD)
    MSComm1.Output = buffer
  End If
Shaq:
    Next k
 Next j
 no:
End Sub

Private Sub Command2_Click()
Form1.Show
Unload Form5
End Sub

Public Function bin(dValue As Long) As String
  Dim result As String
  Dim remainder As Long
  result = ""
  Do
    remainder = dValue - (2 * Int(dValue / 2))
    result = Str(remainder) & result
    dValue = Int(dValue / 2)
    result = Trim(result)
  Loop While (dValue > 0)
done:
  bin = result
  End Function

Public Function dec(bValue As String) As Long
  Dim dit As String
```

```
  Dim j1 As Integer
  Dim result As Long
  Dim remainder As Long
  Dim index As Long
   result = 0
    For j1 = 1 To Len(bValue)
      dit = Mid(bValue, j1, 1)
      index = Val(dit) * (2 ^ (Len(bValue) - j1))
      result = result + index
   dec = result
   Next j1
done:
   dec = result

End Function

Private Sub Command3_Click()
 Form1.Show
 Unload Form5
End Sub

Private Sub Form_Load()
 MSComm1.CommPort = 1
 MSComm1.PortOpen = True
Text1.Text = Form3.x
Text2.Text = Form3.y
End Sub

Private Sub Form_Unload(Cancel As Integer)
   MSComm1.PortOpen = False
End Sub
```

```
;***********************************
; Define User Register

;***********************************
POINTER     EQU   31H
CHECK       EQU   32H
STORE_DATA  EQU   34H
XORY        EQU   35H
PNEOF       EQU   36H
STEP_X      EQU   37H
STEP_X_1    EQU   38H
STEP_Y      EQU   39H
STEP_Y_1    EQU   3AH
STEP_M      EQU   3BH
STEP_M_1    EQU   3CH
CTPORT_X    EQU   3DH
CTPORT_Y    EQU   3FH
STEP_Z      EQU   50H
STEP_Z_1    EQU   51H
CTPORT_Z    EQU   52H
INDEX       EQU   54H
;***********************************
        ORG   0000H
START:
        CLR   P1.0
        CLR   P1.1
        CLR   P1.2
        ACALL HOME
;***********************************
        MOV   TMOD,#021H
; T1 8BIT T0 16 BIT
        MOV   TH1,#0FDH
; 9600 BPS TIMER1
        MOV   TL1,#0FDH
        MOV   IE,#10010000B
; EA ES ENABLE
        SETB  TR1
; START TIMER1
        MOV   SCON,#040H
; SELECT SERIAL TIMER MODE 1
        SETB  REN
BEFORE:
        MOV   POINTER,#0
        MOV   CHECK,#0
GET_DATUM:
        CLR   RI
        JNB   RI,$
        MOV   STORE_DATA,SBUF
        MOV   A,POINTER
        MOV   R0,#INDEX
        ADD   A,R0
        MOV   R0,A
        MOV   @R0,STORE_DATA
        INC   POINTER
        MOV   A,CHECK
        CJNE  A,#0,GET_DATUM_1
        MOV   A,STORE_DATA
        CJNE A,#0FEH,GET_DATUM_1
        ;endoffile คือ FEH
        AJMP    MAIN
GET_DATUM_1:
        CJNE
        A,#0FFH,GET_DATUM_2
        AJMP    MAIN
GET_DATUM_2:
        CJNE
        A,#0FDH,GET_DATUM_3
```

```
        AJMP    MAIN
GET_DATUM_3:
        INC     CHECK
        MOV     A,CHECK
        CJNE    A,#3,CLEAR_CHECK
        AJMP    GET_DATUM
CLEAR_CHECK:
        MOV     CHECK,#0
        AJMP    GET_DATUM
;*****************************************
MAIN:
        MOV     XORY,#0
        MOV     POINTER,#0
        CLR     P3.2
MAIN_1:
        MOV     A,XORY
        CJNE    A,#2,MAIN_3
        MOV     XORY,#0
MAIN_3:
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     PNEOF,@R0
        MOV     A,PNEOF
        CJNE    A,#50H,MAIN_4
        AJMP    POSITIVE
MAIN_4:
        CJNE    A,#4EH,MAIN_5
        AJMP    NEGATIVE
MAIN_5:
        CJNE    A,#0FFH,MAIN_6
        SETB    P3.2
        AJMP    BEFORE
MAIN_6:
        CJNE    A,#0FDH,MAIN_7
        SETB    P3.2
        ACALL   HOME_X
        ACALL   HOME_Y
        ACALL   ROT_Z
        ACALL   ROT_Z_E
        AJMP    OVER_1
MAIN_7:
        SETB    P3.2
        AJMP    OVER
;*************************************
POSITIVE:
        MOV     A,XORY
        CJNE    A,#0,POSITIVE_Y
        AJMP    POSITIVE_X
;*****************************************
POSITIVE_X:
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     STEP_X_1,@R0
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     STEP_X,@R0
        SETB    P1.0
        ACALL   ROT_X_P
        CLR     P1.0
```

```
        INC     POINTER
        INC     XORY
        AJMP    MAIN_1
;*****************************************
POSITIVE_Y:
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     STEP_Y_1,@R0
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     STEP_Y,@R0
        SETB    P1.1
        ACALL   ROT_Y_P
        CLR     P1.1
        SETB    P1.2
        ACALL   ROT_Z
        ACALL   ROT_Z_1
        ACALL   HOME_Z
        CLR     P1.2
        INC     POINTER
        INC     XORY
        AJMP    MAIN_1
;*****************************************
NEGATIVE:
        MOV  A,XORY
        CJNE  A,#0,NEGATIVE_Y
        AJMP  NEGATIVE_X
;*****************************************
NEGATIVE_X:
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     STEP_X_1,@R0
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     STEP_X,@R0
        SETB    P1.0
        ACALL   ROT_X_N
        CLR     P1.0
        INC     POINTER
        INC     XORY
        AJMP    MAIN_1
;*****************************************
NEGATIVE_Y:
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
        MOV     STEP_Y_1,@R0
        INC     POINTER
        MOV     A,POINTER
        MOV     R0,#INDEX
        ADD     A,R0
        MOV     R0,A
```

```
        MOV     STEP_Y,@R0
        SETB    P1.1
        ACALL    ROT_Y_N
        CLR     P1.1
        SETB    P1.2
        ACALL   ROT_Z
        ACALL   ROT_Z_1
        ACALL   HOME_Z
        CLR     P1.2
        INC     POINTER
        INC     XORY
        AJMP    MAIN_1
;*****************************************
ROT_X_P:
        MOV     STEP_M,STEP_X
        MOV     STEP_M_1,STEP_X_1
        MOV     A,STEP_M
        CJNE    A,#0,GO_X_P
NOT_GO_X_P:
        MOV     A,STEP_M_1
        CJNE    A,#0,GO_X_P
        AJMP    EROT_X_P
GO_X_P:
        MOV     A,CTPORT_X
CJNE    A,#0FFH,NOT_SIXTEEN_X_P
        MOV     CTPORT_X,#15
NOT_SIXTEEN_X_P:
        MOV     A,CTPORT_X
        MOV     DPTR,#PORT0
        MOVC    A,@A+DPTR
        MOV     P0,A
        MOV     A,CTPORT_X
        MOV     DPTR,#PORT2
        MOVC    A,@A+DPTR
        MOV     P2,A
        DEC     CTPORT_X
        ACALL   RETRIVE_STEPM
        MOV     A,STEP_M_1
        CJNE    A,#0FFH,X_P
        JMP     EROT_X_P
X_P:
        ACALL   DELAY_5ms
;SET DELAY TIME
        AJMP    GO_X_P
EROT_X_P:
        RET
;*****************************************
ROT_Y_P:
        MOV     STEP_M,STEP_Y
        MOV     STEP_M_1,STEP_Y_1
        MOV     A,STEP_M
        CJNE    A,#0,GO_Y_P
NOT_GO_Y_P:
        MOV     A,STEP_M_1
        CJNE    A,#0,GO_Y_P
        AJMP    EROT_Y_P
GO_Y_P:
        MOV     A,CTPORT_Y
CJNE    A,#16,NOT_SIXTEEN_Y_P
        MOV     CTPORT_Y,#0
NOT_SIXTEEN_Y_P:
        MOV     A,CTPORT_Y
        MOV     DPTR,#PORT0
        MOVC    A,@A+DPTR
        MOV     P0,A
        MOV     A,CTPORT_Y
        MOV     DPTR,#PORT2
        MOVC    A,@A+DPTR
```

