เครื่องกัดลายแผ่นวงจรพิมพ์ควบคุมด้วยคอมพิวเตอร์

โดย

นาย ทรงวิทย์ เจริญรวยวัฒนา รหัส 33504011

นาย อภินันท์ เย็มกลัดนาค รหัส 33504047

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ภาควิชา ฟิสิกส์ประยุกต์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

มีนาคม 2537

Computerized Printed Circuit Board Etching Machine

Mr.songwit Charoenruaywattana

Mr.Apinan Khemgladnark

A Special Project Submitted in Partial Fulfillment of the

Requirement for the Degree of bachelor of science

Department of Applied Physics

Faculty of science

King Mongkut's Institute of Technology Ladkrabang

March 1994

หัวข้อโครงงานพิเศษ เครื่องกัดลายแผ่นวงจรพิมพ์ควบคุมด้วยคอมพิวเตอร์

โดย นายทรงวิทย์ เจริญรวยวัฒนา

นายอภินันท์ เข็มกลัดนาค

ภาควิชา ฟิสิกส์ประยุกต์

อาจารย์ที่ปรึกษา ผศ.ดร.วราวุฒิ เถาลัดดา

อ.อนุชิต จารุวนาวัฒน์

ปีการศึกษา 2536

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร-ลาดกระบัง ได้อนุมัติให้นับรายงานโครงงานฉบับพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

_____ หัวหน้าภาควิชาฟิสิกส์ประยุกต์

(ผศ. ปรีชา เทียนสมประสงค์)

คณะกรรมการสอบโครงงานพิเศษ

_____ ประธานกรรมการ

(ผศ.ดร.วราวุฒิ เถาลัดดา)

_____ กรรมการ

(อ.อนุชิต จารุวนาวัฒน์)

_____ กรรมการ

(ผศ.ดร.เสน่ห์ เอกะวิภาต)

_____ กรรมการ

(ผศ.สุวรรณ คูสำราญ)

ลิขสิทธิ์ของภาควิชาฟิสิกส์ประยุกต์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

หัวข้อโครงงานพิเศษ	เครื่องกัดลายแผ่นวงจรพิมพ์ควบคุมด้วยคอมพิวเตอร์
นักศึกษา	นายทรงวิทย์ เจริญรวยวัฒนา
	นายอภินันท์ เย็มกลัดนาค
อาจารย์ที่ปรึกษา	ผศ.ดร.วราวุฒิ เถาลัดดา
	อาจารย์ อนุชิต จารุวนาวัฒน์
ภาควิชา	ฟิสิกส์ประยุกต์
ปีการศึกษา	2536

บทคัดย่อ

ในการพัฒนาวงจรอิเล็กทรอนิกส์ หลังจากออกแบบและทดลองทางอิเล็กทรอนิกส์แล้ว จะต้องทำการออกแบบลายวงจร แล้วจึงผ่านขบวนการกัดลายวงจรบนแผ่นวงจรพิมพ์ซึ่งใช้วิธีทางเคมี โดยการนำลายพิมพ์ของวงจรไปแช่ในสารเคมีเพื่อกัดทองแดงส่วนที่ไม่ต้องการออกซึ่งเป็นขบวนการที่ต้องทำโดยบุคลากรที่มีความชำนาญ นอกจากนี้วิธีการทางเคมียังเป็นวิธีที่สิ้นเปลืองสารเคมี และสารเคมีที่ใช้ยังทำลายสิ่งแวดล้อมอีกด้วย

โครงงานนี้จึงได้จัดทำขึ้นมาเพื่อใช้สร้างแผ่นวงจรพิมพ์โดยใช้วิธีการแกะสลัก การทำงานจะถูกควบคุมโดยไมโครคอมพิวเตอร์ ผู้ใช้เพียงแต่ป้อนวงจรอิเล็กทรอนิกส์ที่ออกแบบไว้เข้าเครื่องคอมพิวเตอร์ จากนั้นไมโครคอมพิวเตอร์จะควบคุมเครื่องกัดลายวงจรลงบนแผ่นวงจรพิมพ์โดยตรง วิธีการนี้จะช่วยให้ผลิตแผ่นวงจรได้ง่ายและประหยัดค่าใช้จ่ายกว่าวิธีการทางเคมี

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี เนื่องจากได้รับความช่วยเหลือจากหลาย ๆ ฝ่าย คณะผู้จัดทำขอขอบพระคุณ ดร.วราวุฒิ เถาลัดดา และ อาจารย์อนุชิต จารุวนาวัฒน์ ซึ่งเป็นที่ปรึกษาโครงงาน อาจารย์วิโรจน์ เตชะวิญญูธรรม (สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ) ได้ให้คำปรึกษาและจัดทำชุดขับเคลื่อน คุณสุชาติ ตุลยาภรณ์โชติ(บริษัท เคเคซี) ที่ให้คำปรึกษาทางด้านการเลือกใช้บอลสกรู (ball screw) และ linear system และสุดท้ายขอขอบคุณเพื่อนๆทุกท่านที่ให้กำลังใจ

คณะผู้จัดทำโครงงาน

นาย ทรงวิทย์ เจริญรวยวัฒนา

นาย อภินันท์ เย็มกลัดนาค

มีนาคม 2537

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงานพิเศษ

ในการพัฒนาวงจรอิเล็กทรอนิกส์ หลังจากการออกแบบและทดลองเรียบร้อยแล้ว การสร้างเครื่องต้นแบบจะต้องทำการออกแบบลายวงจร แล้วจึงผ่านขบวนการกัดลายวงจรบนแผ่นวงจรพิมพ์โดยวิธีการทางเคมี โดยขั้นแรกจะต้องพิมพ์ลายวงจรลงบนแผ่นวงจรพิมพ์ จากนั้นจะต้องนำไปแช่ในสารเคมีเพื่อกัดลายวงจรตามที่ต้องการ วิธีการนี้จะต้องทำโดยบุคคลากรที่มีความชำนาญ นอกจากนี้วิธีการทางเคมียังเป็นวิธีที่สิ้นเปลืองทั้งสารเคมี และเวลาอีกทั้งสารเคมีเหล่านั้นยังทำลายสิ่งแวดล้อมอีกด้วย จะเห็นได้ว่าวิธีการนี้ไม่เหมาะสมในการสร้างเครื่องต้นแบบ

โครงงานวิจัยนี้เป็นการออกแบบและจัดสร้างเครื่องกัดลายวงจรลงบนแผ่นวงจรพิมพ์โดยใช้วิธีทางกลแทนวิธีทางเคมีได้แก่วิธีการแกะสลัก โดยการทำงานจะถูกควบคุมโดยไมโครคอมพิวเตอร์ ผู้ใช้เพียงแต่ป้อนวงจรอิเล็กทรอนิกส์ที่ออกแบบไว้เข้าเครื่องคอมพิวเตอร์จากนั้นไมโครคอมพิวเตอร์จะวาดลายวงจร และส่งข้อมูลไปยังเครื่องกัดลายวงจรให้กัดลายวงจรนั้นลงบนแผ่นวงจรพิมพ์โดยตรง วิธีการนี้จะช่วยให้ผลิตแผ่นวงจรพิมพ์สำหรับเครื่องต้นแบบได้ง่าย และรวดเร็วกว่าวิธีการทางเคมี ดังนั้นในโครงงานนี้จึงได้จัดทำขึ้นมาเพื่อทำการวิจัย และเพื่อที่จะทำการพัฒนาต่อไป

โดยเรียกโครงงานนี้ว่า "เครื่องกัดลายแผ่นวงจรพิมพ์ควบคุมด้วยคอมพิวเตอร์" (Computerized Printed Circuit Board Etching Machine)

1.2 วัตถุประสงค์ของโครงงาน

เพื่อออกแบบสร้างและทดสอบเครื่องกัดแผ่นวงจรพิมพ์ควบคุมด้วยคอมพิวเตอร์รวมทั้งการพัฒนาโปรแกรมคอมพิวเตอร์เพื่อควบคุมการทำงานของเครื่องที่สร้างขึ้น

1.3 ขอบเขตของโครงงานพิเศษ

1. ทำการออกแบบและสร้างชุดขับเคลื่อน 3 แกน (XYZ-TABLE)
2. ออกแบบและสร้างชุดควบคุม(HARDWARE)พร้อมโปรแกรมคอมพิวเตอร์ (SOFTWARE)
3. ทดสอบการทำงาน
4. หาข้อบกพร่องและแนวทางที่จะพัฒนาต่อไป

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาและรวบรวมข้อมูลเกี่ยวกับการออกแบบชุดขับเคลื่อน 3 แกน (XYZ-TABLE)
2. จัดหามอเตอร์ที่เหมาะสมสำหรับชุดขับเคลื่อน 3 แกน
3. จัดหาวัสดุอุปกรณ์ที่ต้องการใช้ในการสร้างชุดขับเคลื่อน 3 แกน
4. จัดทำชุดควบคุมและโปรแกรมคอมพิวเตอร์ที่ใช้ในการควบคุม
5. ศึกษาการทำงานของโปรแกรมโปรเทล(PROTEL)
6. จัดทำโปรแกรมที่ใช้ในการส่งผ่านข้อมูลจากโปรแกรมโปรเทล (PROTEL) ไปยังชุดขับเคลื่อน 3 แกน
7. จัดหาวัสดุที่จะนำมาใช้ในการกัดแผ่นปริ๊นท์
8. ทดสอบการทำงานของแต่ละส่วนของระบบ
9. วิเคราะห์และสรุปผล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. เป็นพื้นฐานในการพัฒนาเทคโนโลยีทางด้านการควบคุมระบบเครื่องจักรกลแบบอัตโนมัติด้วยระบบคอมพิวเตอร์ต่อไปในอนาคต
2. สามารถสร้างชุดขับเคลื่อนที่ในระนาบ XYZ ด้วยรูปแบบต่าง ๆ โดยการควบคุมด้วยคอมพิวเตอร์ ซึ่งชุดขับเคลื่อนนี้สามารถนำไปประยุกต์ใช้ในงานหลายๆอย่าง อาทิเช่น LASER SCANNING,LASER MARKING,เครื่องตัดโลหะ ฯลฯ

3. เป็นพื้นฐานในการสร้างบุคคลกรทางด้านนี้ให้แพร่หลาย เนื่องจากในปัจจุบัน บุคคลากรทางด้านนี้มีค่อนข้างน้อย

4. ในเชิงพาณิชย์สามารถที่จะผลิตออกมาขายได้ ปัจจุบันมีขายในต่างประเทศแล้ว แต่มีราคาค่อนข้างสูง

บทที่ 2

การออกแบบระบบขับเคลื่อน 3 แกน(XYZ-TABLE)

2.1 การเลือกระบบขับเคลื่อน

ระบบขับเคลื่อนที่ใช้กันอยู่ในปัจจุบัน แบ่งออกเป็น 2 ประเภทใหญ่ๆคือ

1.ระบบขับเคลื่อนด้วยสายพาน เช่นใช้สายพานหรือลวดสลิงเป็นตัวขับเคลื่อนชิ้นงาน ซึ่งมีข้อดีคือมีราคาถูก และสร้างได้ง่ายแต่มีข้อเสียคือการบ่งชี้ตำแหน่งผิดพลาดได้ง่าย

2.ระบบขับเคลื่อนแบบพันเกลียว เช่น เกลียวหนอนแบบลูกปืน (BALL SCREW) ซึ่งมีข้อดี คือ มีความเที่ยงตรงของตำแหน่งสูงมาก แต่มีข้อเสีย คือ ราคาแพงเนื่องจากวัตถุประสงค์ของโครงงานนี้เพื่อสร้างระบบขับเคลื่อน 3 แกน สำหรับใช้วางแผ่นปริ๊นท์ที่ใช้ในการกัดเซาะผิวทองแดงบนแผ่นปริ๊นท์ ดังนั้นความเที่ยงตรงแม่นยำของตำแหน่งจึงเป็นสิ่งสำคัญอันดับแรก จึงทำให้การทำโครงงานพิเศษนี้เลือกใช้ระบบขับเคลื่อนแบบบอลสกรู (BALL SCREW)

หลังจากเลือกระบบขับเคลื่อนแล้ว ในขั้นตอนต่อไปเป็นการออกแบบระบบขับเคลื่อน ซึ่งไดละแกรมของระบบขับเคลื่อนจะต้องกำหนดตัวแปรที่ต้องการขึ้นมาก่อน ซึ่งในที่นี้ได้แก่

1.ระยะทางที่ขับเคลื่อนได้ (TRAVEL LENGTH)

2.ความละเอียดในการขับเคลื่อน

การกำหนดระยะทางที่เคลื่อนที่ทางแกน X และแกน Y ซึ่งในที่นี้ได้กำหนดไว้ให้ทางแกน X มีระยะทางเคลื่อนที่ได้ 200 ม.ม. และทางแกน Y มีระยะทางที่เคลื่อนที่ได้ 200 ม.ม. (สำหรับระยะทางที่เคลื่อนที่ก็คือระยะความยาวของบอลสกรู ต้องคำนึงถึงความยาวของบอลสกรูที่มีขายหรือหาซื้อได้ตามท้องตลาด)

2.2 การเลือกชนิดของมอเตอร์

หลังจากที่เลือกระบบขับเคลื่อนได้แล้ว จะต้องทำการเลือกชนิดของมอเตอร์ที่ต้องการใช้ในการขับบอลสกรูที่ได้เลือกใช้ในระบบขับเคลื่อนดังที่กล่าวแล้วในการเลือกชนิดของมอเตอร์นั้นจะต้องคำนึงถึงคุณสมบัติของมอเตอร์แต่ละชนิด ตามความชำนาญของผู้ใช้และความยากง่ายในการหาซื้อ ซึ่งจะพิจารณาแต่ละชนิดดังต่อไปนี้

(1) มอเตอร์กระแสตรง (DC MOTOR)

มอเตอร์ชนิดนี้เป็นที่ยอมรับว่าความต่อเนื่องของการเคลื่อนที่นิ่มนวลดีพอสมควร แต่ต้องคำนึงถึงปัญหาในการควบคุม ได้แก่

1.ไม่สามารถทราบถึงตำแหน่งของมุมที่หมุนไปของแกนมอเตอร์ ดังนั้นการที่จะต่อใช้งานร่วมกับระบบคอมพิวเตอร์จึงเป็นสิ่งที่ยุ่งยาก

2.ปัญหาเรื่องแรงเฉื่อยที่เกิดขึ้นกับมอเตอร์กระแสตรง เพราะมอเตอร์กระแสตรงไม่สามารถเบรคได้เองโดยลำพัง ต้องอาศัยการเบรคจากระบบภายนอกซึ่งเป็นเรื่องที่ยุ่งยากในการควบคุม

จากปัญหาทั้ง 2 ข้อจะเห็นว่าการเลือกใช้มอเตอร์แบบนี้ แล้วระบบควบคุมจะต้องมีขนาดใหญ่ ซับซ้อนและยากต่อการควบคุม ถึงแม้ว่ามอเตอร์แบบนี้จะหาซื้อได้ง่ายก็ตาม แต่โดยเหตุผลรวมแล้วจะไม่เลือกใช้มอเตอร์ชนิดนี้ในโครงงานนี้

(2) เซอร์โวมอเตอร์กระแสตรง (DC servo motor)

มอเตอร์ชนิดนี้ใช้งานได้ดีกับระบบขับเคลื่อน [1,2] มอเตอร์แบบนี้จะหาซื้อได้ยากภายในประเทศ เท่าที่หาซื้อได้ในท้องตลาดจะเป็นมอเตอร์เก่าซึ่งไม่ทราบสมบัติเฉพาะตัวของมัน จึงเป็นการยากในการจัดทำชุดควบคุมมอเตอร์ ดังนั้นในโครงงานพิเศษนี้จึงไม่เลือกใช้มอเตอร์ชนิดนี้

(3) สเต็ปปิ้งมอเตอร์ (STEPPING MOTOR)

มอเตอร์ชนิดนี้เป็นมอเตอร์ที่มีการหมุนในลักษณะเป็นสเต็ป ดังนั้นจะเห็นได้ว่ามีการกระตุกไปทีละสเต็ป ซึ่งถ้ามองอย่างผิวเผินอาจคิดว่าไม่เหมาะสมสำหรับการใช้งาน เพราะจะทำให้การเลื่อนชิ้นงานของมีการกระตุกไปเป็นช่วงทำให้ไม่มีความต่อเนื่องในการเคลื่อนที่ ปัญหานี้สามารถแก้ไขได้โดยอาศัยเทคนิคพิเศษเสริมเพิ่มเข้าไปนอกเหนือจากการใช้งานปกติ ทำให้มีการขับเคลื่อนทีละ 1/n step เทคนิคนี้เรียกว่า "การจ่ายกระแสแบบพัลส์" และมีการใช้มอเตอร์ชนิดนี้ในระบบต่าง ๆ อย่างแพร่หลายและหาซื้อได้ในท้องตลาดค่อนข้างง่าย ดังนั้นในโครงงานนี้จึงเลือกใช้สเต็ปปิ้งมอเตอร์เป็นตัวขับบอลสกรู ซึ่งสเต็ปปิ้งมอเตอร์ที่หาซื้อได้โดยทั่วไปเป็นแบบ 200 สเต็ปต่อรอบ หรือ 1.8 องศาต่อสเต็ป ดังนั้นการหมุนของมอเตอร์ในแต่ละรอบมีเพียง 200 สเต็ปเท่านั้น ซึ่งการหมุนแต่ละสเต็ปค่อนข้างหยาบ จึงจำเป็นต้องออกแบบชุดควบคุมการหมุนของมอเตอร์ให้ละเอียดยิ่งขึ้น ต้องกำหนดความละเอียดในการเคลื่อนที่ของมอเตอร์โดยการกำหนดสเต็ปที่ต้องการในหนึ่งรอบ

ซึ่งในโครงงานนี้ต้องการควบคุมมอเตอร์ให้มอเตอร์เคลื่อนที่ได้ 400 สเต็ปต่อรอบนั่นคือมีความละเอียด 0.9 องศาต่อหนึ่งสเต็ป

2.3 การเลือกขนาดของบอลสกรู

หลังจากที่กำหนดแล้วว่าต้องการสร้างชุดควบคุมมอเตอร์ให้มอเตอร์เคลื่อนที่ได้ 400 สเต็ปต่อรอบ ขั้นตอนต่อไปจะเป็นการเลือกขนาดของบอลสกรู ในการออกแบบเพื่อใช้ขนาดของบอลสกรู และจำนวนพิช(Pitch)หรือระยะระหว่างเกลียวของบอลสกรูนั้น โดยทั่วไปบริษัทผู้ผลิตจะผลิตบอลสกรูที่มีระยะระหว่างเกลียว 2.00-20.00 ม.ม.ซึ่งเป็นระยะทางที่บอลสกรูจะขับเคลื่อนชิ้นงานได้ โครงงานนี้ได้เลือกใช้บอลสกรูที่มีระยะระหว่างเกลียว 2 ม.ม. หมายความว่าบอลสกรูหมุนไปหนึ่งรอบชิ้นงานจะเคลื่อนที่ได้ 2 ม.ม. นั่นคือถ้ามอเตอร์เคลื่อนที่ 1 รอบ หรือ 400 สเต็ปชิ้นงานจะเคลื่อนที่ได้ 2 ม.ม. แต่ถ้ามอเตอร์หมุนไปหนึ่งสเต็ปชิ้นงานจะเคลื่อนที่ 2 ม.ม./400 = 5 ไมโครเมตร ซึ่งเป็นระยะการเคลื่อนที่ที่มีความละเอียดสูงพอต่อการเคลื่อนที่ของมอเตอร์แต่ละสเต็ป ดังนั้นจึงใช้บอลสกรูที่มีระยะระหว่างเกลียว 2 ม.ม. ส่วนเส้นผ่าศูนย์กลางนั้นไม่สนใจมากนักเพราะโครงงานนี้ไม่ต้องการ critical speed มาก

2.4 การเลือกขนาดของมอเตอร์

ก่อนที่จะเลือกขนาดของมอเตอร์ที่ใช้ในการขับบอลสกรู นั้นจะต้องทำการคำนวณหาแรงบิดทั้งหมดที่ต้องใช้ในการขับเคลื่อนบอลสกรูเสียก่อนซึ่งเมื่อทำการออกแบบส่วนต่าง ๆ มาถึงขณะนี้ สามารถสรุปข้อกำหนดคุณสมบัติเฉพาะตัวของระบบขับเคลื่อนนั้นๆได้จึงสามารถที่จะคำนวณหา Torque ได้ดังนี้

Torque ทั้งหมดที่ต้องใช้ในการขับบอลสกรู

$$T_{Total} = T_{Weight} + T_{Friction} + T_{Cutting} + T_{Acceleration}$$

1.คำนวณหา $T_{Friction}(T_F)$

T_F= Torque เนื่องจากแรงเสียดทานที่เกิดขึ้น

$$[T_{F(Table)} + T_{F(Screw)}] \cdot i$$

เมื่อ i = Gear ratio =1 สำหรับ direct couple ระหว่างมอเตอร์กับบอลสกรู(นั่นคือไม่ใช้ gear box)

$T_{F(Table)} = mg\mu h/2\pi$

m = มวลของชิ้นงาน + มวลของแท่นวางชิ้นงาน + มวลของชุดยึดเคลื่อนแกน Y ที่วางอยู่บนแกน X

g = 9.81 m/s^2

μ = สัมประสิทธิ์ของแรงเสียดทานระหว่าง steel กับ steel

h = ระยะระหว่างเกลียว 2 mm

โดยการคิดค่า T_F นี้เป็นค่าที่ได้จากประสบการณ์ของวิศวะกรที่ทำงานทางด้านนี้

2.คำนวณหาค่า T_{Weight} (T_w)สำหรับแกน X

T_w =Torque เนื่องจากน้ำหนักที่บอลสกรูต้องยึดในแนวแกน X

= mg sin α hi/2

= Axis angle ของบอลสกรู

สำหรับ Horizontal axis ,sin α =0

สำหรับ Vertical axis,sin α = 1

ในที่นี้แกนของบอลสกรูอยู่ในแนวราบ ,sin α =0 ดังนั้น T_w = 0

3.คำนวณหา $T_{Cutting}$

ในกรณีของ MACHINE TOOLS เช่น MILLING MACHINE จะมีการสัมผัสระหว่างดอกสว่านกัดโลหะกับชิ้นงาน ดังนั้น Torque ที่เกิดขึ้นระหว่างการกัดโลหะแต่ในกรณีของโครงงานนี้ก็เหมือนกันจะมี Torque เกิดระหว่างหัวกัดเอ็นมิลล์ (end mill)กับทองแดงที่อยู่บนแผ่นปริ๊นท์

4. $T_{Acceleration}(T_A)$

ในกรณีที่ยังไม่มีการเคลื่อนที่ จะทำให้ T_A = 0 แต่จากข้อ 1-3 สามารถคำนวณหาขนาดของมอเตอร์ได้อย่างคร่าวๆแล้ว

แต่เนื่องจากในโครงงานนี้มีงบประมาณจำกัด ดังนั้นมอเตอร์ที่ใช้จึงใช้ของเก่าที่หาซื้อได้ตามท้องตลาดทั่วไป โดยใช้มอเตอร์ดังนี้ Stepping Motor Type 103G770-3B ของบริษัท Sanyo

บทที่ 3

การสร้างระบบขับเคลื่อน 3 แกน (XYZ-TABLE)

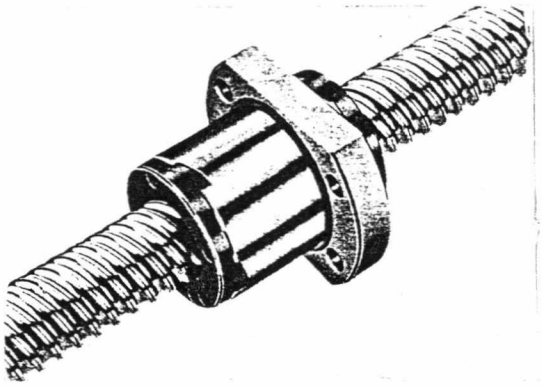
3.1 วัสดุอุปกรณ์ของระบบขับเคลื่อน

หลังจากที่ได้ทำการออกแบบและคำนวณหาขนาดของส่วนต่าง ๆ ของระบบขับเคลื่อนเป็นที่เรียบร้อยแล้ว ขั้นตอนต่อไปเป็นการหาวัสดุอุปกรณ์ของระบบขับเคลื่อนโดยจะทำการแบ่งออกเป็น 2 ส่วนคือ

3.1.1 ทางแกน XY

1.บอลสกรู (BALL SCREW)

เป็นแบบ Precision-Rolled Thread ดังแสดงในรูปที่ 3.1 วัสดุที่ใช้ทำบอลสกรูเป็น stainless stell ขนาดเส้นผ่าศูนย์กลาง 10 mmและความยาวของบอลสกรูทางแกน X และแกน Y 300 mm (รายละเอียดของบอลสกรูดูในเอกสารอ้างอิง)



รูปที่ 3.1 ลักษณะของ Ball Screw ที่นำมาใช้งาน

2.Thrust bearing

เป็นแบริ่งสำหรับใช้ยึดปลายของบอลสกรูเข้ากับแท่นยึด (Ball screw block) ในแต่ละแกนต้องใช้แบริ่งยึดปลายของบอลสกรูยึดกับแท่นแกนละ 1 ตัว ดังนั้นบอลสกรูในแต่ละแกนจะต้องใช้ thrust bearing จำนวน 2 ตัว

