ปีการศึกษา 2532

เครื่องเจาะ PCB อัตโนมัติ

โดย

นายธีรพันธุ์ ศิริสุนทรไพบูลย์ 29-1071

นายบูลย์แรงค์ รุจนะไกรกานต์ 29-1103

นายวีระศักดิ์ โต๊ะสุวรรณาณิช 29-1181

อาจารย์ที่ปรึกษา

อาจารย์ขนิษฐา แซ่ตั้ง

อาจารย์ธำรงศักดิ์ สุกใส

ปริญญานิพนธ์ปีการศึกษา 2532
ภาควิชา อิเลคทรอนิคส์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง เครื่องเจาะ PCB อัตโนมัติ (Automatic PCB Driller)
ผู้จัดทำ
1. นายธีรพันธุ์ ศิริสุนทรไพบูลย์ 29.1071
2. นายบูลย์แรงค์ รุจนะไกรกานต์ 29.1103
3. นายวีระศักดิ์ โต๊ะสุวรรณาณิช 29.1181

.....อาจารย์ที่ปรึกษา
(อาจารย์ขนิษฐา แซ่ตั้ง)

.....อาจารย์ที่ปรึกษา
(อาจารย์ธำรงศักดิ์ สุกใส)

เครื่องเจาะ PCB อัตโนมัติ

ธีรพันธุ์ ศิริสุนทรไพบูลย์
บูลย์แรงค์ รุจนะไกรกานต์
วีระศักดิ์ โต๊ะสุวรรณวณิช
อาจารย์ ขนิษฐา แซ่ตั้ง อาจารย์ที่ปรึกษา
อาจารย์ ธำรงศักดิ์ สุกใส อาจารย์ที่ปรึกษา

บทคัดย่อ

เครื่องเจาะ PCB อัตโนมัติ (automatic PCB driller)นี้ ทำงานด้วยไมโครโปรเซสเซอร์ โดยอาศัยหลักการทางลอจิกที่มีสถานะเพียงสองสถานะคือ 0 และ 1 มาสวิทช์ (switch) ทรานซิสเตอร์ ที่ใช้สำหรับขับสเต็ปปิ้งมอเตอร์ ซึ่งควบคุมการเคลื่อนที่ในแต่ละแกน และเบิด/ปิดรีเลย์ซึ่งใช้ขับมอเตอร์ลูกเบี้ยวและสว่านในการเจาะ การทำงานทั้งหมดนี้จะถูกควบคุมโดยชุดคำสั่งหรือซอฟแวร์ (software)

โครงงานนี้เป็นโครงงานต้นแบบที่สร้างขึ้น เพื่อประยุกต์ใช้งานไมโครโปรเซสเซอร์ในการควบคุมการเคลื่อนที่ของแมคคานิคส์ (mechanics) ต่างๆ ซึ่งมีความสำคัญอย่างยิ่งสำหรับงานอุตสาหกรรมในปัจจุบัน

AUTOMATIC PCB DRILLER

Thiraphant Sirisoonthornphibul
Boonnarong Rujchanakraikarn
Weerasak Tohsuwanwanich
Kanittha Saetaung Advisor
Thamrongsak Suksai Advisor

Abstract

Automatic PCB Driller is controlled with microprocessor, which has only two logics that are 0 and 1. This characterization is used for switching transistors that drive stepping motors in each axis ,and closing/opening relays that drive the driller section. All functions must be done by the set of instructions (software).

This project is a model which is the appliction of microprocessor for controlling the mechanical movement. It is very necessary in industrial now.

สารบัญ

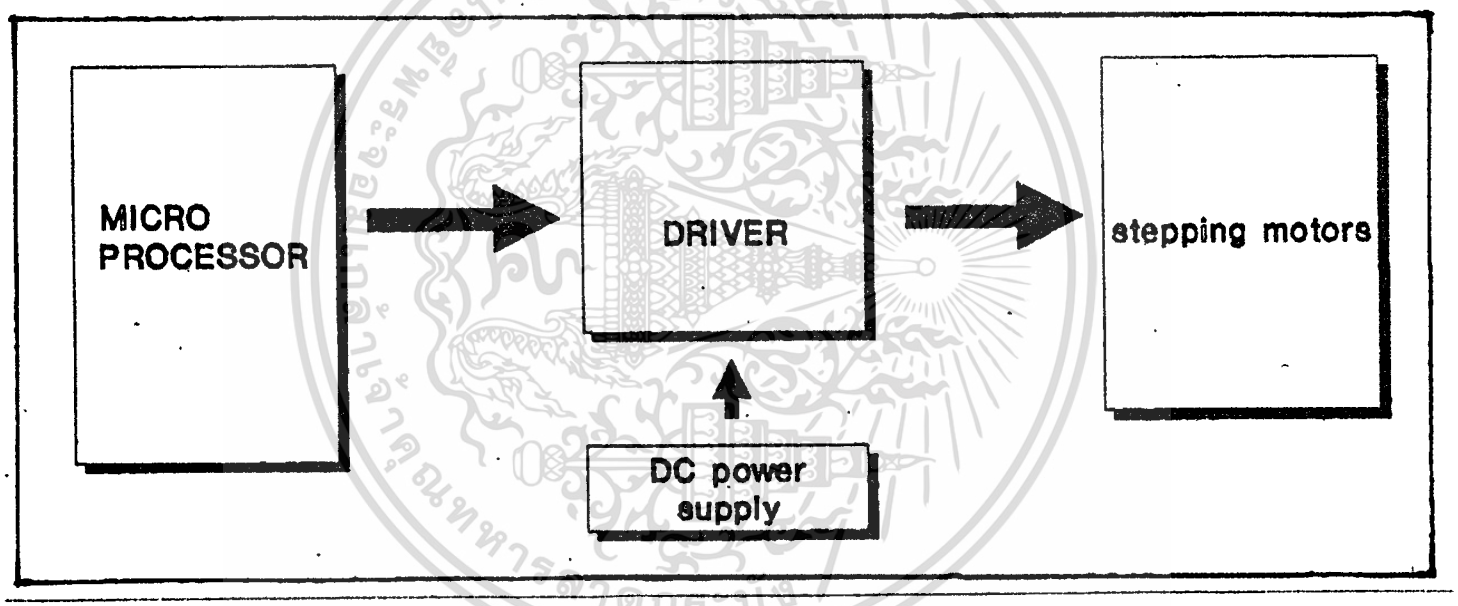
บทที่ 1
บทนำ

ปัจจุบันงานทางด้านอุตสาหกรรมกำลังเติบโตอย่างขึ้นมาก งานแทบทุกด้านต้องการเทคโนโลยีเข้ามาสนับสนุน ไม่ว่าจะเป็นงานทางด้านการผลิต การตรวจสอบคุณภาพ งานออกแบบ ฯลฯ สำหรับอุตสาหกรรมทางด้านอิเลคทรอนิคส์นับเป็นสิ่งจำเป็นอย่างยิ่งเพื่อตอบสนองความต้องการในการใช้เทคโนโลยี

สิ่งที่หลีกเลี่ยงไม่ได้สำหรับงานทางด้านอิเลคทรอนิคส์ก็คือแผ่นพิมพ์ลายวงจร (PCB-Printed circuit Board)ซึ่งในแผ่น PCB จะต้องมีการเจาะรูเพื่อที่จะใส่อุปกรณ์ต่างๆ เช่น ความต้านทาน คาปาซิเตอร์ ไอซี ฯลฯ ถ้าเป็นงานในระดับอุตสาหกรรมแล้วความรวดเร็ว ความแม่นยำ เป็นสิ่งที่ต้องคำนึงถึง

ในรูปที่ 1 เป็นบล็อกไดอะแกรมของเครื่องเจาะ PCB อัตโนมัติ สำหรับแนวความคิดในการสร้าง คือ สเต็ปปิ้งมอเตอร์สามารถที่จะทำให้แมคคานิคส์เคลื่อนที่ได้อย่างมีทิศทาง และแม่นยำ เพราะจะทราบตำแหน่งของโรเตอร์ได้ตลอดเวลา ส่วนไมโครโปรเซสเซอร์สามารถที่จะประมวลผลได้อย่างรวดเร็ว สำหรับคุณสมบัติที่สำคัญของเครื่องเจาะ PCB อัตโนมัติคือ

- แสดงตำแหน่งโคออดิเนท ด้วยหน่วยแสดงผล
- สามารถควบคุมการเคลื่อนที่ด้วยฟังก์ชั่นคีย์ (Function Key)
- จดจำโคออดิเนทได้
- สามารถทำงานแบบอัตโนมัติได้



รูปที่ 1.1 แสดงบล็อกไดอะแกรมของเครื่องเจาะ PCB อัตโนมัติ

บทที่ 2
ทฤษฎีเบื้องต้นของสเต็ปปิ้งมอเตอร์ (Stepping Motor)

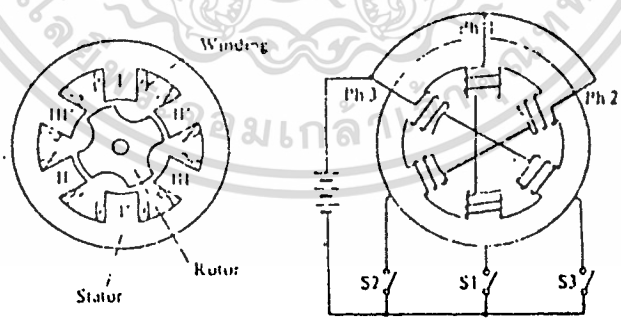
สเต็ปปิ้งมอเตอร์ (Stepping Motor) เป็นดีซีมอเตอร์ชนิดหนึ่งที่ไม่ต้องอาศัยแปรงถ่าน การทำงานของสเต็ปปิ้งมอเตอร์เราจะใช้ วงจรซีเควนเชียล (Seguenc.al) เป็นตัวกำเนิดสัญญาณขับผ่านภาคขับกำลัง (Power Drive) เข้าสู่สเต็ปปิ้งมอเตอร์ซึ่งเป็นการทำงานในลักษณะลูปเปิด (Open Loop)

สเต็ปปิ้งมอเตอร์ สามารถแบ่งได้เป็น 3 ชนิดตามลักษณะโรเตอร์ ดังนี้

1. แบบแอคทีฟ (Active Type)
   แบ่งออกเป็น - แม่เหล็กถาวร (Permanent Magnetic)
   - ดีซีเอนนิไจซ์ (DC Energize)
2. แบบวีอาร์ (VR-Reactive Variable)
3. แบบไฮบริดจ์ (HYBRID) เป็นการรวมทั้ง VR และ PM เข้าด้วยกัน

2.1 โครงสร้างของสเต็ปปิ้งมอเตอร์

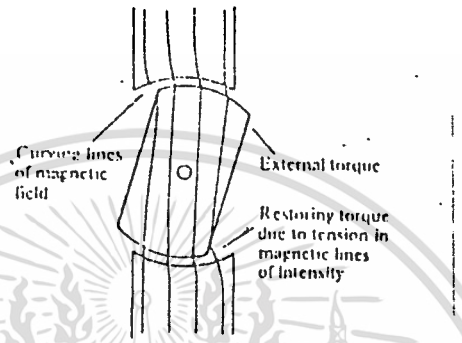
ภายในสเต็ปปิ้งมอเตอร์ประกอบด้วย สเตเตอร์ (Stator) ,โรเตอร์ (Rotor) และ ขดลวด (Winding) ประกอบกันดังแสดงดังรูปที่ 2.1



รูปที่ 2.1 ภาพตัดขวางของสเต็ปปิ้งมอเตอร์ 3 เฟส

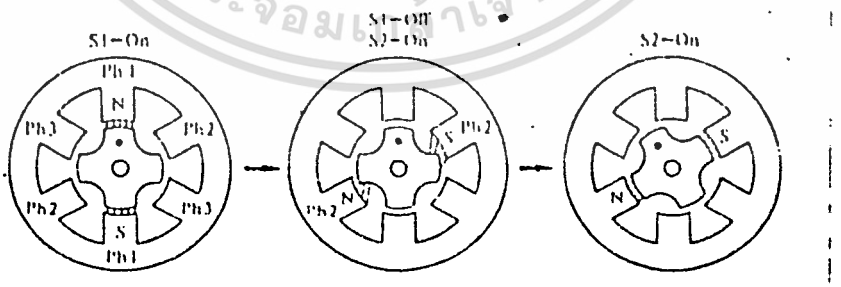
เนื่องจากสเต็ปปิ้ง มอเตอร์ แบบ VR นี้ โรเตอร์เป็นเหล็กอ่อน ซึ่งมีคุณสมบัติคือ พยายามปรับตัวเองให้อยู่ในแนวที่เส้นแรงแม่เหล็กผ่านมากที่สุด ดังเช่นแสดง

ในรูปที่ 2.2 เมื่อเกิดเส้นแรงแม่เหล็กขึ้นที่สเตเตอร์ ตัดผ่านโรเตอร์ ตัวโรเตอร์ก็จะพยายามปรับตัวเองให้เส้นแรงแม่เหล็กตัวผ่านตัวโรเตอร์มากที่สุด โดยที่โรเตอร์จะหมุนตัวเองและจะทำให้เกิดมุมของการหมุนขึ้น ทั้งนี้มอเตอร์จะหยุดหมุนเมื่อเส้นแรงแม่เหล็กตัดผ่านตัวมันมากที่สุด ดังเช่นรูปที่ 2.2



รูปที่ 2.2 เส้นแรงแม่เหล็กที่ทำให้เกิดแรงบิด

ดังนั้นจึงสามารถทำให้สเต็ปปิ้งมอเตอร์ หมุนได้โดยอาศัยหลักการนี้ แต่ต้องให้เกิดเส้นแรงแม่เหล็กเกิดขึ้นถัดไปเรื่อยๆ ดังแสดงอยู่ในรูปที่ 2.3 ,รูปที่ 2.4 และรูปที่ 2.5 ซึ่งแสดงการหมุนของมอเตอร์

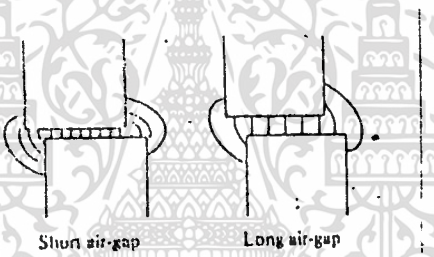


รูปที่ 2.3 แสดงการเคลื่อนที่ทีละ สเต็ป เมื่อกระตุ้นเฟส 1และ2

2.2 โครงสร้างพื้นฐานสเต็ปปิ้งมอเตอร์แบบ VR

ช่องอากาศ (Air Gap) ระหว่างซี่ สเตเตอร์ ซึ่งจะแปรผกผันกับ แรงบิด และความแน่นอนของตำแหน่ง ดังนั้นถ้าต้องการให้ สเต็ปปิ้ง มอเตอร์ มีแรงบิดสูง และมีตำแหน่งแน่นอน ก็จะต้องมีช่องอากาศช่วงแคบๆ

ในรูป 2.6 เป็นการเปรียบเทียบระหว่าง สเต็ปปิ้งมอเตอร์ ที่มีช่องอากาศกว้างกับ สเต็ปปิ้งมอเตอร์ที่มีช่องอากาศแคบ จะเห็นว่าสเต็ปปิ้งมอเตอร์ที่มี ช่องอากาศแคบ มีความหนาแน่นของเส้นแรงแม่เหล็กสูงกว่า ดังนั้นแรงบิด (torque) จึงมากกว่า และความแน่นอนของตำแหน่ง (accuracy in positioning)จึงสูงกว่ามอเตอร์ที่มีช่องอากาศกว้าง



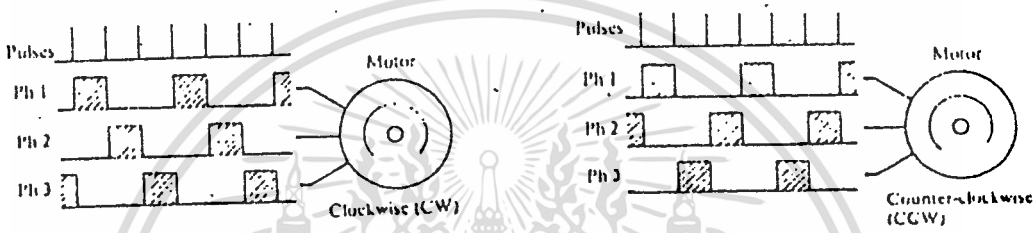
รูปที่ 2.6 เปรียบเทียบเส้นแรงแม่เหล็กระหว่าง มอเตอร์ที่มีช่องอากาศแคบ และที่มีช่องอากาศกว้าง

ในการทำงานของ สเต็ปปิ้งมอเตอร์นั้น ตำแหน่งของโรเตอร์จะไม่หยุดทันทีที่โรเตอร์หมุนจนเส้นแรงแม่เหล็กตัดผ่านมากที่สุด เนื่องจากโรเตอร์ขณะที่หมุนจะมีแรงเฉื่อยเกิดขึ้น จึงทำให้เกิด ชูต (Shoot) ของระยะขจัด

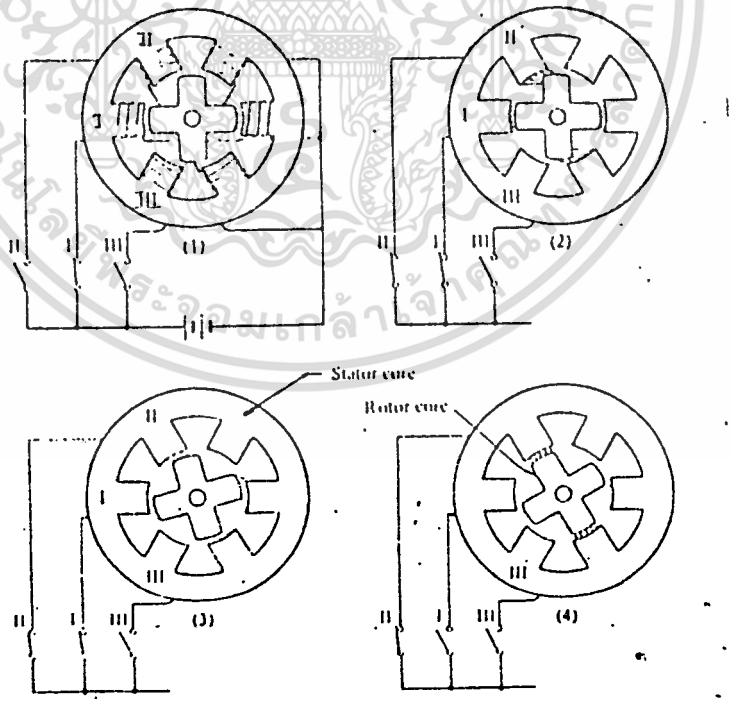
จากนั้นโรเตอร์จึงหมุนกลับให้อยู่ในตำแหน่งที่เส้นแรงแม่เหล็กตัดผ่านมากที่สุด ดังรูปที่ 2.7 แสดงการเปลี่ยนตำแหน่งในแต่ละ สเต็ป ทั้งนี้ชูตที่เกิดขึ้นยังขึ้นอยู่กับช่องอากาศ และ เป็นการขับแบบใด