```
        MOV     P2,A
        INC     CTPORT_Y
        ACALL   RETRIVE_STEPM
        MOV     A,STEP_M_1
        CJNE    A,#0FFH,Y_P
        JMP     EROT_Y_P
Y_P:
        ACALL   DELAY_5ms
        AJMP    GO_Y_P
EROT_Y_P:
        RET
;*****************************************
ROT_X_N:
        MOV     STEP_M,STEP_X
        MOV     STEP_M_1,STEP_X_1
        MOV     A,STEP_M
        CJNE    A,#0,GO_X_N
NOT_GO_X_N:
        MOV     A,STEP_M_1
        CJNE    A,#0,GO_X_N
        AJMP    EROT_X_N
 GO_X_N:
        MOV     A,CTPORT_X
CJNE  A,#16,NOT_SIXTEEN_X_N
        MOV  CTPORT_X,#0
NOT_SIXTEEN_X_N:
        MOV     A,CTPORT_X
        MOV     DPTR,#PORT0
        MOVC    A,@A+DPTR
        MOV     P0,A
        MOV     A,CTPORT_X
        MOV     DPTR,#PORT2
        MOVC    A,@A+DPTR
        MOV     P2,A
        INC     CTPORT_X
        ACALL   RETRIVE_STEPM
        MOV     A,STEP_M_1
        CJNE    A,#0FFH,X_N
        JMP     EROT_X_N
X_N:
        ACALL   DELAY_5ms
;SET DELAY TIME
        AJMP    GO_X_N
EROT_X_N:
        RET
;*****************************************
ROT_Y_N:
        MOV     STEP_M,STEP_Y
        MOV     STEP_M_1,STEP_Y_1
        MOV     A,STEP_M
        CJNE    A,#0,GO_Y_N
NOT_GO_Y_N:
        MOV     A,STEP_M_1
        CJNE    A,#0,GO_Y_N
        AJMP    EROT_Y_N
 GO_Y_N:
        MOV     A,CTPORT_Y
CJNE  A,#0FFH,NOT_SIXTEEN_Y_N
        MOV     CTPORT_Y,#15
NOT_SIXTEEN_Y_N:
        MOV     A,CTPORT_Y
        MOV     DPTR,#PORT0
        MOVC    A,@A+DPTR
        MOV     P0,A
        MOV     A,CTPORT_Y
        MOV     DPTR,#PORT2
        MOVC    A,@A+DPTR
        MOV     P2,A
```

```
        DEC     CTPORT_Y
        ACALL   RETRIVE_STEPM
        MOV     A,STEP_M_1
        CJNE    A,#0FFH,Y_N
        JMP     EROT_Y_N
Y_N:
        ACALL   DELAY_5ms
;SET DELAY TIME
        AJMP    GO_Y_N
EROT_Y_N:
        RET
;*****************************************
PORT0:
DB31,30,27,22,0,6,11,14,15,14,11,6,16,
22,27,30
PORT2:
DB16,22,27,30,31,30,27,22,0,6,11,14,1
5,14,11,6
PORT1: DB 156,180,228,204
;*****************************************
ROT_Z:
        MOV     STEP_Z,#90
        MOV     STEP_Z_1,#0
        MOV     STEP_M,STEP_Z
        MOV     STEP_M_1,STEP_Z_1
        MOV     A,STEP_M
        CJNE    A,#0,GO_Z
NOT_GO_Z:
        MOV     A,STEP_M_1
        CJNE    A,#0,GO_Z
        AJMP    EROT_Z
GO_Z:
        MOV     A,CTPORT_Z
        CJNE    A,#4,NOT_SIXTEEN_Z
        MOV     CTPORT_Z,#0
NOT_SIXTEEN_Z:
        MOV     A,CTPORT_Z
        MOV     DPTR,#PORT1
        MOVC    A,@A+DPTR
        MOV     P1,A
        INC     CTPORT_Z
        ACALL   RETRIVE_STEPM
        MOV     A,STEP_M_1
        CJNE    A,#0FFH,Z
        JMP     EROT_Z
Z:
        ACALL   DELAY_10ms
;SET DELAY TIME
        AJMP    GO_Z
EROT_Z:
        RET
;*****************************************
ROT_Z_1:
        MOV     STEP_Z,#30
        MOV     STEP_Z_1,#0
        MOV     STEP_M,STEP_Z
        MOV     STEP_M_1,STEP_Z_1
        MOV     A,STEP_M
        CJNE    A,#0,GO_Z_1
NOT_GO_Z_1:
        MOV     A,STEP_M_1
        CJNE    A,#0,GO_Z_1
        AJMP    EROT_Z_1
GO_Z_1:
        MOV     A,CTPORT_Z
CJNE    A,#4,NOT_SIXTEEN_Z_1
        MOV     CTPORT_Z,#0
NOT_SIXTEEN_Z_1:
```

```
        MOV     A,CTPORT_Z
        MOV     DPTR,#PORT1
        MOVC    A,@A+DPTR
        MOV     P1,A
        INC     CTPORT_Z
        ACALL   RETRIVE_STEPM
        MOV     A,STEP_M_1
        CJNE    A,#0FFH,Z_1
        JMP     EROT_Z_1
Z_1:
        ACALL   DELAY_100ms
;SET DELAY TIME
        AJMP    GO_Z_1
EROT_Z_1:
        RET
;*****************************************
ROT_Z_E:
        MOV     STEP_Z,#20
        MOV     STEP_Z_1,#0
        MOV     STEP_M,STEP_Z
        MOV     STEP_M_1,STEP_Z_1
        MOV     A,STEP_M
        CJNE    A,#0,GO_Z_E
NOT_GO_Z_E:
        MOV     A,STEP_M_1
        CJNE    A,#0,GO_Z_E
        AJMP    EROT_Z_E

GO_Z_E:
        MOV     A,CTPORT_Z
        CJNE
        A,#4,NOT_SIXTEEN_Z_E
        MOV     CTPORT_Z,#0
NOT_SIXTEEN_Z_E:
        MOV     A,CTPORT_Z
        MOV     DPTR,#PORT1
        MOVC    A,@A+DPTR
        MOV     P1,A
        INC     CTPORT_Z
        ACALL   RETRIVE_STEPM
        MOV     A,STEP_M_1
        CJNE    A,#0FFH,Z_E
        JMP     EROT_Z_E
Z_E:
        ACALL   DELAY_100ms
;SET DELAY TIME
        AJMP    GO_Z_E
EROT_Z_E:
        RET
;*************************************
HOME:
        SETB    P1.2
        ACALL   HOME_Z
        CLR     P1.2
        SETB    P1.0
        ACALL   HOME_X
        CLR     P1.0
        SETB    P1.1
        ACALL   HOME_Y
        CLR     P1.1
        RET

;*****************************************
HOME_X:
        MOV  A,CTPORT_X
        CJNE A,#16,HOME_X_1
        MOV  CTPORT_X,#0
;*****************************************
```

```
HOME_X_1:
        MOV    A,CTPORT_X
        MOV  DPTR,#PORT0
        MOVC A,@A+DPTR
        MOV  P0,A
        MOV  A,CTPORT_X
        MOV  DPTR,#PORT2
        MOVC A,@A+DPTR
        MOV  P2,A
        INC  CTPORT_X
        JB   P3.5,HOME_X_2
        ACALL  DELAY_3ms
        AJMP HOME_X
HOME_X_2:
        RET
;*****************************************
HOME_Y:
        MOV  A,CTPORT_Y
        CJNE A,#0FFH,HOME_Y_1
        MOV  CTPORT_Y,#15
HOME_Y_1:
        MOV    A,CTPORT_Y
        MOV  DPTR,#PORT0
        MOVC A,@A+DPTR
        MOV  P0,A
        MOV  A,CTPORT_Y
        MOV  DPTR,#PORT2
        MOVC A,@A+DPTR
        MOV  P2,A
        DEC  CTPORT_Y
        JB   P3.6,HOME_Y_2
        ACALL  DELAY_3ms
        AJMP HOME_Y
HOME_Y_2:
        RET
;*****************************************
HOME_Z:
        MOV  A,CTPORT_Z
        CJNE A,#0FFH,HOME_Z_1
        MOV  CTPORT_Z,#3
HOME_Z_1:
        MOV    A,CTPORT_Z
        MOV  DPTR,#PORT1
        MOVC A,@A+DPTR
        MOV  P1,A
        DEC  CTPORT_Z
        JB   P3.7,HOME_Z_2
        ACALL  DELAY_5ms
        AJMP HOME_Z
HOME_Z_2:
        RET
;*****************************************
DELAY_100ms:  MOV 7,#100
 DELAY_100ms_1:MOV 6,#0E6H
 DELAY_100ms_2: NOP
                NOP
       DJNZ R6,DELAY_100ms_2
      DJNZ   R7,DELAY_100ms_1
       RET
DELAY_1ms:  MOV 7,#1
 DELAY_1ms_1:MOV 6,#0E6H
 DELAY_1ms_2: NOP
                NOP
   DJNZ   R6,DELAY_1ms_2
    DJNZ   R7,DELAY_1ms_1
   RET
DELAY_3ms:  MOV 7,#3
```

```
 DELAY_3ms_1:MOV  6,#0E6H
  DELAY_3ms_2: NOP
                NOP
   DJNZ   R6,DELAY_3ms_2
   DJNZ   R7,DELAY_3ms_1
   RET
DELAY_5ms:  MOV  7,#5
 DELAY_5ms_1:MOV  6,#0E6H
  DELAY_5ms_2: NOP
                NOP
   DJNZ   R6,DELAY_5ms_2
    DJNZ   R7,DELAY_5ms_1
    RET
DELAY_10ms:  MOV  7,#10
 DELAY_10ms_1:MOV  6,#0E6H
  DELAY_10ms_2: NOP
                NOP
     DJNZ   R6,DELAY_10ms_2
    DJNZ   R7,DELAY_10ms_1
     RET
;************************************
RETRIVE_STEPM_1:
   DEC  STEP_M_1
   RET