3.motor coupling

ทำด้วยเหล็กเป็นรูปทรงกระบอกเจาะรูให้อยู่ในแนวเดียวกัน เพื่อใช้สำหรับต่อโดยตรง (direct coupling) ระหว่างแกนของมอเตอร์กับแกนของบอลสกรู

4.motor

เป็นสเต็ปปิ้งมอเตอร์ของบริษัท SANYO รุ่น 103G770-38 มีจำนวนสเต็ปเท่ากับ 200 สเต็ปต่อรอบ หรือ 1.8 องศาต่อสเต็ป

5.Linear system

เป็นอุปกรณ์ที่ใช้สำหรับรองรับตัวบอลสกรู และใช้ช่วยในการลดแรงเสียดทางภายในระหว่างที่แท่นทำงาน

6.ตัวแท่น XY

อุปกรณ์ที่ใช้ในการทำแท่นนั้นเป็นอลูมิเนียมแล้วมาทำการหล่อขึ้นรูปตามแบบในภาคผนวก เหตุที่เลือกใช้อลูมิเนียมเพราะว่ามีน้ำหนักเบา,ไม่เป็นสนิม และมีความสวยงาม โครงสร้างของระบบขับเคลื่อนทางแกน X และ แกน Y และภาพถ่ายของแท่นที่สร้างขึ้นแสดงดังรูปที่ 3.2 - รูปที่ 3.4

3.1.2 ทางแกน Z

1.เกลียวตางหมู 4 ปาก

โลหะที่นำมาใช้ทำนั้นเป็น STAINLESS STELL มีหน้าที่ใช้เป็นเกลียวสำหรับขับเคลื่อนทางแกน Z เหตุที่ใช้เกลียวแบบนี้เพราะว่าทางด้านแกน Z ไม่มีความจำเป็นต้องใช้ความละเอียดสูง

2.thrust bearing

ใช้ลักษณะเดียวกับในทางแกน XY

3.motor

เป็นสเต็ปปิ้งมอเตอร์ของบริษัท SANYO DENKI รุ่น 103-771-16 และมีจำนวนสเต็ปเท่ากับ 200 สเต็ปต่อรอบ หรือ 1.8 องศ์ศาต่อสเต็ป

4.ตัวแทนแกน Z

ใช้วัสดุเช่นเดียวกันกับทางแกน XY

5.เพลา (shaft)

เป็นเพลา solid shaft ขนาดเส้นผ่านศูนย์กลาง 5 mm โดยเพลาแต่ละอันจะถูกสวมผ่าน linear compact set สำหรับระบบขับเคลื่อนแกน Z จะมี compact linear set จำนวน 2 ตัว เพื่อทำหน้าที่ยึดเพลาในแกน Z กับแท่นรองระบบขับเคลื่อน

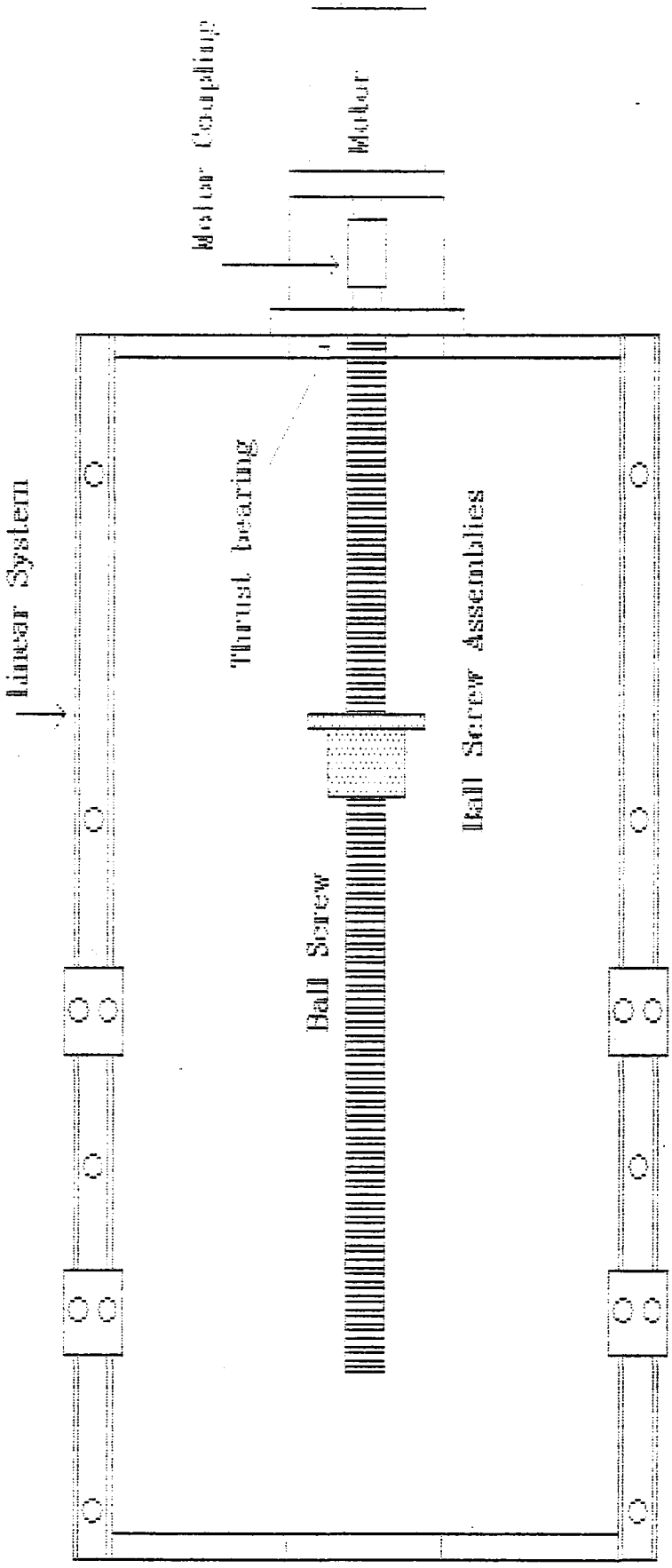
6.Linear compact set

ทำหน้าที่ช่วยทำให้การเคลื่อนที่มีความราบเรียบ และช่วยลดแรงเสียดทานระหว่างโลหะด้วยเพราะว่ามีลูกปืนอยู่ข้างใน

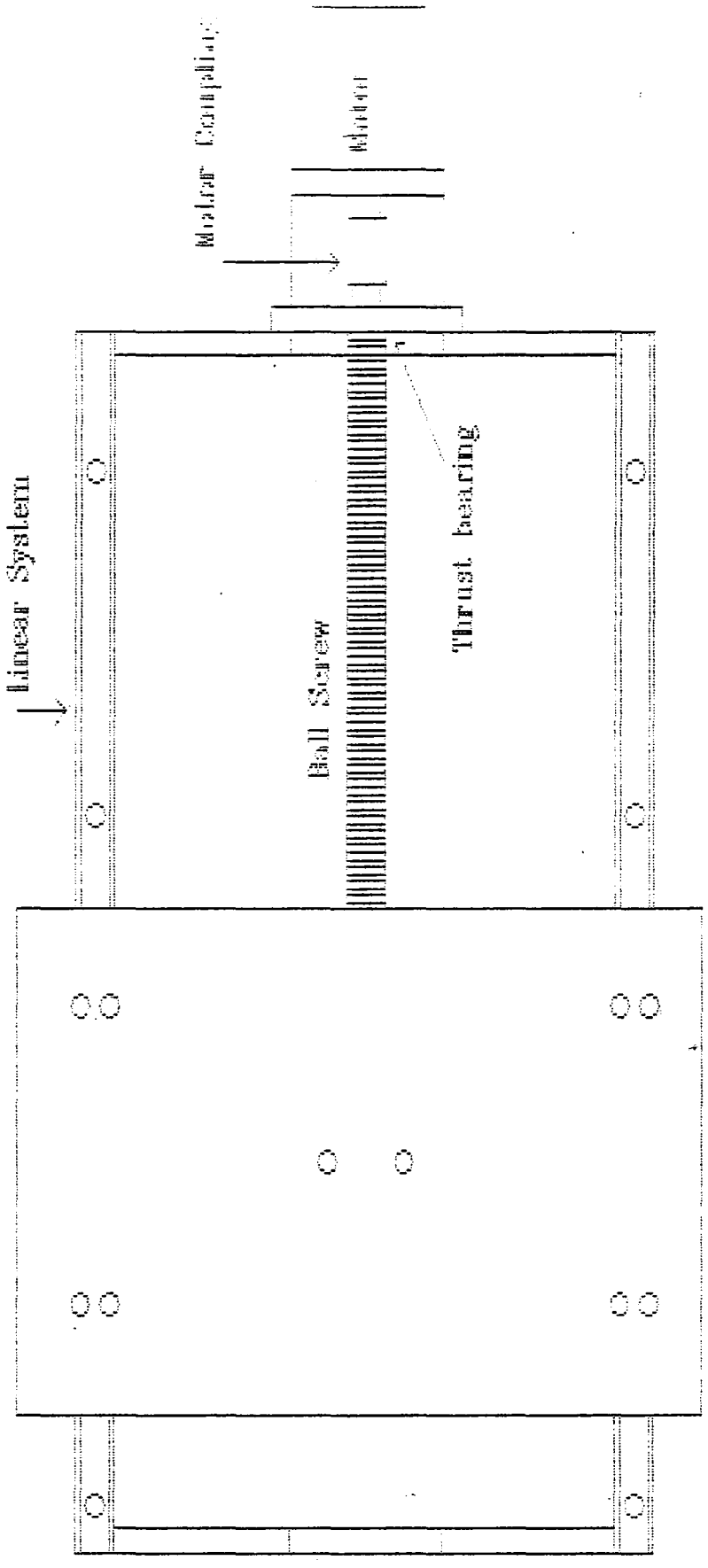
โครงสร้างและภาพถ่ายของระบบขับเคลื่อนแกน Z แสดงดังรูปที่ 3.5 และ 3.6

3.2 อุปกรณ์ที่ใช้ในการกัดปริ๊นท์

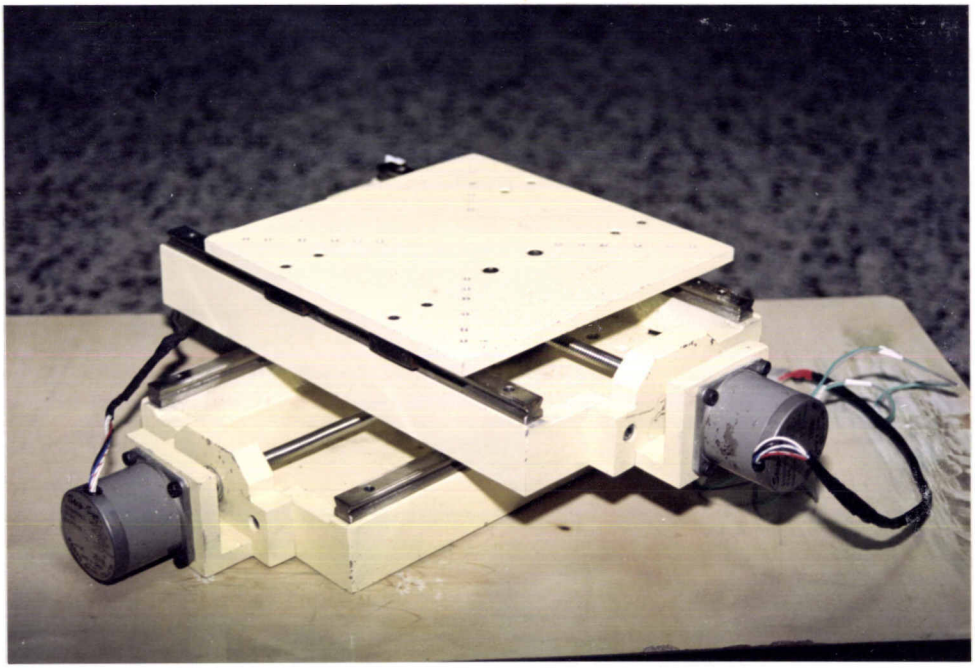
ในการกัดเซาะแผ่นปริ๊นท์นั้นจะต้องใช้หัวกัดที่มีขนาดเล็กและมีความคมมาก อีกทั้งตัวมอเตอร์ที่ใช้ในการหมุนหัวกัด ในโครงงานพิเศษนี้ใช้เป็นดีซีมอเตอร์ขนาด 24 โวลท์ มีความเร็วรอบประมาณ 3700 rpm ซึ่งความเร็วรอบของมอเตอร์นี้มีความสำคัญมากสำหรับความคมของเส้นที่ต้องการจะกัด (โดยความเร็วรอบนี้มาจากแค็ตตาล็อกของบริษัท OSG) ดังนั้นหัวกัดปริ๊นท์นี้จำเป็นต้องใช้หัวกัดที่เรียกว่า endmill ของบริษัท OSG (รายละเอียดดูในภาคผนวก)



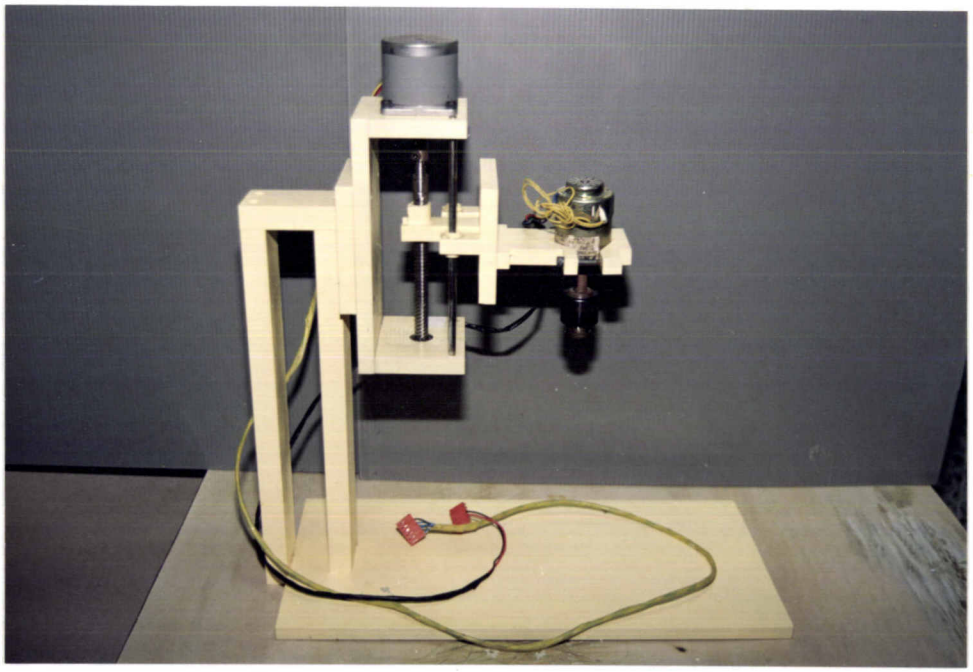
รูปที่3.2 ภาพเขียนของระบบขับเคลื่อนทางแกน X



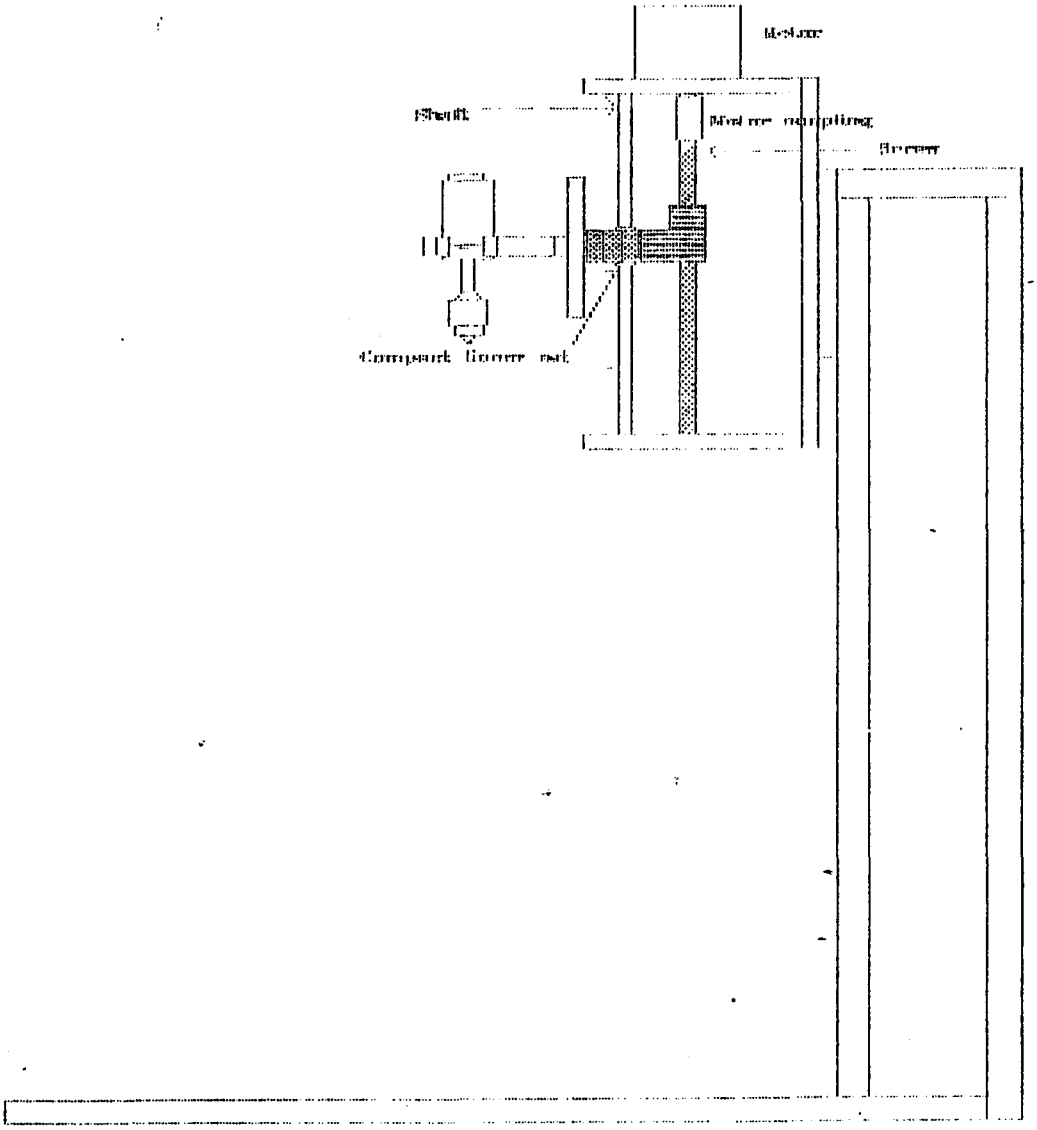
รูปที่ 3.3 ภาพเขียนของระบบขับเคลื่อนทางแกน Y



รูปที่ 3.4 ภาพถ่ายระบบขับเคลื่อน XY



รูปที่ 3.5 ภาพถ่ายระบบขับเคลื่อนทางแกน Z



รูปที่ 3.6 ภาพวาดของระบบขับเคลื่อนแกน Z

3.3 หลักการทำงานของระบบขับเคลื่อน

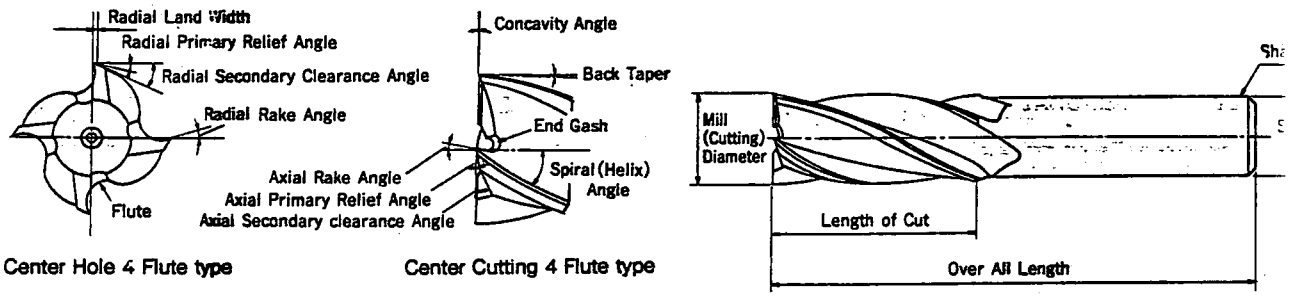
ระบบขับเคลื่อนที่จัดทำขึ้นมานั้นจัดแบ่งหน้าที่การทำงานออกเป็นสองส่วนคือ

3.3.1 ทางแกน XY

หน้าที่ของระบบขับเคลื่อนในทางแกน XY นั้นมีหน้าที่เพื่อทำการเลื่อนชิ้นงานไปในแนวแกน X และแกน Y ตามคำสั่งของโปรแกรมที่จัดทำขึ้น

3.3.2 ทางแกน Z

ทางแกน Z นี้ทำหน้าที่เพียงยึดตัวมอเตอร์ที่ใช้หมุนหัว endmill และใช้เป็นตัวเลื่อนหัว endmill ขึ้นลง ลักษณะของหัว endmill แสดงดังรูปที่ 3.7



รูปที่ 3.7 ภาพแสดงลักษณะของหัว endmill

บทที่ 4

การออกแบบและสร้างส่วนควบคุมพร้อมระบบเชื่อมต่อกับคอมพิวเตอร์

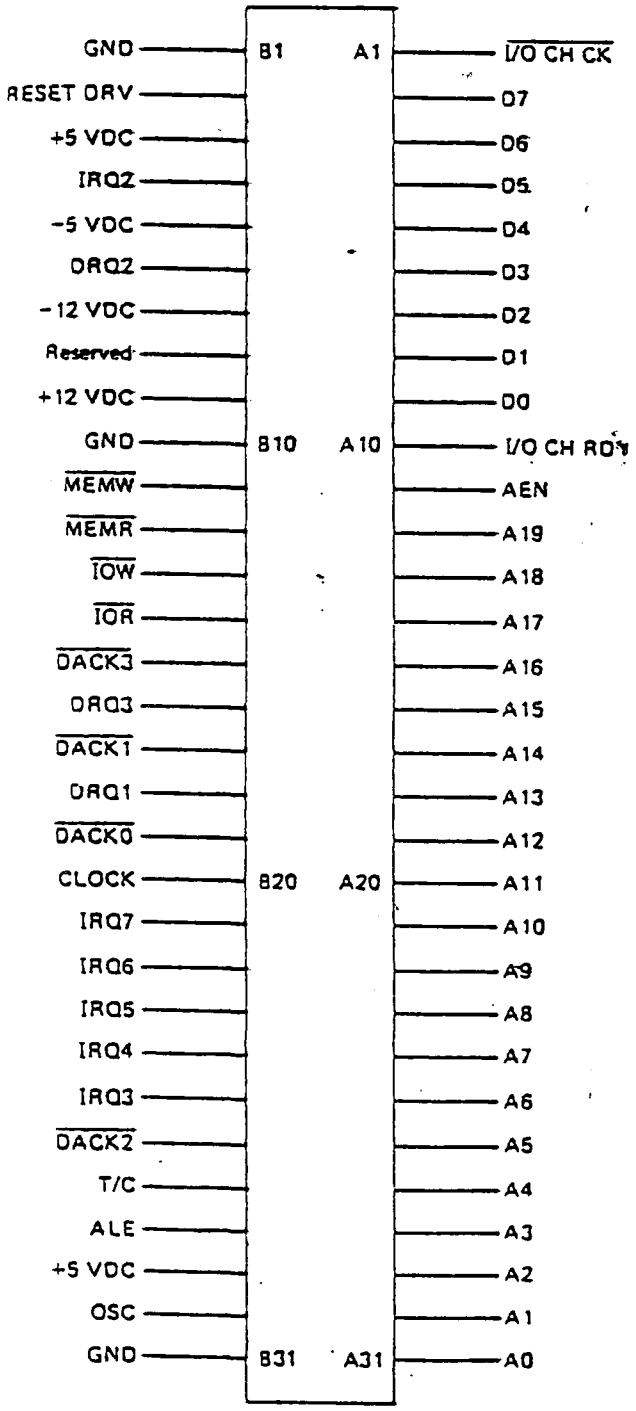
4.1 การเชื่อมต่อไมโครคอมพิวเตอร์

IBM PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเทอร์เฟส(interface circuit) เข้าไปในภายหลังได้ โดยผ่านทางสล็อตที่อยู่บนเมนบอร์ดซึ่งมีจำนวน 8 สล็อต แต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็นสองด้าน ด้านละ 31 ขา แต่ละขาของสล็อตเหล่านี้จะต่อเชื่อมขาสัญญาณต่างๆบนเมนบอร์ด ทำให้การสร้างวงจรอินเทอร์เฟสกับ IBM PC สามารถทำได้สะดวก ซึ่งสัญญาณที่เชื่อมต่อกับขาของสล็อตเหล่านี้จะประกอบไปด้วย สัญญาณของบัสแอดเดรส (Address Bus),บัสข้อมูล (Data Bus),บัสควบคุมสำหรับการอ่าน/เขียนข้อมูลจากหน่วยความจำ หรือ พอร์ท I/O ,สัญญาณสำหรับการขออินเทอร์รัพท์ของวงจรอินเทอร์เฟส ,เส้นสัญญาณสำหรับการขอ DMA ,สัญญาณฐานเวลา (Timing Signal) ต่าง ๆ ที่ใช้ในระบบ, เส้นสัญญาณแสดงการขอรีเฟรชหน่วยความจำ และสัญญาณสำหรับตรวจสอบความผิดพลาด(I/O CHECK)