ทิศทางของการหมุนขึ้นอยู่กับการถูกขับ (Drive) ว่าจะให้ไปในทางใดและเมื่อต้อง การให้มอเตอร์หยุดในตำแหน่งที่ต้องการก็หยุดการขับโรเตอร์ มอเตอร์ก็จะหยุด ณ ตำแหน่งสุดท้ายที่มีการขับโรเตอร์ ดังนั้นเราจะสามารถทราบได้ว่าตำแหน่งของมอเตอร์อยู่ที่ไหน โดยการนับพัลส์ (Pulse) ที่ป้อนให้มอเตอร์ และใช้สูตรดังนี้

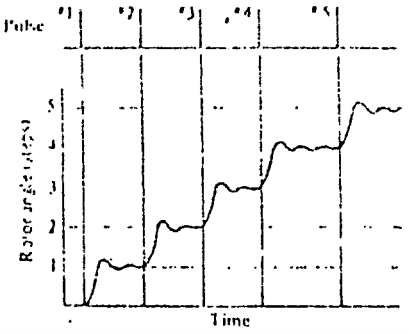
มุมที่เปลี่ยนไป = Step Angle * จำนวนพัลส์



รูปที่ 2.4 แสดงการกระตุ้นแบบเดินหน้าและถอยหลัง



รูปที่ 2.5 แสดงการทำงานของสเต็ปปิ้งมอเตอร์



รูปที่ 2.7 กราฟแสดงตำแหน่งในแต่ละสเต็ป

สำหรับค่า T (Step Response) ขึ้นอยู่กับอัตราการสวิทช์ของกระแสในขดลวด สเตเตอร์ จำนวนขั้นต่อการหมุนรอบโรเตอร์ (SPR) หาได้ดังนี้

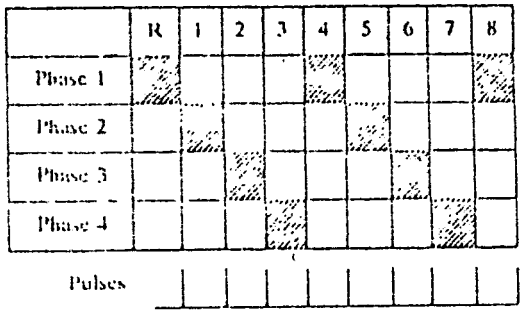
$$STEP/REVOLUTION\ (SPR) = 360/SA = m*Nr$$

โดยที่ SA = ค่ามุมในแต่ละขั้น (Sterp Angle) หน่วนเป็นองศา
M = จำนวนเฟส
Nr = จำนวนซี่ของโรเตอร์

2.3 การกระตุ้นเฟสขดลวดสเตเตอร์

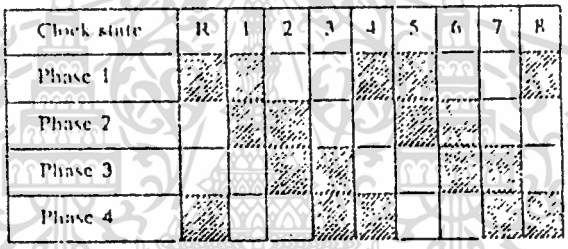
ดังที่ทราบกันอยู่แล้วว่าในการทำให้ สเต็ปปิ้งมอเตอร์ หมุนได้นั้นจะต้องกระตุ้นเฟสในทิศทางกลับกัน ซึ่งในการกระตุ้นเฟสของ สเตเตอร์เราไม่จำเป็นต้องกระตุ้นเฟสเพียงแค่เฟสเดียว อาจจะกระตุ้นทีละสองเฟสก็ได้ ในการกระตุ้นเฟส สเตเตอร์ มี 3 แบบคือ

- กระตุ้นเฟสเดียว (Single Phase Excitation)
- กระตุ้นสองเฟส (Two Phase Excitation)
- กระตุ้นเฟสครึ่ง (One end Two Phase Excitation)
  เป็นการกระตุ้นแบบผสมที่ใช้แบบเฟสเดียวร่วมกับแบบสองเฟส

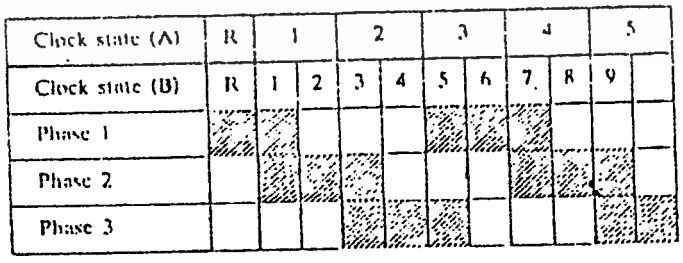


Note: Symbol R indicates 'reset'.

รูปที่ 2.8 แสดงซีเควนเชียลของการกระตุ้นแบบเฟสเดียว



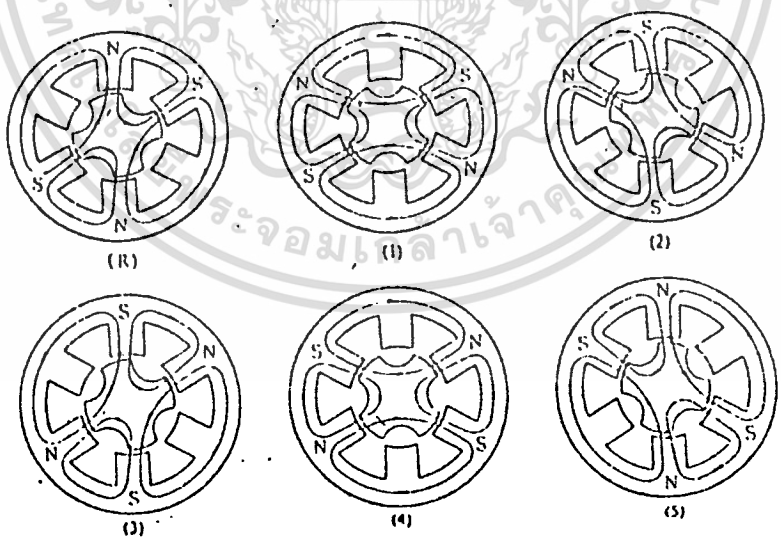
รูปที่ 2.9 แสดงซีเควนเชียลของการกระตุ้นแบบสองเฟส



รูปที่ 2.10 แสดงการกระตุ้นแบบผสม

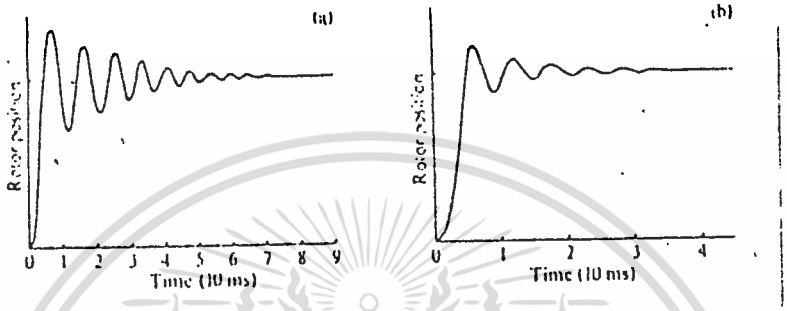
ในการกระตุ้นสเต็ปปิ้งมอเตอร์แบบสองเฟส เส้นแรงแม่เหล็กจะไม่ผ่านแกนเหล็กเป็นเส้นตรงเลยทีเดียวเหมือนแบบเฟสเดียว แต่จะวกกลับมาเข้าสู่แกนทางด้านข้างๆ ดังแสดงในรูปที่ 2.11 และเส้นแรงส่วนหนึ่งจะมาจากแกนตรงข้ามกัน แสดงในรูปที่ 2.12

รูปที่ 2.11 เส้นแรงแม่เหล็กเมื่อกระตุ้นแบบสองเฟส



รูปที่ 2.12 ลักษณะเส้นแรงแม่เหล็กเมื่อขับมอเตอร์แบบสองเฟส

การขับสเต็ปปิ้งมอเตอร์แบบกระตุ้นทีละสองเฟสนี้มีลักษณะเดียวกันกับการ ขับแบบกระตุ้นเฟสเดียว แต่ในการกระตุ้นแต่ละครั้งนั้นจะกระตุ้นทีละ สองเฟส จะต่างกันที่ช่วงทรานส์เซียน(Transient Response)ในการเปลี่ยนตำแหน่งแต่ละขั้นโดยสังเกตจากรูปที่ 2.13



รูปที่ 2.13 แสดงข้อแตกต่างของทรานส์เซียนของการกระตุ้นแบบเฟสเดียว และ สองเฟส

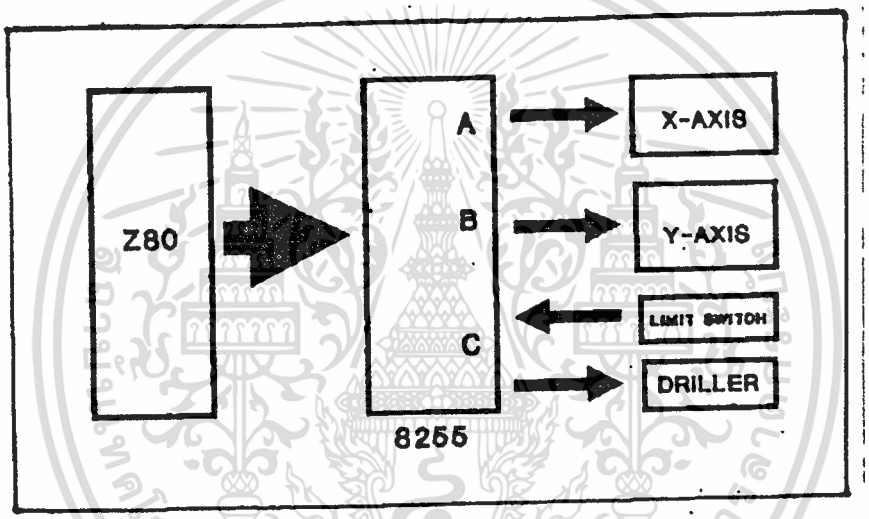
ซึ่งแสดงให้เห็นว่าขับแบบเฟสเดียวนั้นเกิดออสซิเลชั่นแดมป์ (Oscillation Damps) ในช่วงตอบสนองของทรานส์เซียนมากกว่าการขับแบบสองเฟส ความเร็วของทั้งสองแบบจะเท่ากัน แต่การกระตุ้นแบบสองเฟสจะใช้กระแสมากกว่า 2 เท่า

จากหลักการการทำงานของสเต็ปปิ้งมอเตอร์ตามที่ได้กล่าวมา ได้นำข้อดีคือสามารถรู้ตำแหน่งของสเต็ปปิ้งมอเตอร์ได้ตลอดเวลาอย่างแม่นยำมาประยุกต์ใช้งานโดยควบคุมด้วยไมโครโปรเซสเซอร์ (Microprocessor) มาสร้างเป็นต้นแบบเครื่องเจาะแผ่นวงจร พร้อมกันนี้ได้สร้างวงจรส่วนขับสเต็ปปิ้งมอเตอร์ และใช้โปรแกรมภาษาแอสเซมบลี้ควบคุมการทำงาน

บทที่ 3
วงจรขับสเต็ปปิ้งมอเตอร์ และแมคคานิคส์

3.1 โครงสร้างทางฮาร์ดแวร์ของวงจรไดรฟ์เวอร์

วงจรไดรฟ์เวอร์เป็นวงจรที่ต่ออยู่ระหว่าง ซีพียู และ แมคคานิคส์ เพื่อที่จะสามารถควบคุมการทำงานของแมคคานิคส์ให้เป็นไปตามความต้องการ สำหรับโครงสร้างฮาร์ดแวร์ของวงจรไดรฟ์เวอร์ได้แสดงด้วยบล็อกไดอะแกรมในรูปที่ 3.1 และรายละเอียดของวงจรได้แสดงไว้ในรูปที่ 3.2.1 และวงจรที่สร้างเสร็จสมบูรณ์แสดงไว้ในรูปที่ 3.2.2



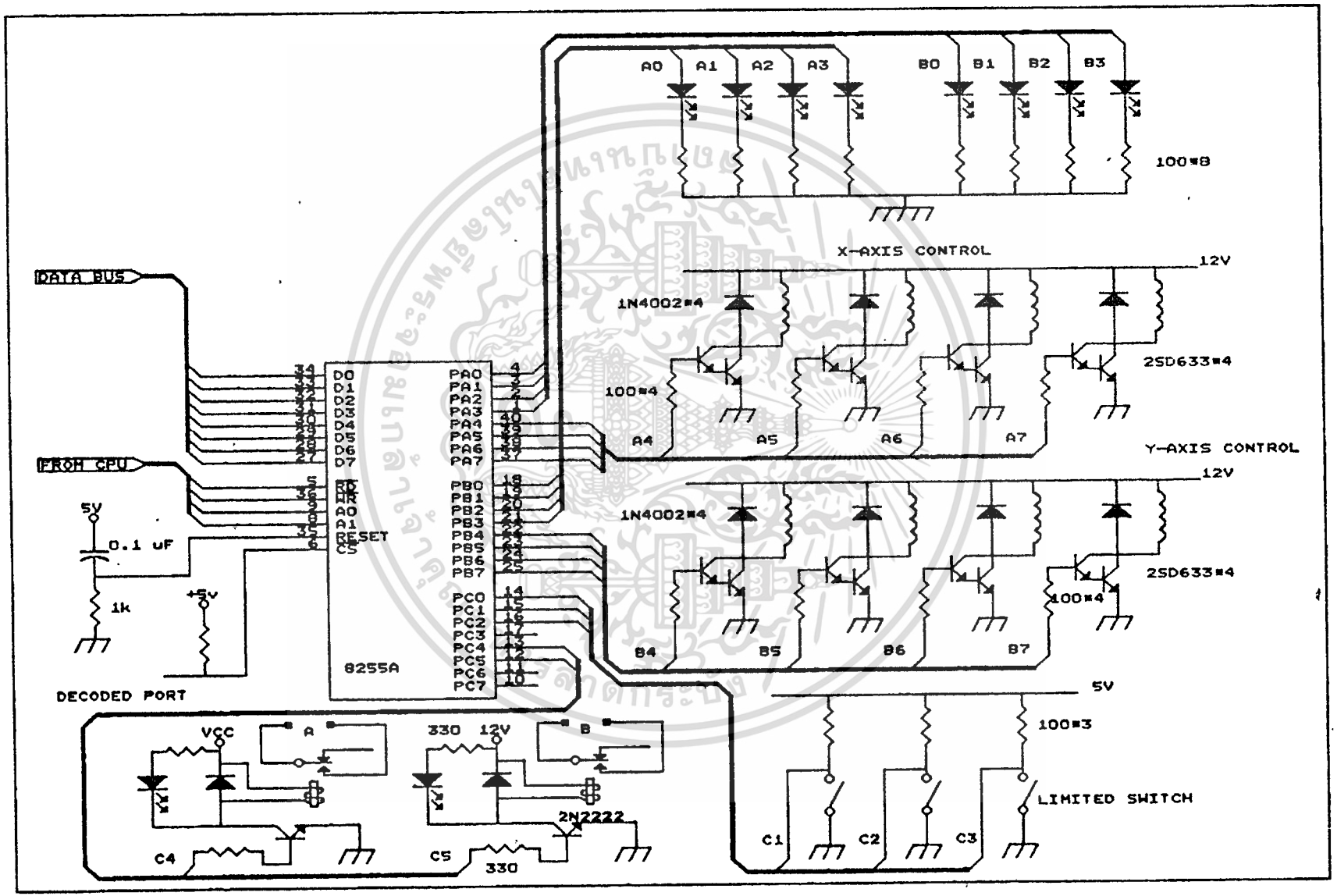
รูปที่ 3.1 บล็อกไดอะแกรมวงจรไดรฟ์เวอร์

3.2 Z80 ไมโครโปรเซสเซอร์ (MICROPROCESSOR)

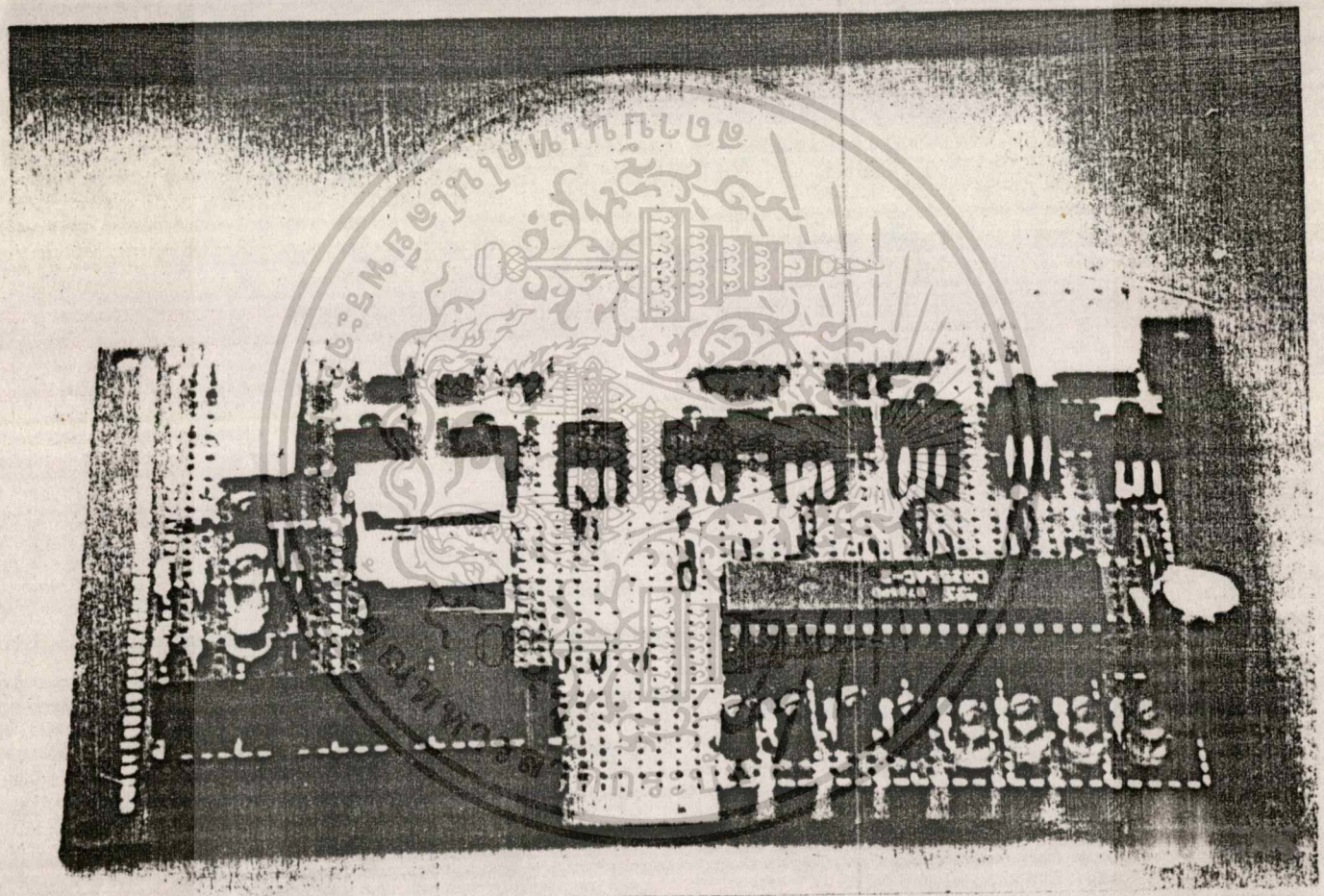
Z80 ซีพียู เป็นไมโครโปรเซสเซอร์ขนาด 8 บิทที่นิยมแพร่หลายมาก เพราะใช้งานได้ง่าย และมีราคาถูก Z80 ได้รับการพัฒนามาจาก 8080 แต่มีข้อดีเหนือกว่า 8080 คือ มีชุดคำสั่งมากกว่า 8080 ถึง 78 คำสั่ง ,รีจิสเตอร์มากกว่า 12 รีจิสเตอร์ และ 8080 ไม่สามารถนำไปใช้งานตัวเดียวได้ต้องมี ตัวกำเนิดสัญญาณนาฬิกา ,ตัวควบคุมระบบแต่ใน Z80 ได้รวมเอาลักษณะพื้นฐานทั้งหมดไว้ในชิพเดียวกัน และเพิ่มประสิทธิภาพทางฮาร์ดแวร์ ซอฟแวร์ และการอินเตอร์เฟส ให้สูงขึ้น