RETRIVE_STEPM:
    DEC   STEP_M
    MOVA,STEP_M
 CJNE   A,#0FFH,RETRIVE_STEPM_2
     JMP RETRIVE_STEPM_1
 RETRIVE_STEPM_2:
         RET
;************************************
OVER:
        AJMP    START
OVER_1:
        AJMP    OVER_1
END
```

National Semiconductor

September 1996

LMD18245
3A, 55V DMOS Full-Bridge Motor Driver

General Description

The LMD18245 full-bridge power amplifier incorporates all the circuit blocks required to drive and control current in a brushed type DC motor or one phase of a bipolar stepper motor. The multi-technology process used to build the device combines bipolar and CMOS control and protection circuitry with DMOS power switches on the same monolithic structure. The LMD18245 controls the motor current via a fixed off-time chopper technique.

An all DMOS H-bridge power stage delivers continuous output currents up to 3A (6A peak) at supply voltages up to 55V. The DMOS power switches feature low $R_{DS(ON)}$ for high efficiency, and a diode intrinsic to the DMOS body structure eliminates the discrete diodes typically required to clamp bipolar power stages.

An innovative current sensing method eliminates the power loss associated with a sense resistor in series with the motor. A four-bit digital-to-analog converter (DAC) provides a digital path for controlling the motor current, and, by extension, simplifies implementation of full, half and microstep stepper motor drives. For higher resolution applications, an external DAC can be used.

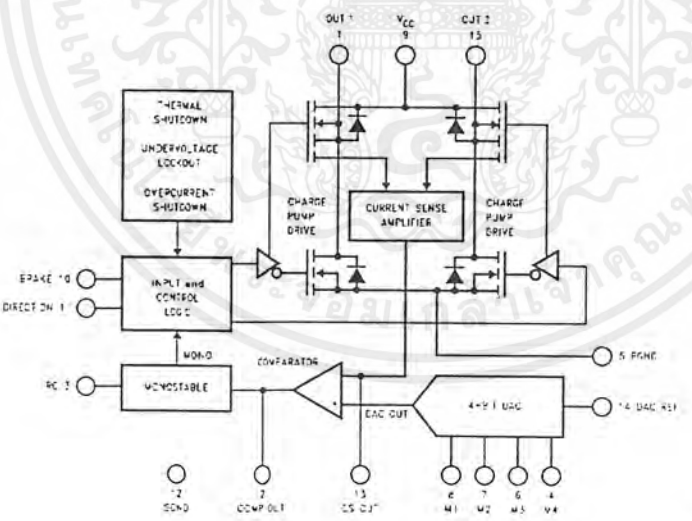
Features

- DMOS power stage rated at 55V and 3A continuous
- Low $R_{DS(ON)}$ of typically 0.3Ω per power switch
- Internal clamp diodes
- Low-loss current sensing method
- Digital or analog control of motor current
- TTL and CMOS compatible inputs
- Thermal shutdown (outputs off) at T_J = 155°C
- Overcurrent protection
- No shoot-through currents
- 15-lead TO-220 molded power package

Applications

- Full, half and microstep stepper motor drives
- Stepper motor and brushed DC motor servo drives
- Automated factory, medical and office equipment

Functional Block and Connection Diagram (15-Lead TO-220 Molded Power Package (T))



TL/H/11878–1

Order Number LMD18245T
See NS Package Number TA15A

RRD-B30M96/Printed in U. S. A.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

DC Voltage at:
- OUT 1, V_{CC}, and OUT 2: +60V
- COMP OUT, RC, M4, M3, M2, M1, BRAKE, DIRECTION, CS OUT, and DAC REF: +12V

DC Voltage PGND to SGND: ±400mV
Continuous Load Current: 3A
Peak Load Current (Note 2): 6A
Junction Temperature ($T_{J(max)}$): +150°C
Power Dissipation (Note 3):
- TO-220 (T_A = 25°C, Infinite Heatsink): 25W
- TO-220 (T_A = 25°C, Free Air): 3.5W

ESD Susceptibility (Note 4): 1500V
Storage Temperature Range (T_S): −40°C to +150°C
Lead Temperature (Soldering, 10 seconds): 300°C

Operating Conditions (Note 1)

Temperature Range (T_J) (Note 3): −40°C to +125°C
Supply Voltage Range (V_{CC}): +12V to +55V
CS OUT Voltage Range: 0V to +5V
DAC REF Voltage Range: 0V to +5V
MONOSTABLE Pulse Range: 10 µs to 100 ms

Electrical Characteristics

The following specifications apply for V_{CC} = +42V, unless otherwise stated. **Boldface limits apply over the operating temperature range, −40°C ≤ T_J ≤ +125°C.** All other limits apply for T_A = T_J = 25°C. (Note 2)

Symbol	Parameter	Conditions	Typical (Note 5)	Limit (Note 5)	Units (Limits)
I_{CC}	Quiescent Supply Current	DAC REF = 0V, V_{CC} = +20V	8	15	mA mA (max)
POWER OUTPUT STAGE					
$R_{DS(ON)}$	Switch ON Resistance	I_{LOAD} = 3A	0.3	0.4 **0.6**	Ω (max) Ω (max)
		I_{LOAD} = 6A	0.3	0.4 **0.6**	Ω (max) Ω (max)
V_{DIODE}	Body Diode Forward Voltage	I_{DIODE} = 3A	1.0	1.5	V V(max)
T_{rr}	Diode Reverse Recovery Time	I_{DIODE} = 1A	80		ns
Q_{rr}	Diode Reverse Recovery Charge	I_{DIODE} = 1A	40		nC
$t_{D(ON)}$	Output Turn ON Delay Time				
	Sourcing Outputs	I_{LOAD} = 3A	5		µs
	Sinking Outputs	I_{LOAD} = 3A	900		ns
$t_{D(OFF)}$	Output Turn OFF Delay Time				
	Sourcing Outputs	I_{LOAD} = 3A	600		ns
	Sinking Outputs	I_{LOAD} = 3A	400		ns
t_{ON}	Output Turn ON Switching Time				
	Sourcing Outputs	I_{LOAD} = 3A	40		µs
	Sinking Outputs	I_{LOAD} = 3A	1		µs
t_{OFF}	Output Turn OFF Switching Time				
	Sourcing Outputs	I_{LOAD} = 3A	200		ns
	Sinking Outputs	I_{LOAD} = 3A	80		ns
t_{pw}	Minimum Input Pulse Width	Pins 10 and 11	2		µs
t_{DB}	Minimum Dead Band	(Note 6)	40		ns

Electrical Characteristics The following specifications apply for V_{CC} = +42V, unless otherwise stated. **Boldface limits apply over the operating temperature range, −40°C ≤ T_J ≤ +125°C.** All other limits apply for T_A = T_J = 25°C. (Note 2) (Continued)

Symbol	Parameter	Conditions	Typical (Note 5)	Limit (Note 5)	Units (Limits)
CURRENT SENSE AMPLIFIER					
	Current Sense Output	I_{LOAD} = 1A (Note 7)	250	200 **175** 300 **325**	µA (min) µA (min) µA (max) µA (max)
	Current Sense Linearity Error	0.5A ≤ I_{LOAD} ≤ 3A (Note 7)	±6	±9	% %(max)
	Current Sense Offset	I_{LOAD} = 0A	5	20	µA µA (max)
DIGITAL-TO-ANALOG CONVERTER (DAC)					
	Resolution			**4**	Bits (min)
	Monotonicity			**4**	Bits (min)
	Total Unadjusted Error		0.125	0.25 **0.5**	LSB (max) LSB (max)
	Propagation Delay		50		ns