นอกจากเส้นสัญญาณเหล่านี้แล้ว สล็อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่างๆที่ใช้ในระบบด้วยคือ +5Vdc,-5Vdc,+12Vdc และ -12Vdc สัญญาณทั้งหมดบนสล็อตจัดเรียงตำแหน่งตามรูปที่ 4.1

4.1.1 การแบ่งแอดเดรสของ IBM/PC

ใน IBM/PCมีการจัดสรรการจัดแอดเดรสต่าง ๆ ไว้แน่นอนซึ่งได้แสดงไว้ดังตารางที่ 4.1 ส่วนที่จะนำมาใช้ในการ decode address นั้นจะอยู่ในส่วนของ Prototype Card ซึ่งมีแอดเดรสอยู่ในช่วง 300-31F ทั้งหมด 32 พอร์ทแอดเดรส



รูปที่ 4.1 ระบบบัสของ IBM PC

ตารางที่ 4.1 การแบ่งแอดเดรสของ IBM/PC

Hex range	Usage	
000-00F	DMA chip 8237A-5	
020-021	Interrupt 8259A	
040-043	Timer 8253-5	Assigned to
060-063	PPI 8255A-5	system board
080-083	DMA page registers	components
0Ax	NMI mask register	
0Cx	Reserved	
0Ex	Reserved	
100-1FF	Not usable	
200-20F	Game control	
210-217	Expansion unit	
220-24F	Reserved	
278-27F	Reserved	
2F0-2F7	Reserved	Assigned to
2F8-2FF	Asynchonous communication(2)	feature card
300-31F	Phototype card	ports
320-32F	Fixed disk	
378-37F	Printer	
380-38C	SDLC communication	
380-389	Binary synchonous communication(2)	
3A0-3A9	Binary synchonous communication(1)	
380-38F	IBM monochrome display printer	
3C0-3CF	Reserved	

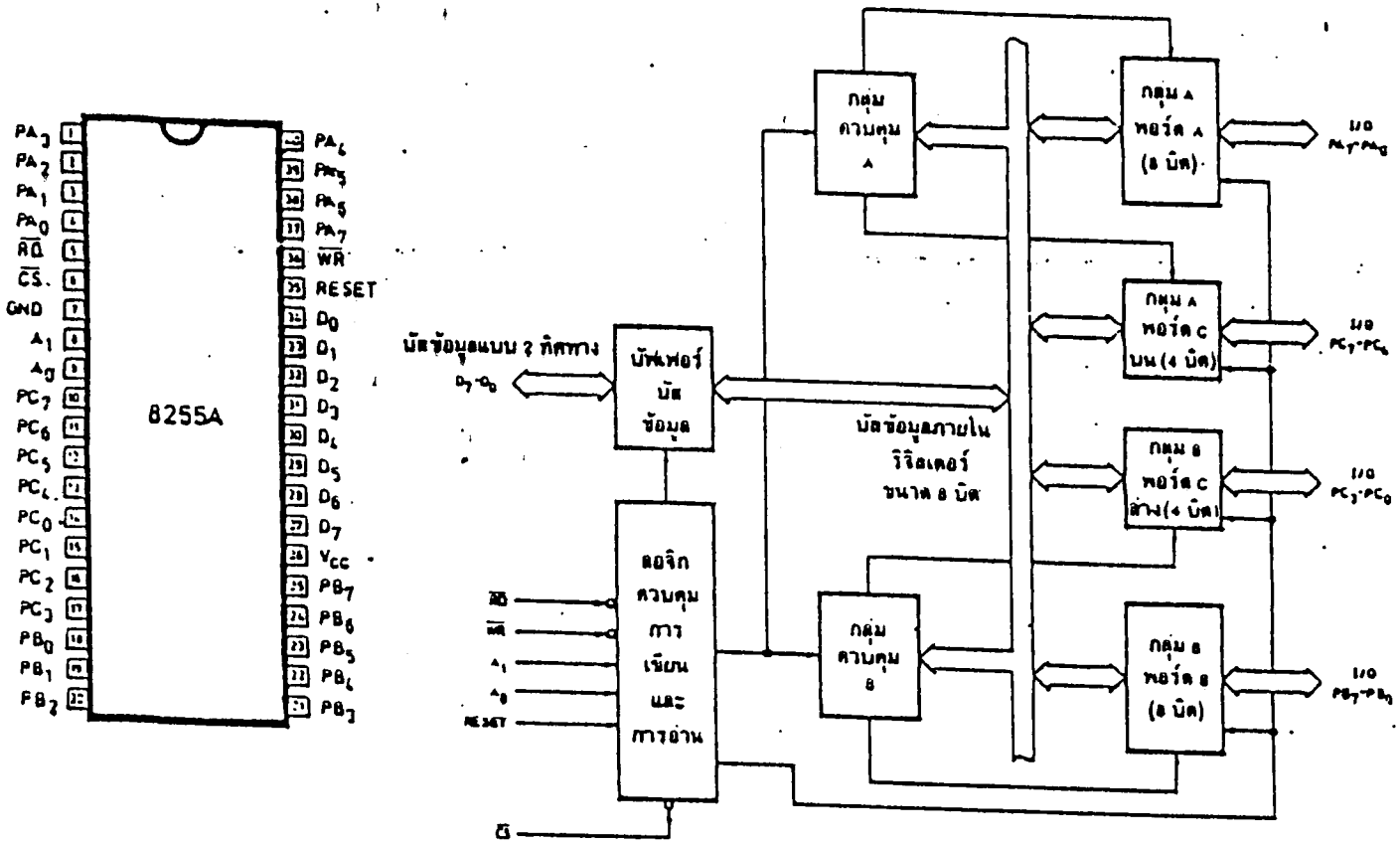
ตารางที่ 4.1 การแบ่งแอดเดรสของ IBM/PC (ต่อ)

Hex range	Usage
3D0-3DF	Colour/graphics
3E0-3E7	Reserved
3F0-3F7	Diskette
3F8-3FF	Asynchonous communications(1)

เมื่อทราบรายละเอียดภายในของ IBM/PC แล้วว่ามีสล็อตอะไรบ้าง และมีการจัดแบ่งแอดเดรสไว้อย่างไร ดังนั้นก็จะนำไปสู่การเชื่อมต่อ IBM/PC กับ 8255 ดังนี้

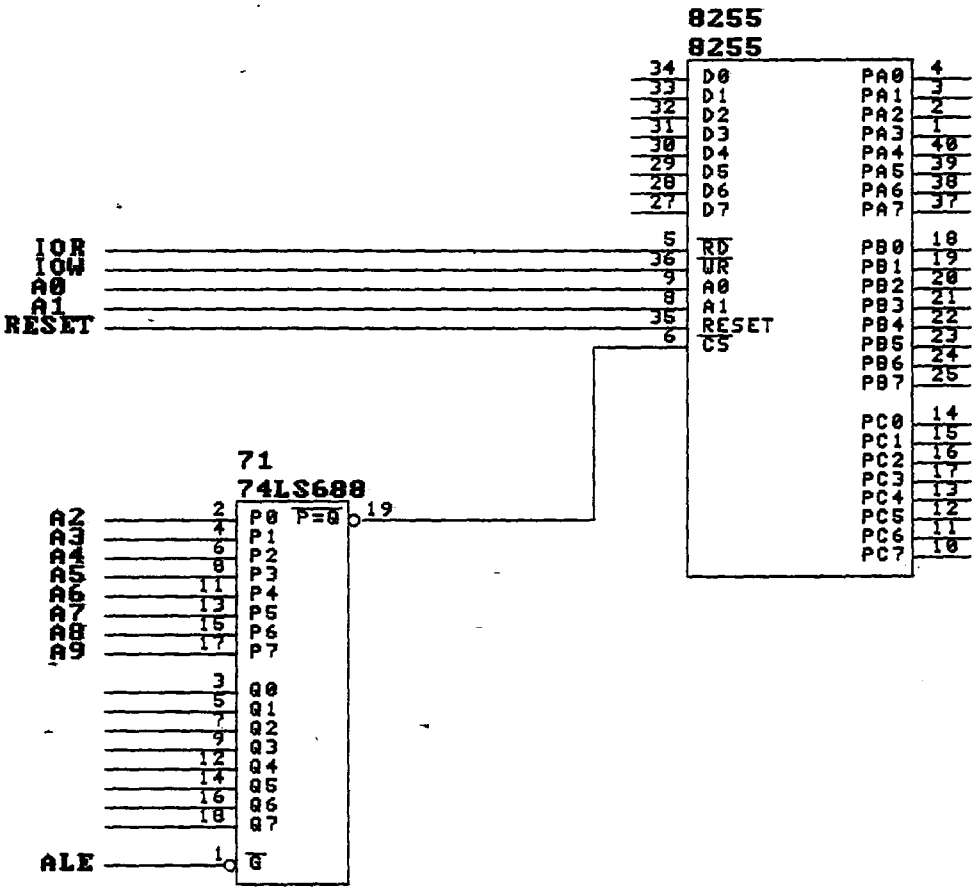
4.1.2 การ decode address

ในโครงงานนี้จะทำการ decode แอดเดรสที่ 300 นำมาใช้ในการเชื่อมต่อกับ 8255 PPI ซึ่งแอดเดรสเหล่านี้อยู่ในส่วนของ prototype card หลังจากที่ได้วงจร decode เสร็จเรียบร้อยแล้วจึงนำวงจร decode นี้มาเชื่อมต่อกับ 8255 แต่ก่อนที่จะทำการเชื่อมต่อกับ 8255 นั้นจะต้องรู้ถึงโครงสร้างของมันคร่าวๆดังนี้คือ 8255 เป็นไอซีที่มี 40 ขา ได้รับการออกแบบให้มีสัญญาณเพื่อเชื่อมต่อกับ 8088 ซึ่งตัว 8255 เองนั้นเป็นไอซีที่ต่อเป็นพอร์ทให้ไมโครโปรเซสเซอร์ได้ 3 พอร์ท คือ PORT A,PORT B และ PORT C สามารถแบ่งได้ออกเป็น 2 ส่วนคือ พอร์ท c ล่างและ พอร์ท c บน ที่พิเศษคือพอร์ททุกพอร์ทเป็นไปได้ทั้งอินพุทและเอาต์พุท แผนผังโครงสร้างของไอซี 8255 แสดงดังรูปที่ 4.2



รูปที่ 4.2 แผนผังโครงสร้างของไอซี 8255

เมื่อทราบลักษณะโครงสร้างภายในแล้วจึงนำเอาวงจร decode มาเชื่อมต่อกันเข้ากับ 8255 ดังแสดงวงจรดังรูปที่ 4.3



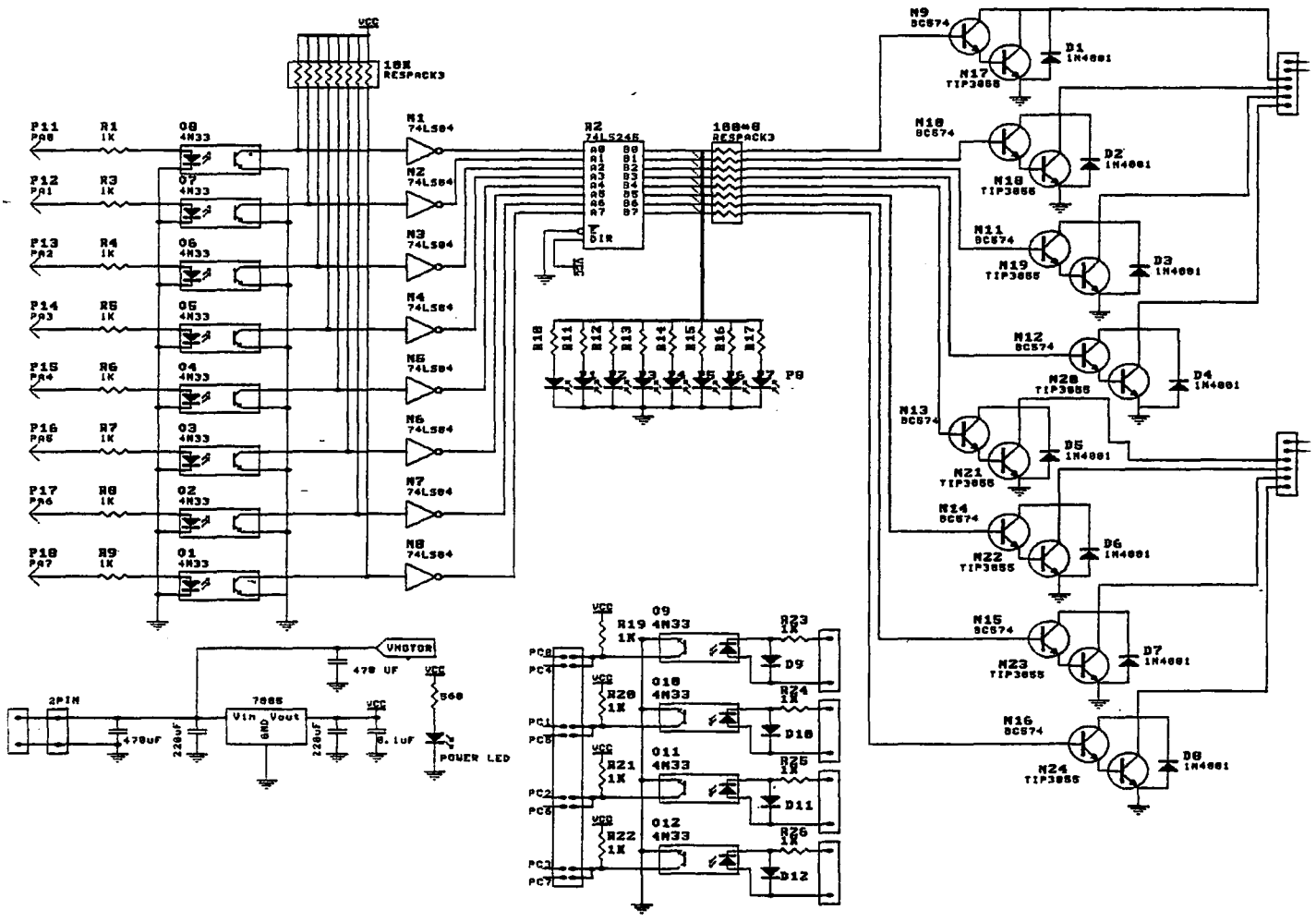
รูปที่ 4.3 การต่อ 8255 กับวงจร decode

ซึ่งวงจรดังกล่าวจะมีการใช้เรียกพอร์ทของ 8255 ดังนี้

		A11.....A0			
พอร์ท	A	300H	0011	0000	0000
พอร์ท	B	301H	0011	0000	0001
พอร์ท	C	302H	0011	0000	0010
คอนโทรพอร์ท		303H	0011	0000	0011

4.2 ชุดขับสเต็ปปิ้งมอเตอร์ (STEPPING MOTOR DRIVER)

4.2.1 ET-SMCC STEPPING MOTOR DRIVER CONTROL วงจรแสดงดังรูป 4.4

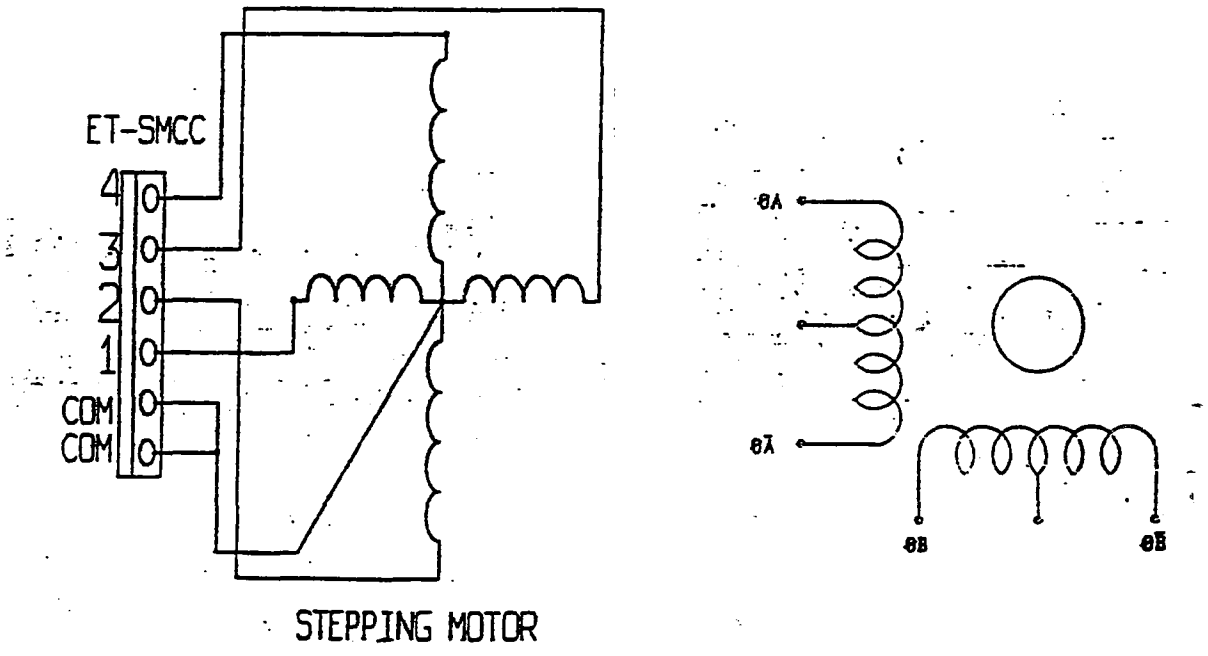


รูปที่ 4.4 แสดงวงจรของ ET-SMCC

ET-SMCC จะเป็นการ์ดต่อเข้ากับพอร์ทของ 8255 หรือพอร์ทอื่นๆด้วยก็ได้โดยใช้ในการต่อควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ โดยจะสามารถต่อกับสเต็ปปิ้งมอเตอร์ได้ 2 ตัวต่อหนึ่งพอร์ท หรือจะต่อพ่วงอีกหนึ่งการ์ด เพื่อใช้กับสเต็ปปิ้งมอเตอร์ 4 ตัวต่อพอร์ท 8255 จำนวนหนึ่งตัวก็ได้ นอกจากนี้ ET-SMCC จะมีอินพุท ในรูปลักษณะของออปโตอีก 4 อินพุท สามารถต่อกับสวิทต่างๆได้อีกด้วย เช่นต่อกับไมโครสวิทตรวจการหมุนของสเต็ปปิ้งมอเตอร์ว่าถึงจุดที่ต้องการแล้วหรือยัง เป็นต้น

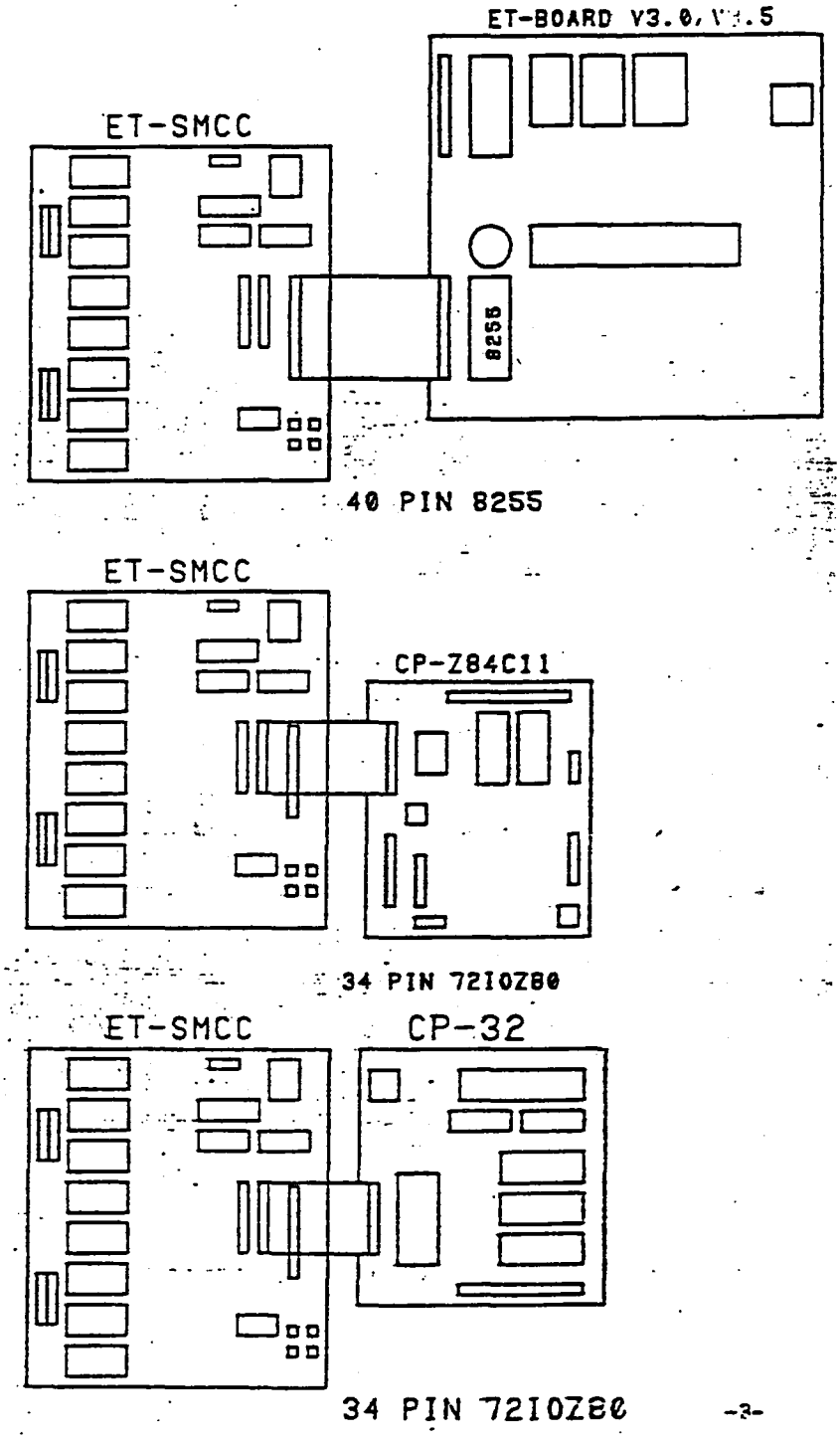
การติดตั้งอุปกรณ์ต่างๆของ ETT

1.สามารถต่อกับสเต็ปปิ้งมอเตอร์ ตามรูปแบบขดของสเต็ปปิ้งนั้นได้ดังรูปที่ 4.5 โดยต่อมอเตอร์ได้ 2 ชุด



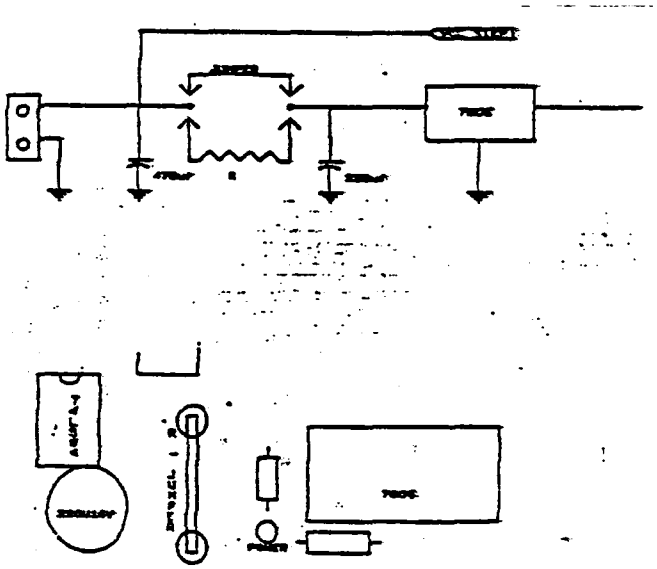
รูปที่ 4.4 การต่อสเต็ปปิ้งมอเตอร์กับ ET-SMCC

2.ต่อร่วมกับการ์ดต่างๆของบริษัท ETT ทาง connector 72IOZ80(32 ขา) หรือต่อกับ 8255 ดังรูป 4.5