โครงสร้างภายในของ ซีพียู หรือ หน่วยประมวลผลกลาง ได้แสดงไว้โดยบล็อกไดอะแกรมในรูปที่ 3.3 ซึ่งแต่ละบล็อกมีลักษณะการทำงานดังนี้คือ

3.2.1 **Arithemetic Logic Unit (ALU)** เป็นหน่วยที่ทำหน้าที่ในการคำนวณฟังก์ชันทางคณิตศาสตร์ และการกระทำฟังก์ชันทางลอจิก



รูปที่ 3.2.1 แสดงวงจรไดรฟ์เวอร์



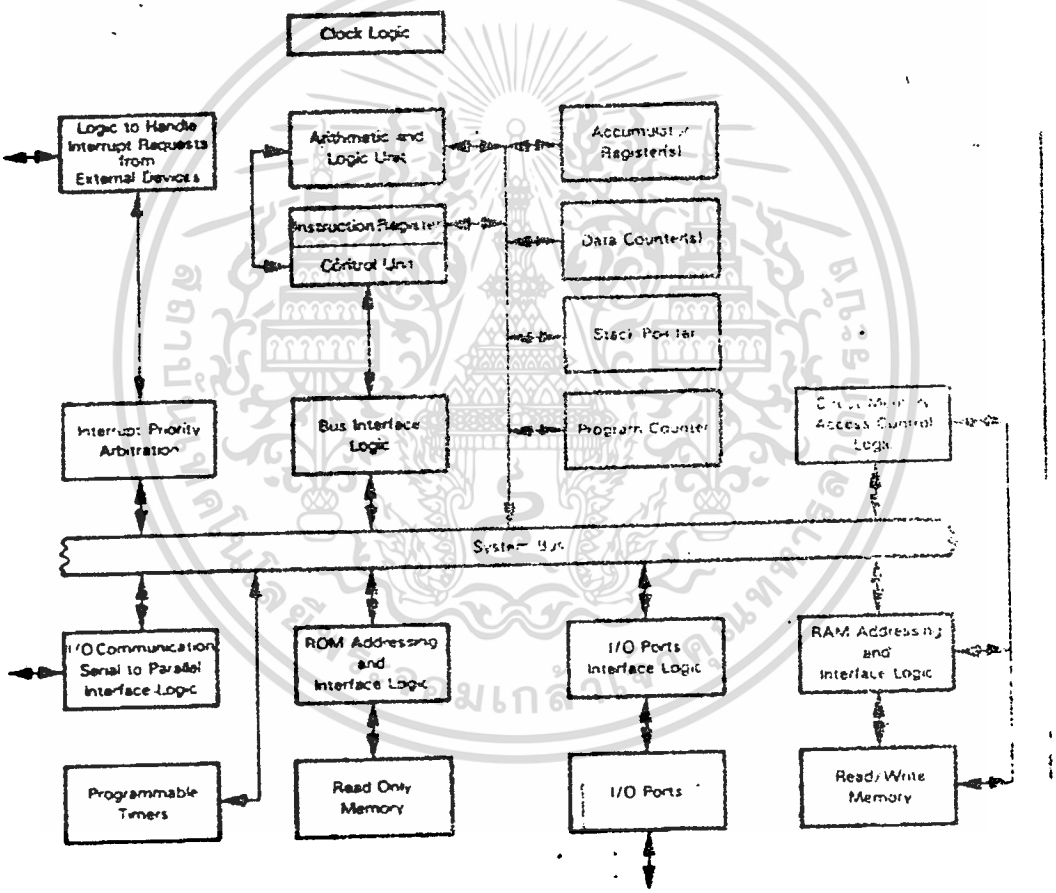
รูปที่ 3.2.2 แสดงอุปกรณ์ของวงจรขับแมคคานิคส์ (Driver)

3.2.2 **Control Unit** เป็นหน่วยที่ทำหน้าที่ส่งสัญญาณไปควบคุมอุปกรณ์ต่างๆที่ต่ออยู่กับ ซีพียู ให้ทำงานได้อย่างถูกต้อง

3.2.3 **Data Bus** เป็นบัสสองทางที่ใช้ส่งผ่านข้อมูลระหว่าง ซีพียู และ อุปกรณ์อื่นๆภายในระบบ โดยที่ Z80 มีดาต้าบัส 8 เส้น

3.2.4 **Control Bus** เป็นบัสทางเดียว ที่ใช้ส่งสัญญาณควบคุม แก่อุปกรณ์ต่างๆภายในระบบ

3.2.5 **Address Bus** เป็นบัสทางเดียว ใช้ส่งค่าแอดเดรสจาก ซีพียู ออกไปยังหน่วยความจำเพื่อระบุตำแหน่งที่ต้องการรับและส่งข้อมูล

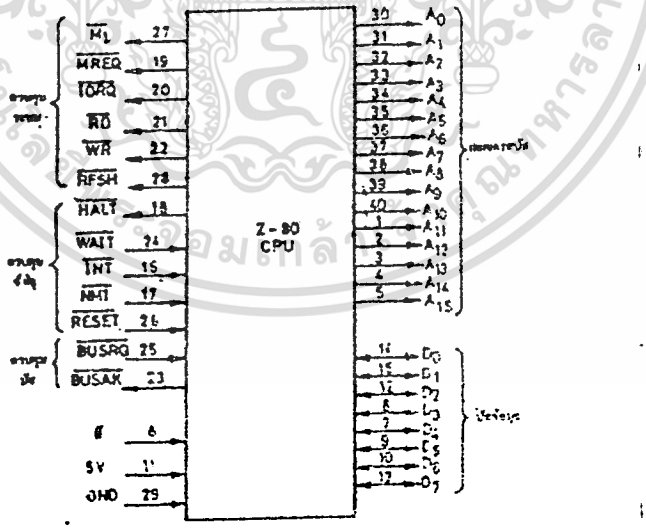


รูปที่ 3.3 แสดงบล็อกไดอะแกรมของ Z80

สำหรับโครงสร้างภายนอกเป็น แพคเกจ 4 ขาแบบ DIP ซึ่งรายละเอียดของขาได้แสดงไว้ในรูปที่ 3.4 และมีดังนี้คือ

A0-A15 เป็นขาสัญญาณที่สามารถบ่งบอกตำแหน่งได้ถึง 65536 ตำแหน่ง A0-A7 จะแสดงตำแหน่งของพอร์ทที่ Z80 ต้องการติดต่อด้วย

D0-D7	เป็นขาสัญญาณอินพุทและเอาท์พุทแบบสองทิศทาง ใช้เพื่อส่งผ่านข้อมูลระหว่าง Z80 กับหน่วยความจำหรืออุปกรณ์ I/O
$\overline{M1}$	จะแอคทีฟเมื่อ Z80 ทำการเฟทช์ออพโคดคำสั่ง
$\overline{MREQ}$	เป็นสายสัญญาณเพื่อติดต่อระหว่าง Z80 กับหน่วยความจำ
$\overline{RD}$	เป็นขาเอาท์พุท แอคทีฟเมื่อ Z80 ต้องการอ่านข้อมูล
$\overline{WR}$	เป็นขาเอาท์พุท และจะแอคทีฟเมื่อ Z80 ต้องการส่งข้อมูลออก
$\overline{RFSH}$	จะทำงานเมื่อ A0-A6 ของบัสแอดเดรสให้ค่ารีเฟรสออกมา
$\overline{HALT}$	การสั่ง NOP (no operation) และจะหลุดจาก Halt ได้เมื่อถูกรีเซ็ทหรือถูกอินเทอร์รัพท์
$\overline{WAIT}$	เป็นสัญญาณอินพุทเพื่อให้อุปกรณ์ภายนอกทำงานให้ทันกับ Z80
$\overline{INT}$	สัญญาณขออินเทอร์รัพ (interrupt reguest)
$\overline{NMI}$	เป็นสัญญาณขออินเทอร์รัพท์ที่มีความสำคัญสูงกว่า $\overline{INT}$
$\overline{RESET}$	การรีเซ็ทจะทำให้โปรแกรมเคาว์เตอร์เป็น 0000 และในสภาวะการรีเซ็ทนี้ บัสแอดเดรสและบัสข้อมูลจะเป็น อิมพิแดนซ์สูง (high impedance)
$\overline{BUSRQ}$	เป็นสัญญาณที่ให้แก่ Z80 เมื่อต้องการใช้บัสข้อมูลและบัสแอดเดรส
$\overline{BUSAK}$	เป็นการตอบสนองสัญญาณ $\overline{BUSRQ}$

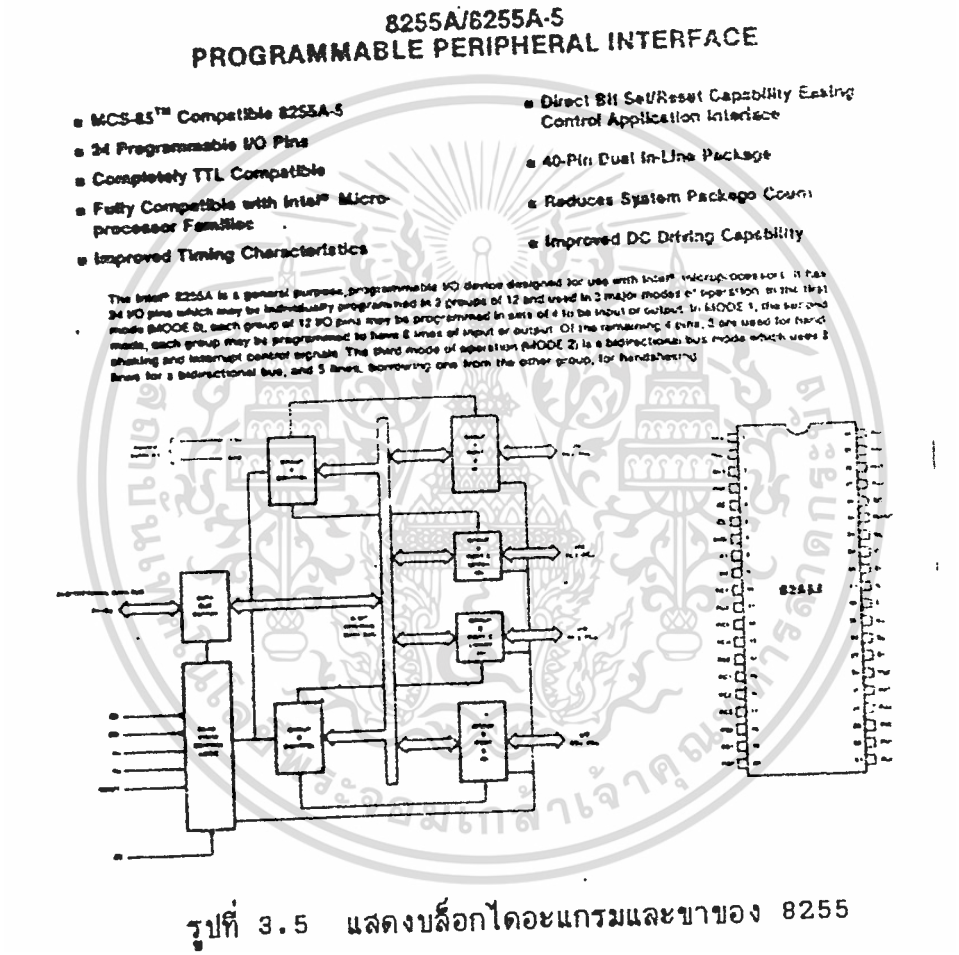


รูปที่ 3.4 แสดงลักษณะของขาไอซี Z-80 ซีพียู

การที่จะนำ Z80 ไปใช้งานได้จะต้องมีอุปกรณ์อื่นร่วมด้วย และการทำงานของ Z80 จะถูกควบคุมโดยชุดคำสั่ง หรือที่เรียกว่า ซอฟแวร์ ซึ่งซอฟแวร์ของระบบนี้จะได้กล่าวไว้ในบทที่ 4 ต่อไป

3.3 8255 PIA

8255 PIA (Programmable Interface Adapter) เป็นอุปกรณ์ LSI บรรจุอยู่ใน Package ขนาด 40 ขาแบบ DIP (Dual-in-Line Package) เริ่มผลิตโดยบริษัท Intel Cooperation โดยเริ่มแรกมีจุดประสงค์จะใช้งานร่วมกับ 8080 แต่ในภายหลังได้มีการทำงานร่วมกับ ซีพียู เบอร์ต่างๆรวมทั้ง Z80 ด้วย ในรูปที่ 3.5 แสดงบล็อกไดอะแกรมและขาของ 8255



รูปที่ 3.5 แสดงบล็อกไดอะแกรมและขาของ 8255

หน้าที่ของแต่ละบล็อกมีดังนี้คือ บล็อกกลุ่มแรกได้แก่ บล็อกจำนวน 4 บล็อกที่อยู่ทางด้านขวาของรูป ซึ่งจะเป็นส่วนที่เชื่อมต่อกับอุปกรณ์ภายนอกอื่นโดยมีสาย PA0-PA7, PB0-PB7 เป็นทางผ่านของข้อมูลระหว่างอุปกรณ์ภายนอกกับ 8255 สายสัญญาณเหล่านี้จะถูกแบ่งออกเป็น 3 I/O พอร์ท ได้แก่ พอร์ท A(PA),พอร์ท B(PB),พอร์ท C(PC) แต่ละพอร์ทเป็นไปได้ทั้งอินพุทและเอาท์พุท และแต่ละบล็อกจะมีสายสัญญาณเชื่อมต่อกับบัสข้อมูลภายในของ 8255

บล็อกกลุ่มที่ 2 คือ ชุดควบคุมเอ และ ชุดควบคุมบี ซึ่งเป็นตัวกำหนดลักษณะของทั้ง

3 I/O พอร์ท (8255 มีลักษณะการทำงานที่แตกต่างกันอยู่ 3 โหมด สามารถกำหนดได้โดยการโปรแกรมส่ง คอนโทรลเวิร์ด-Control Word ให้กับ 8255) พอร์ท C จะประกอบด้วยพอร์ทขนาด 4 บิท 2 พอร์ท กลุ่มหนึ่งจะถูกควบคุมโดย ชุดคอนโทรลเอ (Group A Control) และอีกกลุ่มหนึ่งจะถูกควบคุมโดยชุดคอนโทรลบี (Group B Control)

บล็อกกลุ่มสุดท้ายที่จะกล่าวถึงได้แก่ บัฟเฟอร์ของดาต้าบัส และลอจิกควบคุมสัญญาณ ซึ่งบล็อกเหล่านี้จะเป็นส่วนที่ติดต่อกับ ซีพียู และลอจิกควบคุมข้อมูลจะเป็นส่วนที่ควบคุมให้ข้อมูลเข้าหรือออกจากริจิสเตอร์ภายใน ตัวที่ถูกต้องและในเวลาที่เหมาะสม

รายละเอียดในการจัดเรียงขาของ 8255

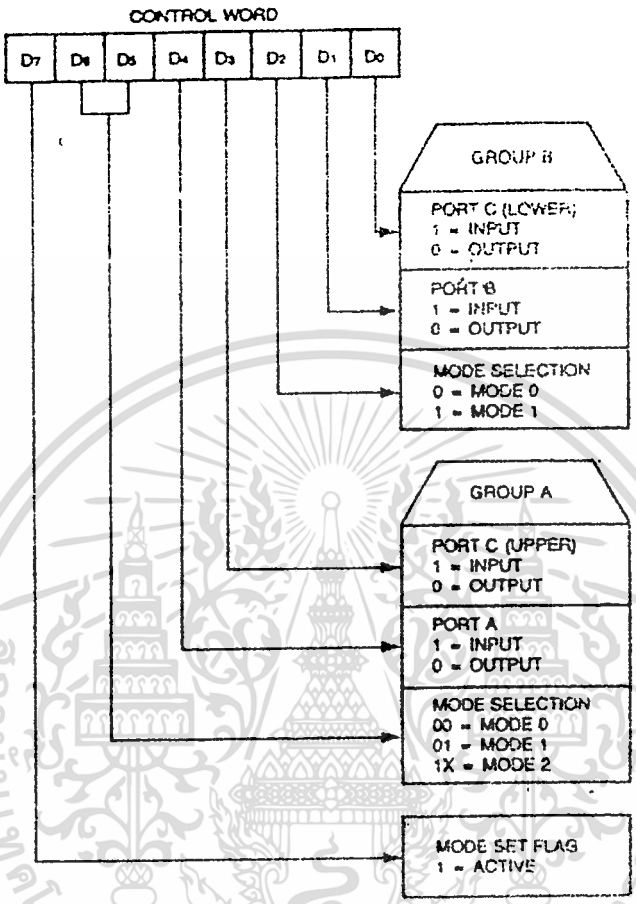
D0-D7	เป็นสายข้อมูล อินพุท/เอาท์พุทแบบสองทิศทาง เป็นทางผ่านข้อมูลระหว่างพอร์ทต่างๆของ 8255 กับบัสข้อมูลของ Z80
$\overline{\text{CS}}$	(chip select) เมื่อขานี้มีลอจิกเป็น 0 แล้ว ซีพียู จะสามารถติดต่อเพื่ออ่านหรือเขียนกับ 8255 ได้
$\overline{\text{RD}}$	จะต้องแอคทีฟร่วมกับสัญญาณ CS คือเป็นลอจิก "0" แล้ว ซีพียู จะสามารถอ่านข้อมูลออกไปได้
$\overline{\text{WR}}$	เช่นเดียวกับ $\overline{\text{RD}}$ แต่เป็นการส่งข้อมูลให้แก่ 8255
RESET	เมื่อมีลอจิกเป็น 1 จะทำการเซ็ทพอร์ทของ 8255 ทุกพอร์ท
A0-A1	จะเป็นตัวกำหนดริจิสเตอร์ภายในของ 8255
PA0-PA7	
PB0-PB7	ทั้งสองพอร์ทจะเป็น I/O ติดต่อกับอุปกรณ์ภายนอกอื่นๆ
PC0-PC7	ขาสัญญาณเหล่านี้จถูกใช้เป็นพอร์ท I/O ขนาด 8 บิทใช้ต่อกับอุปกรณ์ภายนอกอื่นๆ แต่ กลุ่มขาของสัญญาณเหล่านี้สามารถแบ่งออกเป็น 2 กลุ่มแต่ละกลุ่มมีขนาด 4 บิท กลุ่มแรกจะใช้ควบคิม PB0-PB7 และอีกกลุ่มจะใช้ควบคุม PA0-PA7