I_{REF}	DAC REF Input Current	DAC REF = +5V	−0.5	± **10**	µA µA (max)
COMPARATOR AND MONOSTABLE					
	Comparator High Output Level		6.27		V
	Comparator Low Output Level		88		mV
	Comparator Output Current Source Sink		 0.2 3.2		 mA mA
t_{DELAY}	Monostable Turn OFF Delay	(Note 8)	1.2	2.0	µs µs (max)
PROTECTION AND PACKAGE THERMAL RESISTANCES					
	Undervoltage Lockout, V_{CC}			5 8	V (min) V (max)
T_{JSD}	Shutdown Temperature, T_J		155		°C
θ_{JC} θ_{JA}	Package Thermal Resistances Junction-to-Case, TO-220 Junction-to-Ambient, TO-220		 1.5 35		 °C/W °C/W
LOGIC INPUTS					
V_{IL}	Low Level Input Voltage			−**0.1** **0.8**	V (min) V (max)
V_{IH}	High Level Input Voltage			**2** **12**	V (min) V (max)
I_{IN}	Input Current	V_{IN} = 0V or 12V		± **10**	µA (max)

Electrical Characteristics

The following specifications apply for V_{CC} = +42V, unless otherwise stated. **Boldface limits apply over the operating temperature range, −40°C ≤ T_J ≤ +125°C.** All other limits apply for T_A = T_J = 25°C. (Note 2) (Continued)

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Electrical specifications do not apply when operating the device outside the rated Operating Conditions.

Note 2: Unless otherwise stated, load currents are pulses with widths less than 2 ms and duty cycles less than 5%.

Note 3: The maximum allowable power dissipation at any ambient temperature is $P_{Max} = (125 - T_A)/\theta_{JA}$, where 125°C is the maximum junction temperature for operation, T_A is the ambient temperature in °C, and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. Exceeding P_{max} voids the Electrical Specifications by forcing T_J above 125°C. If the junction temperature exceeds 155°C, internal circuitry disables the power bridge. When a heatsink is used, θ_{JA} is the sum of the junction-to-case thermal resistance of the package, θ_{JC}, and the case-to-ambient thermal resistance of the heatsink.

Note 4: ESD rating is based on the human body model of 100 pF discharged through a 1.5 kΩ resistor. M1, M2, M3 and M4, pins 8, 7, 6 and 4 are protected to 800V.

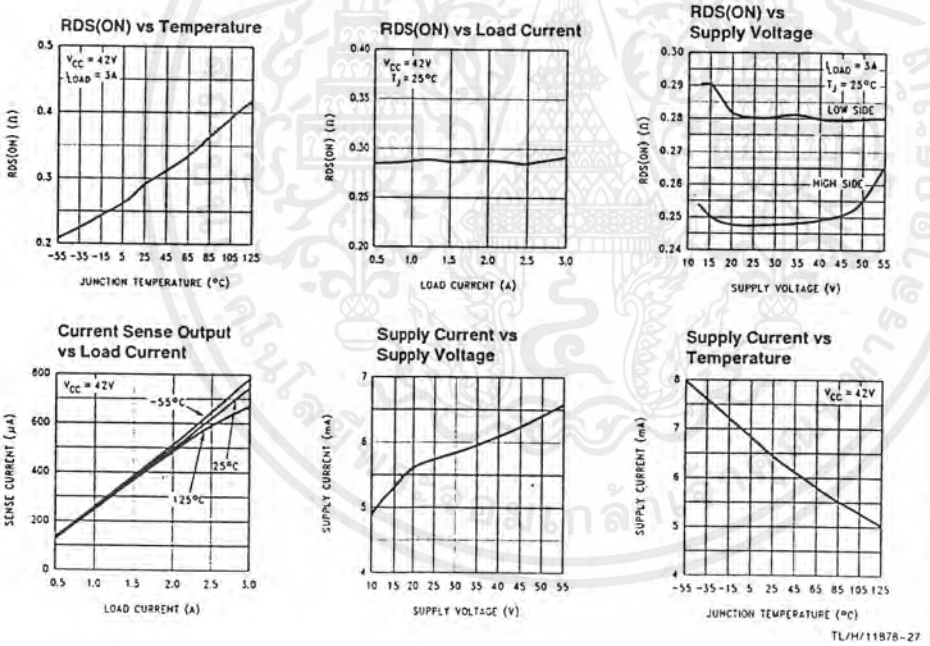
Note 5: All limits are 100% production tested at 25°C. Temperature extreme limits are guaranteed via correlation using accepted SQC (Statistical Quality Control) methods. All limits are used to calculate AOQL (Average Outgoing Quality Level). Typicals are at T_J = 25°C and represent the most likely parametric norm.

Note 6: Asymmetric turn OFF and ON delay times and switching times ensure a switch turns OFF before the other switch in the same half H-bridge begins to turn ON (preventing momentary short circuits between the power supply and ground). The transitional period during which both switches are OFF is commonly referred to as the dead band.

Note 7: (I_{LOAD}, I_{SENSE}) data points are taken for load currents of 0.5A, 1A, 2A and 3A. The current sense gain is specified as I_{SENSE}/I_{LOAD} for the 1A data point. The current sense linearity is specified as the slope of the line between the 0.5A and 1A data points minus the slope of the line between the 2A and 3A data points all divided by the slope of the line between the 0.5A and 1A data points.

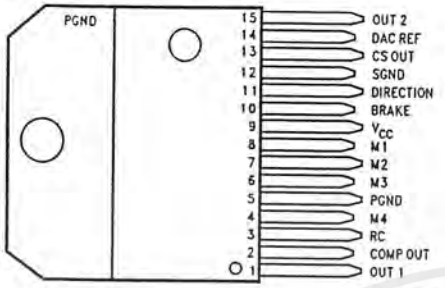
Note 8: Turn OFF delay, t_{DELAY}, is defined as the time from the voltage at the output of the current sense amplifier reaching the DAC output voltage to the lower DMOS switch beginning to turn OFF. With V_{CC} = 32V, DIRECTION high, and 200Ω connected between OUT1 and V_{CC}, the voltage at RC is increased from 0V to 5V at 1.2V/μs, and t_{DELAY} is measured as the time from the voltage at RC reaching 2V to the time the voltage at OUT 1 reaches 3V.

Typical Performance Characteristics



TL/H/11878–27

Connection Diagram



Top View

15-Lead TO-220 Molded Power Package
Order Number LMD18245T
See NS Package Number TA15A

Pinout Descriptions (See Functional Block and Connection Diagrams)

Pin 1, OUT 1: Output node of the first half H-bridge.

Pin 2, COMP OUT: Output of the comparator. If the voltage at CS OUT exceeds that provided by the DAC, the comparator triggers the monostable.

Pin 3, RC: Monostable timing node. A parallel resistor-capacitor network connected between this node and ground sets the monostable timing pulse at about 1.1 RC seconds.

Pin 5, PGND: Ground return node of the power bridge. Bond wires (internal) connect PGND to the tab of the TO-220 package.

Pins 4 and 6 through 8, M4 through M1: Digital inputs of the DAC. These inputs make up a four-bit binary number with M4 as the most significant bit or MSB. The DAC provides an analog voltage directly proportional to the binary number applied at M4 through M1.

Pin 9, V_{CC}: Power supply node.

Pin 10, BRAKE: Brake logic input. Pulling the BRAKE input logic-high activates both sourcing switches of the power bridge—effectively shorting the load. See Table I. Shorting the load in this manner forces the load current to recirculate and decay to zero.

Pin 11, DIRECTION: Direction logic input. The logic level at this input dictates the direction of current flow in the load. See Table I.

Pin 12, SGND: Ground return node of all signal level circuits.