รูปที่ 4.5 การต่อ ET-SMCC กับการ์ด 8255

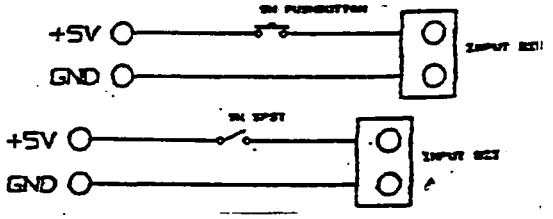
3.การต่อแหล่งจ่ายไฟ กับ ET-SMCC และ สเต็ปปิ้งมอเตอร์จากการ์ด ET-SMCC นั้นจะมีส่วนจ่าย Power ให้กับอุปกรณ์บนบอร์ดจาก Power Supply ชุดเดียวกันกับที่จ่ายให้สเต็ปปิ้งมอเตอร์โดยจะต้องใช้ 7805 ปรับไฟจาก +12 โวลต์ หรือ +24 โวลต์ ตามลำดับไฟที่ป้อนให้สเต็ปปิ้งมอเตอร์ โดยอาจจะต้องใช้ตัวความต้านทานต่อแทนสาย JUMP ถ้าในกรณีต่อกับสเต็ปปิ้งมอเตอร์ที่ต้องการไฟมากกว่า 12 โวลต์โดยควรต่อตัวความต้านทานแทนสาย JUMP (ในกรณีที่ใช้ 24 โวลต์ ให้ต่อตัวความต้านทาน 50 โอมม์ 5 วัตต์) ดังแสดงตามรูปที่ 4.6



รูปที่ 4.6 การต่อ Power Supply กับ ET-SMCC

โดยวงจรภายใน ET-SMCC (ไม่รวมสเต็ปปิ้งมอเตอร์)ใช้กระแสไฟสูงสุด 250 mA

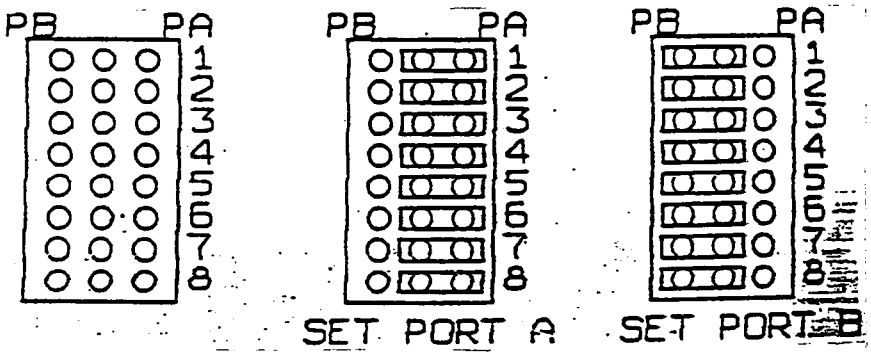
4.ถ้ามีอินพุทจะต้องต่อร่วมกับบอร์ด ET-SMCC โดยต่อกับอินพุทออร์ทที่มีลักษณะเป็นออปโตดังรูปที่ 4.7 โดยใช้ Power Supply +5 โวลต์ และถ้าต้องการใช้เป็นอินพุท 24 โวลต์ก็ให้เปลี่ยนเป็นความต้านทาน 1K ในส่วนของอินพุทเป็นความต้านทานค่า 3.3 K



รูปที่ 4.7 การใช้งานพอร์ทอินพุทของบอร์ด ET-SMCC

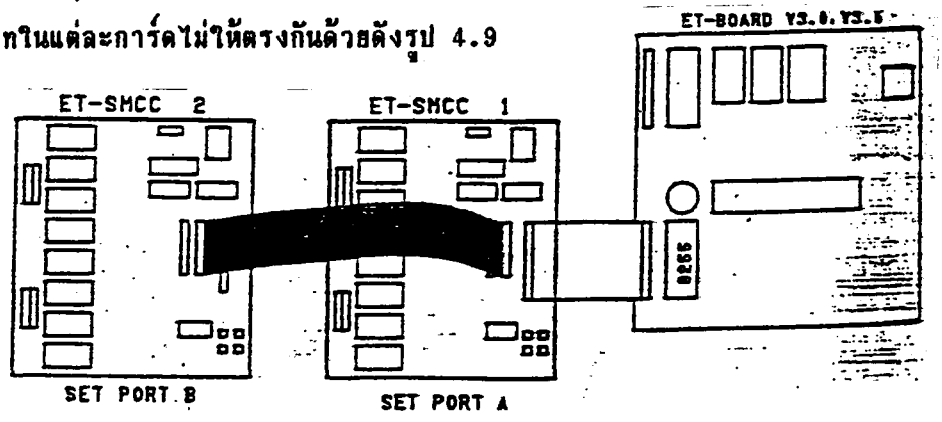
การ decode พอร์ท

บอร์ด ET-SMCC จะมี JUMPER อยู่ 2 ชุด โดยชุดแรกจะเป็นการเลือกว่าสเต็ปปิ้งมอเตอร์จะใช้พอร์ทใดในการควบคุมคือจะใช้พอร์ท A หรือพอร์ท B ก็ได้ดังรูป 4.8



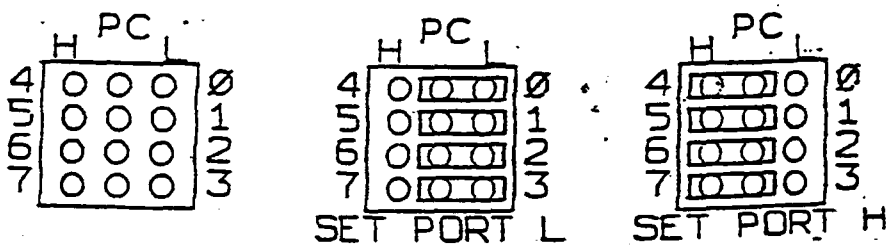
รูปที่ 4.8 การเซ็ตจตำแหน่งของ JUMPER เพื่อเลือกพอร์ทใช้งาน

จากลักษณะนี้สามารถเลือกพอร์ท A หรือ B ได้ทำให้ ET-SMCC สามารถต่อร่าง ET-SMCCอีกชุดหนึ่งได้จากขั้ว 34 ขา(72IOZ80) ได้อีกโดยต้องเซ็ตการใช้พอร์ทในแต่ละพอร์ทในแต่ละการ์ดไม่ให้ตรงกันด้วยดังรูป 4.9



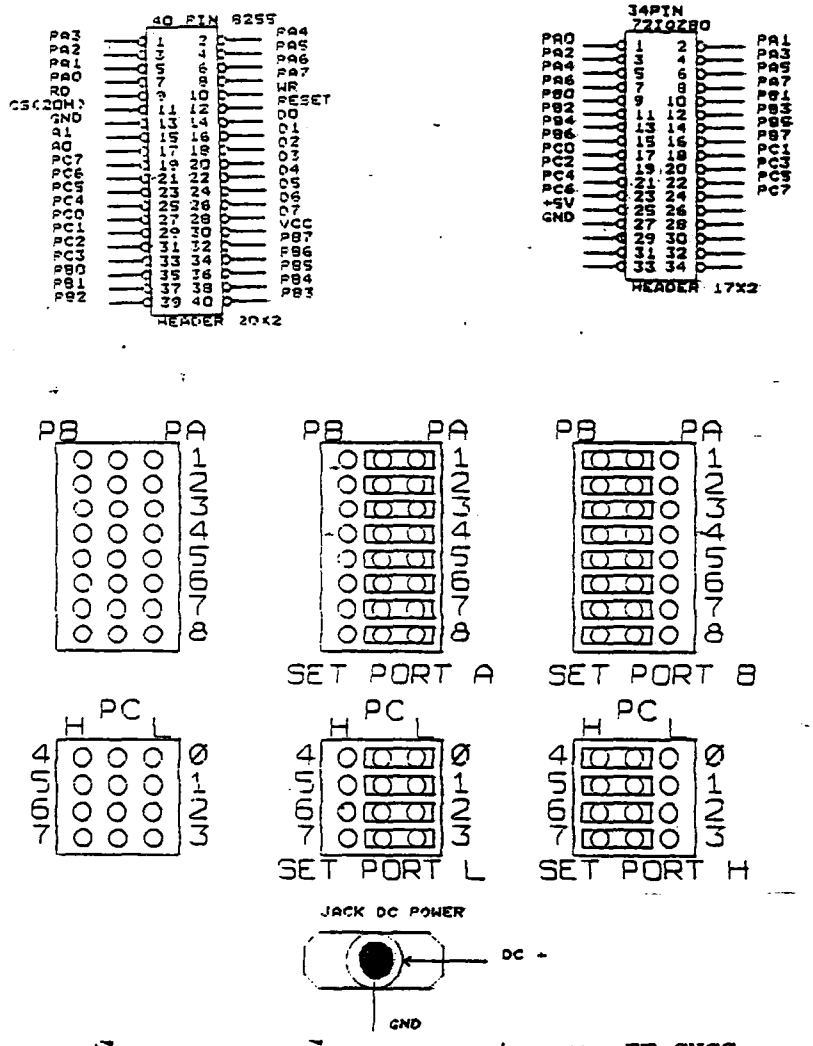
รูปที่ 4.9 การต่อบอร์ด ET-SMCC เพื่อใช้งานร่วมกัน

ชุดที่2จะเป็นส่วนของอินพุทพอร์ท โดยสามารถเลือกว่าจะรับอินพุทจากพอร์ท C ด้วย 4 บิทสูง หรือ 4 บิทต่ำ ดังรูป 4.10



รูปที่ 4.10 การเลือกตำแหน่งของพอร์ทอิกพุท

การจัดวางตำแหน่งของ connector ที่ต่อกับ ET-SMCC แสดงดังรูปที่ 4.11



รูปที่ 4.11 รายละเอียดขา connector ของ ET-SMCC

การควบคุมและใช้งานสเต็ปปิ้งมอเตอร์

สามารถสั่งงานให้สเต็ปปิ้งมอเตอร์หมุนได้ 3 แบบ ตามกระแสไฟที่ป้อนแก่เฟสต่างๆ ดังนี้

1.แบบจ่ายกระแสไฟให้เฟสเดียวานเวียนกันไป หรือ ONE EXCITATION หรือแบบ HALF DRIVE คือเอาท์พุทจะเป็น 0001,0010,0100,1000

ตารางที่ 4.2 แสดงการขับแบบ Half step

	OUT4	OUT3	OUT2	OUT1
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

การ EXCITATION แบบนี้แรงบิดจะมีน้อย

2.แบบจ่ายกระแสไฟทีเดียว 2 เฟส หรือ TWO-EXCITATION หรือ FULL STEPคือ 0011,0110,1100,1001 หมุนเวียนกันไปแบบนี้แรงบิดจะมาก

ตารางที่ 4.3 แสดงการขับแบบ Full step

	OUT4	OUT3	OUT2	OUT1
1	0	0	1	1
2	0	1	1	0
3	1	1	0	0
4	1	0	0	1

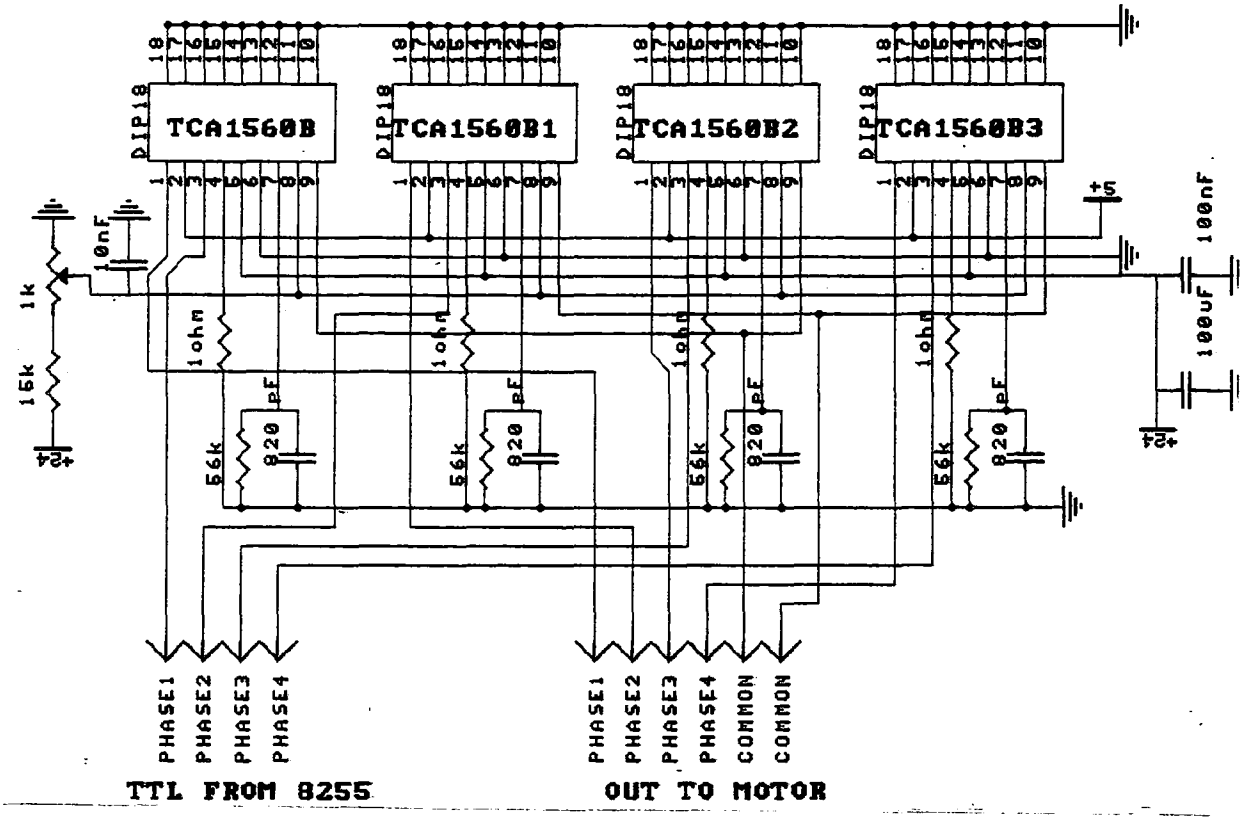
3.แบบจ่ายกระแสไฟ 1 เฟสสลับกับ 2 เฟสเรียกว่า แบบONE-TWO EXCITATION นี้จำนวนสเต็ปจะเพิ่มขึ้นเป็น 2เท่าของสองแบบแรกแต่แรงบิดเฉลี่ยจะน้อย การหมุนจะเป็น 0001,0011,0010,0110,0100,1100,1000,1001

ตารางที่ 4.4 แสดงการขับแบบ One-two motor driver

	OUT4	OUT3	OUT2	OUT1
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	0	0
6	1	1	0	0
7	1	0	0	0
8	1	0	0	1

4.2.2 ใช้ IC TCA1560B เป็นชุดขับสเต็ปปิ้งมอเตอร์

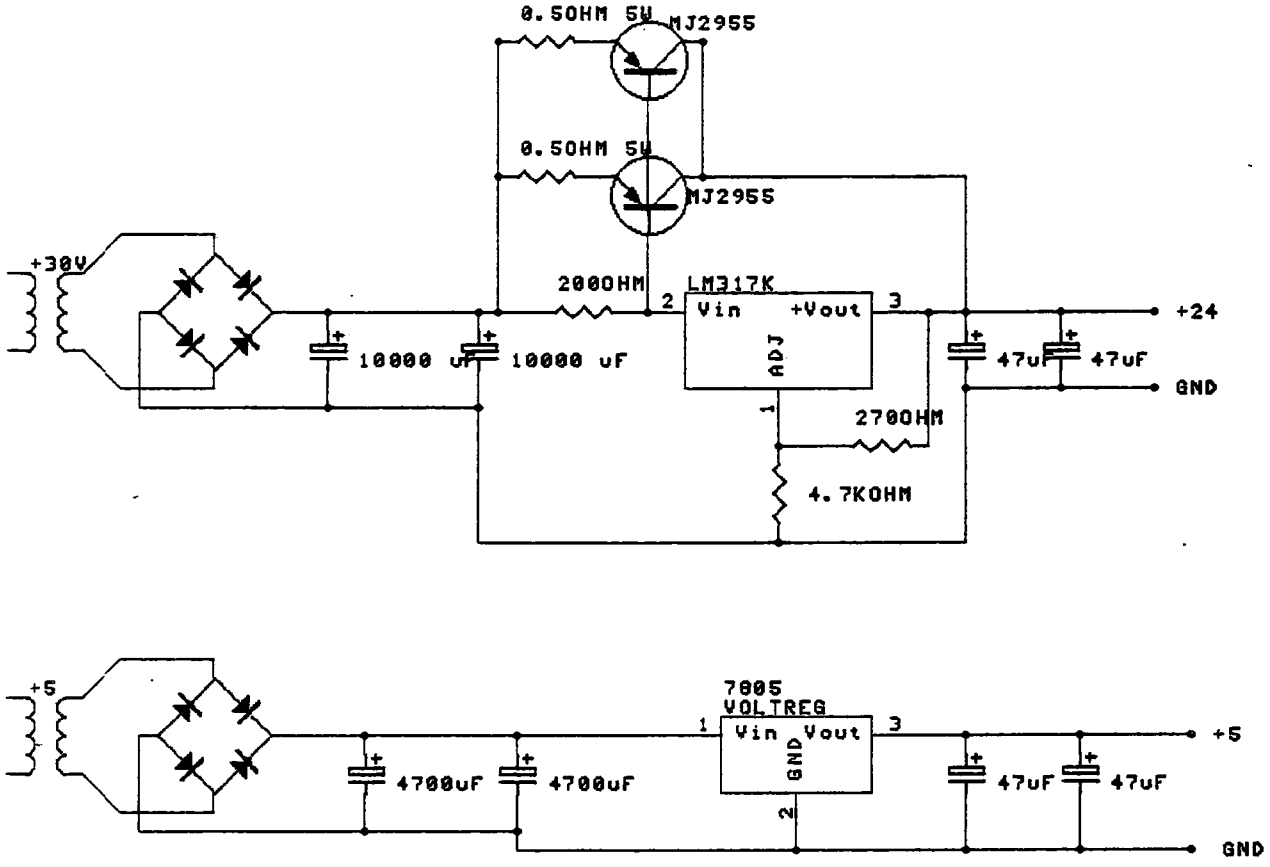
TCA1560B เป็นไอซีที่ใช้ขับสเต็ปปิ้งมอเตอร์ของบริษัท Siemen การใช้งานในโครงงานนี้แสดงตามวงจรในรูปที่ 4.12



รูปที่ 4.12 วงจรของชุดขับสเต็ปปิ้งมอเตอร์โดยใช้ TCA1560B

4.3 ภาคจ่ายกำลัง (POWER SUPPLY)

เนื่องจากการทำงานของสเต็ปปิ้งมอเตอร์จะกำหนดกระแสที่สเต็ปปิ้งมอเตอร์แต่ละตัวนั้นๆต้องการ ดังนั้นการทำภาคจ่ายกำลัง(POWER SUPPLY)จะต้องทำให้เพียงพอแก่ความต้องการของสเต็ปปิ้งมอเตอร์ ในโครงงานนี้ใช้สเต็ปปิ้งมอเตอร์ที่กินกระแส 0.22 แอมป์/เฟส ทั้งหมด 3 ตัว จึงกินกระแสประมาณ 3 แอมป์ จึงได้ทำการออกแบบภาคจ่ายกำลัง(POWER SUPPLY)ที่จ่ายประมาณ 3 แอมป์ ถึงจะเพียงพอแก่ความต้องการของสเต็ปปิ้งมอเตอร์ ดังรูปที่ 4.13



รูปที่ 4.13 วงจร POWER SUPPLY

บทที่ 5

ชุดโปรแกรมควบคุม

5.1 การทำงานของโปรแกรม

โปรแกรมที่ใช้ในโครงงานนี้เขียนโดยใช้ภาษาปาสคาล ซึ่งเป็นภาษาที่ใช้งานง่าย และเป็นที่นิยมใช้กันอย่างกว้างขวาง ภาพโดยรวมของโปรแกรมจะทำงานตาม flowchart รูปที่ 5.1ขนาดของลายวงจรที่สามารถนำมาใช้กับเครื่องต้องมีขนาดไม่เกิน 6.4" X 4.8" การทำงานในส่วนที่สำคัญสามารถแบ่งได้เป็น 4 ส่วนใหญ่ ๆ คือ

- การ load file
- การวาดภาพของลายวงจร
- การหาขอบของภาพลายวงจร
- การควบคุมแท่นเพื่อทำแผ่น PCB

การ load file

เป็นส่วนการอ่านไฟล์ที่เป็นลายวงจรที่ผู้ใช้ออกแบบไว้บนโปรแกรมโปรเทล ขั้นตอนการทำงานของโปรแกรมในส่วนนี้เป็นตาม flowchart รูปที่ 5.2 การทำงานโดยรวมของโปรแกรมในส่วนนี้ คือ การอ่านข้อมูลจากไฟล์ที่ต้องการ และจัดการกับข้อมูลที่อ่านเข้าเพื่อแยกข้อมูล ออกเป็นส่วนต่าง ๆ เช่น ส่วนข้อมูลที่เป็นการลากเส้น การวางขาอุปกรณ์ เพื่อเป็นการใช้หน่วยความจำของคอมพิวเตอร์ให้ประสิทธิภาพสูงสุดการเก็บข้อมูลต่างที่อ่านเข้ามาจากไฟล์จึงใช้การนำข้อมูลมาจัดเก็บแบบ Links List

การวาดภาพของลายวงจร

เป็นส่วนการสร้างภาพของลายวงจรบนจอภาพจากข้อมูลที่อ่านเข้ามา ในการสร้างภาพของลายวงจรจะต้องเป็นภาพที่กลับจากซ้ายเป็นขวาจากภาพของวงจรที่ออกแบบบนโปรแกรมโปรเทลเพื่อให้ได้ตำแหน่งของขาอุปกรณ์ที่ถูกต้อง การสร้างภาพที่ใช้ในโปรแกรมนี้จะสร้างเฉพาะส่วนที่สร้างจากคำสั่ง Place-Trace,Place-Via,Place-Fill,Place-Component จากโปรแกรมโปรเทล

การหาขอบของภาพลายวงจร

เป็นส่วนที่จัดการกับภาพของลายวงจรที่สร้างขึ้น เนื่องจากการทำงานในส่วนการทำแผ่น PCB ในโครงงานพิเศษนี้เป็นการตัดชั้นของทองแดงที่เคลือบบนชิ้นงาน จึงต้องใช้วิธีหาขอบรอบเส้นของลายวงจร ส่วนของการหาขอบเนื่องจากภาพที่สร้างขึ้นเป็นภาพสีเดียวในการหาขอบจึงเลือกให้วิธีทดสอบสีกับจุดที่อยู่รอบ ๆ ว่าจุดนั้นเป็นขอบของลายวงจรหรือไม่ ขั้นตอนการทำงานในการขอบของลายวงจรแสดงดัง flowchart รูปที่ 5.3

การควบคุมแท่นเพื่อทำแผ่น PCB

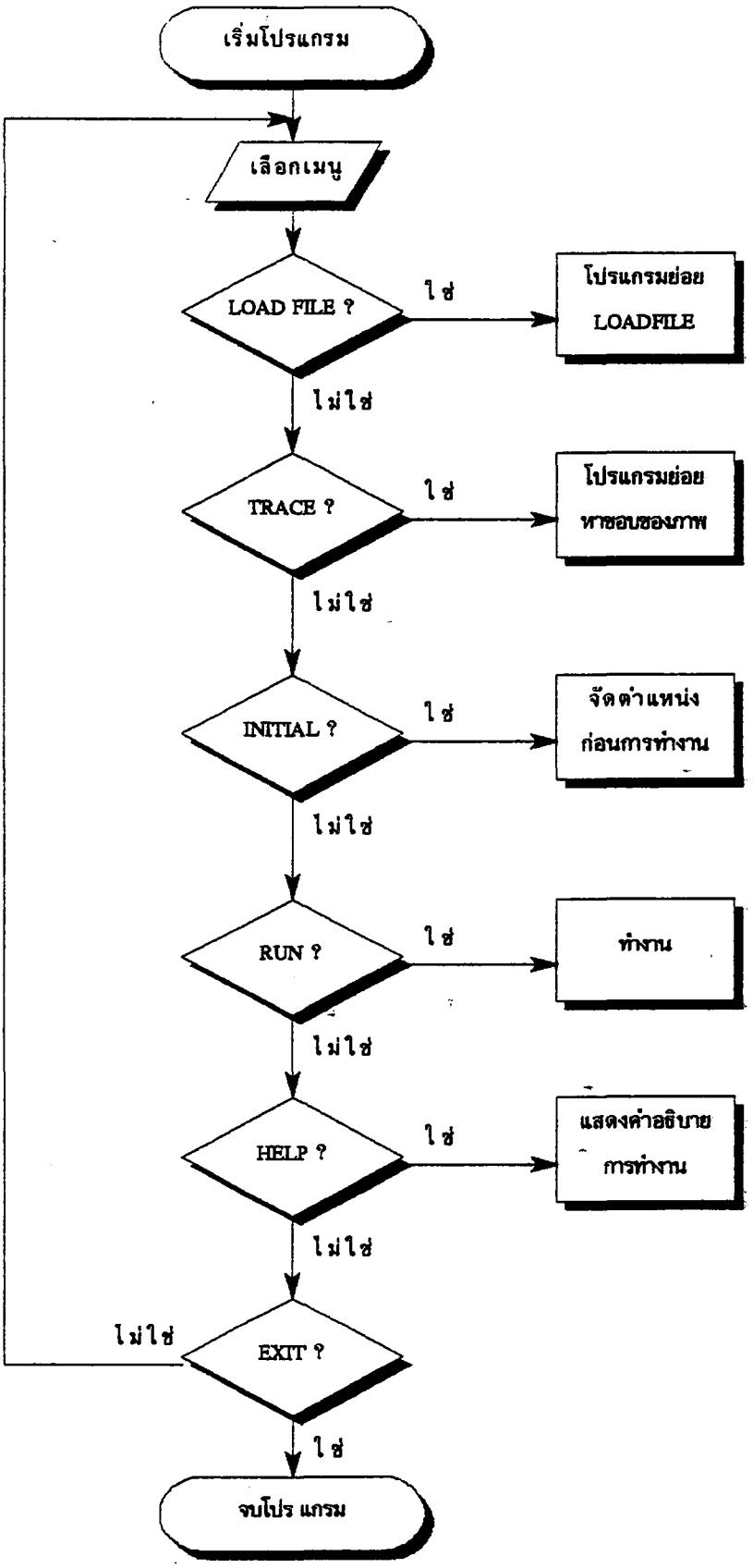
เป็นส่วนที่ทำหน้าที่ทำแผ่น PCB จากภาพของลายวงจรที่ผ่านการหาขอบของภาพแล้วการทำงานของโปรแกรมในโครงงานนี้เพียงสร้างเส้นตรงได้เท่านั้น คือ โปรแกรมจัดการกับภาพรูปทรงต่าง ๆ โดยแบ่งเป็นเส้นตรงที่ยาวที่สุดต่อเนื่องกันจนเป็นรูปทรงต่าง ดังแสดงตาม flowchart รูปที่ 5.4