<u>การต่อ 8255 กับ Z80</u>

ในการต่อ 8255 เข้ากับ Z80 นั้น สัญญาณต่างๆจะเหมือนกับที่ Z80 ติดต่อกับ I/O อื่นๆ แต่เนื่องจาก 8255 มีขาอินพุท A0 และ A1 อยู่แล้วโดยจะต่อตรงกับแอดเดรสบัส นั่นคือ 8255 สามารถที่จะถอดพอร์ทแอดเดรสได้ 4 ค่า (2^2)

สำหรับ โค้ดสำหรับเลือกของเครื่องเจาะ PCB นี้คือ 81H ,80H เป็นพอร์ท A 81H เป็นพอร์ท B ,82H เป็นพอร์ท C และ 83H เป็นคอนโทรลพอร์ท และในการใช้งานของ 8255 ในระบบนี้ได้กำหนดให้ทำงานในโหมด 0 ซึ่งรายละเอียดในการโปรแกรมให้

8255 ทำงานในโหมดต่างๆ ได้แสดงไว้ในรูปที่ 3.6



รูปที่ 3.6 แสดงรายละเอียดแต่ละบิทของรีจีสเตอร์ควบคุมของ 8255

การใช้งาน 8255 ในกรณีนี้ต้องการให้ พอร์ท A เป็นเอาท์พุท ,พอร์ท B เป็นเอาท์พุท,พอร์ท C ให้ 4 บิทล่างเป็นเอาท์พุท และ พอร์ท C ให้ 4 บิทบนเป็นอินพุท ดังนั้นจึง กำหนดให้ตั้งแต่ D1-D6 เป็น "0" หมด ,ให้ D1 เป็น "1"เพราะเป็นอินพุท และให้ D7 เป็น "1"เพื่อ กำหนดแฟล็ก จากการกำหนดทั้งหมดนี้ทำให้ได้คำสั่งควบคุมเท่ากับ "81H" คำสั่งของ Z80 ที่จะเซ็ทให้ 8255 อยู่ในลักษณะดังกล่าวได้แก่

```
LD   A,80H   ; control word
OUT (83H),A  ; send to control port
```

ข้อมูลที่ส่งให้แก่ พอร์ท A และ B ของ 8255 จะเป็นสัญญาณที่ส่งให้แก่ขาเบสของทรานส์ซิสเตอร์ซึ่งจะต่อกับขดลวดของสเต็ปปิ้งมอเตอร์ ดังนั้นเมื่อต้องการให้ขดลวดของสเต็ปปิ้งมอเตอร์ขดใดแอคทีฟ ก็จะให้ลอจิก "1"ไปที่บิทนั้น ซึ่งรูปแบบการส่งข้อมูลเพื่อหมุน

สเต็ปปิ้งมอเตอร์ เป็นดังนี้คือ

โหมด	ค่าที่ส่ง	ค่าที่ส่ง (Binary)
หมุนขวา (ตามลำดับ)	33H	0011 0011
	66H	0110 0110
	CCH	1100 1100
	99H	1001 1001
หมุนซ้าย (ตามลำดับ)	99H	1001 1001
	CCH	1100 1100
	66H	0110 0110
	33H	0011 0011

ตาราง 3.1 แสดงข้อมูลที่ส่งเพื่อหมุนสเต็ปปิ้งมอเตอร์

จากตารางที่ 3.1 จะเห็นว่าเราสามารถที่จะควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ไปในทิศทางที่เราต้องการได้ โดยการกำหนดค่าที่จะส่งออกไปทาง พอร์ท A (หรือ B) ดังที่แสดงไว้ในตาราง ซึ่งเป็นการกระตุ้นมอเตอร์ แบบ " Two Phase Excitation Sequential "

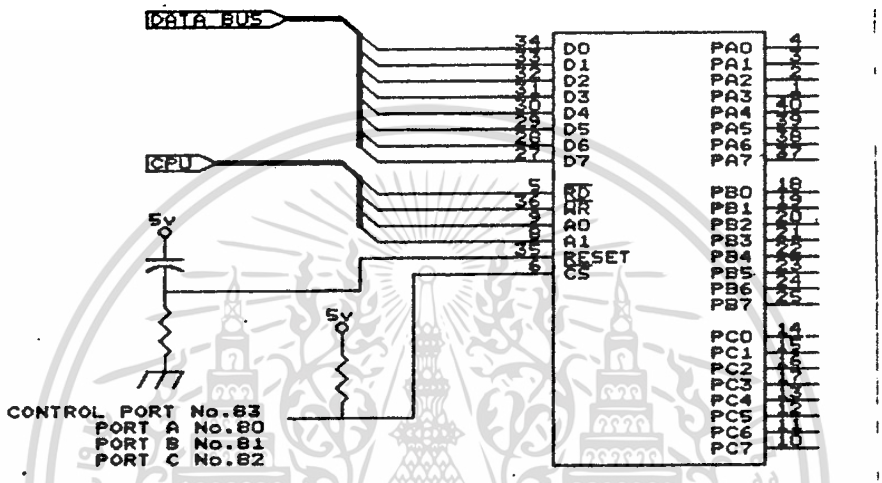
ส่วนในพอร์ท C นั้น 4 บิทล่างเป็นอินพุทเพื่อตรวจสอบสถานะในแต่ละแกนว่ามาอยู่ที่ตำแหน่ง 0 หรือไม่โดยที่

บิท	หน้าที่
C0	ตรวจสอบตำแหน่งในแกน X
C1	ตรวจสอบตำแหน่งในแกน Y
C2	ตรวจสอบตำแหน่งของสว่าน
C3	-

ตาราง 3.2 แสดงการใช้แต่ละบิทในพอร์ท C ล่าง

และพอร์ท C ใน 4 บิทบน (PC4-PC7) จะต่อกับทรานส์ซิสเตอร์ซึ่งต่อกับกับรีเลย์ เพื่อทำหน้าที่ปิดหรือเปิดส่วนของสว่านให้ทำงาน โดยที่ PC4 จะเป็นตัวควบคุมสว่าน และ PC5 จะควบคุมลูกเบี้ยว

จากที่กล่าวมาทั้งหมด จะเห็นได้ว่าเราสามารถที่จะควบคุมการทำงานทั้งหมดได้โดยส่งข้อมูลให้แก่ 8255 ซึ่งจะทำให้ระบบทำงานได้ตามต้องการ

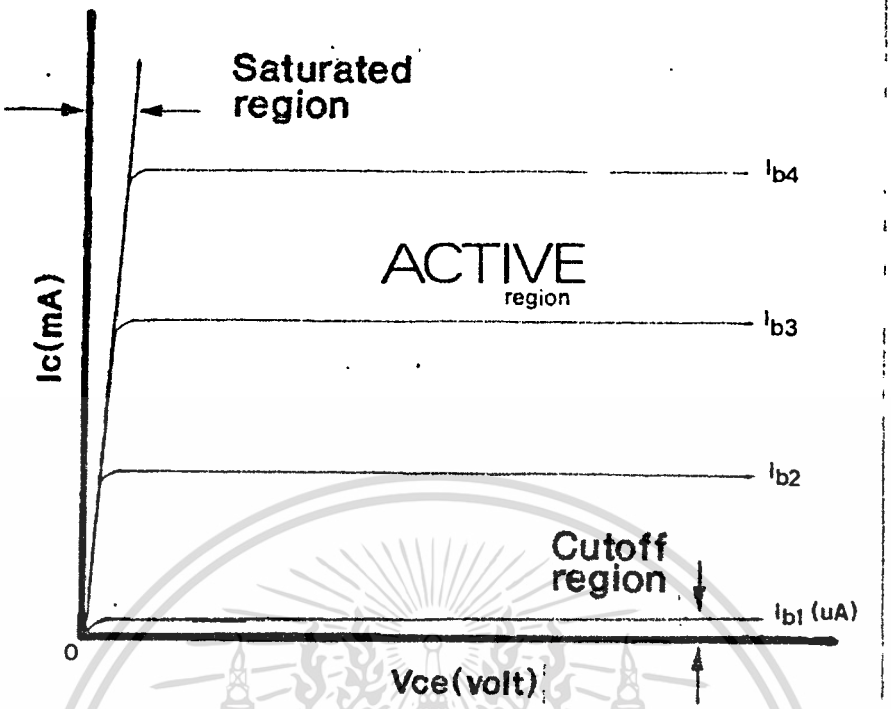


รูปที่ 3.7 แสดงการต่อระหว่าง 8255 และ Z80

3.4 สวิทช์ชิ่งทรานซิสเตอร์ (SWITCHING TRANSISTOR)

การป้อนไฟให้กับขดลวดเพื่อให้ สเต็ปปิ้งมอเตอร์ หมุนได้นั้น จำเป็นต้องใช้กระแสมาก แต่ในระบบดิจิตอลนั้นระดับของกระแสจะมีค่าต่ำมาก ดังนั้นเพื่อที่จะให้สัญญาณที่ส่งมาจาก 8255 สามารถควบคุมการทำงานของระบบได้ จึงได้นำเอา ทรานซิสเตอร์มาทำหน้าที่สองประการคือ ตัวขับ (drive) กระแสและเป็นตัวสวิทช์ชิ่ง (switching) โดยอาศัยหลักการทำงานของทรานซิสเตอร์ ในย่านคัดออฟ (cutoff) และย่านอิ่มตัว (saturated)

ในรูปที่ 3.8 เป็นกราฟแสดงคุณสมบัติของวงจรทรานซิสเตอร์ที่ต่อแบบอิมิเตอร์ร่วม (common-emitter characteristic curves) เมื่อทรานซิสเตอร์ทำงานอยู่ในแถบที่ต่ำกว่า I_{B1} จะเรียกย่านนี้ว่า"Cutoff" ทำให้ กระแสไม่มี หรือกระแสเบสจะเป็น 0 และทำให้ไม่มีกระแสไหลผ่าน CE ของทรานซิสเตอร์ ในกรณีนี้ I_c จะประมาณ I_{CEO} มีค่าต่ำกว่า 1 uA. เมื่อทรานซิสเตอร์ทำงานในย่าน"Active" จะได้ $I_c = h_{FE} \times I_B$ ซึ่งจะมีการขยายกระแส และเมื่อทรานซิสเตอร์ทำงานในย่าน "saturated" จะทำให้กระแสสามารถไหลผ่านจากไลน์ลงกราวด์ไปได้ โดยที่ $R_{...} = V_{...} / I_c$



รูปที่ 3.8 กราฟแสดงคุณสมบัติวงจรอิมิเตอร์ร่วม

จากหลักการดังกล่าว เมื่อป้อนลอจิก "0" ให้แก่ขาเบสของทราสซิสเตอร์จะทำให้ทราสซิสเตอร์อยู่ในสภาวะคัตออฟ ไม่สามารถนำกระแสได้ และเมื่อป้อนลอจิก "1" ให้แก่ทรานซิสเตอร์ ก็จะทำให้ทรานซิสเตอร์ทำงานอยู่ในย่านอิ่มตัว และสามารถนำกระแสได้ จากรูปที่ 3.8 จะเห็นได้ว่า I_B มีหน่วยเป็นไมโครแอมป์ แต่ I_C นั้นมีหน่วยเป็นมิลลิแอมป์ ซึ่งค่าของ I_C ได้จาก h_{FE} (current gain) คูณกับ I_B ซึ่งสามารถที่จะ exist ขดลวดของสเต็ปปิ้งมอเตอร์ได้

ในการสร้างวงจรไดรฟ์เวอร์นี้จะต้องขับกระแสมาก ดังนั้นจึงใช้ดาร์ลิงตันทรานซิสเตอร์เบอร์ 2SD633 ซึ่งจะทำมีอัตราการขยายกระแส เป็น $h_{FE} \times h_{FE}$

3.5 อุปกรณ์ที่ต่อกับ 8255

1) LED 4 ตัวแรกต่อเข้ากับขา PA0-PA3 ของ 8255 เพื่อแสดงการทำงานของสเต็ปปิ้งมอเตอร์ในแกน X ตามคำสั่งจาก CPU
2) ทรานซิสเตอร์ เบอร์ 2SD633 (ดาลิงตันคอมมอนอิมิเตอร์) 4 ตัวแรกต่อเข้ากับ PA4-PA7 โดยที่ขาของ คอลเลคเตอร์ ต่ออยู่กับคอยล์ของ สเต็ปปิ้งมอเตอร์ตามลำดับ และขา อิมิเตอร์ ต่อเข้ากับกราวน์ของวงจรเพื่อใช้ขับสเต็ปปิ้งมอเตอร์ในแกน x ให้ทำงานตามคำสั่ง
3) LED อีก 4 ตัวทำหน้าที่แสดงสถานะในแกน Y รายละเอียดเช่นเดียวกับในข้อ 1)

โดยต่ออยู่กับ PB0-PB3

4) ทรานซิสเตอร์ เบอร์ 2SD633 (ดาร์ลิงตันคอมมอนอิมิเตอร์) 4 ตัวแรกต่อเข้ากับ PB4-PB7 โดยที่ขาของ คอลเลคเตอร์ ต่ออยู่กับคอยล์ของ สเต็ปปิ้ง มอเตอร์ ตามลำดับ และขา อิมิเตอร์ ต่อเข้ากับกราวน์ของวงจรเพื่อใช้ขับสเต็ปปิ้งมอเตอร์ ในแกน Y ให้ทำงานตามคำสั่ง

5) ทรานซิสเตอร์ 2N2222 ซึ่งรับสัญญาณจาก PC4,PC5 ของ 8255 เข้าทางขาเบส และขา คอลเลคเตอร์ต่อกับ รีเลย์ ซึ่งเป็นตัวขับสว่านเจาะ และลูกเบี้ยวตามลำดับ โดยที่ขา อิมิเตอร์ ต่อลงกราวน์ มี LED แสดงสถานะการทำงานของริเลย์ คือ เมื่อ รีเลยี ตัวใดทำงาน LED ตัวนั้นจะสว่าง แสดงว่ากำลังขับรีเลย์ หรือ ลูกเบี้ยวอยู่

6) ขา RESET ต่ออยู่กับไฟ 5v โดยมีความต้านทาน 1K พูลดาวน์ (Pull Down)ไว้

7) ขา PC0-PC2 ของ 8255 ต่อกับลิมิตสวิทซ์ (Limit switch) ของเครื่องเจาะแผ่นปริ้นซ์ถ้า ลิมิตสวิทซ์อยู่ในสถานะปิด (close) โปรแกรมจะสั่งให้สเต็ปปิ้งมอเตอร์นั้นหยุดทำงาน

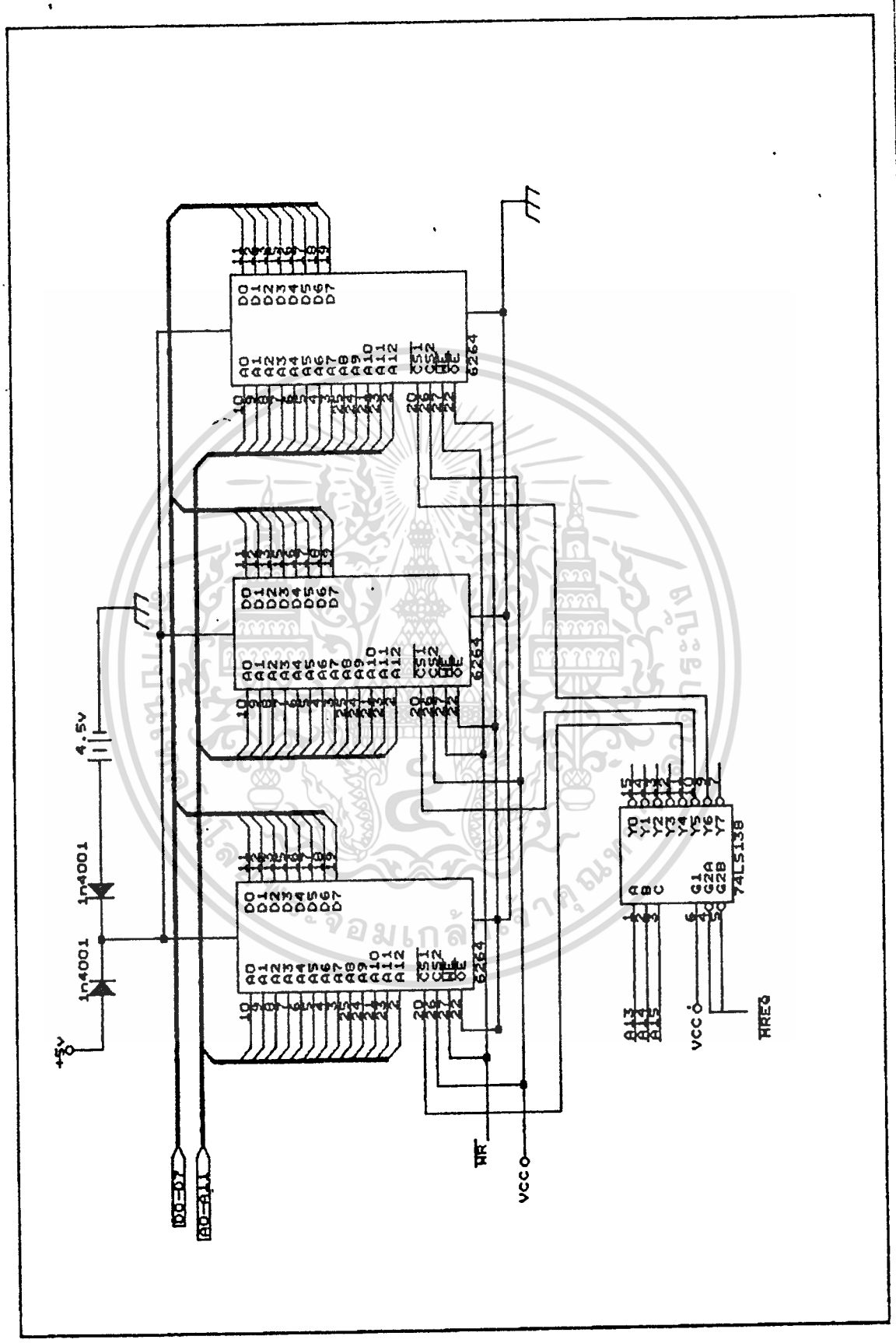
3.6 แรมแพค (RAM PACK)