Pin 13, CS OUT: Output of the current sense amplifier. The current sense amplifier sources 250 μA (typical) per ampere of total forward current conducted by the upper two switches of the power bridge.

Pin 14, DAC REF: Voltage reference input of the DAC. The DAC provides an analog voltage equal to $V_{DAC\ REF} \times D/16$, where D is the decimal equivalent (0–15) of the binary number applied at M4 through M1.

Pin 15, OUT 2: Output node of the second half H-bridge.

TABLE I. Switch Control Logic Truth Table

BRAKE	DIRECTION	MONO	Active Switches
H	X	X	Source 1, Source 2
L	H	L	Source 2
L	H	H	Source 2, Sink 1
L	L	L	Source 1
L	L	H	Source 1, Sink 2

X = don't care

MONO is the output of the monostable.

Functional Descriptions

TYPICAL OPERATION OF A CHOPPER AMPLIFIER

Chopper amplifiers employ feedback driven switching of a power bridge to control and limit current in the winding of a motor *(Figure 1)*. The bridge consists of four solid state power switches and four diodes connected in an H configuration. Control circuitry (not shown) monitors the winding current and compares it to a threshold. While the winding current remains less than the threshold, a source switch and a sink switch in opposite halves of the bridge force the supply voltage across the winding, and the winding current increases rapidly towards V_{CC}/R (*Figures 1a* and *1d*). As the winding current surpasses the threshold, the control circuitry turns OFF the sink switch for a fixed period or *off-time*. During the off-time, the source switch and the opposite upper diode short the winding, and the winding current recirculates and decays slowly towards zero (*Figures 1b* and *1e*). At the end of the off-time, the control circuitry turns back ON the sink switch, and the winding current again increases rapidly towards V_{CC}/R (*Figures 1a* and *1d* again). The above sequence repeats to provide a current chopping action that limits the winding current to the threshold *(Figure 1g)*. Chopping only occurs if the winding current reaches the threshold. During a change in the direction of the winding current, the diodes provide a decay path for the initial winding current (*Figures 1c* and *1f*). Since the bridge shorts the winding for a fixed period, this type of chopper amplifier is commonly referred to as a *fixed off-time chopper.*

Functional Descriptions (Continued)

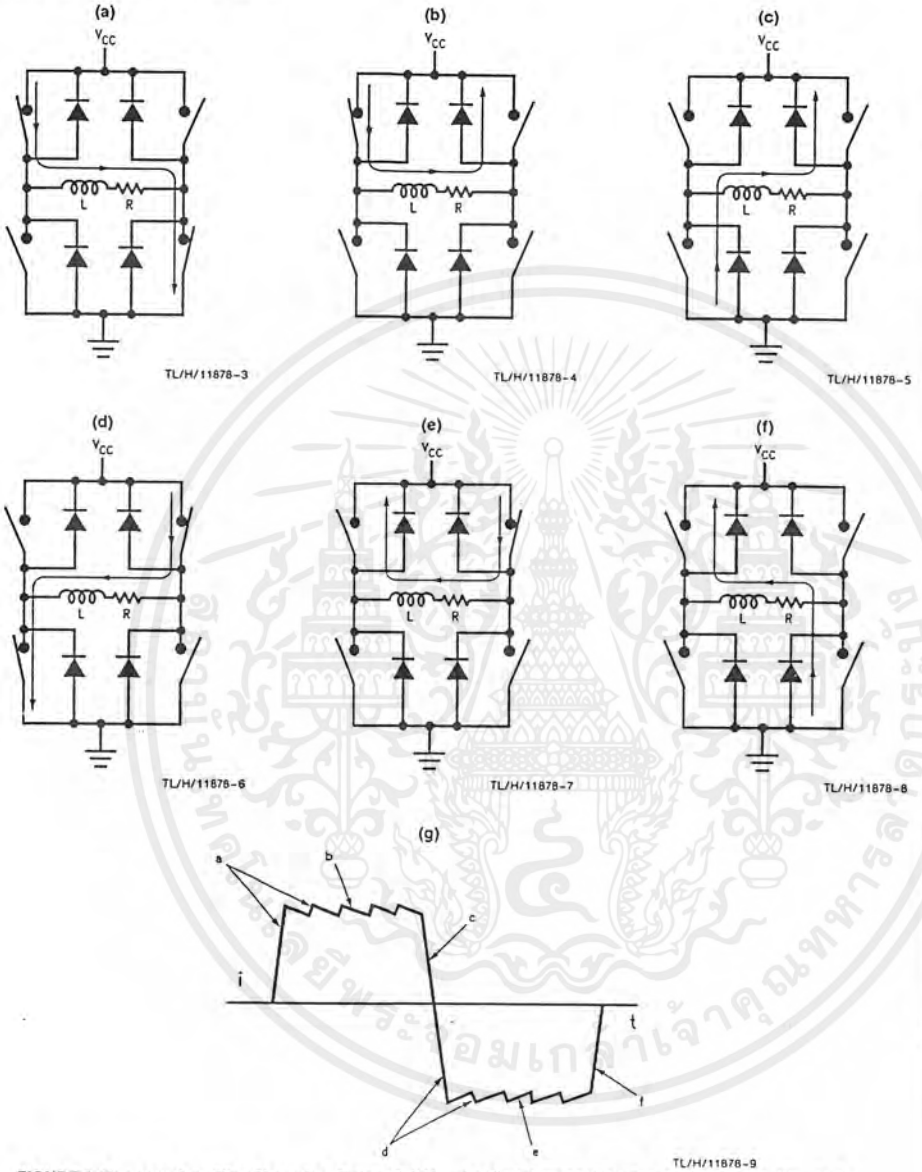


FIGURE 1. Chopper Amplifier Chopping States: Full V_{CC} Applied Across the Winding (a) and (d), Shorted Winding (b) and (e), Winding Current Decays During a Change in the Direction of the Winding Current (c) and (f), and the Chopped Winding Current (g)

Functional Descriptions (Continued)

THE LMD18245 CHOPPER AMPLIFIER

The LMD18245 incorporates all the circuit blocks needed to implement a fixed off-time chopper amplifier. These blocks include: an all DMOS, full H-bridge with clamp diodes, an amplifier for sensing the load current, a comparator, a monostable, and a DAC for digital control of the chopping threshold. Also incorporated are logic, level shifting and drive blocks for digital control of the direction of the load current and braking.

THE H-BRIDGE

The power stage consists of four DMOS power switches and associated body diodes connected in an H-bridge configuration *(Figure 2)*. Turning ON a source switch and a sink switch in opposite halves of the bridge forces the full supply voltage less the switch drops across the motor winding. While the bridge remains in this state, the winding current increases exponentially towards a limit dictated by the supply voltage, the switch drops, and the winding resistance. Subsequently turning OFF the sink switch causes a voltage transient that forward biases the body diode of the other source switch. The diode clamps the transient at one diode drop above the supply voltage and provides an alternative current path. While the bridge remains in this state, it essentially shorts the winding and the winding current recirculates and decays exponentially towards zero. During a change in the direction of the winding current, both the switches and the body diodes provide a decay path for the initial winding current *(Figure 3)*.

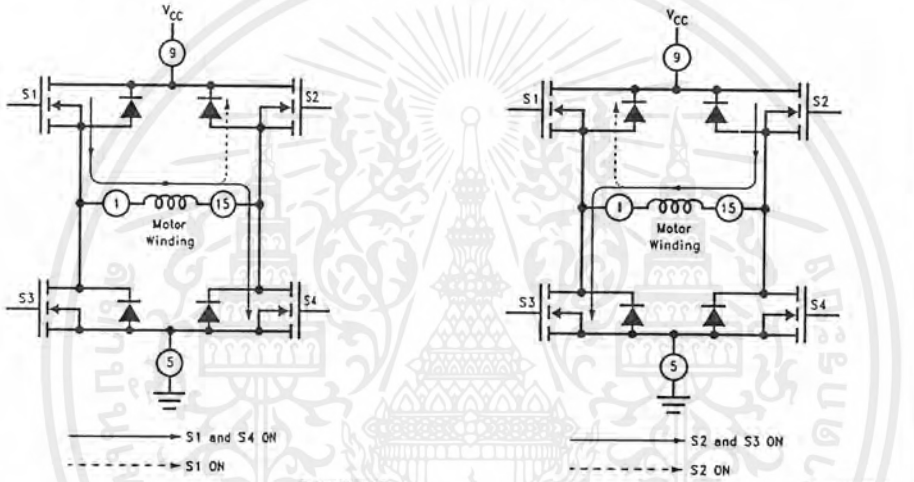


FIGURE 2. The DMOS H-Bridge

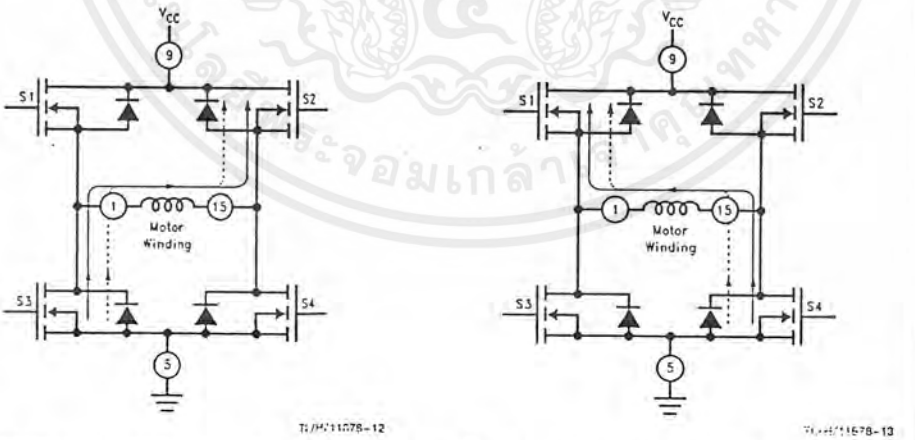


FIGURE 3. Decay Paths for Initial Winding Current During a Change in the Direction of the Winding Current

Functional Descriptions (Continued)

THE CURRENT SENSE AMPLIFIER

Many transistor cells in parallel make up the DMOS power switches. The current sense amplifier *(Figure 4)* uses a small fraction of the cells of both upper switches to provide a unique, low-loss means for sensing the load current. In practice, each upper switch functions as a 1x sense device in parallel with a 4000x power device. The current sense amplifier forces the voltage at the source of the sense device to equal that at the source of the power device; thus, the devices share the total drain current in proportion to the 1:4000 cell ratio. Only the current flowing from drain to source, the forward current, registers at the output of the current sense amplifier. The current sense amplifier, therefore, sources 250 μA per ampere of total forward current conducted by the upper two switches of the power bridge.