5.2 การใช้โปรแกรม

การใช้งานโปรแกรมสามารถทำได้โดยการสั่งงานทางเมนูดังรูปที่ 5.5 โดยมีตัวเลือกดังนี้

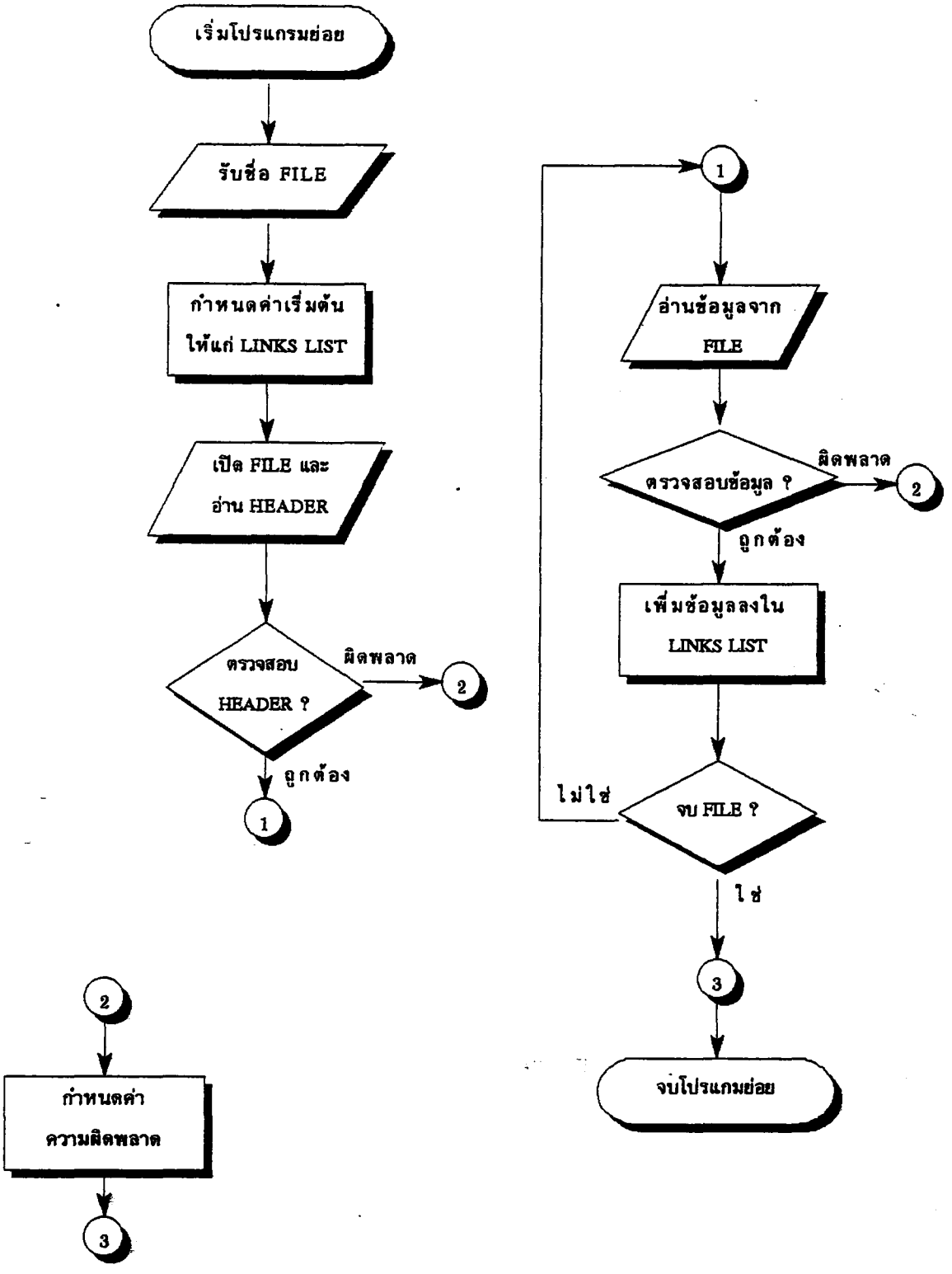
- ตัวเลือก File
- ตัวเลือก Trace
- ตัวเลือก Initial
- ตัวเลือก Run
- ตัวเลือก Dimension
- ตัวเลือก Help
- ตัวเลือก Exit

MAIN PROGRAM



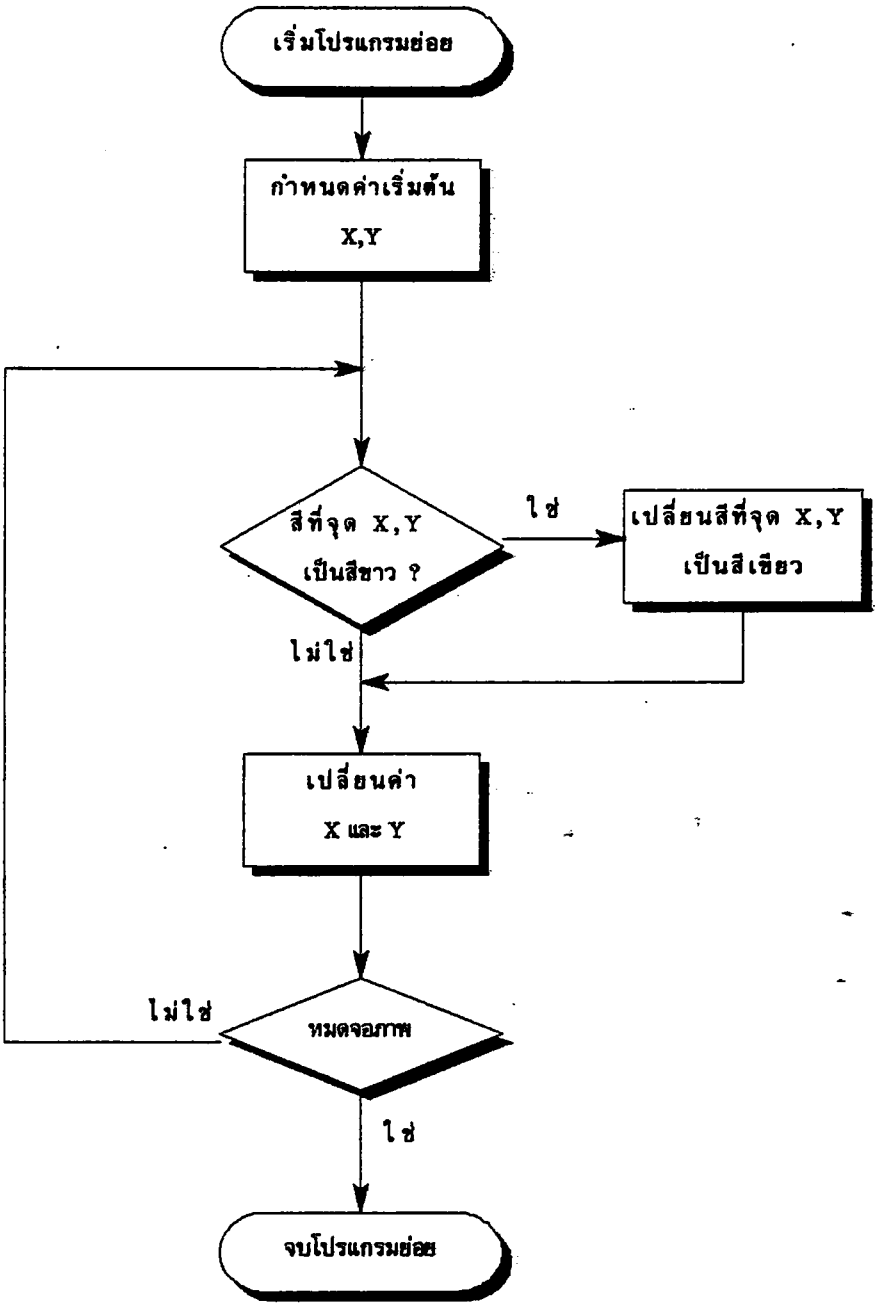
รูปที่ 5.1 Flowchart ของโปรแกรมที่พัฒนาขึ้น

โปรแกรมย่อย LOADFILE



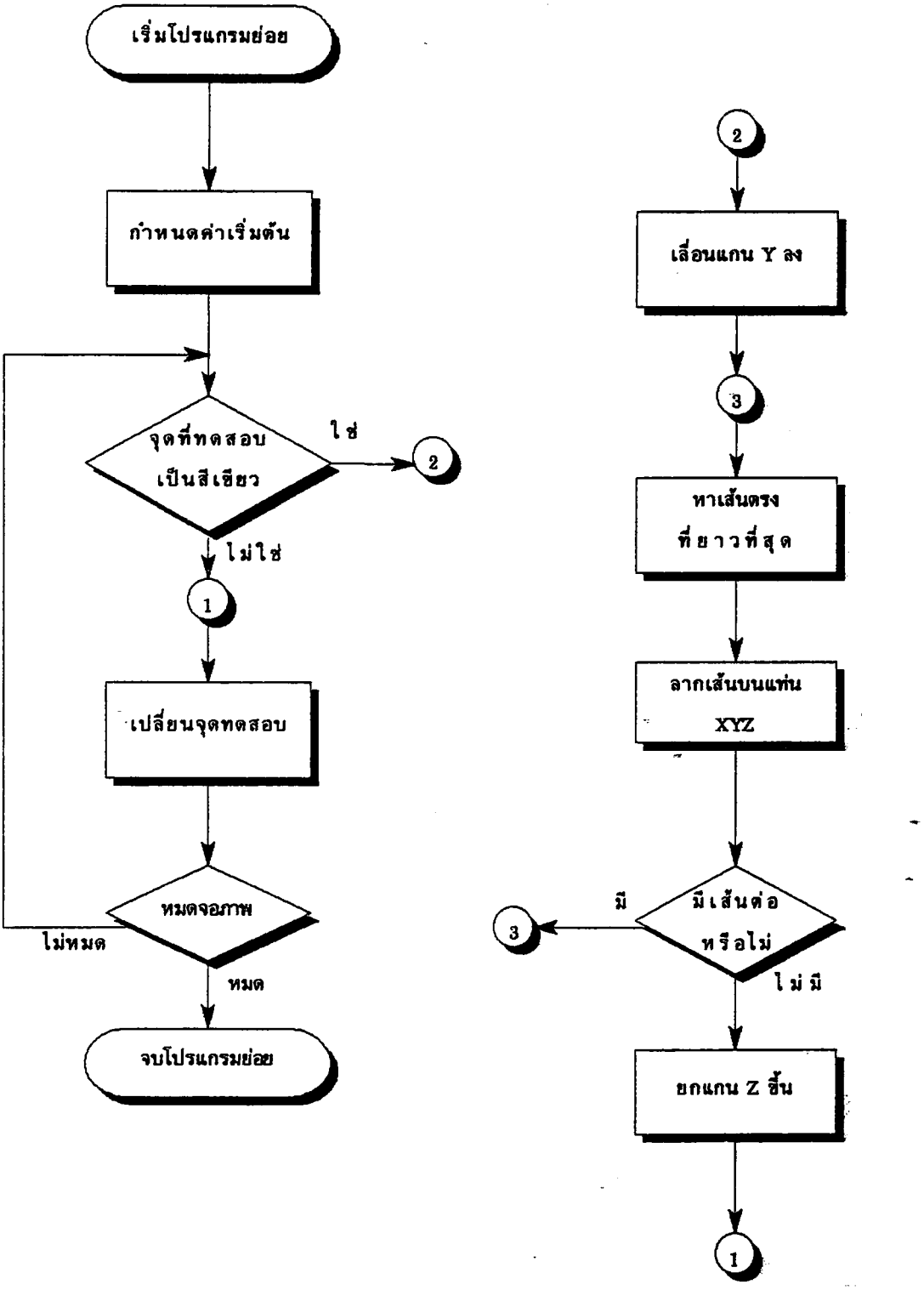
รูปที่ 5.2 Flowchart ของโปรแกรมในส่วนการ Load File

โปรแกรมย่อยหาขอบของภาพ

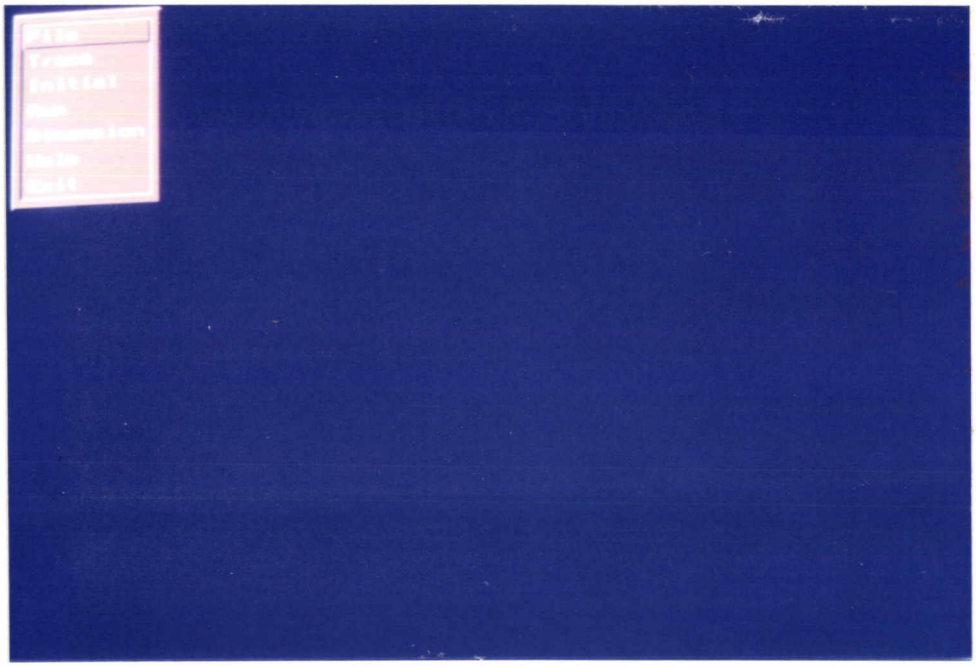


รูปที่ 5.3 Flowchart ของโปรแกรมในส่วนการหาขอบของภาพ

โปรแกรมย่อย RUN



รูปที่ 5.4 Flowchart ของโปรแกรมในส่วนการทำแผ่น PCB

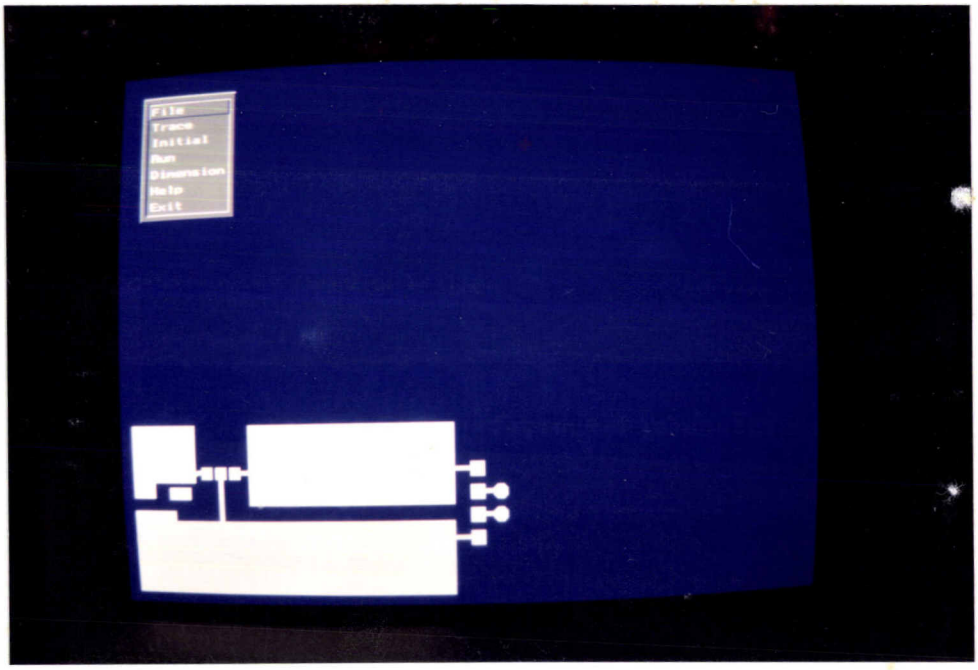


รูปที่ 5.5 รูปแสดงเมนูของโปรแกรม

ตัวเลือก File

เมื่อเลือกตัวเลือกนี้โปรแกรมจะรอรับชื่อไฟล์ที่ต้องการนำมาเป็นต้นแบบในการทำแผ่นPCB หากผู้ใช้ป้อนชื่อไฟล์ โดยไม่ระบุสกุลของไฟล์โปรแกรมจะกำหนดให้สกุลของไฟล์ที่ป้อนเข้ามาเป็น ".PCB" และหากชื่อไฟล์ที่ป้อนเข้ามามีเครื่องหมาย "*" หรือ "?" ซึ่งอาจทำให้มีไฟล์หลายไฟล์ที่สอดคล้องกับชื่อไฟล์ที่ป้อนเข้ามา โปรแกรมจะแสดงชื่อไฟล์ทั้งหมดนั้นออกมาให้ผู้ใช้เลือกไฟล์ที่จะอ่านเข้ามา เมื่ออ่านไฟล์เข้ามาเรียบร้อยไม่มีข้อผิดพลาดโปรแกรมจะแสดงลายวงจรบนจอภาพดังรูปที่ 5.6 ข้อผิดพลาดที่สามารถเกิดขึ้นได้ในการอ่านไฟล์เข้ามาคือ

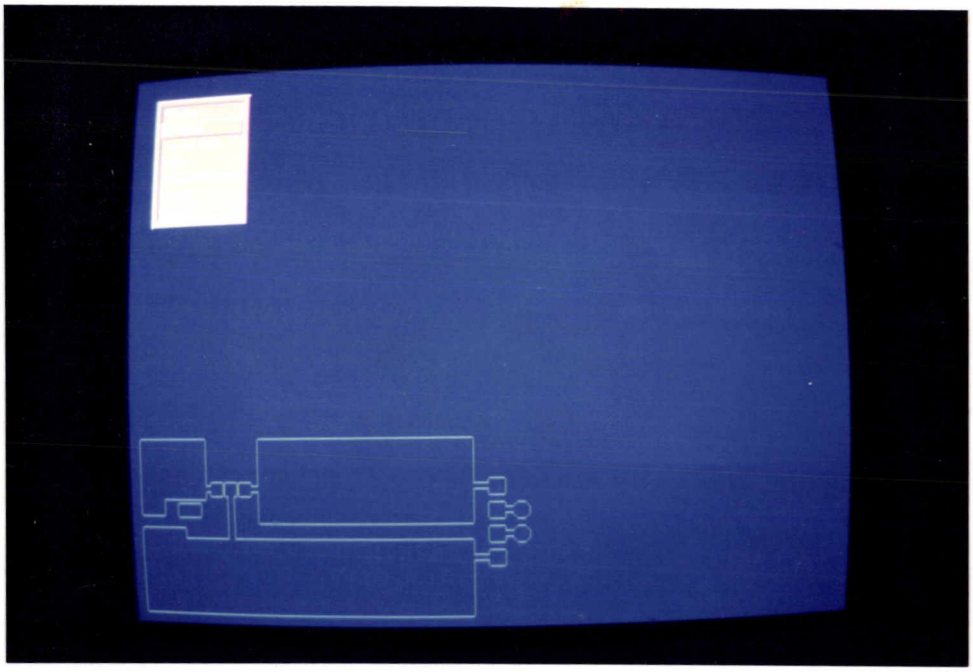
- ไฟล์ที่อ่านเข้ามาไม่ใช่ไฟล์ของวงจรที่ออกแบบบนโปรแกรม Protel
- ไฟล์ที่อ่านเข้ามาไม่สมบูรณ์เมื่ออ่านเข้าไปแล้วไม่สามารถดีความข้อมูลได้



รูปที่ 5.6 แสดงการวาดรูปลายวงจรบนจอภาพโดยกลับซ้ายเป็นขวา

ตัวเลือก Trace

เป็นตัวเลือกเพื่อหาขอบของลายวงจรก่อนการทำแผ่น PCB ในการทำงานจะใช้เวลาประมาณ 3 นาที แสดงดังรูป 5.7



รูป 5.7 แสดงภาพของลายวงจรที่ผ่านขบวนการหาขอบภาพแล้ว

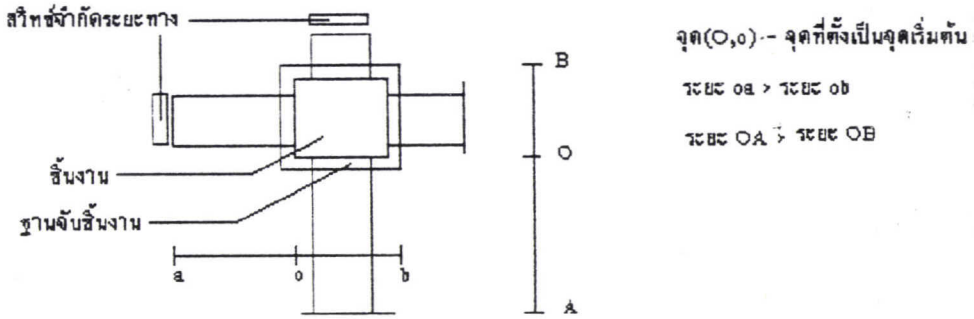
ตัวเลือก Initial

เป็นตัวเลือกที่ใช้ในการเตรียมความพร้อมก่อนทำแผ่น PCB โดยก่อนเริ่มการทำแผ่น PCB ต้องทำตามขั้นดังนี้

- กำหนดตำแหน่งเริ่มต้น(มุมล่างซ้าย)ของชิ้นงานโดยโปรแกมจะถือว่าตำแหน่งนั้นเป็นจุด (0,0) โดยต้องให้ระยะการเคลื่อนที่ของแท่นด้านแกน Y ให้มากกว่าขนาดทางแกน Y ของลายวงจร และให้ระยะการเครื่อนที่ของแท่นด้านแกน X ให้มากกว่าขนาดทางแกน X ของลายวงจรหากระยะทางเหลือไม่พอจะทำให้แท่นเคลื่อนที่ไปชน สวิทช์ที่ใช้จำกัดการเคลื่อนที่ของแท่นเพื่อป้องบอลสกรูเสียหาย การเคลื่อนที่เพื่อกำหนดตำแหน่งควบคุมทางคีย์ลูกศร การจัดตำแหน่งในการ Initial แสดงดังรูปที่ 5.8

- กำหนดความลึกของหัวกัดที่จะใช้ในการทำงาน โดยควบคุมผ่านคีย์ Page Up (เลื่อนหัวกัดขึ้น) และ คีย์ Page Down(เลื่อนหัวกัดลง)