เนื่องจากหน่วยความจำบน ET-board ที่ใช้ทำงานอยู่มีจำนวนจำกัด หน่วยความจำส่วนหนึ่งใช้สำหรับเขียนโปรแกรมควบคุมการทำงาน แต่ในการทำงานแบบอัตโนมัติมีความจำเป็นอย่างยิ่งที่ต้องใช้เนื้อที่ของหน่วยความจำเป็นจำนวนมาก เพื่อใช้ในการเก็บโคออดิเนท ดังนั้นจึงต้องขยายหน่วยความจำออกมา โดยใช้ แรม 6264 ทั้งหมด 3 ตัว โดยที่ แรม แต่ละตัวมี แอดเดรส ดังนี้คือ

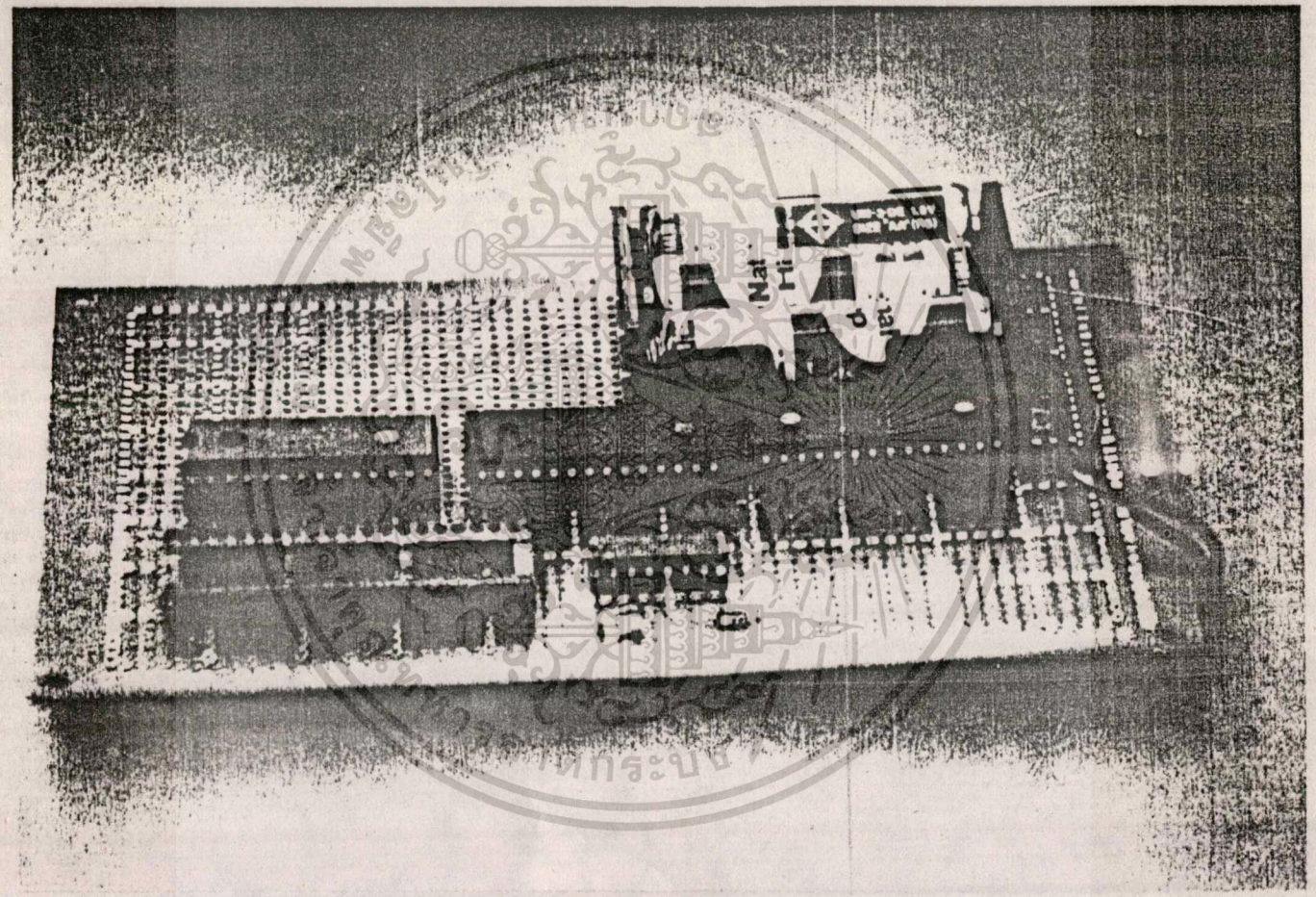
แรม	แอดเดรส
1	8000-9FFF
2	A000-BFFF
3	C000-DFFF

ตาราง 3.3 แสดงแอดเดรสของ แรม แต่ละตัว

โดยใช้ไอซี 74LS138 เป็นตัวดีโคดพอร์ทหน่วยความจำ และวงจรได้แสดงไว้ในรูป 3.9.1 และวงจรที่สร้างสมบูรณ์แสดงดังรูปที่ 3.9.2



รูปที่ 3.9.1 แสดงวงจรแรมแพค



รูป 3.9.2 แสดงอุปกรณ์ของวงจรแรมแพค

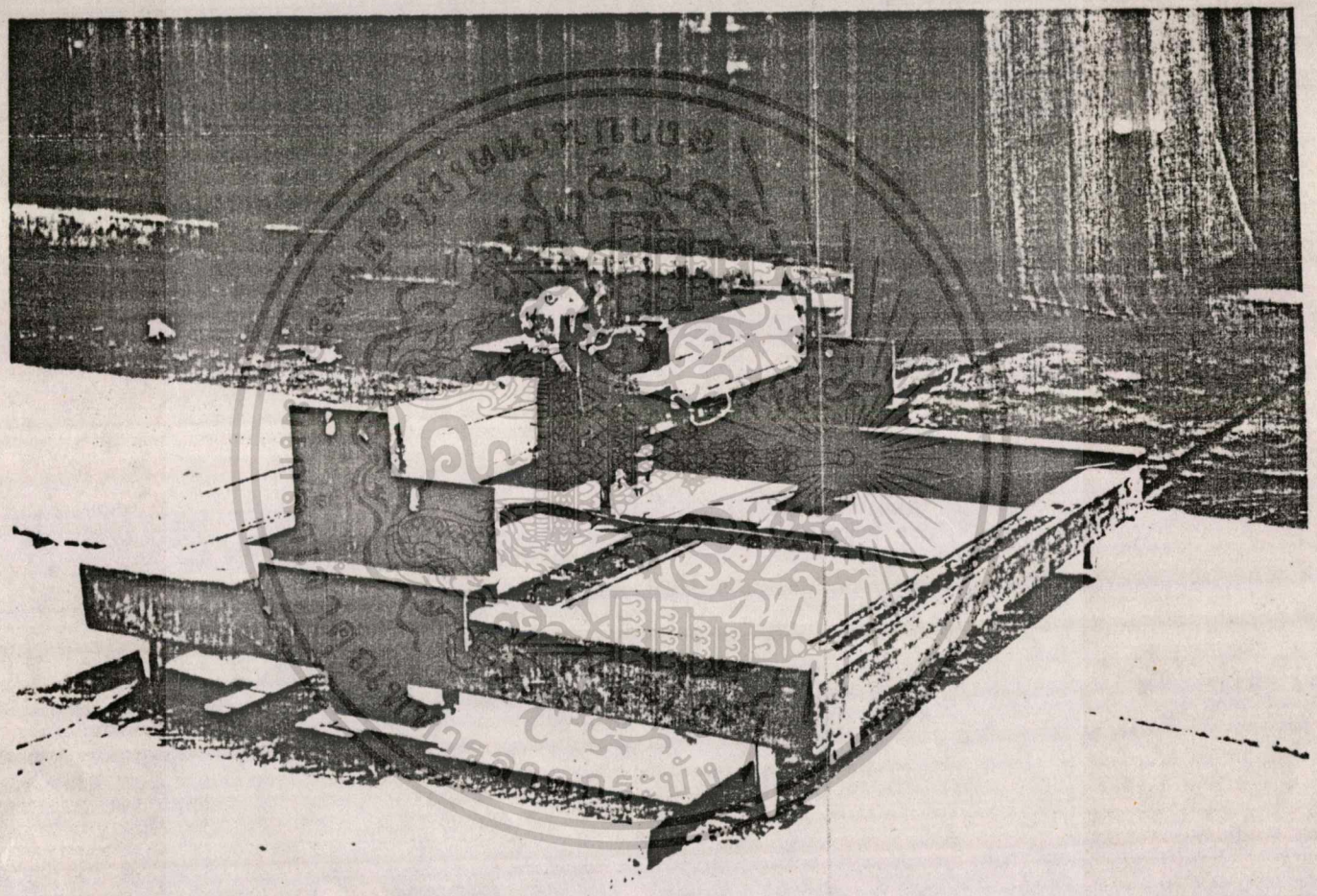
3.7 แท่นเจาะ

แท่นเจาะเป็นส่วนของแมคคานิคส์ โดยมีองค์ประกอบที่สำคัญคือ

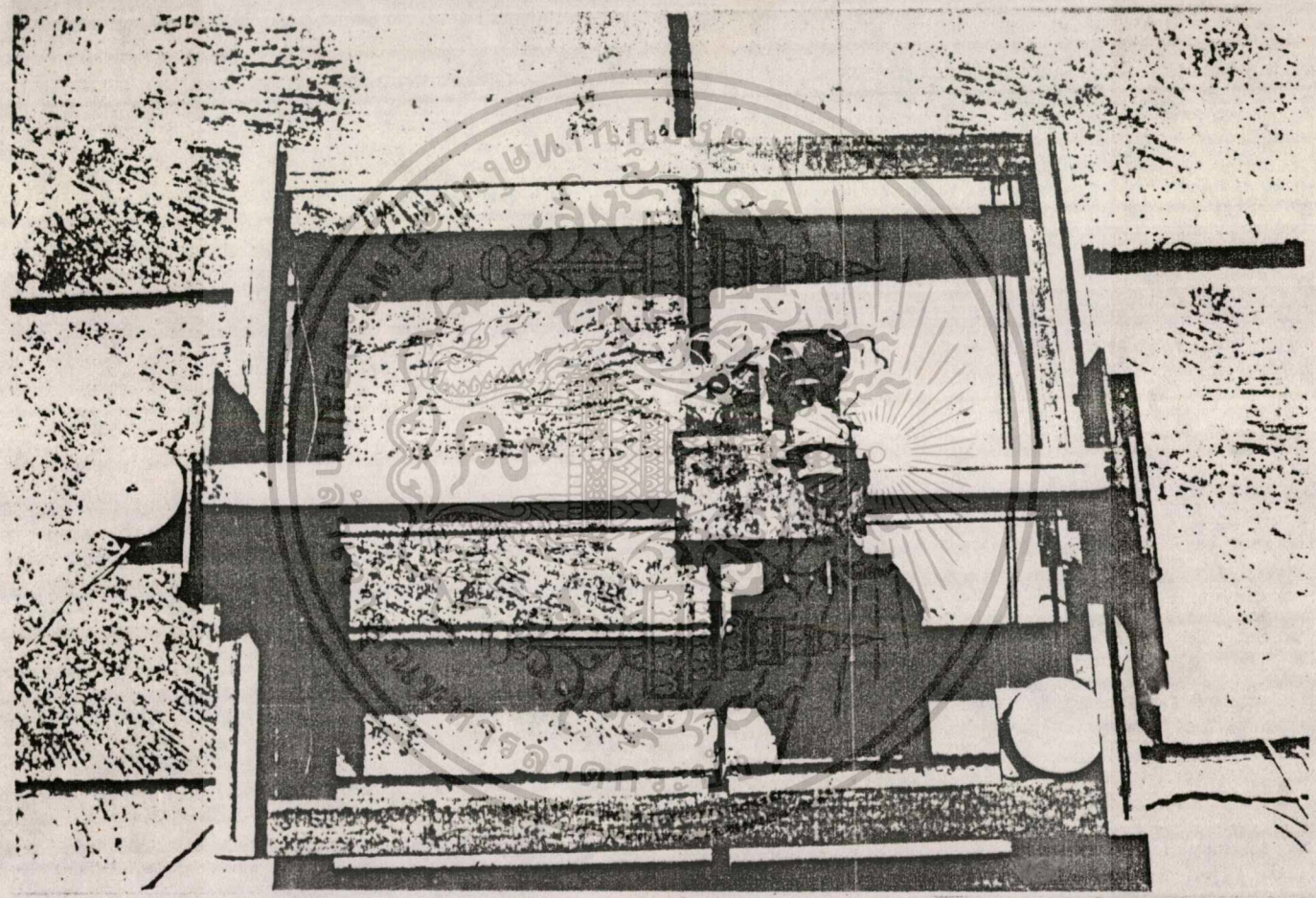
- สเต็ปปิ้งมอเตอร์ที่ควบคุมการเคลื่อนที่แกน x
- สเต็ปปิ้งมอเตอร์ที่ควบคุมการเคลื่อนที่แกน y
- ดีซีมอเตอร์ขับสว่านและลูกเบี้ยว
- ลิมิตสวิทซ์ตรวจสอบสถานะของการเคลื่อนที่

ในรูปที่ 3.10 เป็นภาพแสดงแท่นเจาะซึ่งมีส่วนประกอบตามที่กล่าวมา ,ในรูปที่ 3.11 จะแสดงตำแหน่งการวางสเต็ปปิ้งมอเตอร์เพื่อควบคุมการเคลื่อนที่ของแต่ละแกน และในรูปที่ 3.12 แสดงส่วนของสว่านอันประกอบด้วย สว่านและลูกเบี้ยว

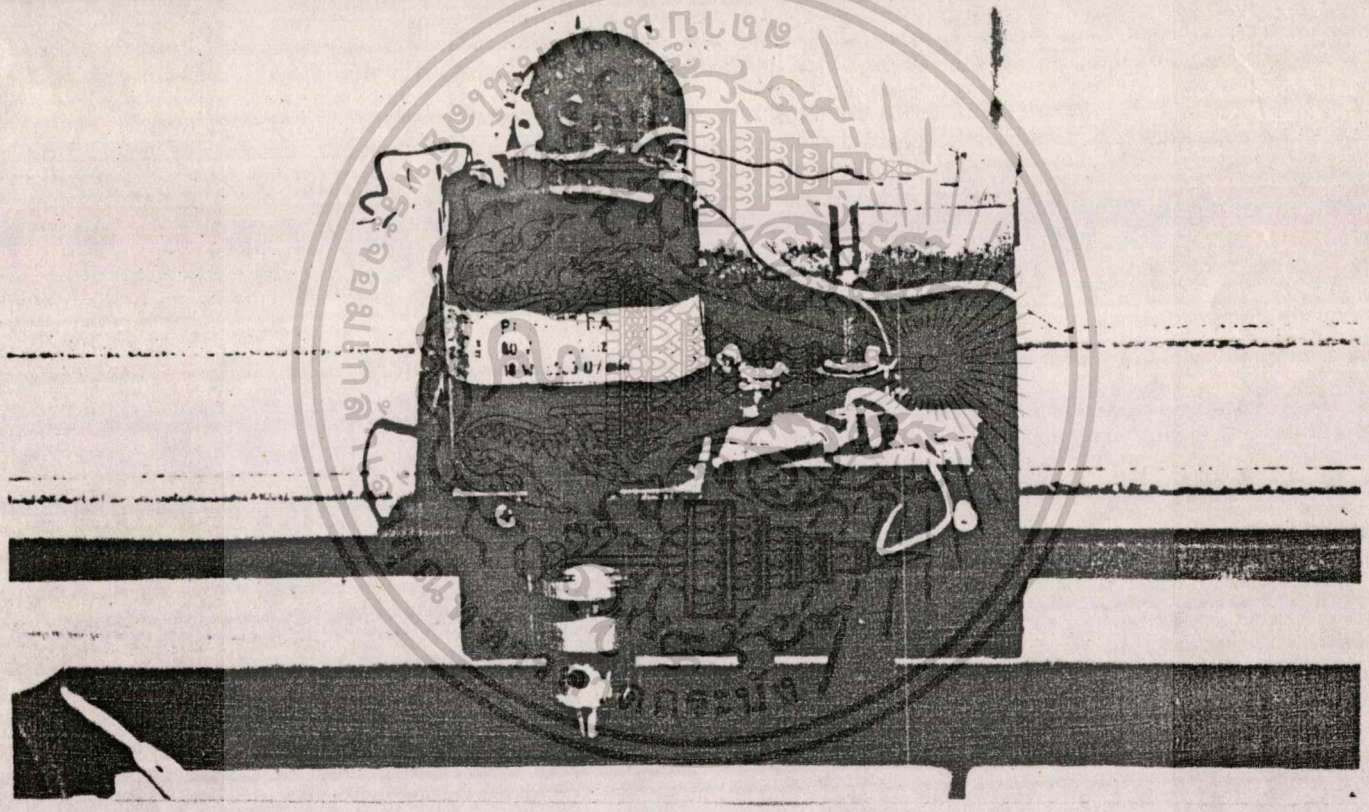
การเคลื่อนที่ในแนวแกน x และ แกน y สามารถที่จะทราบได้ว่าเคลื่อนที่มายังตำแหน่งเริ่มต้นหรือไม่โดยดูจากสถานะของลิมิตสวิทซ์ในแต่ละแกน เช่นเดียวกับสว่านเมื่อเคลื่อนที่ลงมาเจาะด้วยการหมุนของลูกเบี้ยว พอครบรอบก็จะกลับมาตำแหน่งเดิมและ ชนลิมิตสวิทซ์ โปรแกรมก็จะสั่งหยุดหมุนทั้งมอเตอร์ของสว่านและลูกเบี้ยว โดยการตรวจสอบสถานะในแต่ละแกนนั้น จะอ่านค่าเข้ามาทางพอร์ท c ของ 8255 ใน 4 บิทบน ทำให้สามารถทราบได้ว่า



รูป 3.10 แสดงลักษณะของแท่นเจาะ



รูป 3.11 แสดงตำแหน่งการวางสเต็ปปิ้งมอเตอร์ในแต่ละแกน



รูป 3.12 แสดงส่วนของสว่าน

คีย์	หน้าที่
C	เพิ่มค่าของ x
8	ลดค่าของ x
F	เพิ่มค่าของ y
B	ลดค่าของ y
D	เพิ่มค่าของ x และเพิ่มค่าของ y
9	เพิ่มค่าของ x และลดค่าของ y
E	ลดค่าของ x และเพิ่มค่าของ y
A	ลดค่าของ x และลดค่าของ y
4	ทำงานอัตโนมัติ
5	เก็บโคออดิเนท
6	เจาะ
7	จัดเรียงลำดับ

ตาราง 4.1 หน้าที่ของแต่ละคีย์

4.2 รูทีนแปลงค่า (CONVERT ROUTINE)

โปรแกรมส่วนจะทำหน้าที่ในการแปลงค่าของ x และy ในบัฟเฟอร์ (Buffer) ซึ่งเป็นตัวเลขฐานสิบหก เก็บอยู่ที่แอดเดรส 1300-1303

ในตอนแรกจะทำการแปลงเป็นตัวเลขฐานสิบ โดยเรียก Utility SubRoutine H - D มีโค้ด คือ 0A

เมื่อทำการแปลงได้เลขฐานสิบ แล้วจะทำการแยกตัวเลขดังกล่าวออกเป็นสามหลัก

- หลักร้อยของ x อยู่ที่ แอดเดรส 1304 ,ของ y อยู่ที่ แอดเดรส 1307
- หลักสิบ ของ x อยู่ที่ แอดเดรส 1305 ,ของ y อยู่ที่ แอดเดรส 1308
- หลักหน่วยของ y อยู่ที่ แอดเดรส 1306 ,ของ y อยู่ที่ แอดเดรส 1309