The sense current develops a potential across R_S that is proportional to the load current; for example, per ampere of load current, the sense current develops one volt across a 4 kΩ resistor (the product of 250 μA per ampere and 4 kΩ). Since chopping of the load current occurs as the voltage at CS OUT surpasses the threshold (the DAC output voltage), R_S sets the gain of the chopper amplifier; for example, a 2 kΩ resistor sets the gain at two amperes of load current per volt of the threshold (the reciprocal of the product of 250 μA per ampere and 2 kΩ). A quarter watt resistor suffices. A low value capacitor connected in parallel with R_S filters the effects of switching noise from the current sense signal.

While the specified maximum DC voltage compliance at CS OUT is 12V, the specified operating voltage range at CS OUT is 0V to 5V.

THE DIGITAL-TO-ANALOG CONVERTER (DAC)

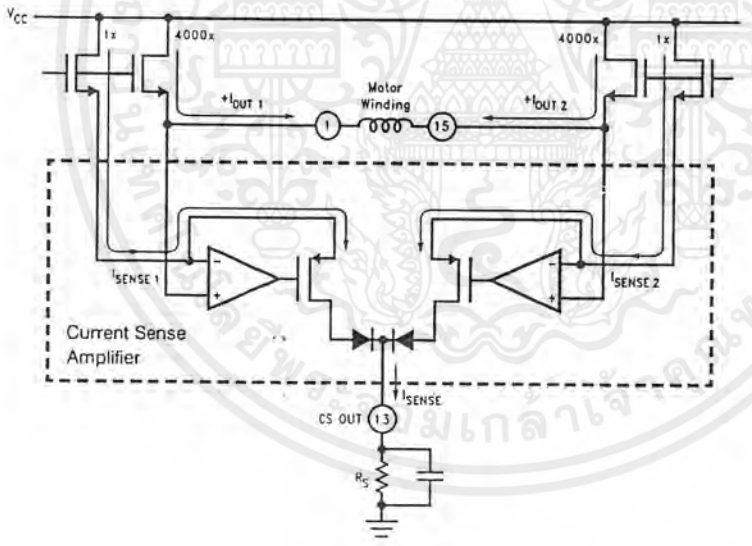
The DAC sets the threshold voltage for chopping at $V_{DAC\ REF} \times D/16$, where D is the decimal equivalent (0–15) of the binary number applied at M4 through M1, the digital inputs of the DAC. M4 is the MSB or most significant bit. For applications that require higher resolution, an external DAC can drive the DAC REF input. While the specified maximum DC voltage compliance at DAC REF is 12V, the specified operating voltage range at DAC REF is 0V to 5V.

THE COMPARATOR, MONOSTABLE AND WINDING CURRENT THRESHOLD FOR CHOPPING

As the voltage at CS OUT surpasses that at the output of the DAC, the comparator triggers the monostable, and the monostable, once triggered, provides a timing pulse to the control logic. During the timing pulse, the power bridge shorts the motor winding, causing current in the winding to recirculate and decay slowly towards zero (*Figures 1b* and *1e* again). A parallel resistor-capacitor network connected between RC (pin #3) and ground sets the timing pulse or off-time at about 1.1 RC seconds.

Chopping of the winding current occurs as the voltage at CS OUT exceeds that at the output of the DAC; so chopping occurs at a winding current threshold of about

$$(V_{DAC\ REF} \times D/16) \div ((250 \times 10^{-6}) \times R_S))\ \text{amperes.}$$



TL/H/11678–14

FIGURE 4. The Source Switches of the Power Bridge and the Current Sense Amplifier

Applications Information

POWER SUPPLY BYPASSING

Step changes in current drawn from the power supply occur repeatedly during normal operation and may cause large voltage spikes across inductance in the power supply line. Care must be taken to limit voltage spikes at V_{CC} to less than the 60V Absolute Maximum Rating. At a change in the direction of the load current, the initial load current tends to raise the voltage at the power supply rail (*Figure 3* again). Current transients caused by the reverse recovery of the clamp diodes tend to pull down the voltage at the power supply rail.

Bypassing the power supply line at V_{CC} is required to protect the device and minimize the adverse effects of normal operation on the power supply rail. Using both a 1 μF high frequency ceramic capacitor and a large-value aluminum electrolytic capacitor is highly recommended. A value of 100 μF per ampere of load current usually suffices for the aluminum electrolytic capacitor. Both capacitors should have short leads and be located within one half inch of V_{CC}.

OVERCURRENT PROTECTION

If the forward current in either source switch exceeds a 12A threshold, internal circuitry disables both source switches, forcing a rapid decay of the fault current *(Figure 5)*. Approximately 3 μs after the fault current reaches zero, the device restarts. Automatic restart allows an immediate return to normal operation once the fault condition has been removed. If the fault persists, the device will begin cycling into and out of thermal shutdown. Switching large fault currents may cause potentially destructive voltage spikes across inductance in the power supply line; therefore, the power supply line must be properly bypassed at V_{CC} for the motor driver to survive an extended overcurrent fault.

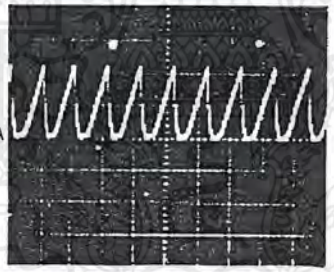
In the case of a locked rotor, the inductance of the winding tends to limit the rate of change of the fault current to a value easily handled by the protection circuitry. In the case of a low inductance short from either output to ground or between outputs, the fault current could surge past the 12A shutdown threshold, forcing the device to dissipate a substantial amount of power for the brief period required to disable the source switches. Because the fault power must be dissipated by only one source switch, a short from output to ground represents the worst case fault. Any overcurrent fault is potentially destructive, especially while operating with high supply voltages ($\geq$30V), so precautions are in order. Sinking V_{CC} for heat with 1 square inch of 1 ounce copper on the printed circuit board is highly recommended. The sink switches are not internally protected against shorts to V_{CC}.