- กดคีย์ Enter เป็นการรับค่าทั้งหมดที่กำหนด

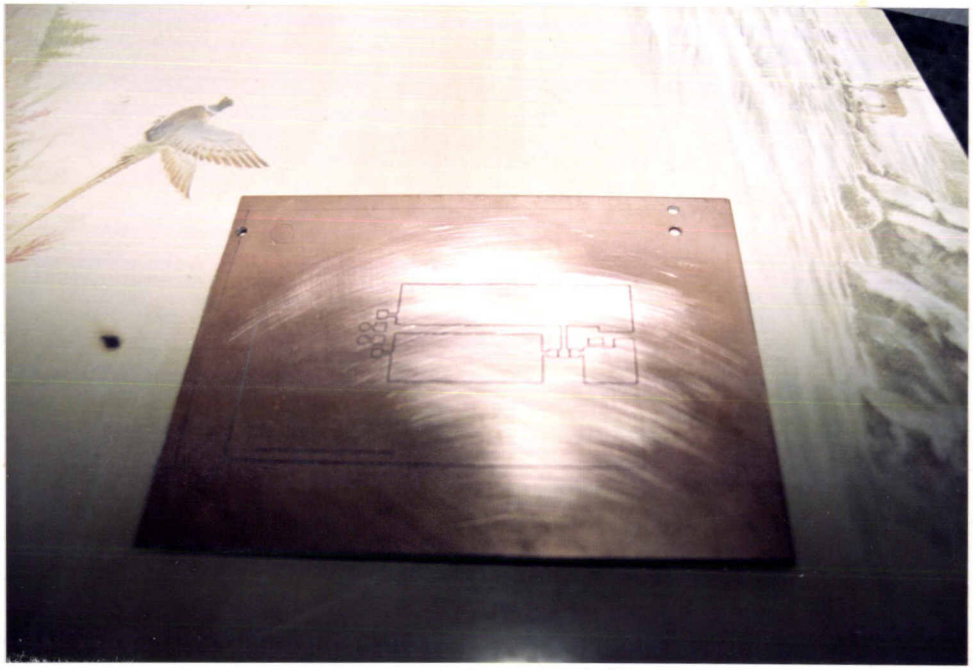


รูป 5.8 แสดงตำแหน่งในการ Initial

ตัวเลือก Run

เป็นการเริ่มทำแผ่น PCB จากรูปต้นแบบบนจอภาพ ระหว่างการทำงานหากแท่นเคลื่อนที่ไปชนทำให้สวิทช์ที่ใช้จำกัดระยะทางทำงานจะทำให้โปรแกรมหยุดทำงาน แต่ถ้าหากผู้ใช้ต้องการหยุดการทำงานสามารถหยุดได้โดยกดปุ่ม Ctrl-Break ระยะเวลาใน

การทำแผ่น PCB แต่ละครั้งจะขึ้นกับความละเอียดของลายวงจรโดยที่ลายวงจรที่มีขนาดใหญ่แต่มีรายละเอียดน้อยอาจใช้เวลาในการทำน้อยกว่าลายวงจรที่มีขนาดเล็กกว่าแต่มีลายละเอียดมาก เมื่อการทำงานในส่วนนี้เรียบร้อยจะได้ชิ้นงานตามรูปที่ 5.9



รูปที่ 5.9 แสดงแผ่น PCB เมื่อสิ้นสุดการทำงาน

ตัวเลือก Dimension

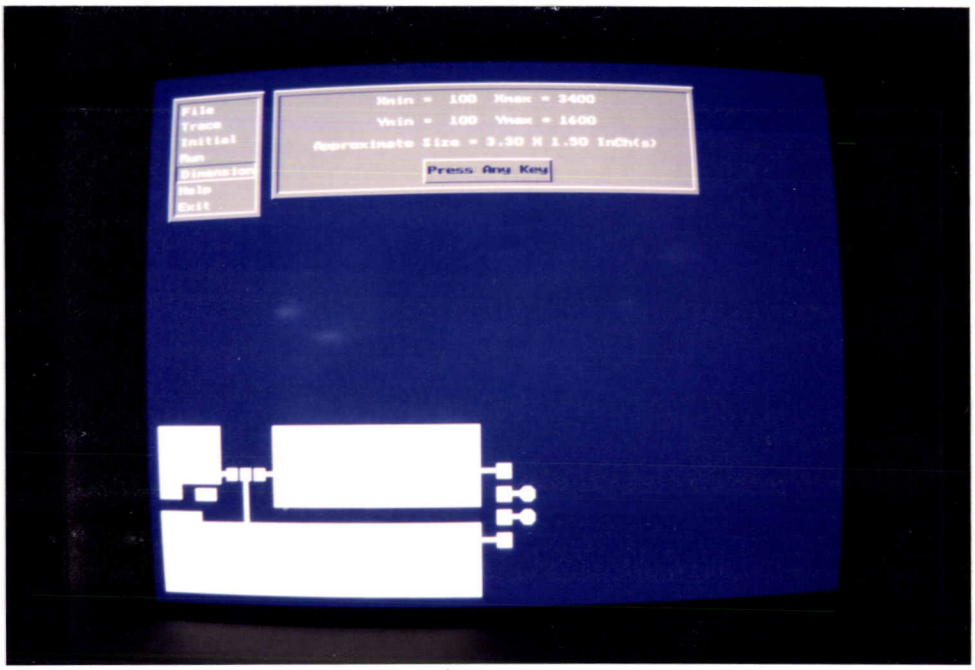
เป็นตัวเลือกให้โปรแกรมแสดงขนาดโดยประมาณของลายวงจรที่วาดบนจอภาพเพื่อนำขนาดไปใช้ในขั้นตอนการ Initial แสดงดังรูปที่ 5.10

ตัวเลือก Help

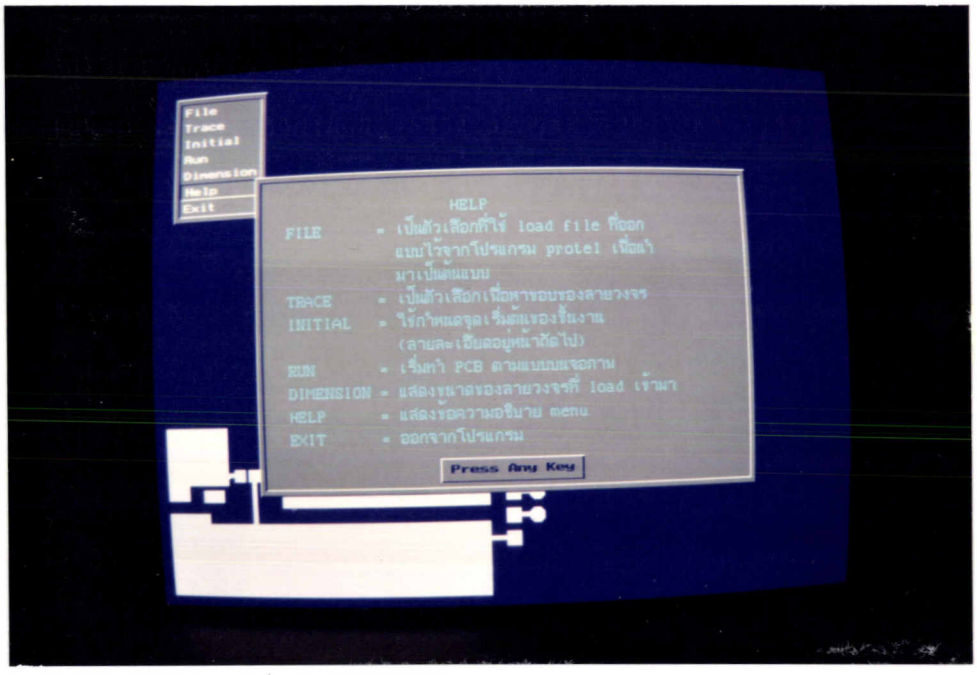
ตัวเลือกนี้จะแสดงรายละเอียดอย่างคร่าว ๆ ของตัวเลือกต่าง ๆ ในการใช้โปรแกรมดังรูปที่ 5.11

ตัวเลือก EXit

ใช้เมื่อต้องการออกจากโปรแกรม



รูปที่ 5.10 ขนาดของลายวงจรที่แสดงบนจอภาพ



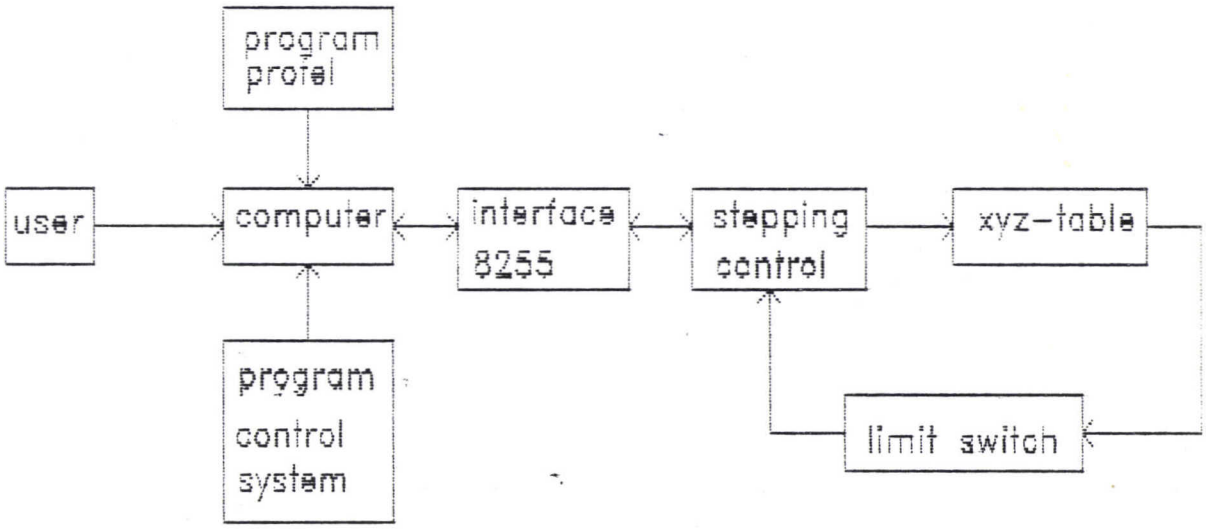
รูปที่ 5.11 แสดงจอภาพเมื่อเลือกตัวเลือก Help

บทที่ 6

การทดลองและผลการทดลอง

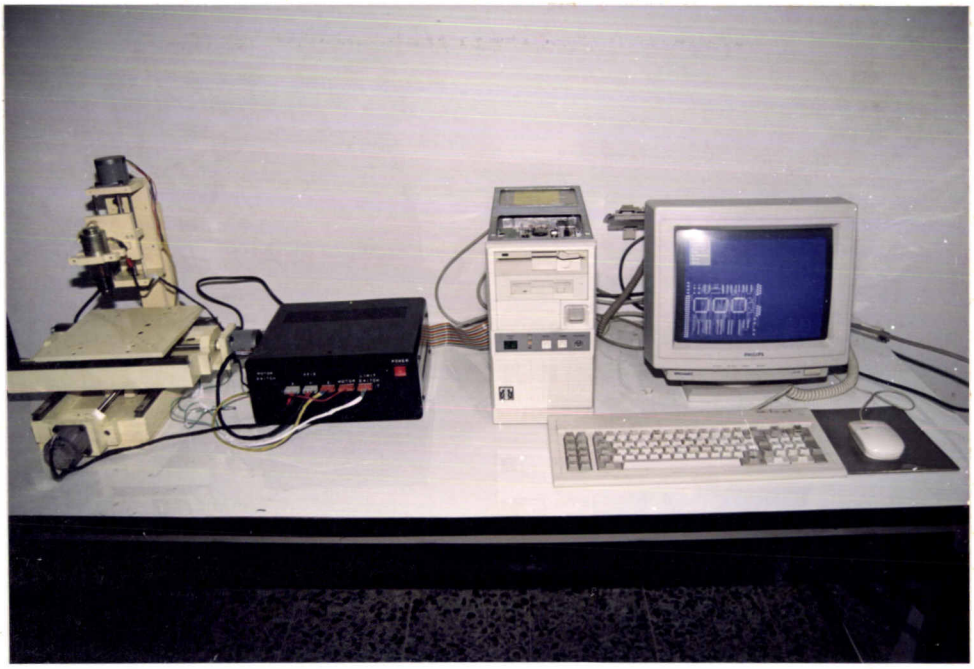
6.1 การติดตั้งและทดสอบระบบขับเคลื่อน(XYZ-table)

หลังจากที่ทำการออกแบบและสร้างระบบขับเคลื่อนรวมถึงระบบควบคุมการขับเคลื่อนมอเตอร์เสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปเป็นการติดตั้งส่วนต่างๆเข้าด้วยกันเป็นระบบที่สมบูรณ์ เพื่อทำการทดสอบการทำงานต่อไป



รูปที่ 6.1 ภาพถ่ายของ XYZ-TABLE พร้อมระบบขับเคลื่อน
ซึ่งควบคุมด้วยระบบคอมพิวเตอร์

จากรูปที่ 6.1 จะทำการทดสอบการทำงานของชุดขับเคลื่อนควบคุมด้วยระบบคอมพิวเตอร์ที่สร้างเสร็จเรียบร้อยแล้ว โดยขั้นตอนในการทำงานจะมีไดอะแกรมดังนี้



รูปที่ 6.2 ไดอะแกรมของระบบควบคุมการทำงานของระบบ

6.1.1 ขั้นตอนการทำงานของระบบ

1.ผู้ใช้จะต้องทำการออกแบบลายวงจรให้เรียบร้อยผ่านทางโปรแกรม Protel ให้เรียบร้อย (โดยการออกแบบนี้มีข้อจำกัดคือต้องออกแบบให้อยู่ในด้านเดียวกัน)

2.ต่อจากนั้นลายวงจรที่ออกแบบไว้เรียบร้อยแล้ว จะถูกนำไปสู่ขั้นตอนของโปรแกรมที่ทำการเขียนไว้เรียบร้อย โดยขั้นตอนของโปรแกรมได้อธิบายไว้แล้วในข้างต้น

3.หลังจากนั้นจะเป็นการส่งผ่านข้อมูลออกทางชุดอินเทอร์เฟส 8255 เพื่อเป็นการควบคุมการทำงานของชุดขับเคลื่อนทั้ง 3 แกน

4.ขั้นสุดท้ายนั้นจะเป็นการใช้หัวกัดเอ็นมิลกัดลายของทองแดง จนงานเสร็จเรียบร้อยตามที่ต้องการ

ส่วนของ feed back นี้มีไว้เพื่อเป็นการป้องกันไม่ให้การเคลื่อนที่ของแท่นออกนอกขอบเขตที่กำหนดไว้เพื่อป้องกันความเสียหายเนื่องมาจากการทำงานของแท่น โดยในกรณีที่มีการเคลื่อนที่เกินความยาวของแท่น อุปกรณ์ที่เป็นไมโครสวิทช์ที่ติดอยู่ที่แกน Xและ Y โดยถ้าแท่นเคลื่อนที่ชนกับสวิทช์จะไปทำการอินเทอร์รัพบอกให้ระบบหยุดทำงาน

6.2 การทดสอบการทำงานร่วมกันทั้ง 3 แกน

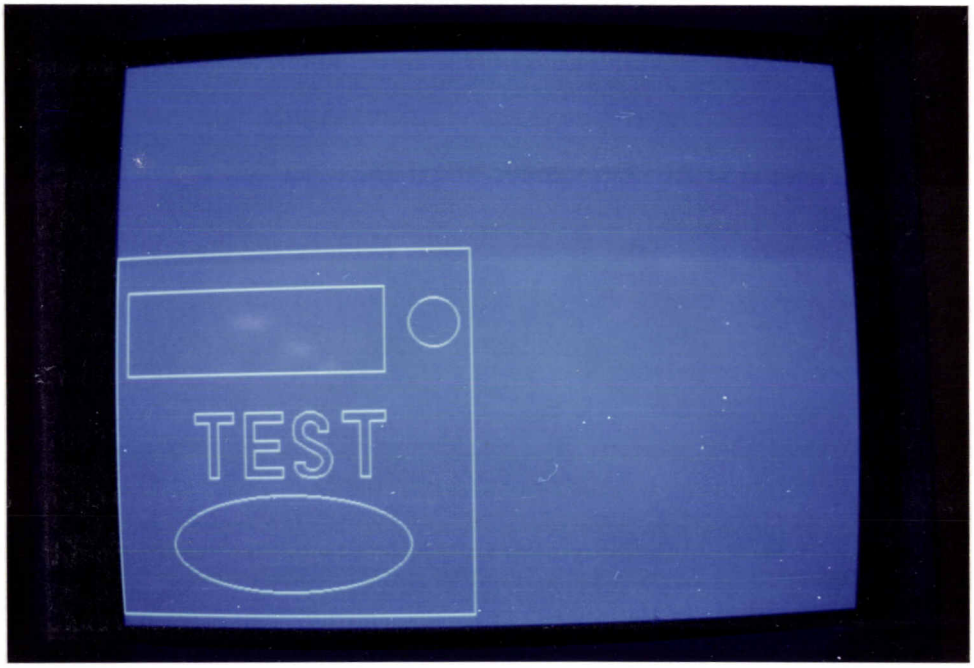
ในขั้นตอนของการทดสอบนี้สามารถที่จะทำการทดสอบ โดยการสร้างภาพที่ต้องการขึ้นบนจอภาพแล้วทดลองส่วนของโปรแกรมในส่วนการทำแผ่น PCB สร้างภาพลายเส้นที่เหมือนบนจอภาพรูปที่ 6.3 แสดงภาพลายเส้นต้นแบบที่ใช้ในการทดสอบ และรูปที่ 6.4 แสดงผลของการทดสอบสร้างภาพลายเส้นตามต้นแบบ

6.3 การทดสอบความถูกต้องของการลากเส้น

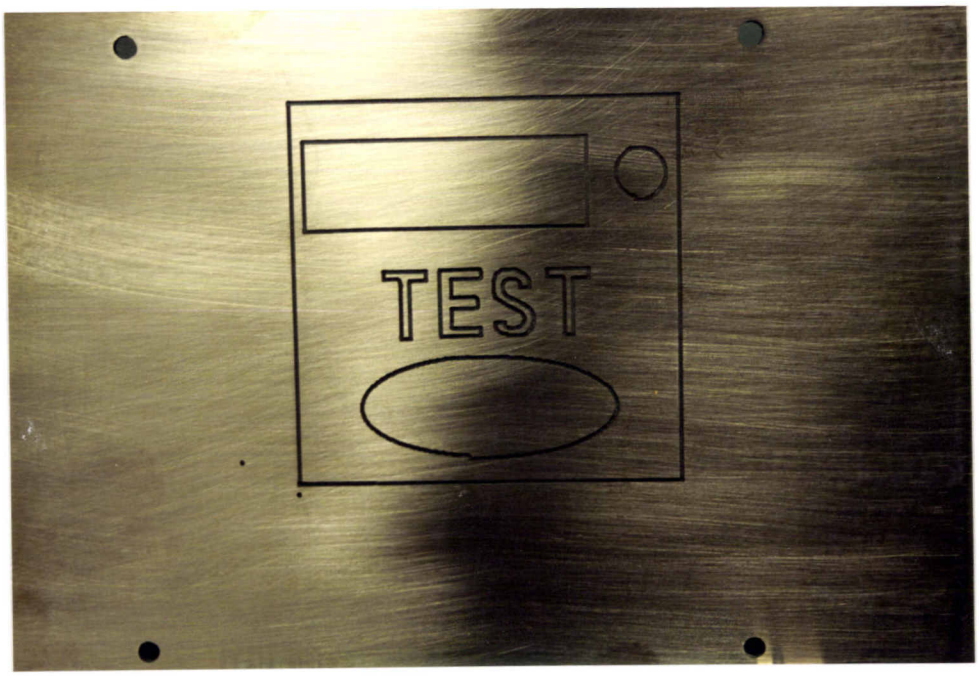
การทดสอบนี้เป็นการทดสอบความถูกต้องในการลากเส้นจาก XYZ-Table โดยสร้างภาพสี่เหลี่ยมจตุรัสขนาด 3",2" และ 1" ซ้อนกันตามลำดับ จากนั้นใช้โปรแกรมในส่วนการทำแผ่น PCB สร้างภาพตามแบบบนจอภาพและวัดความยาวของเส้นที่ลาก รูปที่ 6.5 แสดงภาพลายเส้นต้นแบบที่ใช้ทดสอบ รูปที่ 6.6 แสดงผลการทดสอบที่ได้

6.4 การทดลองสร้างแผ่น PCB

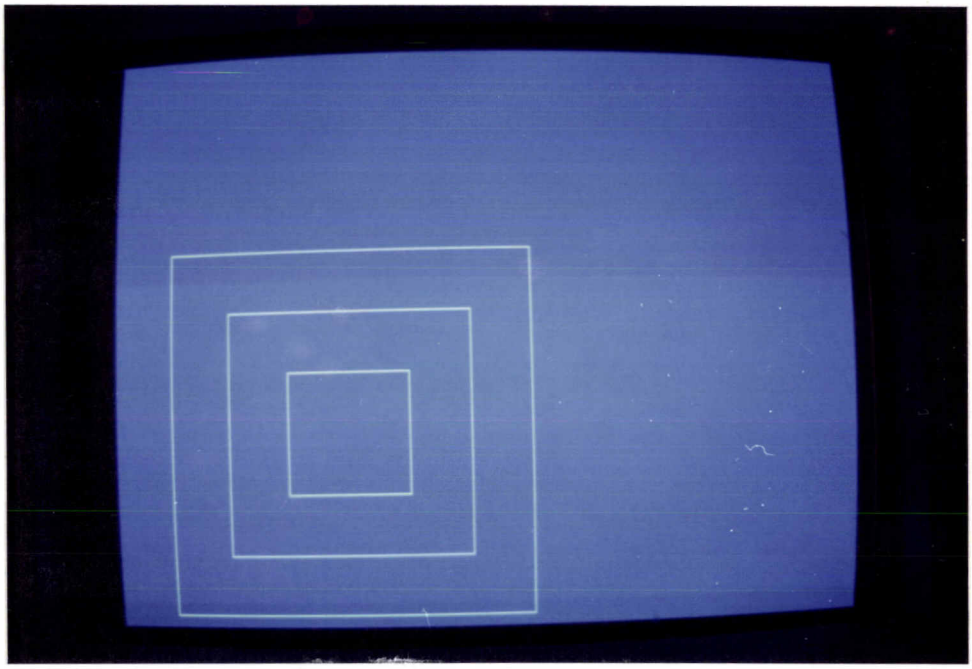
เป็นการทดสอบระบบทั้ง Hardware และ Software ในโครงงานนี้ โดยนำลายวงจรที่ออกแบบจากโปรแกรม Protel และทำงานทดสอบการทำงานของโปรแกรมทุกส่วน รูป 6.7 แสดงลายวงจรที่นำมาทดสอบ และรูป 6.8 แสดงแผ่น PCB ที่ได้จากการทำงานของเครื่องที่สร้างขึ้น



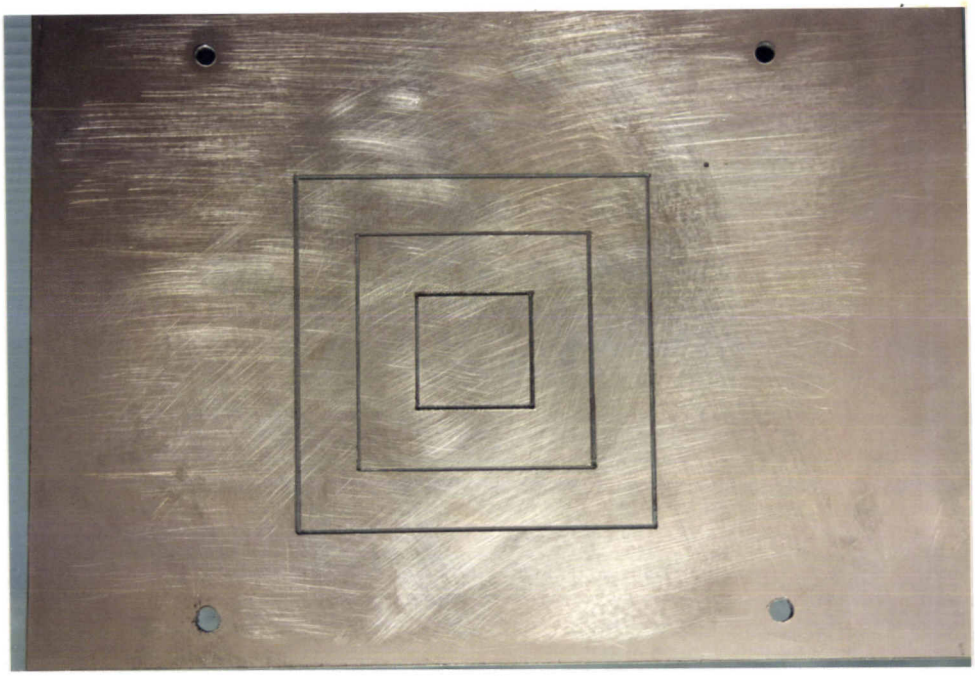
รูปที่ 6.3 ภาพลายเส้นที่ใช้ในการทดสอบการทำงานของแท่น



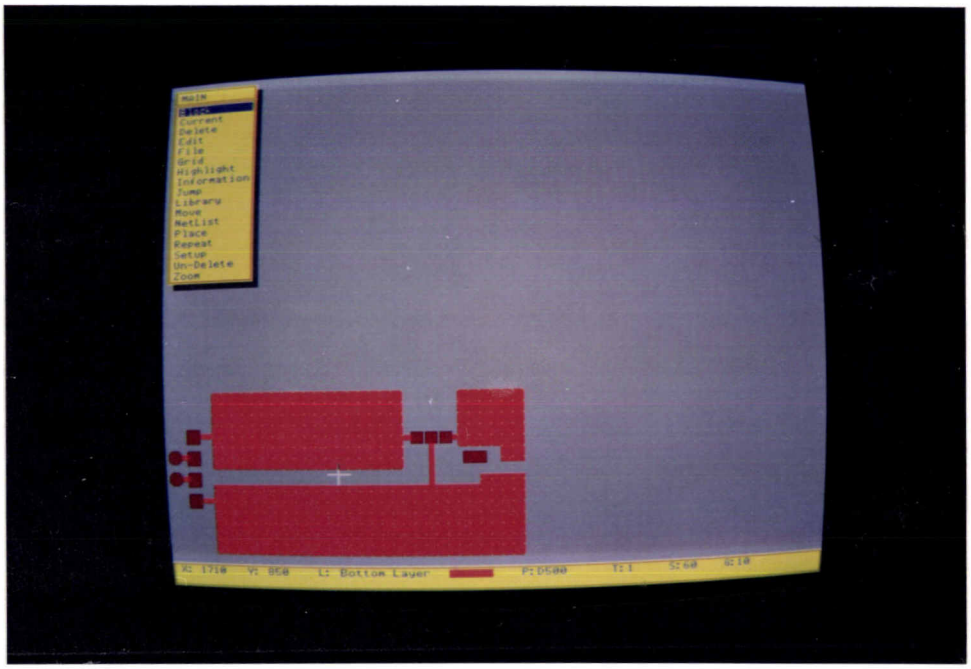
รูปที่ 6.4 ผลการทอสอบจากภาพต้นแบบรูปที่ 6.3