หลังจากที่ได้ค่าเหล่านี้มาแล้ว จะเรียก "Digit Routine" เพื่อทำการแปลงจากเลขฐานสิบให้เป็นโค้ดแสดงผล (Display Code) ดังแสดงในรูปที่ 4.3

4.3 โปรแกรมส่วนควบคุมการเคลื่อนที่

จากตารางที่ 4.1 จะเห็นว่า คีย์ที่ควบคุมการเคลื่อนมีทั้งหมดมี 8 คีย์ คือ C,8,F B,D,9,E,A ซึ่งจะครอบคลุมการเคลื่อนที่ได้ทั้ง 8 ทิศทาง ดังในรูปที่ 4.4

สำหรับการเคลื่อนที่นั้น ถ้าเป็นการเพิ่มค่าจะต้องเช็คก่อนว่าค่าที่เพิ่มนั้นเป็นค่าที่มากที่สุดหรือไม่ (ค่าที่มากที่สุดคือ 999 หรือ 03E7 ในฐานสิบหก) โดยมีรูทีนที่ใช้ตรวจสอบคือ "CHKMX" หากพบว่ายังไม่เป็นค่าที่มากที่สุด ก็จะทำการเพิ่มค่าและสั่งให้สเต็ปปิ้งมอเตอร์ของแกนนั้นๆหมุนไปทางขวา และถ้าเป็นค่าที่มากที่สุดแล้วก็จะไม่มีการเพิ่มค่า

ในทางตรงกันข้าม ถ้าเป็นการลดค่าก็จะต้องเช็คว่าค่าที่ลดนั้นเป็นค่าน้อยที่สุดหรือไม่ (ค่าที่น้อยที่สุดคือ 000) โดยมีรูทีนที่ใช้ตรวจสอบคือ "CHKMN" และหากพบว่ายังไม่เป็นค่าที่น้อยที่สุด ก็จะลดค่าและสั่งให้สเต็ปปิ้งมอเตอร์หมุนไปทางซ้าย

4.4 มาร์ค (MARK)

โปรแกรมนี้จะทำงานเมื่อมีการกดคีย์ "5" เมื่อเลื่อนสว่านมายังตำแหน่งที่ต้องการและต้องการที่จะเก็บโคออดิเนทดังกล่าวไว้เพื่อทำการเจาะอัตโนมัติต่อไป

ในการเก็บโคออดิเนททุกครั้งจะมีการนับจำนวนจุดไว้ด้วย โดยจำนวนจุดจะเก็บอยู่ที่ แอดเดรส 1360 และ แอดเดรส ที่ใช้เก็บโคออดิเนทจะถูกเก็บไว้ที่ แอดเดรส 1362 ดังแสดงในรูปที่ 4.5

4.5 เจาะ (DRILL)

ชุดเจาะของเครื่องเจาะ PCB นี้มีอยู่ 3 ส่วนคือ

- มอเตอร์ขับลูกเบี้ยว
- มอเตอร์ขับสว่าน
- ลิมิตสวิทช์

การทำงานในตอนแรก โปรแกรมจะทำให้สว่านหมุนก่อน (โดยส่งค่า 10 ที่พอร์ทC) หลังจากนั้นจึงสั่งให้ทั้งมอเตอร์ของสว่านและลูกเบี้ยวหมุน (ส่งค่า 30 ที่พอร์ท C) ซึ่งลูกเบี้ยวจะทำสว่านเคลื่อนที่ลงไปเจาะในตำแหน่งที่ต้องการ ในขณะเดียวกันก็จะต้องมีการเช็คสถานะของลิมิตสวิทช์ (อินพอร์ท c และเช็คบิทที่ 2) เมื่อสว่านเคลื่อนที่กลับมาอยู่ในตำแหน่งเดิม สถานะของลิมิตสวิทช์จะเป็น 0 ก็จะสั่งให้มอเตอร์ของลูกเบี้ยวและสว่านหยุดหมุน (โดยส่งค่า 00 ไปที่พอร์ท C) ดังแสดงในรูปที่ 4.6

4.6 โปรแกรมเรียงลำดับ

หลังจากที่มีการเก็บโคออดิเนทต่างๆมา เพื่อความสะดวกในการเคลื่อนที่เมื่อจะใช้งานแบบอัตโนมัติ จึงนำเอาโคออดิเนทเหล่านั้นมาเรียงลำดับ โดยเรียงลำดับค่าของ x จากน้อยไปมาก

หลักการในการเรียงลำดับ คือกำหนดค่าที่เป็นตัวเช็คตั้งแต่ 000 (น้อยที่สุด)จนถึง 999 (มากที่สุด) นำค่านี้ไปเปรียบเทียบกับโคออดิเนทที่เก็บเอาไว้ เมื่อพบว่าเท่ากัน

ก็จะดึงโคออดิเนทนั้นไปเก็บไว้ในที่ใหม่ ดังนั้นก็จะได้โคออดิเนทชุดเดิมแต่เรียงลำดับ x อยู่ในที่ใหม่ ตัวอย่างเช่น ถ้าเก็บโคออดิเนทไว้ 3 จุดคือ

จุดที่หนึ่ง 123,230
จุดที่สอง 060,100
จุดที่สาม 400,340.

โดยค่าที่เป็นตัวเช็คเริ่มจาก 000 เมื่อเปรียบเทียบกับจุดทั้งสามแล้วไม่มีค่าเท่าก็จะเพิ่มไปเรื่อยๆ จนถึงค่า x=060 จะเท่ากับจุดที่สอง ก็จะนำโคออดิเนทนี้ไปเก็บเป็นจุดแรกในที่ใหม่ และก็เพิ่มไปจนค่าตัวเช็คเท่ากับ 123 ซึ่งตรงกับจุดแรก ก็จะนำโคออดิเนทนี้เป็นเก็บเป็นจุดที่สอง และจุดที่สามตามลำดับ ดังนั้นเมื่อเรียงเสร็จแล้วจะได้ลำดับดังนี้

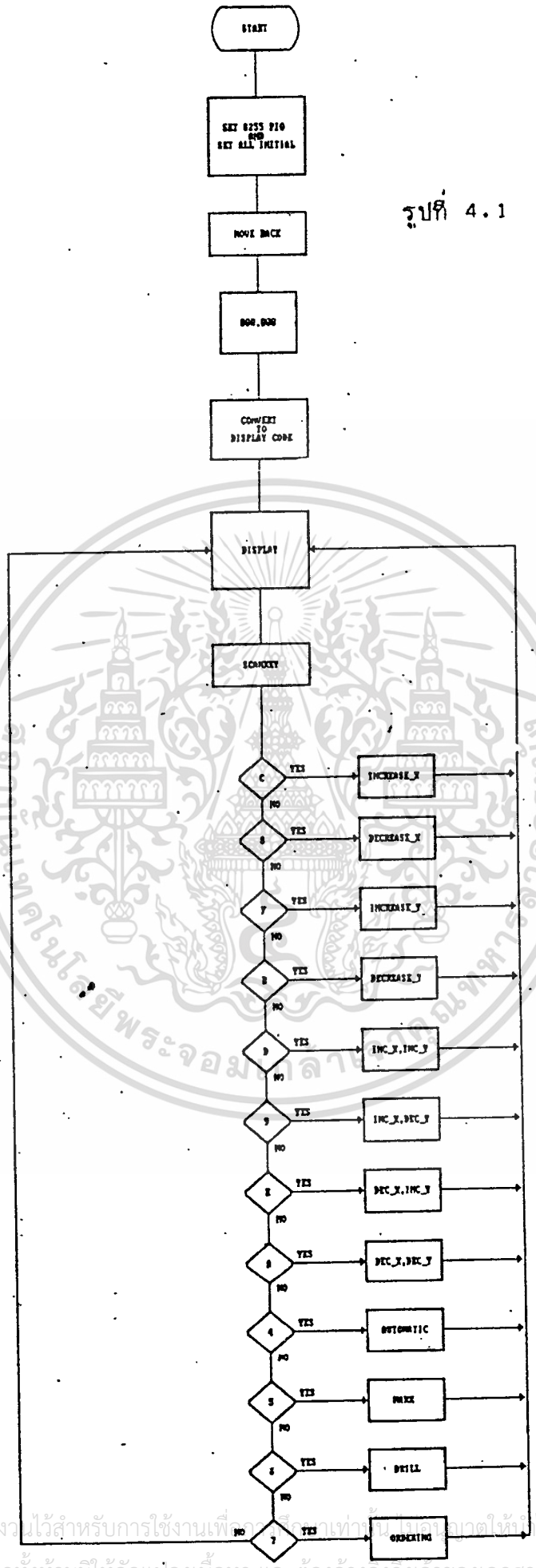
จุดที่หนึ่ง 060,100
จุดที่สอง 123,230
จุดที่สาม 400,340

จะเห็นได้ว่าโคออดิเนทได้ถูกจัดใหม่โดยเรียงค่าของ x จากน้อยไปมาก ซึ่งขั้นตอนต่างๆ ได้แสดงๆไว้ในรูปที่ 4.7

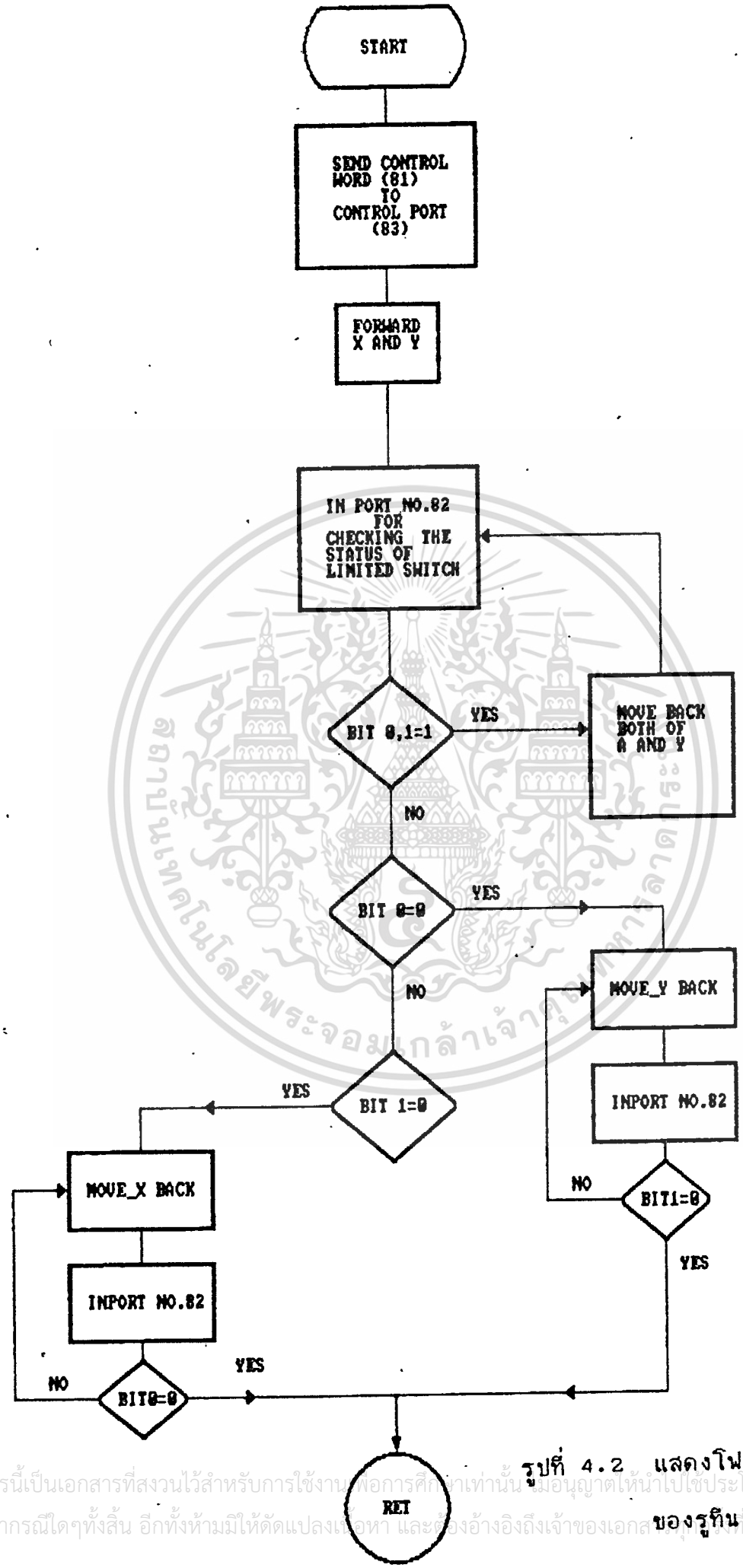
4.7 อัตโนมัติ (AUTOMATIC)

โปรแกรมนี้ได้แสดงไว้ในรูปที่ 4.8 หลักการสำคัญของโปรแกรมส่วนนี้ คือ การทำให้สว่านเลื่อนไปยังโคออดิเนทที่เก็บตำแหน่งนั้นเอาไว้ และทำการเคลื่อนที่ไปเจาะได้เองจนหมดทุกจุด

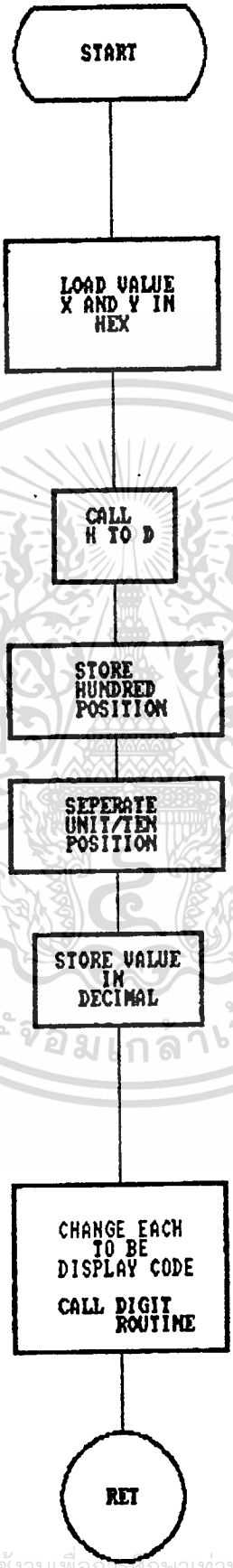
เมื่อโคออดิเนทต่างๆถูกจัดเรียบร้อยแล้ว โปรแกรมจะนำจุดเหล่านั้นมาลบกันเพื่อหาระยะทางที่จะเคลื่อนที่ไป เนื่องจาก x นั้นถูกเรียงลำดับจากน้อยมามากเรียบร้อยแล้ว ดังนั้นในการเคลื่อนที่ในแกน x จะเพิ่มขึ้นตลอด แต่สำหรับในแกน y จะต้องแยกพิจารณาเป็น 3 กรณีคือ y ที่ไม่มีการเปลี่ยนแปลง,y ที่เพิ่มขึ้น ,y ที่ลดลง โดยเฉพาะกรณีที่ y มีค่าลดลง เมื่อลบแล้วจะต้องนำผลลัพธ์ที่ได้มาหาระยะที่แท้จริงโดยทำ 2'คอมพลีเมนต์ ดังนั้นระบบจะสามารถเคลื่อนที่ไปยังโคออดิเนทต่างๆได้อย่างถูกต้อง



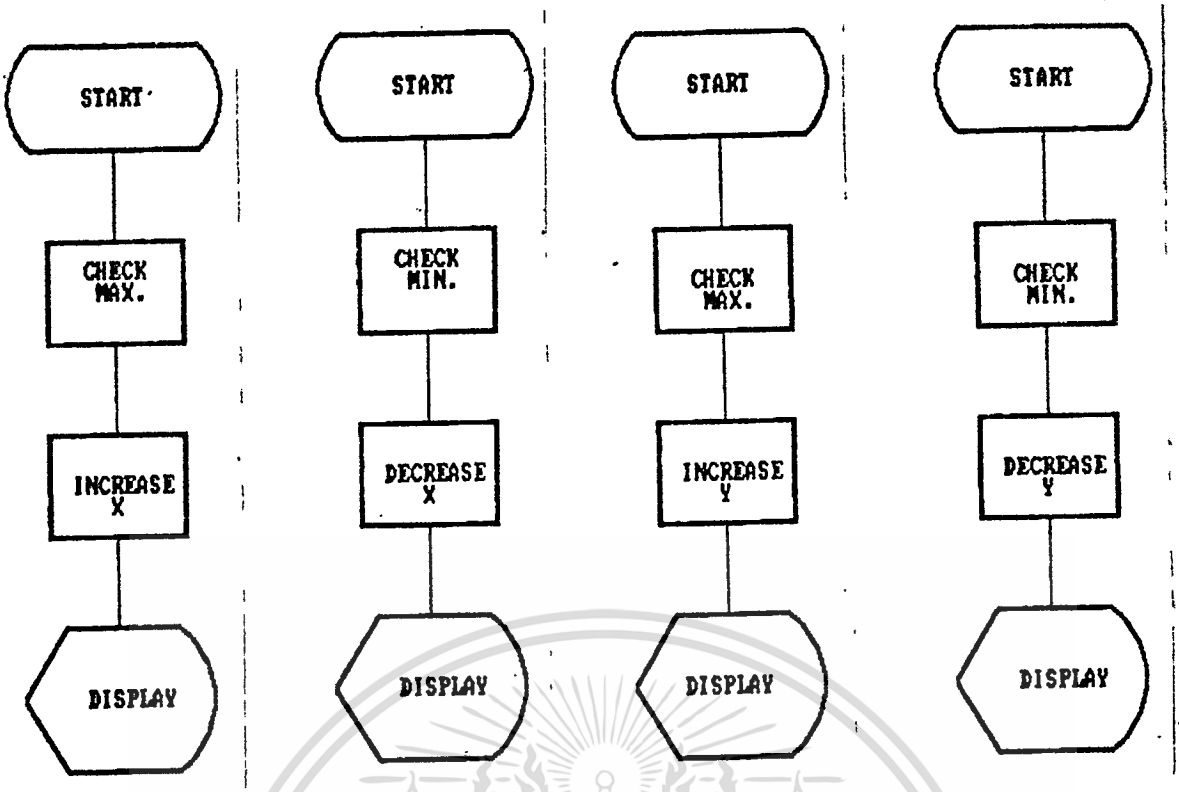
รูปที่ 4.1 แสดงโฟวส์ชาร์ต
ของโปรแกรมหลัก