THERMAL SHUTDOWN

Internal circuitry senses the junction temperature near the power bridge and disables the bridge if the junction temperature exceeds about 155°C. When the junction temperature cools past the shutdown threshold (lowered by a slight hysteresis), the device automatically restarts.

UNDERVOLTAGE LOCKOUT

Internal circuitry disables the power bridge if the power supply voltage drops below a rough threshold between 8V and 5V. Should the power supply voltage then exceed the threshold, the device automatically restarts.



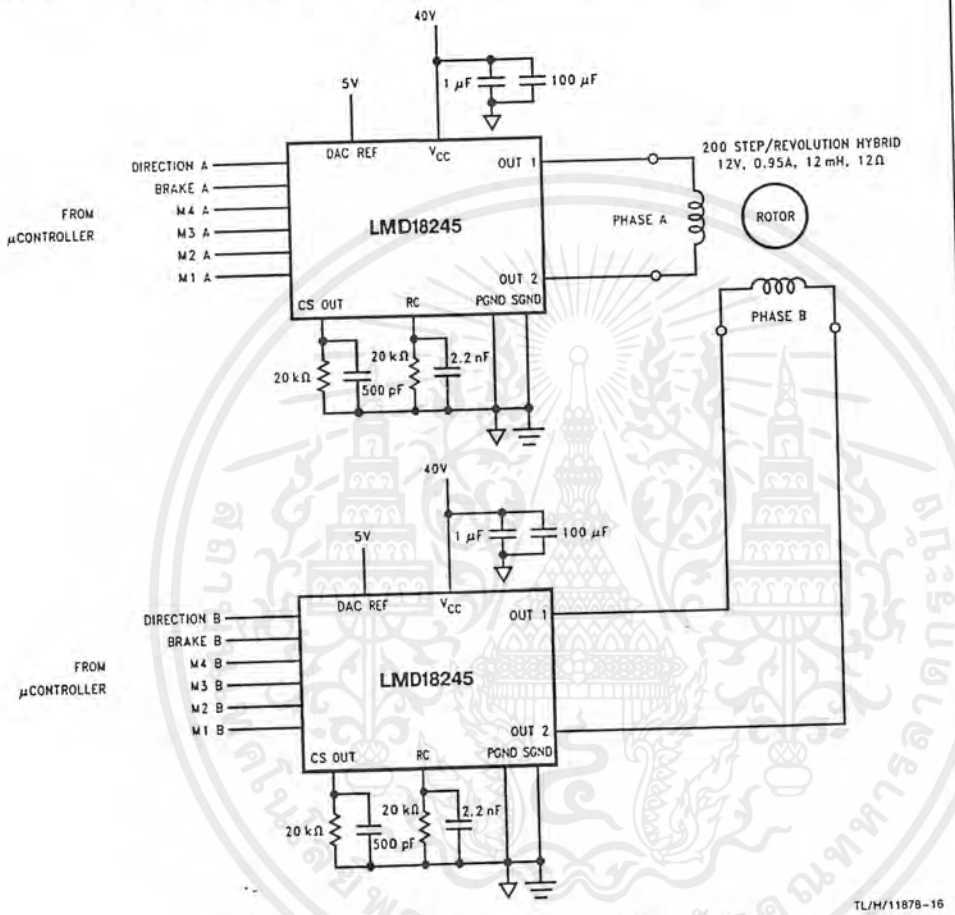
TL/H/11878–15

Trace: Fault Current at 5A/div
Horizontal: 20 μs/div

FIGURE 5. Fault Current with V_{CC} = 30V, OUT 1 Shorted to OUT 2, and CS OUT Grounded

The Typical Application

Figure 6 shows the typical application, the power stage of a chopper drive for bipolar stepper motors. The 20 kΩ resistor and 2.2 nF capacitor connected between RC and ground set the off-time at about 48 μs, and the 20 kΩ resistor connected between CS OUT and ground sets the gain at about 200 mA per volt of the threshold for chopping. Digital signals control the thresholds for chopping, the directions of the winding currents, and, by extension, the drive type (full step, half step, etc.). A μprocessor or μcontroller usually provides the digital control signals.



TL/H/11878–16

FIGURE 6. Typical Application Circuit for Driving Bipolar Stepper Motors

The Typical Application (Continued)

ONE-PHASE-ON FULL STEP DRIVE (WAVE DRIVE)

To make the motor take full steps, windings A and B can be energized in the sequence

$$A \rightarrow B \rightarrow A^* \rightarrow B^* \rightarrow A \rightarrow \ldots,$$

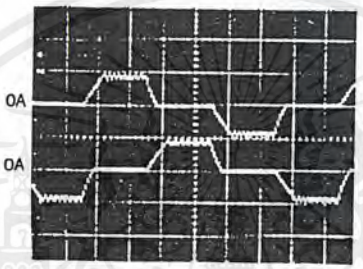
where A represents winding A energized with current in one direction and A* represents winding A energized with current in the opposite direction. The motor takes one full step each time one winding is de-energized and the other is energized. To make the motor step in the opposite direction, the order of the above sequence must be reversed. *Figure 7* shows the winding currents and digital control signals for a wave drive application of the typical application circuit.

TWO-PHASE-ON FULL STEP DRIVE

To make the motor take full steps, windings A and B can also be energized in the sequence

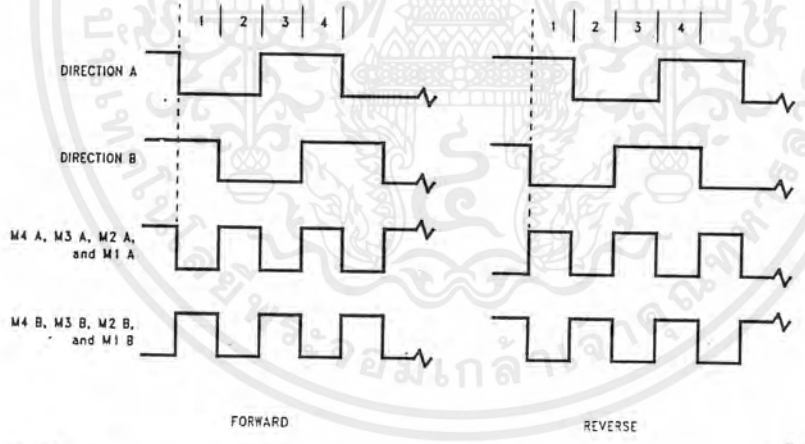
$$AB \rightarrow A^*B \rightarrow A^*B^* \rightarrow AB^* \rightarrow AB \rightarrow \ldots,$$

and because both windings are energized at all times, this sequence produces more torque than that produced with wave drive. The motor takes one full step at each change of direction of either winding current. *Figure 8* shows the winding currents and digital control signals for this application of the typical application circuit, and *Figure 9* shows, for a single phase, the winding current and voltage at the output of the associated current sense amplifier.



TL/H/11678–17

Top Trace: Phase A Winding Current at 1A/div
Bottom Trace: Phase B Winding Current at 1A/div
Horizontal: 1 ms/div
*500 steps/second

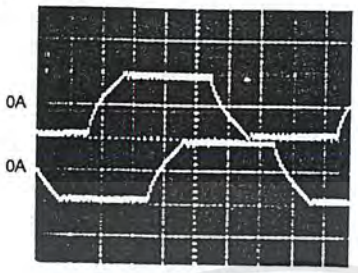


TL/H/11678–18

BRAKE A = BRAKE B = 0

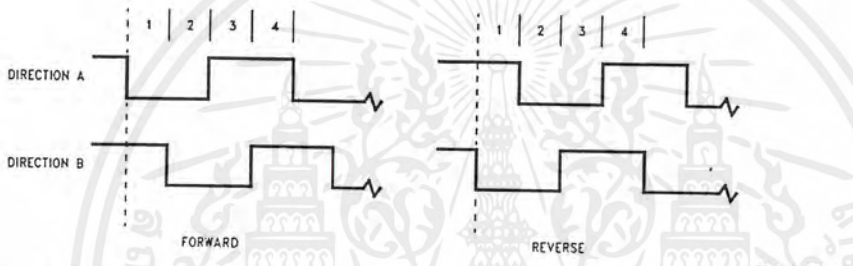
FIGURE 7. Winding Currents and Digital Control Signals for One-Phase-On Drive (Wave Drive)

The Typical Application (Continued)



TL/H/11878–19

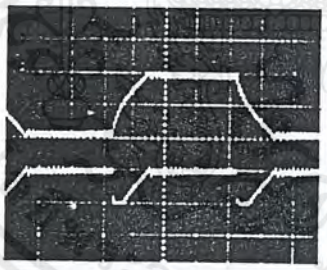
Top Trace: Phase A Winding Current at 1A/div
Bottom Trace: Phase B Winding Current at 1A/div
Horizontal: 1 ms/div
*500 steps/second



TL/H/11878–20

M4 A through M1 A = M4 B through M1 B = 1
BRAKE A = BRAKE B = 0

FIGURE 8. Winding Currents and Digital Control Signals for Two-Phase-On Drive



TL/H/11878–21

Top Trace: Phase A Winding Current at 1A/div
Bottom Trace: Phase A Sense Voltage at 5V/div
Horizontal: 1 ms/div
*500 steps/second

FIGURE 9. Winding Current and Voltage at the Output of the Associated Current Sense Amplifier

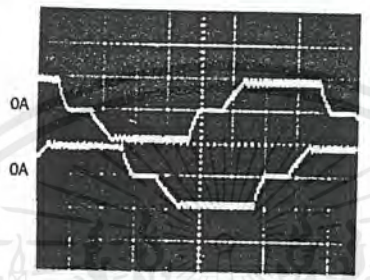
The Typical Application (Continued)

HALF STEP DRIVE WITHOUT TORQUE COMPENSATION

To make the motor take half steps, windings A and B can be energized in the sequence

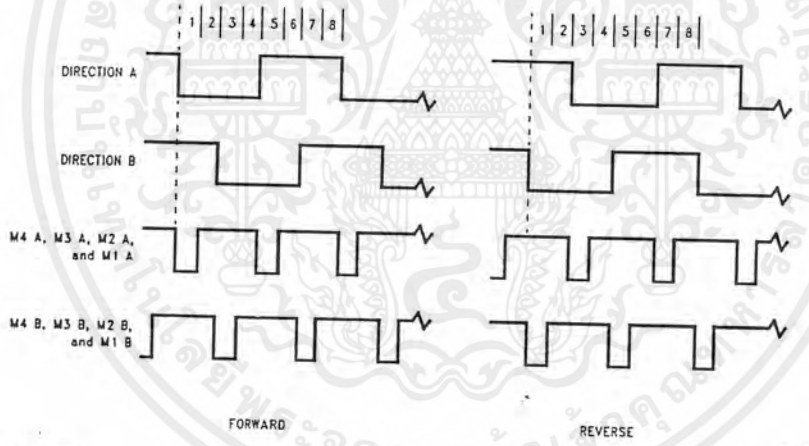
$$A \rightarrow AB \rightarrow B \rightarrow A^*B \rightarrow A^* \rightarrow A^*B^* \rightarrow B^* \rightarrow AB^* \rightarrow A \rightarrow \ldots$$

The motor takes one half step each time the number of energized windings changes. It is important to note that although half stepping doubles the step resolution, changing the number of energized windings from two to one decreases (one to two increases) torque by about 40%, resulting in significant torque ripple and possibly noisy operation. *Figure 10* shows the winding currents and digital control signals for this half step application of the typical application circuit.



TL/H/11878–22

Top Trace: Phase A Winding Current at 1A/div
Bottom Trace: Phase B Winding Current at 1A/div
Horizontal: 1 ms/div
*500 steps/second



BRAKE A = BRAKE B = 0

TL/H/11878–23

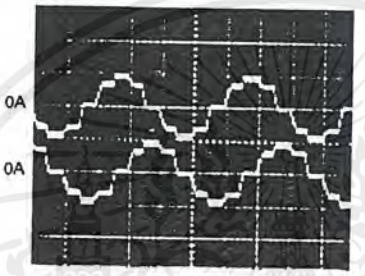
FIGURE 10. Winding Currents and Digital Control Signals for Half Step Drive without Torque Compensation

The Typical Application (Continued)

HALF STEP DRIVE WITH TORQUE COMPENSATION