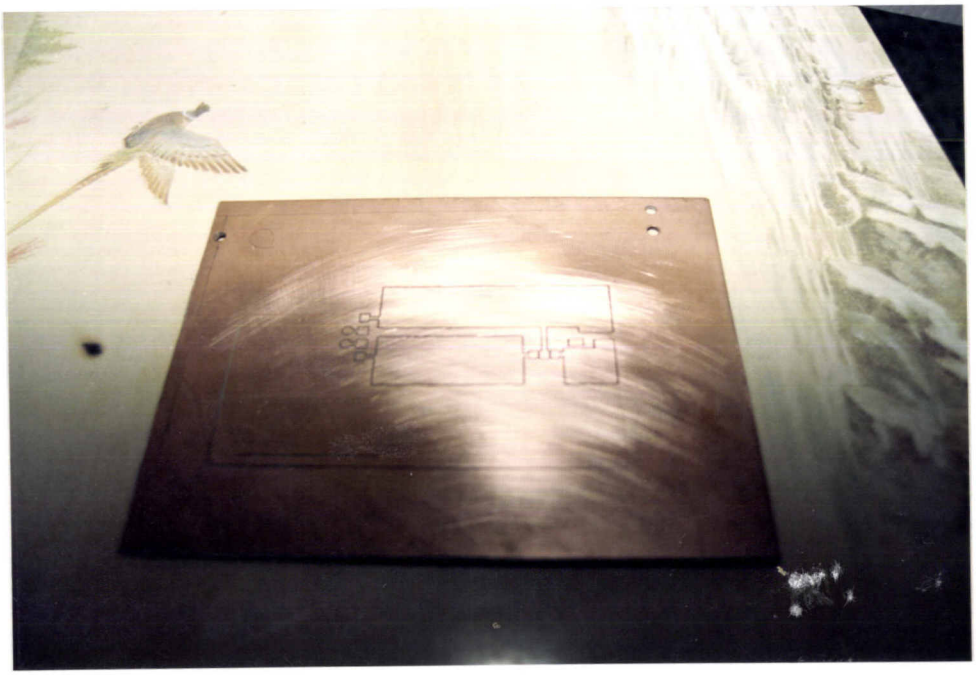
รูปที่ 6.5 ภาพลายเส้นสี่เหลี่ยมจตุรัส ขนาด 3",2" และ 1" ตามลำดับ



รูปที่ 6.6 ผลการทดลองที่ได้จากต้นแบบรูป 6.5



รูปที่ 6.7 ภาพลายวงจรที่ออกแบบบนโปรแกรม Protel



รูปที่ 6.8 แผ่น PCB ที่ได้จากการทำงานของเครื่องที่สร้างขึ้น

บทที่ 7

สรุปและข้อเสนอแนะ

จากการสร้างและพัฒนาเครื่องต้นแบบ ชุดขับเคลื่อน 3 แกนซึ่งขับเคลื่อนด้วย Ball Screw (ระยะ Pitch 2mm) พร้อมระบบควบคุมมอเตอร์ที่ใช้ขับเคลื่อนได้ออกแบบสร้างให้สามารถควบคุมการเคลื่อนที่ของสเต็ปปิ้งมอเตอร์ 400 สเต็ปต่อรอบ ทำให้ความละเอียดในการเคลื่อนที่ของแท่นเท่ากับ 5 um พบว่าในการทดสอบความแม่นยำของตำแหน่งในการเคลื่อนที่ ได้ผลเป็นที่น่าพอใจ ซึ่งทดสอบโดยการสร้างภาพต้นแบบให้เครื่องลากเส้นตามภาพต้นแบบนั้น ส่วนในการทดสอบด้วยการใช้ทำแผ่น PCB ผลที่ได้ดีพอสมควร แต่ก็มีปัญหาเกิดขึ้นดังนี้

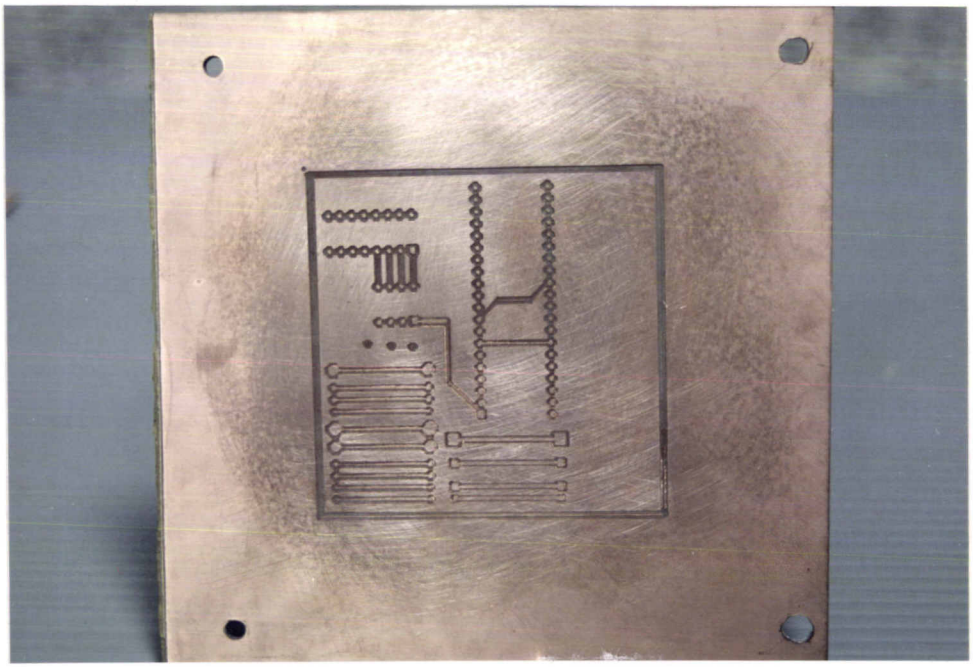
1.จากการทดลองพบว่าการติดตั้งแผ่น PCB ลงบนแท่นเพื่อปฏิบัติงานหากแผ่น PCB ที่ติดตั้งไม่ได้ระดับจะทำให้ความลึกของการกัดเซาะไม่คงที่อาจทำให้ผิวทองแดงที่เคลือบอยู่ไม่ขาดออกจากกัน และระดับความลึกของการกัดเซาะยังมีผลกับขนาดของเส้นบนชิ้นงาน ดังแสดงตามรูปที่ 7.1 จะเห็นได้ว่าด้านซ้ายขนาดของเส้นที่ลากจะมีขนาดใหญ่กว่าเส้นด้านขวา ซึ่งเกิดจากความลึกของการกัดเซาะไม่เท่ากัน และทำให้ลายเส้นด้านซ้ายมีความละเอียดมากกว่าด้านขวาโดยดูจากเส้นขอบทางขวาลายทองแดงระหว่างเส้นที่เป็นขอบรูปถูกกัดออกจนหมดแต่ด้านซ้ายลายทองแดงระหว่างเส้นยังคงเหลืออยู่

2.ความเร็วของแท่นในการเคลื่อนที่ถ้ามากเกินไปจะทำให้เส้นที่ได้ไม่คมชัด และอาจทำให้หัวกัดเกิดความเสียหายได้

3.ความเร็วรอบของมอเตอร์ที่ใช้เมื่อติดตั้งส่วนจับหัวกัดแล้วลดลงจาก คุณสมบัติที่ระบุไว้ทำให้หัวกัดเกิดความเสียหายได้

4.ความร้อนที่เกิดขึ้นในการทำแผ่น PCB ทำให้หัวกัดเกิดความเสียหาย(หัวกัดที่ใช้ในโครงงานนี้ไม่ใช่หัวกัดแผ่น PCB โดยตรง)

แต่อย่างไรก็ตามโครงงานพิเศษนี้ประสบผลสำเร็จตามวัตถุประสงค์ที่ตั้งไว้ในระดับหนึ่ง กล่าวคือถ้าได้มีการแก้ปัญหาดังกล่าวได้คาดว่าจะได้ผลเป็นที่น่าพอใจ



รูปที่ 7.1 แสดงเส้นที่เกิดจากความลึกในการกัดเซาะไม่เท่ากัน

ภาคผนวก

ส่วนของโปรแกรม
Auto pcb.pas

```
Program Auto_PCB;

Uses Crt,Graph,Menu2,Thai2,Dos;

Const Header : String = 'PCB FILE 4';

Type  TrackPtr  = ^TrackType;
      TrackType = Record
                    x1,y1,x2,y2,width,Layer : Integer;
                    Next : TrackPtr;
                  End;
      HeadTrackType = Record
                        Num : Word;
                        First : TrackPtr;
                      End;

      PadPtr    = ^PadType;
      PadType   = Record
                    x,y,lx,ly,form,Layer : Integer;
                    Next : PadPtr;
                  End;
      HeadPadType = Record
                      Num : Word;
                      First : PadPtr;
                    End;

      ViaPtr    = ^ViaType;
      ViaType   = Record
                    x,y,Outside,Inside : Integer;
                    Next : ViaPtr;
                  End;
      HeadViaType = record
                      Num : Word;
                      First : ViaPtr;
                    End;

      FillPtr   = ^FillType;
      FillType  = Record
                    x1,y1,x2,y2,Layer : Integer;
                    Next : FillPtr;
                  End;
      HeadFillType = Record
                       Num : Word;
                       First : FillPtr;
                     End;

Var
   Fi       : Text;
   FileRead,Select: Byte;
   FileName  : String;
   Heap,Bitmap         : Pointer;
   TrackNode,TrackDat : TrackPtr;
   PadNode,PadDat      : PadPtr;
   ViaNode,ViaDat      : ViaPtr;
   FillNode,FillDat    : FillPtr;
   HeadTrack : HeadTrackType;     {Head Of Track-Link}
```

```
    HeadPad    : HeadPadType;        {Head Of Pad-Link}
    HeadVia    : HeadViaType;        {Head Of Via-Link}
    HeadFill   : HeadFillType;       {Head Of Fill-Link}
    Xmin,Ymin,Xmax,Ymax : Word; {Dimension Of PCB}
    XOfPlane,YOfPlane   : Word; {location of Top-Right For EaCh Plane}
    BeginXofPlane,
    BeginYofPlane       : Integer; {location of Bottom-Left of EaCh Plane}
    CurXPlane,CurYPlane : Byte; {Current X & Current Y Plane}
    NumXPlane,NumYPlane : Byte; {Number of X & Number of Y Plane}
    Divider,CheckMake   : Byte; { 1 = 0.025 um }
                                { .....       }
                                { .....       }
                                {10 = 0.25  um }
    ScreenWidth,Screendepth : Word;
    Menu   : MenuType;
    PCBFileFlag,TraceFlag,InitialFlag : Byte;
    MakeExit,Ch : Char;
    Size : Word;
    Behide,Old1b : Pointer;

{$i Trace.icl}
{$i Makepcb2.icl}
{$i Ptform2.icl}

Procedure InitAllHead;
Begin
  headTrack.Num := 0;
  headTrack.First := Nil;
  HeadPad.Num := 0;
  HeadPad.First := Nil;
  headVia.Num := 0;
  headVia.First := Nil;
  headFill.Num := 0;
  headFill.First := Nil;
End;

Procedure AddTrackNode(P1,P2,P3,P4,P5,P6 : Integer);
Begin
  Inc(HeadTrack.Num);
  New(TrackDat);
  With TrackDat^ Do
  Begin
    x1     := P1;
    y1     := P2;
    x2     := P3;
    y2     := P4;
    Width  := P5;
    Layer  := P6;
    If x1 < Xmin Then Xmin := x1;
    If y1 < Ymin Then Ymin := y1;
    if x2 > Xmax Then Xmax := x2;
    If y2 > Ymax Then Ymax := y2;
  End;
  If HeadTrack.First = nil Then HeadTrack.First := TrackDat Else
    TrackNode^.Next := TrackDat;
  TrackNode := TrackDat;
```

```
End;

Procedure AddPadNode(P1,P2,P3,P4,P5,P6 : Integer);
Begin
  Inc(HeadPad.Num);
  New(PadDat);
  With PadDat^ Do
  Begin
    x     := P1;
    y     := P2;
    Lx    := P3;
    Ly    := P4;
    form  := P5;
    Layer := P6;
    If x < Xmin Then Xmin := x Else If x > Xmax Then Xmax := x;
    If y < Ymin Then Ymin := y Else If y > Ymax Then Ymax := y;
  End;
  If HeadPad.First = nil Then HeadPad.first := PadDat
    Else PadNode^.Next := PadDat;
  PadNode := PadDat;
End;

Procedure AddViaNode(P1,P2,P3,P4 : Integer);
Begin
  Inc(HeadVia.Num);
  New(ViaDat);
  With ViaDat^ Do
  Begin
    x       := P1;
    y       := P2;
    OutSide:= P3;
    Inside  := P4;
    If x < Xmin Then Xmin := x Else If x > Xmax Then Xmax := x;
    If y < Ymin Then Ymin := y Else If y > Ymax Then Ymax := y;
  End;
  If HeadVia.First = nil Then  HeadVia.first := ViaDat
    Else ViaNode^.Next := ViaDat;
  ViaNode := ViaDat;
End;

Procedure AddFillNode(P1,P2,P3,P4,P5: Integer);
Begin
  Inc(HeadFill.Num);
  New(FillDat);
  With FillDat^ Do
  Begin
    x1    := P1;
    y1    := P2;
    x2    := P3;
    y2    := P4;
    Layer := P5;
    If x1 < Xmin Then Xmin := x1;
    If y1 < Ymin Then Ymin := y1;
    if x2 > Xmax Then Xmax := x2;
    If y2 > Ymax Then Ymax := y2;
  End;
```

```
  If HeadFill.first = nil Then headFill.first := FillDat
    Else FillNode^.Next := FillDat;
  FillNode := FillDat;
End;


Function FindParameter(NPara,command : Integer):Byte ; {Find N parameter}
Var                                                    {0 - NoError}
  Parameter : Array[1..10] Of Word;                    {1 - Error}
  Ch        : Char;
  i,j,n     : Byte;
  paraStr   : String;
  Code      : Integer;
Begin
  For i := 1 To NPara Do
  Begin
    n := 0;
    ParaStr :='';
    Repeat
      Read(fi,Ch);
      If ((Ch <> #10) And (Ch <> #32)) Then
      Begin
         ParaStr := ParaStr+Ch;
         inc(n);
      End;
    Until (Ch= #10) Or (Ch = #32);
    If Ch =#10 Then Delete(ParaStr,n,1);
    Val(ParaStr,Parameter[i],Code);
    If (Code <> 0) And (i <> 9) Then Begin
                        FindParameter := 1;
                        Exit;
                     End;

  End;
  Case Command Of
   {Track}   1  : AddTrackNode(Parameter[1],Parameter[2],Parameter[3],
                               Parameter[4],Parameter[5],Parameter[6]);
   {Pad}     2  : AddPadNode(Parameter[1],Parameter[2],Parameter[3],
                             Parameter[4],Parameter[5],Parameter[8]);
   {Via}     3  : AddViaNode(Parameter[1],Parameter[2],Parameter[3],
                             Parameter[4]);
   {Fill}    4  : AddFillNode(Parameter[1],Parameter[2],Parameter[3],
                              Parameter[4],Parameter[5]);
  End;
End;


Function ReadPCB(Finame : String) : Byte; {Read PCB File }
Var                                       { 0 - No Error }
     command : String;                    { 1 - Not ProtelFormat}
      error  : Byte;                      { 2 - File Not Found }
                                          { 3 - File Error }
Begin
  {$i-}
  Assign(Fi,Finame);
  Reset(fi);
```

```
  {$I+}
  If Ioresult <> 0 Then Begin ReadPCB := 2;Exit; End;
  Readln(Fi,Command);
  If Command <> Header Then Begin ReadPCB := 1;Close(fi);Exit;End;
  If PCBFileFlag = 1 Then
  Begin
    Release(Heap);
  End;
  Mark(Heap);
  TrackNode := NIL;PadNode   := NIL;ViaNode   := NIL;FillNode  := NIL;
  InitAllHead;
  Xmin := $FFFF;Xmax := $0;
  Ymin := $FFFF;Ymax := $0;
  While Not Eof(Fi) Do
  Begin
    Error := 0;
    Readln(fi,Command);
    If (Command ='FT') Or (Command = 'CT')
      Then Error := FindParameter(6,1) Else
    If (Command ='FP') Or (Command = 'CP')
      Then Error := FindParameter(9,2) Else
    If (Command ='FV') Then Error := FindParameter(4,3) Else
    If (Command ='FF') Then Error := FindParameter(5,4);
    If Error = 1 Then Begin ReadPCB := 3;Close(fi);Exit;End;
  End;
  Close(fi);
  If HeadTrack.Num <> 0 Then TrackNode^.Next := NIL;
  If HeadPad.Num   <> 0 Then PadNode^.Next   := NIL;
  If HeadVia.Num   <> 0 Then ViaNode^.Next   := NIL;
  If HeadFill.Num  <> 0 Then FillNode^.Next  := NIL;
  PCBFileFlag := 1;
  ReadPCB := 0;
End;

Procedure OpenGraph;
Var Gd,Gm : Integer;
Begin
  DetectGraph(Gd,Gm);
  Initgraph(Gd,Gm,'');
End;

Procedure DrawPCB(Layer : Byte);
Begin
   SetColor(15);
   SetFillStyle(1,15);
   TrackNode     := HeadTrack.First;
   PadNode       := HeadPad.First;
   ViaNode       := HeadVia.First;
   FillNode      := HeadFill.First;

   While (TrackNode <> Nil) Do
     Begin
       If TrackNode^.Layer = Layer Then Track(TrackNode^.x1,TrackNode^.y1,
                                              TrackNode^.x2,TrackNode^.y2,
                                               TrackNode^.Width);
       TrackNode := TrackNode^.Next
```

```
    End;
  While (FillNode <> Nil) Do
    Begin
      If FillNode^.Layer = Layer Then Fill(FillNode^.x1,Fillnode^.y1,
                                           FillNode^.x2,Fillnode^.y2);
      Fillnode := Fillnode^.Next;
    End;

  While(PadNode <> Nil) Do
    Begin
      Pad(PadNode^.x,PadNode^.y,
          PadNode^.lx,PadNode^.ly,PadNode^.form);
      PadNode := PadNode^.Next;
    End;
  While(ViaNode <> Nil) Do
    Begin
      Via(Vianode^.x,Vianode^.y,vianode^.Outside,Vianode^.inside);
      ViaNode := Vianode^.Next;
    End;

End;


Procedure ShowDimension;
Var
  XminStr,YminStr,XmaxStr,YmaxStr,WidthStr,HightStr : String;
  Size : Word;
  Behide : Pointer;
  Ch : Char;
Begin
  If PCBFileFlag = 1 Then
  Begin
    Str(Xmin:4,XminStr);
    Str(Ymin:4,YminStr);
    Str(Xmax:4,XmaxStr);
    Str(Ymax:4,YmaxStr);
    Str((Xmax - Xmin)/1000:3:2,WidthStr);
    Str((Ymax-Ymin)/1000:3:2,hightStr);
    size := Imagesize(120,20,500,120);
    Getmem(Behide,size);
    GetImage(120,20,500,120,Behide^);
    BoxUp(120,20,500,120,7);
    BoxDown(125,25,495,115,7);
    SetTextjustify(centertext,centertext);
    OutTextXy(310,35,'Xmin = '+Xminstr+'  Xmax = '+XmaxStr);
    OutTextXY(310,55,'Ymin = '+YminStr+'  Ymax = '+YmaxStr);
    OutTextxy(310,75,'Approximate Size = '+WidthStr+' X '+hightStr+
              ' InCh(s)');
    Ch := ShowMassage(255,90,365,110,7,'Press Any Key','');
    PutImage(120,20,Behide^,normalput);
    Freemem(Behide,size);
  End Else Ch := ShowMassage(120,20,300,50,7,'Load File First !','');
End;

Procedure ShowHelp;
Var Behidge : Pointer;
```

```
    size : Word;
    Pindex,res : Byte;

Procedure Draw;
Begin
  SetFillStyle(1,10);Bar(300,230,310,240);Bar(260,260,270,270);
  Bar(426,288,436,298);
  SetFillStyle(1,1);Bar(275,245,345,300);
  SetFillStyle(8,yellow);Bar(290,260,330,285);
  SetColor(15);
  Line(400,230,400,340);Line(260,340,400,340);
  SetColor(0);
  Line(290,315,332,315);Line(345,315,400,315);
  Line(290,310,290,320);Line(332,310,332,320);Line(400,310,400,320);
  Line(345,310,345,320);
  Line(245,260,245,290);Line(245,300,245,340);
  Line(240,260,250,260);Line(240,290,250,290);Line(240,340,250,340);
  Line(240,300,250,300);
  OutThaiXY(225,265,'A',11);OutThaiXY(227,310,'C',11);
  OutThaiXY(306,320,'B',11);OutThaiXY(370,320,'D',11);
  SetColor(green);Line(325,273,422,273);Line(215,293,291,293);
  SetFillStyle(1,11);Bar(430,310,530,340);
  OutThaiXY(435,315,'C>A ¾Ã¹ D>B',0);
End;

Function ShowPage(page:Byte):Byte;{0-Ready,1-no help file,2-quit}
Var helpfi : text;
    i    : Byte;
    ShowStr,pagestr: String;
    yy      : Integer;
    Ch     : Char;
Begin
  Assign(helpfi,'project.hlp');
  {$i-}
  reset(helpfi);
  {$i+}
  If IOResult = 0 Then
  Begin
    str(page,pagestr);
    Repeat
    Readln(Helpfi,showstr);
    Until showStr = 'P'+pagestr;
    yy := 0;
    BoxDown(105,105,535,375,7);
    For i := 1 To 11 Do
    Begin
      Readln(Helpfi,showstr);
      If ShowStr = 'DRAW' Then Draw Else OutThaixy(130,120+yy,showStr,11);
      yy := yy +20;
    End;
    ShowPage := 0;
    close(helpfi);
    Ch := ShowMassage(260,350,380,370,7,'Press Any Key','');
    if Ch = #27 Then showpage := 2;
  End Else
  Begin
```

```
    Ch := ShowMassage(260,350,380,370,7,'No Help File','');
    ShowPage := 1;
  End;
End;


Begin
    pindex := 0;
    size := Imagesize(100,100,540,380);
    GetMem(Behide,Size);
    GetImage(100,100,540,380,Behide^);
    BoxUp(100,100,540,380,7);
    BoxDown(105,105,535,375,7);
    Repeat
      inc(pindex);
      res := showpage(pindex);
    Until (res = 2) or (res = 1) or (pindex = 2);
    PutImage(100,100,Behide^,normalput);
    Freemem(Behide,size);
End;

Procedure DoTraceItem;
Begin
  If PCBFileFlag = 1 Then
  Begin
    RestoreScr(Menu);
    DoTrace(0,0,ScreenWidth-1,ScreenDepth-1,0);
    TraceFlag := 1;
  End Else Ch := ShowMassage(120,20,300,50,7,'Load File First !','');
End;

Procedure DoInitial;
Var Size : Word;
    Behide : Pointer;
    key   : Char;
Begin
  If TraceFlag = 1 Then
  Begin
    size := Imagesize(100,100,540,380);
    GetMem(Behide,Size);
    GetImage(100,100,540,380,Behide^);
    BoxUp(100,100,540,380,7);
    BoxDown(105,105,535,375,7);
    OutThaixy(130,120,'                    ขะภ»ปเปา',11);
    OutThaixy(130,140,'  - +ฒะฆๆฤฆ "ศ^กหลขๆะป" .ภ'ฆภ.ศ'ฆภป»นฆปนโฉ»ปส +Y
            11);
    OutThaixy(130,160,'  - +ฒะฆๆฤฆ "ศ^กหลศฉ" .ภ'ฆภ.ศ'ฆภป»นฆปนโฉ»ปส -Y',
            11);
    OutThaixy(130,180,'  - +ฒะฆๆฤฆ "ศ^กหลขสโ" .ภ'ฆภ.ศ'ฆภป»นฆปนโฉ»ปส +X'.
            11);
    OutThaixy(130,200,'  - +ฒะฆๆฤฆ "ศ^กหลญะโฤ" .ภ'ฆภ.ศ'ฆภป»นฆปนโฉ»ปส -X
            11);
    OutThaixy(130,220,'  - +ฒะฆๆฤฆ "Page Up" .ภ'ฆภ.ศ'ฆภปฮแสกแฤขๆะป',11)
    OutThaixy(130,240,'  - +ฒะฆๆฤฆ "Page Down" .ภ'ฆภ.ศ'ฆภปฮะสกแฤศฉ',11)
    OutThaixy(130,260,'  - +ฒะฆๆฤฆ "Enter" .ภ'ฆภทกศฉลแผฆฆโทา»ฮปฆฉทฆโฉ ะ
            11);
```

```
Repeat
Key := Readkey;If key = #0 Then key := Readkey;
  Case key Of
    #72  : Begin
             For stt := 1 To 400 Do
             Begin
               Port[Pa] := Data_out1[IndexMotor1] Or
               Data_out2[IndexMotor2];
               Update(IndexMotor1,1);
               Delay(1);
             End;
           End;
    #80  : Begin
             For stt := 1 To 400 Do
             Begin
               Port[PA]:= Data_out1[IndexMotor1]
               Or Data_out2[IndexMotor2];;
               Update(IndexMotor1,0);
               Delay(1);
             End;
           End;
    #75  : Begin
             For stt := 1 To 400 Do
             Begin
               Port[PA] := Data_out2[IndexMotor2]
               Or Data_out1[IndexMotor1];
               Update(IndexMotor2,1);
               DElay(1);
             End;
           End;
    #77  : Begin
             For stt := 1 To 400 Do
             Begin
               Port[PA] := Data_out2[IndexMotor2]
               Or Data_out1[IndexMotor1];
               Update(IndexMotor2,0);
               Delay(1);
             End;
           End;
    #81  : Begin
             For stt := 1 To 5 Do
             Begin
               Port[Pb] := Data_out1[IndexMotor3];
               Update(IndexMotor3,1);
               DELAY(7);
             End;
           End;
    #73  : Begin
             For stt := 1 To 5 Do
             Begin
               Port[Pb] := Data_out1[IndexMotor3];
               Update(IndexMotor3,0);
               DELAY(7);
             End;
           End;
  End;
```

```
    Until key = #13;
    PutImage(100,100,Behide^,normalput);
    Freemem(Behide,size);
    Initialflag := 1;
  End Else Ch := ShowMassage(120,20,300,50,7,'Trace First !','');
End;

Procedure DoRun;
Var Result : Byte;
Begin
  If InitialFlag = 1 Then
  Begin
    Restorescr(menu);
    Result := MakePCB;
    If Result <> 0 Then Ch := ShowMassage(50,20,350,50,7,
                        'criticla error or interrupt by user !','');
    If PCBFileflag = 1 Then
    Begin
      Release(heap);
      PCBFileflag := 0;
    End;
    TraceFlag := 0;
    InitialFlag := 0;
  End Else Ch := ShowMassage(120,20,300,50,7,'Initial First !','');

End;

{$F+,S-,W-}
Procedure New1b;Interrupt;
Begin
  critical := true;
End;
{$F-,S+}



Begin
  {***Initial Value***}
  GetIntVec($1b,Old1b);
  SetIntvec($1b,Addr(new1b));
  Select := loadfont('Normal.fon');
  If Select <> 0 Then Begin
                        Writeln('Thai font Not Found!');
                        Halt;
                      End;
  PCBFileFlag := 0;TraceFlag := 0;InitialFlag := 0;
  Xmin := $ffff;Ymin := $ffff;
  Xmax := 0;Ymax := 0;
  Divider :=  10;
  port[pcc] := $89;
  IndexMotor1 := 1;IndexMotor2 := 1;IndexMotor3 := 1;
  AddMenuItem(Menu,1,'File');
  AddMenuitem(Menu,2,'Trace');
  AddMenuItem(Menu,3,'Initial');
  AddMenuItem(Menu,4,'Run');
  AddMenuItem(Menu,5,'Dimension');
```

```
AddMenuItem(Menu,5,'Help');
AddMenuItem(Menu,7,'Exit');
InitMenu(Menu,20,20,7);
OpenGraph;
ScreenWidth  := GetMaxX + 1;
ScreenDepth  := GetMaxY + 1;
Repeat
  Select := DoMenu(Menu,7);
  Case Select Of
      1{File}      : Begin
                       size := Imagesize(120,20,300,60);
                       Getmem(Behide,size);
                       Getimage(120,20,300,60,Behide^);
                       WriteinboxUp(120,20,300,60,7,190,35,
                       'Enter File name');
                       EnterString(130,45,18,11,filename);
                       PutImage(120,20,Behide^,normalput);
                       Freemem(Behide,size);
                       If filename <> '' Then
                       Begin
                         filename := Showload(120,20,filename);
                         If filename = 'FILENOTFOUND' Then
                           Ch := ShowMassage(120,20,300,50,7,
                           'File Not Found!','') Else
                         If Filename = 'PATHNOTFOUND' Then
                           Ch := ShowMassage(120,20,300,50,7,
                           'Path Not Found!','') Else
                         If filename = 'ESCAPE' Then Else
                         Begin
                           FileRead := ReadPCB(Filename);
                           Case FileRead of
                              1 :   Ch := ShowMassage(120,20,300,50,7,
                                    'Not Protel Format!','');
                              3 :   Ch := ShowMassage(120,20,300,50,7,
                                    'Data Error!','');
                              0 :   Begin
                                     RestoreScr(Menu);
                                     Cleardevice;
                                     DrawPCB(6);
                                    End;
                           End;
                         End;
                       End;
                     End;
      2{Trace}     : Begin
                       DoTraceItem;
                     End;
      3{Initial}   : Begin
                       DoInitial;
                     End;
      4{Run}       : Begin
                       DoRun;
                     End;
      5{Dimension}: ShowDimension;
      6{help}      : ShowHelp;
```

```
      7,0{Exit}    : Begin
                       MakeExit := ShowMassage(120,20,300,50,7,
                       'Are you sure ! (Y/N)','YN');
                     End;
    End;
  Until MakeExit = 'Y';
  If PCBFileflag = 1 Then Release(heap);
  CloseGraph;
  SetIntVec($1b,Old1b);
  Port[Pa] := $00;
  Port[pb] := $00;
End.
```