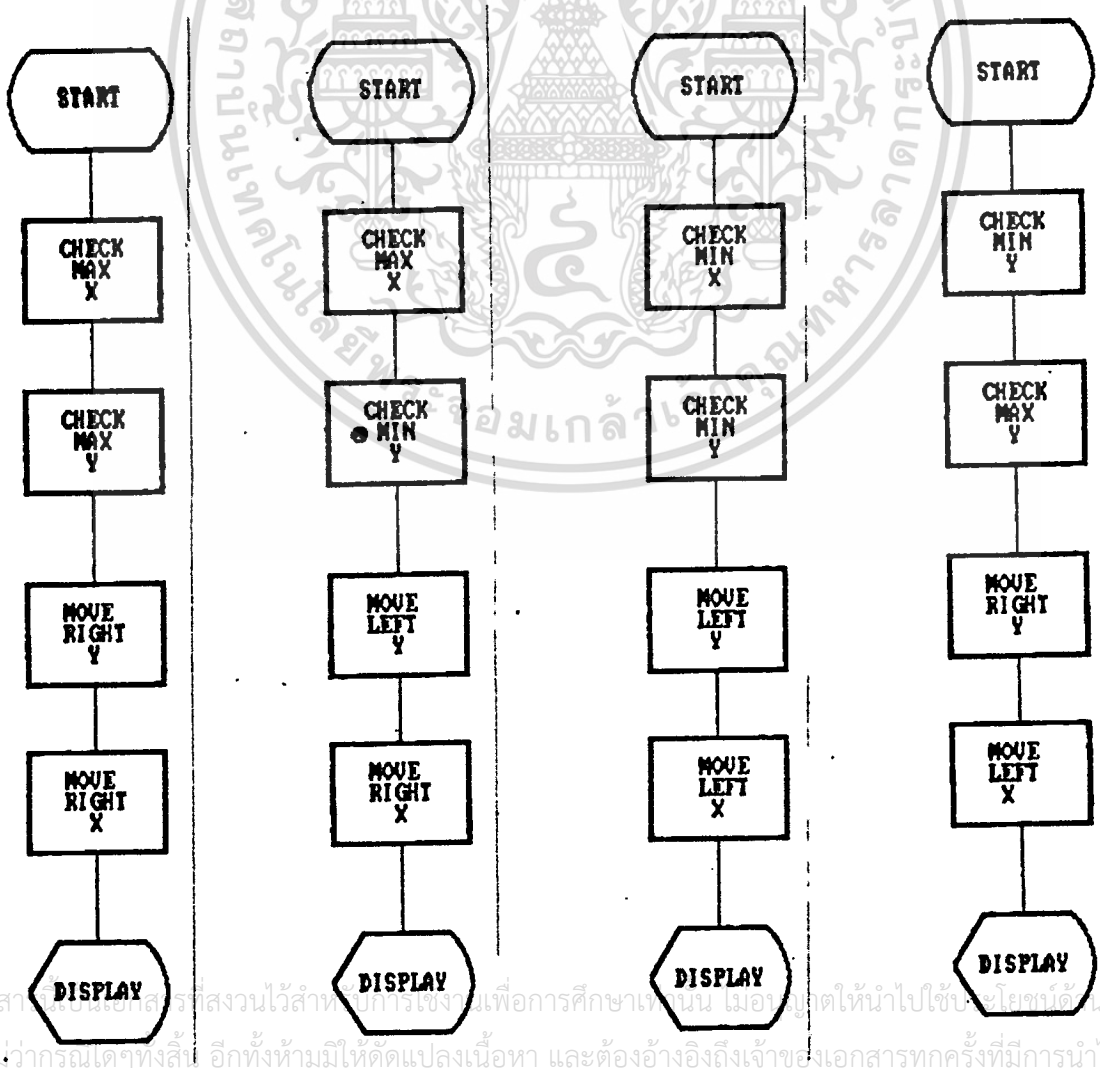
รูปที่ 4.2 แสดงโฟวล์ชาร์ต ของรูทีนเคลื่อนที่กลับ

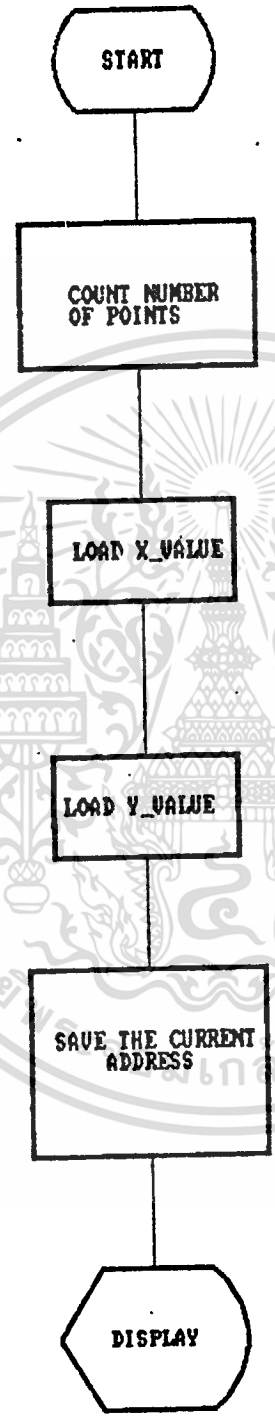


รูปที่ 4.3 แสดงโฟวล์ชาร์ตของรูทีนแปลงฐานเลข

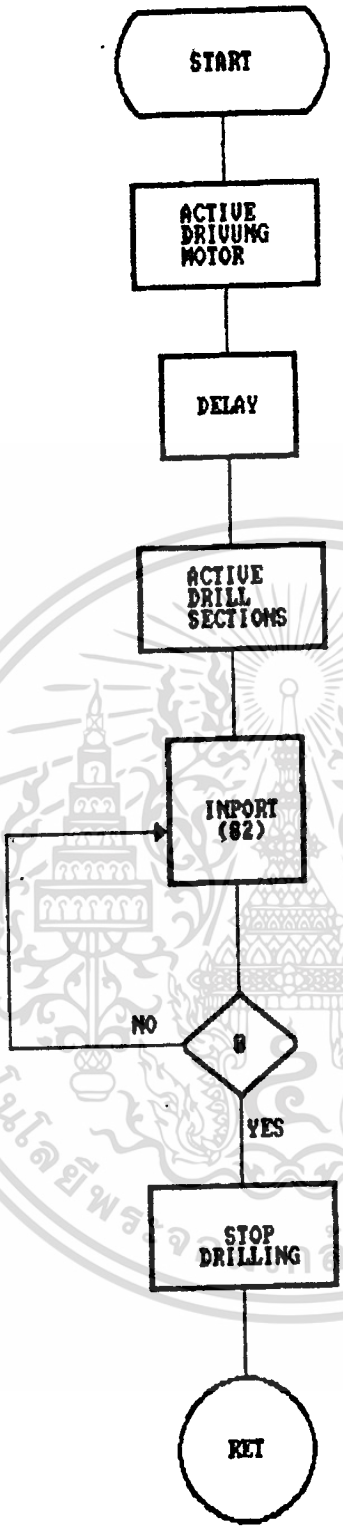


รูปที่ 4.4 แสดงโฟวล์ชาร์ตของโปรแกรมส่วนควบคุมการเคลื่อนที่ 8 ทิศทาง

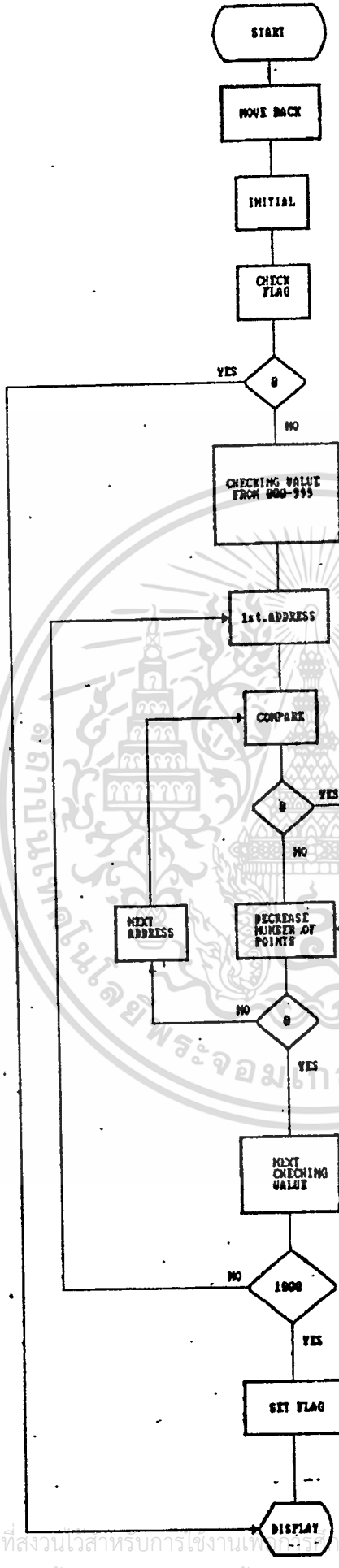




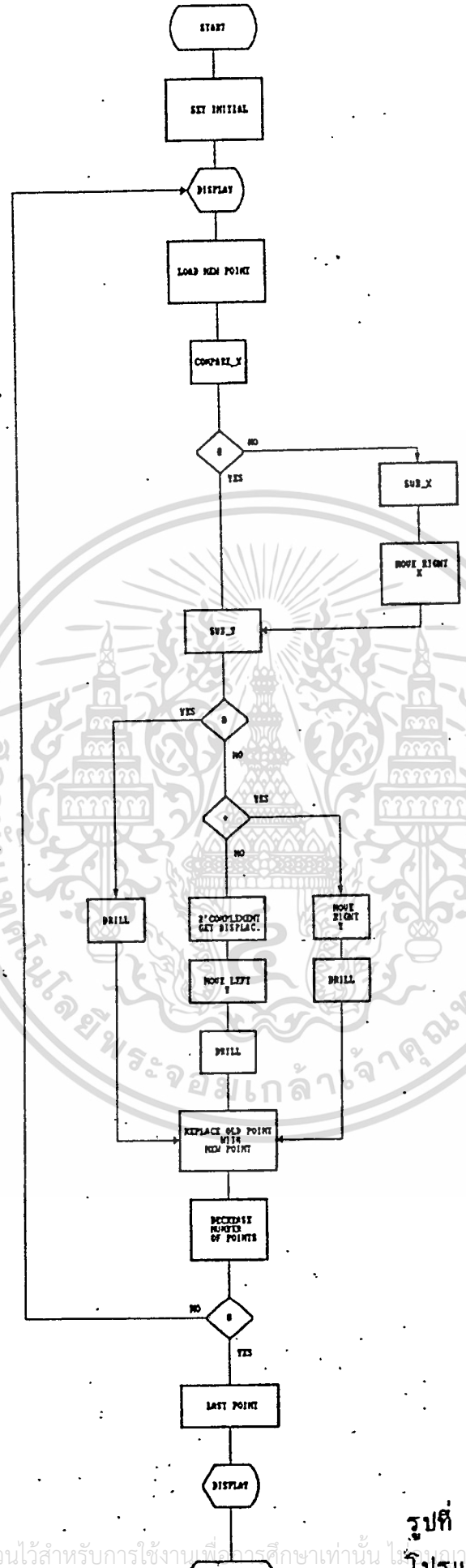
รูปที่ 4.5 แสดงโฟวล์ชาร์ตของโปรแกรมมาร์คจุด



รูปที่ 4.6 แสดงโฟวล์ชาร์ตของโปรแกรมเจาะ



รูปที่ 4.7 แสดงโฟวส์ชาร์ต
ของโปรแกรมส่วนจัดเรียงลำดับ



รูปที่ 4.8 แสดงโฟวล์ชาร์ตของโปรแกรมทำงานแบบอัตโนมัติ

บทที่ 5
บทสรุปและวิจารณ์

การสร้างต้นแบบเครื่องเจาะ PCB อัตโนมัติชุดนี้ ได้ทำการพัฒนาและทดลองเป็นลำดับและขั้นตอนดังนี้คือ

- การออกแบบ และการสร้างวงจรขับสเต็ปปิ้งมอเตอร์
- การออกแบบ และการสร้างวงจรแรมแพค
- การเขียนโปรแกรมในส่วนการแสดงผล และการสแกนคีย์
- การเขียนโปรแกรมควบคุมการเคลื่อนที่ 4 ทิศทาง
- การพัฒนาโปรแกรมควบคุมการเคลื่อนที่ให้เป็น 8 ทิศทาง
- การเขียนโปรแกรมเก็บโคออดิเนท และเจาะ
- การเขียนโปรแกรมส่วนเรียงลำดับ
- การเขียนโปรแกรมทำงานแบบอัตโนมัติ

ซึ่งการทำงานของเครื่องเจาะ PCB นี้สามารถทำงานได้ตามความต้องการ คือ แสดงตำแหน่งโคออดิเนทได้ด้วยหน่วยแสดง ,ควบคุมการเคลื่อนไปในทิศทางต่างๆได้ ,จดจำโคออดิเนทได้ และสามารถที่จะทำงานได้แบบอัตโนมัติ กล่าวคือเคลื่อนที่ไปยังตำแหน่งที่จดจำไว้ และทำการเจาะเองจนหมดทุกจุด ซึ่งการทำงานทั้งหมดนี้เป็นไปตามซอฟแวร์ที่เขียนขึ้น

ระหว่างที่ทำการสร้างเครื่องต้นแบบนี้ ได้ประสบปัญหาหลายประการ คือ อุปกรณ์ทางแมคคานิคส์ที่สร้างขึ้นไม่สอดคล้องกัน อันเนื่องมาจากปัจจัยต่างๆ เช่น มอเตอร์แต่ละตัวจะใช้ไฟเลี้ยงต่างกันมาก ทำให้เกิดความยุ่งยากในการทดลองทุกครั้งเพราะต้องหาแหล่งจ่ายไฟหลายขนาด ปัญหาที่สำคัญอีกประการของแมคคานิคส์คือเรื่องของน้ำหนัก ซึ่งส่วนหัวเจาะมีน้ำหนักมากทำให้การเคลื่อนที่ไม่คล่องตัวนัก ดังนั้นการพัฒนาในส่วนของแมคคานิคส์อาจจะเป็นไปในอีกรูปแบบก็ได้ คือการทำให้แผ่นวงจรเคลื่อนที่ แทนที่จะให้ส่วนของสว่านเคลื่อนที่

นอกจากนี้การใช้ชิงเกิ้ลบอร์ด (Single Board) สำหรับเขียนโปรแกรมควบคุมไม่ค่อยจะสะดวกนักเพราะจะต้องป้อนโปรแกรมเป็นออพโค้ด (Opcode) ซึ่งจะเกิดความยุ่งยากในการตรวจสอบโปรแกรม ซึ่งถ้าเป็นการพัฒนาบนไมโครคอมพิวเตอร์จะสะดวกกว่ามาก อีกประการหนึ่งก็คือหน่วยความจำของชิงเกิ้ลบอร์ดมีอยู่จำกัดแต่สำหรับ

เครื่องเจาะ PCB ที่ทำงานแบบอัตโนมัติต้องใช้หน่วยความจำ จำนวนมากในการเก็บโคออดิเนทต่างๆ แต่ทั้งนี้และทั้งนั้นควรพิจารณาถึงความสะดวกในการติดตั้ง ต้นทุนในการผลิต และการนำไปใช้งานเป็นสำคัญ

อนึ่งผู้จัดทำมีความหวังว่า หลักการในการควบคุมการเคลื่อนที่หรือการทำงานของแมคคานิคส์ด้วยไมโครโปรเซสเซอร์นี้ จะเป็นประโยชน์ในการพัฒนาไปในรูปแบบอื่นๆต่อไป