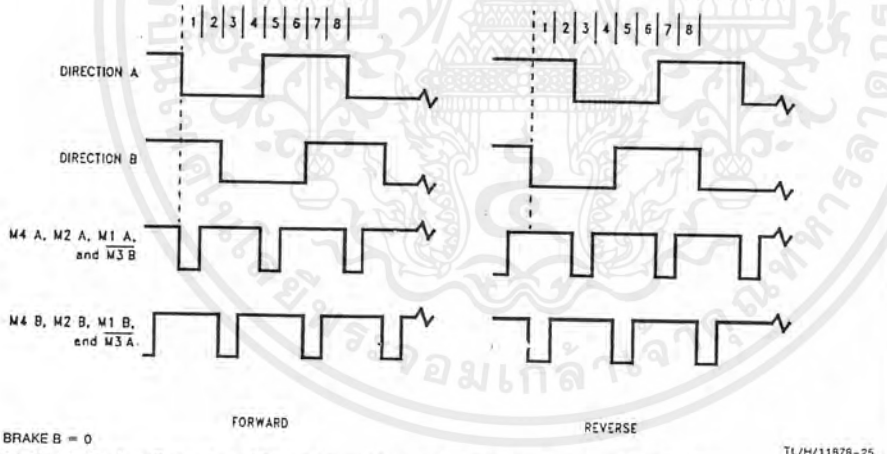
To make the motor take half steps, the windings can also be energized with sinusoidal currents *(Figure 11)*. Controlling the winding currents in the fashion shown doubles the step resolution without the significant torque ripple of the prior drive technique. The motor takes one half step each time the level of either winding current changes. Half step drive with torque compensation is microstepping drive. Along with the obvious advantage of increased step resolution, microstepping reduces both full step oscillations and resonances that occur as the motor and load combination is driven at its natural resonant frequency or subharmonics thereof. Both of these advantages are obtained by replacing full steps with bursts of microsteps. When compared to full step drive, the motor runs smoother and quieter.

Figure 12 shows the lookup table for this application of the typical application circuit. Dividing 90°electrical per full step by two microsteps per full step yields 45° electrical per microstep. α, therefore, increases from 0 to 315° in increments of 45°. Each full 360° cycle comprises eight half steps. Rounding $|\cos\alpha|$ to four bits gives D A, the decimal equivalent of the binary number applied at M4 A through M1 A. DIRECTION A controls the polarity of the current in winding A. *Figure 11* shows the sinusoidal winding currents.



TL/H/11878–24

Top Trace: Phase A Winding Current at 1A/div
Bottom Trace: Phase B Winding Current at 1A/div
Horizontal: 2 ms/div
*500 steps/second



BRAKE A = BRAKE B = 0

TL/H/11878–25

FIGURE 11. Winding Currents and Digital Control Signals for Half Step Drive with Torque Compensation

The Typical Application (Continued)

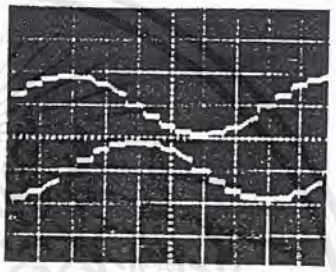
90° ELECTRICAL/FULL STEP ÷ 2 MICROSTEPS/FULL STEP = 45° ELECTRICAL/MICROSTEP

	α	|cos(α)|	D A	DIRECTION A	|sin(α)|	D B	DIRECTION B
|	0	1	15	1	0	0	1
FORWARD	45	0.707	11	1	0.707	11	1
↓	90	0	0	0	1	15	1
	135	0.707	11	0	0.707	11	1
↑	180	1	15	0	0	0	0
REVERSE	225	0.707	11	0	0.707	11	0
|	270	0	0	1	1	15	0
	315	0.707	11	1	0.707	11	0
	REPEAT						

FIGURE 12. Lookup Table for Half Step Drive with Torque Compensation

QUARTER STEP DRIVE WITH TORQUE COMPENSATION

Figure 13 shows the winding currents and lookup table for a quarter step drive (four microsteps per full step) with torque compensation.



TL/H/11878–26

Top Trace: Phase A Winding Current at 1A/div
Bottom Trace: Phase B Winding Current at 1A/div
Horizontal: 2ms/div
*250 steps/second

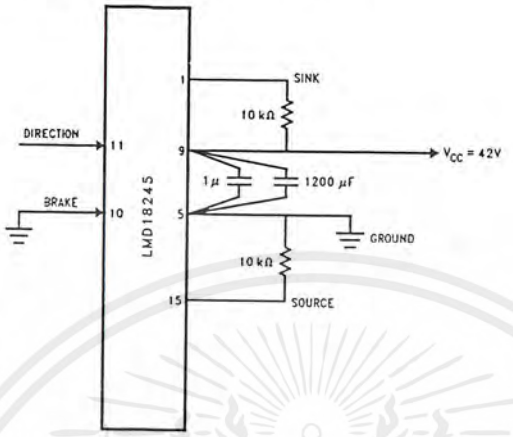
90° ELECTRICAL/FULL STEP ÷ 4 MICROSTEPS/FULL STEP = 22.5° ELECTRICAL/MICROSTEP

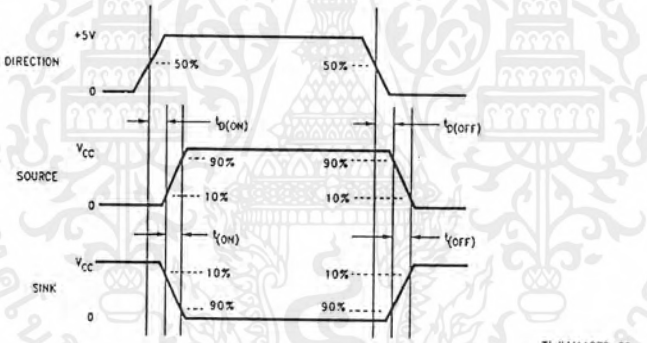
	α	|cos(α)|	D A	DIRECTION A	|sin(α)|	D B	DIRECTION B
	0	1	15	1	0	0	1
	22.5	0.924	14	1	0.383	6	1
|	45	0.707	11	1	0.707	11	1
FORWARD	67.5	0.383	6	1	0.924	14	1
↓	90	0	0	0	1	15	1
	112.5	0.383	6	0	0.924	14	1
↑	135	0.707	11	0	0.707	11	1
REVERSE	157.5	0.924	14	0	0.383	6	1
|	180	1	15	0	0	0	0
	202.5	0.924	14	0	0.383	6	0
	225	0.707	11	0	0.707	11	0
	247.5	0.383	6	0	0.924	14	0
	270	0	0	1	1	15	0
	292.5	0.383	6	1	0.924	14	0
	315	0.707	11	1	0.707	11	0
	337.5	0.924	14	1	0.383	6	0
	REPEAT						

BRAKE A = BRAKE B = 0

FIGURE 13. Winding Currents and Lookup Table for Quarter Step Drive with Torque Compensation

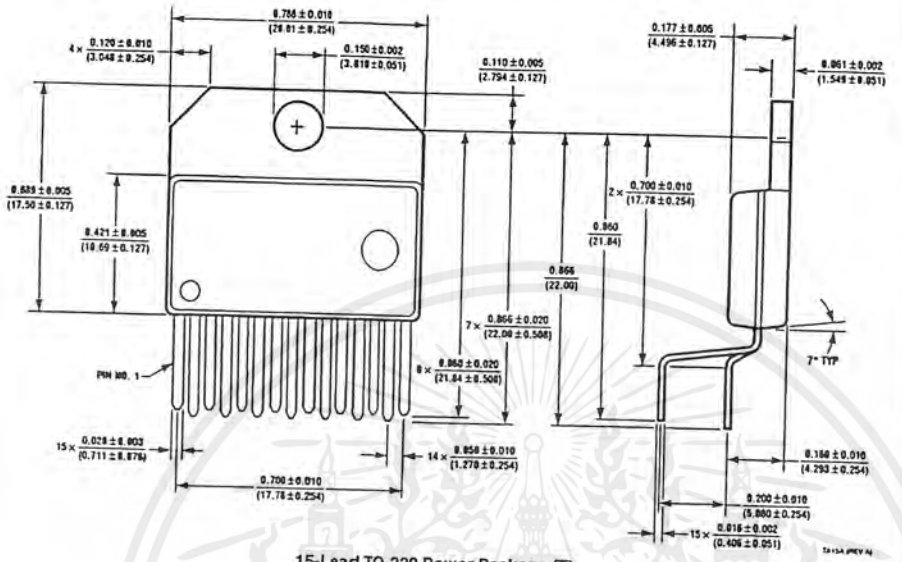
Test Circuit and Switching Time Definitions





TL/H/11878–28

Physical Dimensions inches (millimeters) unless otherwise noted



15-Lead TO-220 Power Package (T)
Order Number LMD18245T
NS Package Number TA15A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

http://www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 180-530 85 85
English Tel: +49 (0) 180-532 78 32
Français Tel: +49 (0) 180-532 93 58
Italiano Tel: +49 (0) 180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2308
Fax: 81-043-299-2408

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลงได้ด้วยความช่วยเหลืออย่างดียิ่งจากหลายๆฝ่าย ซึ่งผู้จัดทำใคร่ขอขอบคุณทุกๆท่านที่มีส่วนร่วมสนับสนุน ช่วยเหลือและแนะนำในทุกๆด้าน

ขอขอบพระคุณ รศ. อิทธิชัย อรุณศรีแสงไชย อาจารย์ที่ปรึกษาโครงงานนี้ ที่ได้เสียสละเวลา ให้คำปรึกษา และข้อเสนอแนะที่เป็นประโยชน์ ตลอดจนห้องทดลองและการทำงาน ให้การทำโครงงานชิ้นนี้สำเร็จลุล่วงได้ด้วยดี

ขอกราบขอบพระคุณบิดา มารดา ผู้ให้โอกาสและคอยให้กำลังใจเสมอมา

ขอขอบพระคุณเพื่อนๆ ห้อง A-404 ที่ช่วยเหลือให้คำปรึกษาและกำลังใจเสมอมา

สุดท้ายขอขอบคุณภาควิชาอิเล็คทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่สนับสนุนการทำโครงงานในครั้งนี้

คุณค่าและประโยชน์อันพึงมีจากรายงานฉบับนี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

พีรวุฒิ คันธวัฒน์

ภาณุมาศ ศิริเตียรทอง

สุรชัย คัมภีร์รัตนอาภา

หนังสืออ้างอิง

1. สุเจตน์ จันทรังษ์, " ไมโครคอนโทรลเลอร์ชิฟเดี่ยว 8051" , โครงการตำราวิชาการ มหาวิทยาลัยมหานคร,178 หน้า,2535
2. ไชยันต์ สุวรรณชีวะศิริ , " สเต็ปปิ้งมอเตอร์และการควบคุม" ,วารสาร เซมิคอนดัคเตอร์ ฉบับ 116 หน้า 102-110
3. สุพันธ์ อนมเชิดชูกิจ ," สเต็ปปิ้งมอเตอร์ " ,วารสาร เซมิคอนดักเตอร์ ฉบับ 126 หน้า 83-89
4. พิพัฒน์ เลาหสงคราม , " ไมโครคอนโทรลเลอร์ MCS-48 MCS-51 " , 2537
5. กฤษดา ใจเย็น ," ความรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปการณ์ภายนอก ผ่านพอร์ตอนุกรม " ,2536
6. ฉัททวุฒิ พืชผล, "คู่มือเรียนวิชวลเบสิก 6" ,2542