ภาคผนวก

```
;CONTROL PROGRAM
;
;THIS IS THE FIRST PROGRAM FOR INITIALING THE SYSTEM
;START AT ADDRESS 1000
;ADDRESS 1340 STORES MOVING STATUS OF X
;ADDRESS 1341 STORES MOVING STATUS OF Y
;AND DELAY ROUTINE IS AT ADDRESS 1350
;
DELAY     :     LD D,FF
LOOP_D    :     LD E,FF
LOOP_A    :     DEC E
                JP NZ,LOOP_A
                DEC D
                JP NZ,LOOP_D
                RET
;
INITIAL   :     LD A,81          ;LOAD CONTROL WORD
                OUT (83),A       ;SEND TO CONTROL PORT
                LD A,99          ;SET FIRST VALUE TO EXIST X,Y
                LD (1340),A
                LD (1340),A
                LD HL,0000
                LD (1360),HL     ;SET COUNTER
                LD HL,A000
                LD (1362),HL     ;1st STORE ADDRESS
                LD HL,C000
                LD (1364),HL     ;2nd STORE ADDRESS
                LD HL,0000
                LD (1366),HL     ;CLEAR FLAG
                CALL BACK
                JP START
;
;
                LD DE,130A       ;DISPLAY WORD "START"
                LD B,6F
```

```
                LD C,06
                LD A,0C
                RST 10
;
;           ********************
;           *  MAIN PROGRAM    *
;           ********************
START       :   LD B,50
CLEAR       :   LD HL,1300
                LD (HL),00      ;CLEAR ALL DATA
                INC HL
                DJNZ,CLEAR
DATA        :   LD HL,0000      ;LOAD INITIAL VALUE 0000
                LD (1300),HL    ;STORE X
                LD (1302),HL    ;STORE Y
                CALL NEXT_X
                CALL NEXT_Y
DISPLAY     :   LD DE,1304      ;FIRST ADDRESS FOR DISPLAY
                LD B,1F         ;SET DELAY TIME
                LD C,06         ;NUMBER OF DIGIT
                LD A,0C         ;LOAD DISPLAY CODE 0C
                RST 10
SCAN_KEY    :   XOR A
                LD (1FFB),A     ;CLEAR ADDRESS 1FFB
                LD A,05         ;LOAD SCAN KEY CODE 05
                RST 10
                LD A,(1FFB)     ;LOAD KEY CODE TO A
                CP 0C
                JP Z,INCR_X     ;0C FOR INCREMENT X
                CP 08
                JP Z,DECR_X     ;08 FOR DECREMENT X
                CP 0F
                JP Z,INCR_Y     ;0F FOR INCREMENT Y
                CP 0B
                JP Z,DECR_Y     ;0B FOR DECREMENT Y
```

```
                CP 0D
                JP Z,INX_INY    ;0D FOR INCR X,INCR Y
                CP 09
                JP Z,INX_DEY    ;09 FOR INCR X,DECR Y
                CP 0E
                JP Z,DEX_INY    ;0E FOR DECR X,INCR Y
                CP 0A
                JP Z,DEX_DEY    ;0A FOR DECR X,DECR Y
                CP 04
                JP Z,AUTO       ;04 FOR AUTOMATIC FUNCTION
                CP 05
                JP Z,MARK       ;05 FOR GET POSOTION POINT
                CP 06
                JP Z,DRILL      ;06 FOR DRILL
                CP 07
                JP Z,ORDER      ;07 FOR SET ORDER OF POINTS
                JP DISPLAY
;
;
;INCREASE AND DECREASE PROGRAM FOR X AND Y
;START AT ADDRESS 1110
;
INCR_X      :   LD HL,(1300)
                CALL CHKMX
                CALL INC_X
                JP DISPLAY
DECR_X      :   LD HL,(1300)
                CALL CHKMN
                CALL DEC_X
                JP DISPLAY
INCR_Y      :   LD HL,(1302)
                CALL CHKMX
                CALL INC_Y
                JP DISPLAY
DECR_Y      :   LD HL,(1302)
```

```
                CALL CHKMN
                CALL DEC_Y
                JP DISPLAY
INX_INY    :    LD HL,(1300)
                CALL CHKMX
                LD HL,(1302)
                CALL CHKMX
                CALL INC_Y
                LD HL,(1300)
                CALL INC_X
                JP DISPLAY
INX_DEY    :    LD HL,(1300)
                CALL CHKMX
                LD HL,(1302)
                CALL CHKMN
                CALL DEC_Y
                LD HL,(1300)
                CALL INC_X
                JP DISPLAY
DEX_INY    :    LD HL,(1300)
                CALL CHKMN
                LD HL,(1302)
                CALL CHKMX
                CALL INC_Y
                LD HL,(1300)
                CALL DEC_X
                JP DISPLAY
DEX_DEY    :    LD HL,(1300)
                CALL CHKMN
                LD HL,(1302)
                CALL CHKMN
                CALL DEC_Y
                LD HL,(1300)
                CALL DEC_X
                JP DISPLAY
```

```
;          ********************************
;          *  NEXT_X AND NEXT_Y ROUTINE   *
;          ********************************
;
;START AT ADDRESS 11AO AND NEXT_Y AT 11BO
;
NEXT_X     :     LD HL,1300       ;LOAD X-VALUE IN HEX.
                 CALL CONVERT_X   ;CHANGE TO BE DISPLAY CODE
POINT_X    :     LD A,(1306)      ;ADD 80 FOR DISPLAY (.)
                 ADD 80
                 LD (1306),A
                 RET
NEXT_Y     :     LD HL,(1302)     ;LOAD Y-VALUE IN HEX.
                 CALL CONVERT_Y   ;CHANGE TO BE DISPLAY CODE-Y
POINT_Y    :     LD A,(1309)
                 ADD 80
                 LD (1309),A
                 RET
;
;          ******************************************
;          *  THE RIGHT AND LEFT ROTATING ROUTINE   *
;          ******************************************

;START AT ADDRESS 11CO FOR RIGHT
;START AT ADDRESS 11DO FOR LEFT
;OX AND OY ARE SCALE OF EACH AXIS
;
RIGHT      :     LD B,OX          ;LOAD EXIST PER SCALE
LOOP_R     :     RLC A            ;ROTATE FOR MOVE RIGHT
                 OUT (C),A        ;OUT PORT
                 CALL DELAY
                 DJNZ,LOOP_R
                 RET
LEFT       :     LD B,OY
LOOP_L     :     RRC A            ;ROTATE FOR MOVE LEFT
```

```
                OUT (C),A
                CALL DELAY
                DJNZ,LOOP_L
                RET
;
;          **********************
;          *  CONVERT ROUTINE   *
;          **********************
;THESE IS CONVERT_X AND CONVERT_Y ROUTINE WHICH IS COMPOSED
;OF 2 SECTIONS
;1...SUB-ROUTINE FOR CHANGE HEX. TO DECIMAL
;2...SUB-ROUTINE FOR CHANGE DECIMAL TO DISPLAY CODE
; THE START ADDRESS IS 1200
;
;
CONVERT_X :     LD E,(HL)
                INC HL
                LD D,(HL)
                LD (1FF2),DE    ;LOAD X(HEX) TO INPUT BUFFER
                LD A,0A         ;LOAD HEX TO DECIMAL CODE
                RST 10
                LD A,(1FF5)     ;LOAD DECIMAL CODE
                LD (1304),A
                LD A,(1FF4)
                LD (1305),A
                LD HL,1305
                LD A,00
                RLD             ;SEPERATE DECIMAL IN 1305
                LD C,(HL)
                LD (HL),A
                LD B,04         ;FOUR TIMES FOR SHIFT BIT
SHIFT     :     SRL C
                DJNZ,SHIFT
                INC HL
                LD (HL),C
```

```
                LD HL,1304
                CALL DIGIT      ;CALL DIGIT ROUTINE FOR CHANGE TO
                RET             ;DISPLAY NUMBER CODE
;
;
CONVERT_Y :     LD E,(HL)
                INC HL
                LD D,(HL)
                LD (1FF2),DE
                LD A,0A
                RST 10
                LD A,(1FF5)
                LD (1307),A
                LD A,(1FF4)
                LD (1308),A
                LD HL,1308
                LD A,00
                RLD
                LD C,(HL)
                LD (HL),A
                LD B,04
SHIFT     :     SRL C
                DJNZ,SHIFT
                INC HL
                LD (HL),C
                LD HL,1307
                CALL DIGIT
                RET
;
;
;SUB-ROUTINE DIGIT
;NUMBER DISPLAY CODE AND THESE STORE AT ADDRESS 1330-1339
;1330 = 3F
;1331 = 06
;1332 = 5B
```

```
;1333 = 4F
;1334 = 66
;1335 = 6D
;1336 = 7D
;1337 = 07
;1338 = 7F
;1339 = 6F
;
DIGIT     :     LD C,03          ;3 DIGITS CHANGE TO NUMBER CODE
MORE      :     LD DE,1330       ;CODE FOR DISPLAY DIGIT '0'
                LD B,00          ;START AT 00
AGAIN     :     LD A,(HL)
                XOR B            ;COMPARE
                JRZ EQUAL        ;IF IT'S THIS CODE,JUMP TO EQUAL
                INC DE
                INC B
                LD A,B
                XOR 0A           ;CHECK THAT IS LESS THAN 0A
                JR NZ,AGAIN
EQUAL     :     LD A,(DE)
                LD (HL),A
                INC HL
                DEC C
                JRNZ MORE
                RET
;
;          ************************
;          *  MOVE BACK ROUTINE   *
;          ************************
;
MOVE_1    :     LD B,05
LOOP_1    :     LD A,(1340)
                RLC A            ;ROTATE 99 TO 33
                OUT (80),A       ;SEND TO PORT A
                OUT (81),A       ;SEND TO PORT B
```

```
                CALL DELAY
                LD (1340),A     ;STORE THE X-STATUS
                LD (1341),A     ;STORE THE Y-STATUS
                DJNZ,LOOP_1
;
SET_ZERO  :     IN A,(82)       ;INPORT C FOR CHECKING LIMIT SW.
                CPL
                AND 03
                JP Z,MVBACK     ;IF X AND Y NOT "1" GO TO MVBACK
                CP 01           ;CHECK BIT 0 (SWITCH 1)
                JP Z,BACK_Y
                CP 02           ;CHECK BIT 1 (SWITCH 2)
                JP Z,BACK_X
MVBACK    :     LD A,(1340)     ;ONE EXIST FOR MOVING RIGHT X
                RRC A
                OUT (80),A
                CALL DELAY
                LD (1340),A
                LD A,(1341)     ;ONE EXIST FOR MOVING RIGHT Y
                RRC A
                OUT (81),A
                CALL DELAY
                LD (1341),A
                JP SET_ZERO
BACK_Y    :     LD A,(1341)
                RRC A           ;ROTATE RIGHT
                OUT (81),A      ;OUT PORT A
                CALL DELAY
                LD (1341),A     ;STORE THE X_STATUS
                IN A,(82)
                BIT 1,A
                JP NZ,BACK_Y
                RET
BACK_X    :     LD A,(1340)
                RRC A           ;ROTATE RIGHT
```

```
                OUT (80),A      ;OUT PORT B
                CALL DELAY
                LD (1340),A     ;STORE THE Y_STATUS
                IN A,(82)
                BIT 0,A
                JP NZ,BACK_X
                RET
;
;           ********************************
;           *  CHECK MAX AND MIN ROUTINE   *
;           ********************************
;START AT ADDRESS
;
CHKMX       :   LD A,H
                CP 03           ;COMPARE HIGH BYTE WITH 03
                JR NZ,GO_1
                LD A,L
                CP E7           ;COMPARE LOW BYTE WITH E7
                JP Z,DISPLAY    ;IF IT IS 03E7,JUMP TO DISPLAY
                RET
CHKMN       :   LD A,L
                CP 00           ;COMPARE HIGH BYTE WITH 00
                JR NZ,GO_2
                LD A,L
                CP 00           ;COMPARE LOW BYTE WITH 00
                JP Z,DISPLAY    ;IF IT IS 0000,JUMP TO DISPLAY
                RET
;
;
;               ************************************************
;               *  INCREMENT X,Y AND DECREMENT X,Y ROUTINE     *
;               ************************************************
;
;INCREASING AND DECREASING VALUE
;AND OUTPORT FOR MOVING
```

```
;
INC_X   :   INC HL
            LD (1300),HL
            LD A,(1340)     ;LOAD X-STATUS
            LD C,80         ;LOAD X-PORT
            CALL RIGHT
            LD (1340),A     ;RESTORE X_STATUS
            CALL NEXT_X
            RET
DEC_X   :   DEC HL
            LD (1300),HL
            LD A,(1304)
            LD C,80
            CALL LEFT
            LD (1340),A
            CALL NEXT_X
            RET
INC_Y   :   INC HL
            LD (1302),HL
            LD A,(1341)     ;LOAD Y-STATUS
            LD C,81         ;LOAD Y-PORT
            CALL RIGHT
            LD (1341),A     ;RESTORE Y-STATUS
            CALL NEXT_Y
            RET
DEC_Y   :   DEC HL
            LD (1302),HL
            LD A,(1341)
            LD C,81
            CALL LEFT
            LD (1341),A
            CALL NEXT_Y
            RET
;
;
```

```
;          ******************
;          * MARK ROUTINE   *
;          ******************
;AT ADDRESS 1500
;
;
DELAY     :     LD D,06
   A      :     LD E,FF
   B      :     DEC E
               JP NZ,B
               DEC D
               JP NZ,A
MARK      :     LD DE,(1360)    ;NUMBER OF POINT
               INC DE          ;COUNTING
               LD (1360),DE    ;RESTORE
               LD DE,(1300)    ;LOAD THE FIRST VALUE OF X
               LD HL,(1362)    ;FIRST ADDRESS
               LD (HL),E       ;MOVE LOW BYTE OF X
               INC HL
               LD (HL),D       ;MOVE HIGH BYTE OF X
               INC HL
               LD DE,(1302)    ;LOAD Y
               LD (HL),E
               INC HL
               LD (HL),D
               INC HL
               LD (1362),HL    ;KEEP THE CURRENT ADDRESS
               JP DISPLAY
;
;          ********************
;          *  DRILL ROUTINE   *
;          ********************
;
;AT ADDRESS 1540
;
```

```
                CALL DRILL
                JP DISPLAY
;
;DRILL IS AT 154A
;
;
DRILL     :     LD A,10          ;10 FOR DRIVE DRILL HEAD
                OUT (82),A       ;OUT TO PORT C
          :     LD B,FF
DELAY_Q         DEC B
                JP NZ,DELAY_Q
CONT      :     LD A,30          ;FOR DRIVE DRILL SET
                OUT (82),A
TIL       :     IN A,(82)        ;FOR CHECK LIMIT SWITCH
                BIT 2,A
                JP NZ,CONT
                RET
;
;
;         **********************************
;         *  ALL POINT ORDERING ROUTINE    *
;         **********************************
; AT ADDRESS 1570
;
                CALL BACK
                LD HL,0000       ;SET INITIAL
                LD (1300),HL
                LD (1302),HL
                CALL NEXT_X
                CALL NEXT_Y
                LD DE,1304
                LD B,FF
                LD C,06
                LD A,0C
                RST 10
```

```
FLAG-CHK  :   LD A,(1366)
              CP 55
              JP Z,DISPLAY
ORDER     :   LD DE,0000      ;1st CHECKING VALUE
   AA     :   LD HL,A000      ;1st STORE ADDRESS
              LD (1362),HL
              LD BC,(1360)    ;NUMBER OF POINTS
   BB     :   LD HL,(1362)    ;FOR CHECK BY COMPARING
              LD A,E
              CP (HL)         ;COMPARE THE FIRST X-VALUE
              JP NZ,NEXT_1
              INC HL
              LD A,D          ;COMPARE THE SECOND X-VALUE
              CP (HL)
              JP  Z,STORE     ;IF OK.JUMP FOR STORE AT NEW PLACE
NEXT      :   INC HL
              INC HL
              INC HL
              LD (1362),HL    ;KEEP CURRENT ADDRESS
NUMPOI    :   DEC BC          ;CHECK POINT
              LD A,B
              OR C
              JP NZ,BB
              INC DE          ;IF POINT NUMBER ISN'T ZERO ,INC.
              LD A,D          ;CHECKING-VALUE UNTILL IT'S 1000
              CP 03
              JP NZ,AA
              LD A,E
              CP E8           ;IF CHECKING-VALUE ISN'T 1000 ,
              JP NZ,AA        ;JUMP TO DO THIS FUNCTION AGAIN
              LD A,55
              LD (1366),A     ;SET FLAG 55 WHEN COMPLETE
              JP DISPLAY
STORE     :   PUSH BC
              PUSH DE
```

```
                LD HL,(1362)    ;LOAD THE OLD PLACE ADDRESS
                LD DE,(1364)    ;LOAD THE NEW PLACE ADDRESS
                LD B,04         ;SET 4 TIMES
LOOP_STR    :   LD A,(HL)
                LD (DE),A
                INC HL
                INC DE
                DJNZ,LOOP_STR
                POP DE
                POP BC
                JP NUMPOI
;
;
NEXT_1      :   INC HL
                JP NEXT
;
;
;           ******************************
;           *  AUTOMATIC MOVING SECTION  *
;           ******************************
;
; AT ADDRESS 1600
; DISPLACEMENT OF  X IS ADDRESS 1370
; DISPLACEMENT OF +Y IS ADDRESS 1372
; DISPLACEMENT OF -Y IS ADDRESS 1374
;
                CALL BACK
BEGIN       :   LD HL,C000
                LD (1364),HL    ;SET 1st ADDRESS FOR MOVING
                LD BC,(1360)    ;LOAD NUMBER OF POINTS
                LD HL,0000
                LD (1300),HL
                LD (1302),HL    ;FOR 1st IS 000,000
SOLO        :   PUSH BC
                CALL NEXT_X
```

```
            CALL NEXT_Y
            LD DE,1304      ;DISPLAY PART
            LD B,6F
            LD C,06
            LD A,0C
            RST 10
            LD HL,(1364)    ;LOAD NEW POINT
            LD A,(HL)
            LD (130A),A
            INC HL
            LD A,(HL)
            LD (130B),A
            INC HL
            LD A,(HL)
            LD (130C),A
            INC HL
            LD A,(HL)
            LD (130D),A
            INC HL
            LD (1304),HL    ;STORE CURRENT ADDRESS
            LD A,(130A)     ;COMPARE X-VALUE IF X ISN'T EQUAL
            LD D,A          ;THE OLD,JUMP TO SUB_X
            LD A,(1300)
            CP D
            JP NZ,SUB_X
            LD A,(130B)
            LD D,A
            LD A,(1301)
            CP D
            JP NZ,SUB_X
SUB_Y       :   ADD A,00
            LD HL,(130C)
            LD DE,(1302)
            SBC HL,DE
            JP Z,RIGHT_X    ;SUBTRACT Y AND GET 0:MOVE ONLY X
```

```
                JP P,RIGHT_Y    ;POSITIVE:MOVE Y IN RIGHT DIM.
LEFT_Y      :   LD A,L
                CPL
                LD (1374),A
                LD A,H
                CPL
                LD (1375),A
                LD A,(1374)     ;2'COMPLEMENT OF -Y
                ADD A,01
                LD (1374),A
                LD A,(1375)
                ADC A,01
                LD (1375),A
                LD DE,(1374)    ;DISPLACEMENT OF LEFT-Y
AGAIN_1     :   LD A,(1341)     ;STATUS OF Y
                LD C,81         ;PORT Y CODE
                CALL LEFT
                LD (1341),A
                DEC DE
                LD A,E
                OR D
                JP NZ,AGAIN_1
                CALL DRILL
                CALL DELAY
STU         :   LD A,(130A)     ;MOVE VALUE
                LD (1300),A     ;LOW BYTE X
                LD A,(130B)
                LD (1301),A     ;HIGH BYTE X
                LD A,(130C)
                LD (1302),A     ;LOW BYTE Y
                LD A,(130D)
                LD (1303),A     ;HIGH BYTE Y
                POP BC
                DEC BC
                LD A,B
```

```
                OR C
                JP NZ,SOLO
                JP LAST
;
; ADDRESS 16D0
;
RIGHT_Y   :     LD A,L
                LD (1372),A
                LD A,H
                LD (1373),A
                LD DE,1372      ;DISPLACEMENT FOR MOVE Y IN RIGHT
AGAIN_2   :     LD A,(1341)     ;STATUS OF Y
                LD C,81
                CALL RIGHT
                LD (1341),A
                DEC DE
                LD A,E
                OR D
                JP NZ,AGAIN_2
                CALL DRILL
                CALL DELAY
                JP STU
;
; ADDRESS 1700
;
SUB_X     :     ADD A,00
                LD HL,(130A)
                LD DE,(1300)
                SBC HL,DE       ;SUBTRACT X
                LD (1370),HL    ;STORE DISPLACEMENT AT 1370
                LD DE,(1370)
AGAIN_3   :     LD A,(1340)
                LD C,80
                CALL RIGHT
                LD (1340),A
```

```
                DEC DE
                LD A,E
                OR D
                JP NZ,AGAIN_3
                JP SUB_Y
;
; ADDRESS 1760
;
RIGHT_X   :     CALL DRILL
                CALL DELAY
                JP STU
;
;
; ADDRESS 1730
;
LAST      :     LD A,(130A)     ;MOVE LAST VALUE
                LD (1300),A     ;LOW BYTE X
                LD A,(130B)
                LD (1301),A     ;HIGH BYTE X
                LD A,(130C)
                LD (1302),A     ;LOW BYTE Y
                LD A,(130D)
                LD (1303),A     ;HIGH BYTE Y
                CALL NEXT_X
                CALL NEXT_Y
                LD DE,1304      ;DISPLAY PART
                LD B,6F
                LD C,06
                LD A,0C
                RST 10
                JP DISPLAY
```

กิตติกรรมประกาศ

กิตติกรรมประกาศ

โครงนี้สำเร็จลงได้ ทั้งนี้ก็ด้วยความช่วยเหลือจากบุคคลหลายฝ่าย จึงขอขอบคุณ

อ.ขนิษฐา แซ่ตั้ง
อ.ธำรงศักดิ์ สุกใส
อ.เกียรติวรรณ ทรงสัตย์
คุณเนติพงษ์ ศรีหิรัญ
คุณเทศพล สอตระกูล
คุณธีระพงษ์ เจริญคุณวิวัฒ

ซึ่งได้ให้คำแนะนำ และความช่วยเหลือเรื่องอุปกรณ์ต่างๆ เป็นผลให้โครงงานและปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วง

หนังสืออ้างอิง

เอกสารอ้างอิงภาษาอังกฤษ

1. Takashi Kenjo, "Stepping Motors and their Microprocessor Controls", Clarendon Press, Oxford ,1984 ,pp 1-55
2. Rodnay Zak, "Programming the Z80 " sybex, USA ,1979
3. James W. Coffron ," Z80 Application " Sybex, USA ,1983
4. Robert F.Coughlin, "Principle and Applications of Semiconductors and Circuits", Prentice-Hall inc.,New Jersey 1971,pp 50-64
5. Z80 ET-Board Version 2.0 User's Manual