ส่วนของโปรแกรม

Make pcb2.icl

```
TYPE    AR_DATA = Array[1..8] Of Byte;
        Point   = Record
                    x : integer;
                    y : integer;
                  End;

CONST   PA = $0300;
        PB = $0301;
        PC = $0302;
        PCC = $0303;

        Data_out1 : Ar_Data = ($01,$03,$02,$06,$04,$0c,$08,$09);
        Data_out2 : Ar_Data = ($10,$30,$20,$60,$40,$c0,$80,$90);

Const  MaxX = 639;
       MaxY = 479;

Var Gd,gm,i :Integer;
    out,Index,di,offset:Byte;
    st,stt   : integer;
    indexMotor1,indexMotor2,indexMotor3 : Integer;
    checktable   : Byte;
    key : char;
    Dx,dy,loop : Integer;
    curxpos,curypos : Integer;
    Bfont : Integer;
    Oldm : point;
    critical : Boolean;

Procedure Update(Var index:Integer;mode:Byte);
Begin
  If Mode = 1 Then
  Begin
    inc(index);
    If index > 8 Then index := 1;
  End
  Else
  Begin
    Dec(index);
    If index < 1 Then index := 8;
  End;
End;

procedure Penup;
Var i : Byte;
Begin
  For i := 1 To 50 Do
  Begin
    Port[Pb] := Data_out1[IndexMotor3];
    Update(IndexMotor3,0);
    DELAY(7);
  End;
End;

Procedure PenDown;
```

```
Var i : Byte;
Begin
  For i := 1 To 50 Do
  Begin
    Port[Pb] := Data_out1[IndexMotor3];
    Update(IndexMotor3,1);
    DELAY(7);
  End;
  Delay(1500);
End;



Procedure  Outtable(m : Point);
Var
  i : Integer;
  D_out : Byte;
Begin
  If (m.y = 1) And (m.x = 0) Then {dir =1 }
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
      port[PA] := D_out;
      update(indexmotor1,0);
      delay(3);
    End;
  End Else
  If (m.y = 1) And (m.x = 1) then  {dir =2}
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
      port[PA] := D_out;
      update(indexmotor1,0);
      update(indexmotor2,1);
      delay(3);
    End;
  End Else
  If (m.y = 0) And (m.x = 1) then  {dir =3}
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
      port[PA] := D_out;
      update(indexmotor2,1);
      delay(3);
    End;
  End Else
  If (m.y = -1) And (m.x = 1) then  {dir =4}
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
```

```
      port[PA] := D_out;
      update(indexmotor1,1);
      update(indexmotor2,1);
      delay(3);
    End;
  End Else
  If (m.y = -1) And (m.x = 0) then  {dir = 5}
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
      port[PA] := D_out;
      update(indexmotor1,1);
      delay(3);
    End;
  End Else
  If (m.y = -1) And (m.x = -1) then  {dir = 6}
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
      port[PA] := D_out;
      update(indexmotor1,1);
      update(indexmotor2,0);
      delay(3);
    End;
  End Else
  If (m.y = 0) And (m.x = -1) then  {dir = 7}
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
      port[PA] := D_out;
      update(indexmotor2,0);
      delay(3);
    End;
  End Else
  If (m.y = 1) And (m.x = -1) then  {dir = 8}
  Begin
    For i := 1 To 5 Do
    Begin
      D_out := Data_out1[indexmotor1] or Data_out2[indexmotor2];
      port[PA] := D_out;
      update(indexmotor1,0);
      update(indexmotor2,0);
      delay(3);
    End;
  End;
  Oldm := m;
End;



Function   Outline(x1,y1,x2,y2 : Integer):Byte; { 0 = Ready }
                                                {other = Erroe}
```

```
Var
  Step,Error,dx,dy,temp : integer;
  mv1,mv2          : point;
  Endoutline : Boolean;
  Check : Byte;
  Label  Stop;
Begin
  mv1.x := 1;mv1.y := 1;mv2 := mv1;
  y1 := (479-y1) * 10;y2 := (479-y2) * 10;    { Change Y Coordinate}
  x1 := x1 * 10;x2 := x2 * 10;
  dx := x2-x1;
  dy := y2-y1;
  If (dx < 0) Then Begin
                     mv1.x := -1;
                     mv2   := mv1;
                     dx := -1 *dx;
                   End;
  If (dy < 0) Then Begin
                     mv1.y := -1;
                     mv2   := mv1;
                     dy := -1 *dy;
                   End;
  If (dx >= dy) Then mv2.y := 0 Else Begin
                                       mv2.x := 0;
                                       Temp  := dx;
                                       dx    := dy;
                                       dy    := Temp;
                                     End;
  Step := dx + dy;
  error := -1*dx;
  dx   := 2 * dx;
  dy   := 2 * dy;
  Endoutline := False;
  critical  := False;
  {Putpixel(x1,y1,15);}
  Repeat
    If Step = 0 Then Endoutline := True Else
    Begin
      error := error + dy;
      if error > 0 Then
      Begin
        x1 := x1 + Mv1.x;
        y1 := y1 + mv1.y;
        {putpixel(x1,y1,15);}
        outtable(mv1);
        error := error -dx;
        step := Step -2;
      End Else
      Begin
      x1 := x1 + Mv2.x;
      y1 := y1 + mv2.y;
      {putpixel(x1,y1,15);}
      outtable(mv2);
      step := Step -1;
      End;
    End;
```

```
  Check := Port[Pc];
  If Check <> $F0 Then critical := True;
  Until Endoutline Or critical;
  If critical = True Then outline := Check Else outline := 0;
End;


Function GetDir(ax,ay:integer):Byte;
{Retrun Direction series point}
{          8   1   2           }
{            \ ¦ /             }
{          7 - x - 3           }
{            / ¦ \             }
{          6   5   4           }
{  0 = no series point          }
Var
  N,S,E,W : Boolean;
  aN,aS,aE,aW : Integer;
Begin
  aN := ay + 1;
  aS := ay - 1;
  aE := ax + 1;
  aW := ax - 1;
  If ( aN <= MaxY) Then N := True Else N := False;
  If ( aS >= 0)          Then S := True Else S := False;
  If ( aE <= MaxX)Then E := True Else E := False;
  If ( aW >= 0)          Then W := True Else W := False;

  If  ((GetPixel(ax,an) = 2) And N)        Then GetDir := 1 Else
  If (((GetPixel(ae,an) = 2) And N) And E) Then GetDir := 2 Else
  If  ((GetPixel(ae,ay) = 2) And E)        Then GetDir := 3 Else
  If (((GetPixel(ae,as) = 2) And E) And S) Then GetDir := 4 Else
  If  ((GetPixel(ax,as) = 2) And S)        Then GetDir := 5 Else
  If (((GetPixel(aw,as) = 2) And W) And S) Then GetDir := 6 Else
  If  ((GetPixel(aw,ay) = 2) And W)        Then GetDir := 7 Else
  If (((GetPixel(aw,an) = 2) And W) And N) Then GetDir := 8 Else
  GetDir := 0;
End;


Procedure NewPoint(Var px,py : integer;Dir : Byte);
Begin
  Case Dir Of
    1 : Inc(Py);
    2 : Begin
          Inc(px);inc(py);
        End;
    3 : inc(px);
    4 : Begin
          Inc(px);Dec(py);
        End;
    5 : Dec(py);
    6 : Begin
          Dec(px);Dec(py);
        End;
    7 : dec(px);
```

```
    8 : Begin
          Dec(px);
          inc(py);
        End;
  End;
End;
```

```
Procedure Findline(fx1,fy1: Integer;Var fx2,fy2 : Integer);
Var
   Tempfx,Tempfy,num : Integer;
   Endline : Boolean;
   Dir,OldDir      : Byte;

Begin
  Tempfx := fx1;
  Tempfy := fy1;
  Endline := False;
  num := 1;
  Repeat
    Dir := GetDir(Tempfx,TempFy);
    If Dir <> 0 Then
    Begin
      inc(num);
      If num > 2 Then
      Begin
        If Dir = OldDir Then
        Begin
          OldDir := Dir;
          Putpixel(TempFx,TempFy,0);
          NewPoint(TempFx,TempFy,Dir);
      End Else
      Begin
        EndLine := True;

        End;
      End Else
      Begin
        OldDir := Dir;
        {PutPixel(TempFx,TempFy,0);}
        NewPoint(TempFx,TempFy,Dir);
      End;
    End Else EndLine := True;
  Until EndLine;
  PutPixel(TempFx,TempFy,0);
  fx2 := Tempfx;
  fy2 := Tempfy;
End;
```

```
Function MakePCB:Byte;
Var x,y,x2,y2 : Integer;
    xloop,yloop : integer;
    EndSeries,critical  : Boolean;
    Test,Lineclose      : Byte;
    label Error;

Begin
  curxpos := 0;curypos := 479;
  PenUp;
  For yloop := MaxY Downto 0 Do
  Begin
    For xloop := 0 To MaxX Do
    Begin
      If GetPixel(xloop,yloop) = 2 Then
      Begin
        x := xloop;y := yloop;
        EndSeries  := False;
        critical  := false;
        CheckTable := outline(curxpos,curypos,x,y);
        If CheckTable <> 0 Then
        Begin
          critical := true;
          Goto Error;
        End;
        PenDown;
        Repeat
          Delay(200);
          findline(x,y,x2,y2);
          checktable := outline(x,y,x2,y2);
          If CheckTable <> 0 Then
          Begin
            critical := True;
            xloop := maxx;
            yloop := maxy;
          End;
          Test := GetDir(x2,y2);
          If Test = 0 Then EndSeries := True
          Else
          Begin
            x := x2;y := y2;
          End;
        Until EndSeries Or critical;
        PenUP;
        curxpos := x2;
        curypos := y2;
      End;
    End;
  End;
  Error:If critical Then Makepcb := 1 Else makepcb := 0;
End;
```

ส่วนของโปรแกรม

Trace.icl

```
Procedure doTrace(x1,y1,x2,y2:Word;Mode:Byte);
Var tx,ty,tc : Integer;
    TD  : Byte;
    c1,c2 : Byte;

Function TestAround(x,y:Integer):Byte;
Var top,left,bottom,right:integer;
Begin
  top := y-1;left := x-1;bottom := y+1;right := x+1;
  If (Top  >= 0)    And (GetPixel(x,y-1) = c2) Then TestARound := 1 Els
  If (Left >= 0)    And (GetPixel(x-1,y) = c2) Then TestAround := 1 Els
  If (Bottom <= 479)And (GetPixel(x,y+1) = c2) Then TestARound := 1 Els
  If (right  <= 639)And (GetPixel(x+1,y) = c2) Then TestAround := 1 Els
  TestAround := 0;
End;

Begin
  Case mode Of
    0 : Begin {detect Outside}
          c1 := 0;
          c2 := 15;
        End;
    1 : Begin {detect Inside}
          c1 := 15;
          c2 := 0;
        End;
  End;
  For ty := y1 To y2 Do
    For tx := x1 To x2 Do
    Begin
      IF GetPixel(tx,ty) = c1 Then
      Begin
        tD := TestAround(tx,ty);
        If td <> 0 {= 1} Then PutPixel(tx,ty,Green);
      End;
    End;
  For ty := 0 To GetMaxy Do
    For tx := 0 To GetMaxX Do
    Begin
      tc := GetPixel(tx,ty);
      If tc = 15 Then PutPixel(tx,ty,0);
    End;
End;
```

ส่วนของโปรแกรม

Ptform2.icl

```
Procedure ChangeY(Var Y : Integer);
Begin
  {Y := (YofPlane -Y) Div Divider;}
  Y := ScreenDepth -(Y Div Divider);
End;

Procedure ChangeX(Var X : Integer);
Begin
  {X := (X-BeginXofPlane)  Div Divider;}
  x := ((Xmax - X) + Xmin) Div Divider;
End;


Function Angel(Slop : Real):Real;
Begin
  If Slop <> 2.0 Then  Angel := ArcTan(Slop)*180/Pi
  Else Angel := 90;
End;

Procedure Track(x1,y1,x2,y2:Integer;Width:integer);
Var
  ac : Integer;
  slop,Ang,Accos,Acsin: Real;
  FT : Array[1..4] Of PointType;
Begin
  Ac := Trunc((Width / 2) / Divider);
  If (x2-x1) <> 0 Then   Slop := (y2-y1)/(x2-x1) Else Slop := 2.0;
  Ang := 90-Angel(Slop);
  ChangeY(y1);
  ChangeY(y2);
  ChangeX(x1);
  Changex(x2);
  Accos := Ac*Cos(Ang);
  AcSin := Ac*sin(Ang);


  FT[1].x := Trunc((X1+AcCos));
  FT[1].y := Trunc((y1-AcSin));

  FT[2].x := Trunc((x2+AcCos));
  Ft[2].y := Trunc((y2-AcSin));

  FT[3].X := Trunc((x2-AcCos));
  FT[3].y := Trunc((y2+AcSin));

  FT[4].X := Trunc((x1-AcCos));
  Ft[4].y := Trunc((y1+AcSin));

  FillPoly(4,FT);
  FillEllipse(x1,y1,Ac,Ac);FillEllipse(x2,y2,Ac,Ac);
End;


Procedure Via(x,y,outside,inside : Integer);
Var OutS,InS : Integer;
```

```
Begin
  changeY(y);
  ChangeX(X);
  OutS := Trunc((OutSide/2) / Divider);
  InS  := Trunc((InSide/2)  Divider);
  FillEllipse(x,y,outs,outs);
  SetfillStyle(1,0);
  FillEllipse(x,y,ins,ins);
  SetfillStyle(1,15);
End;

Procedure Fill(x1,y1,x2,y2:Integer);
Begin
  ChangeY(y1);
  ChangeY(y2);
  ChangeX(x1);
  Changex(x2);
  Bar(x1,y1,x2,y2);
End;

Procedure Pad(x,y,lx,ly,form:Integer);{ No TagetType And Moire Type}

Var Rx,Ry,xx : Integer;
    Oct : Array[1..8] Of PointType;
Begin
  changeY(Y);
  ChangeX(X);
  Rx := Trunc((lx/2) / Divider);Ry := Trunc((ly /2)/divider);
  Case form Of
        1 : FillEllipse(x,y,rx,ry);
        2 : Bar(x-rx,y-ry,x+rx,y+ry);
        3 : Begin {OctType}
             xx := Trunc((lx / Divider) / 4);
             Oct[1].x := x - xx;
             Oct[1].y := y - ry;

             Oct[2].x := x + xx;
             Oct[2].y := y - rx;

             Oct[3].x := x + rx;
             Oct[3].y := y - xx;

             Oct[4].x := x + rx;
             Oct[4].y := y + xx;

             Oct[5].x := x + xx;
             Oct[5].y := y + rx;

             Oct[6].x := x - xx;
             Oct[6].y := y + rx;

             Oct[7].x := x - rx;
             Oct[7].y := y + xx;

             Oct[8].x := x - rx;
             Oct[8].y := y - xx;
```

```
            Fillpoly(8,Oct);
           End;
       4 : Begin
            xx := rx-ry;
            Bar(x-xx,y-ry,x+xx,y+ry);
            pieslice(x-xx,y,90,270,ry);
            pieslice(x+xx,y,0,360,ry);
           End;
  End;
End;
```

บอลสกรู(Ball screw)

หัวกัดเอ็นมิลล์

ดาตาชีท(Data sheet)

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

PNP
2N5875, 2N5876*
NPN
2N5877, 2N5878*

*Motorola preferred device

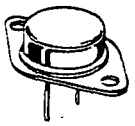
COMPLEMENTARY SILICON HIGH-POWER TRANSISTORS

. . . designed for general-purpose power amplifier and switching applications.

- Low Collector-Emitter Saturation Voltage –
  $V_{CE(sat)}$ = 1.0 Vdc (Max) @ I_C = 5.0 Adc
- Low Leakage Current –
  I_{CEX} = 0.5 mAdc (Max) @ Rated Voltage
- Excellent DC Current Gain –
  h_{FE} = 20 (Min) @ I_C = 4.0 Adc
- High Current Gain – Bandwidth Product –
  f_T = 4.0 MHz (Min) @ I_C = 0.5 A

10 AMPERE
COMPLEMENTARY SILICON
POWER TRANSISTORS

60-80 VOLTS
150 WATTS



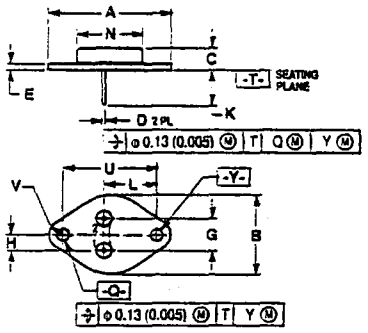
MAXIMUM RATINGS (1)

Rating	Symbol	2N5875 2N5877	2N5876 2N5878	Unit
Collector-Emitter Voltage	V_{CEO}	60	80	Vdc
Collector-Base Voltage	V_{CB}	60	80	Vdc
Emitter-Base Voltage	V_{EB}	5.0		Vdc
Collector Current – Continuous Peak	I_C	10 20		Adc
Base Current	I_B	4.0		Adc
Total Device Dissipation @ T_C = 25°C Derate above 25°C	P_D	150 0.857		Watts W/°C
Operating and Storage Junction Temperature Range	T_J, T_{stg}	-65 to +200		°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Case	θ_{JC}	1.17	°C/W

(1) Indicates JEDEC Registered Data



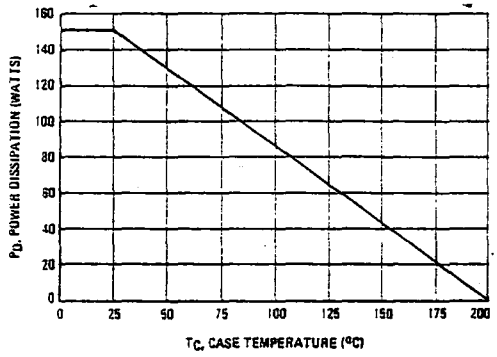
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. ALL RULES AND NOTES ASSOCIATED WITH REFERENCED TO-204AA OUTLINE SHALL APPLY.
4. 001-05 AND -06 OBSOLETE, NEW STANDARD 301-07.

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	39.37 REF		1.550 REF	
B	—	26.67	—	1.050
C	5.35	5.51	0.250	0.335
D	0.97	1.09	0.038	0.043
E	1.40	1.77	0.055	0.070
G	10.92 BSC		0.430 BSC	
H	5.46 BSC		0.215 BSC	
K	11.18	12.19	0.440	0.480
L	16.89 BSC		0.665 BSC	
N	—	21.08	—	0.830
Q	3.84	4.19	0.151	0.165
U	30.15 BSC		1.187 BSC	
V	3.33	4.77	0.131	0.188

STYLE 1:
PIN 1. BASE
2. EMITTER
CASE: COLLECTOR

CASE 1-07
TO-204AA
(TO-3)

FIGURE 1 – POWER DERATING



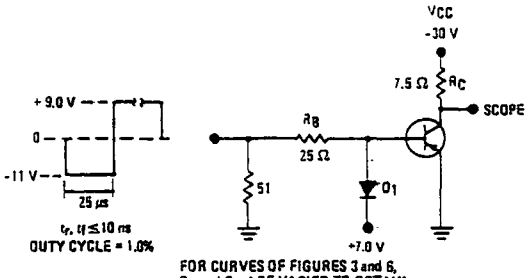
3

*ELECTRICAL CHARACTERISTICS (T_C = 25°C unless otherwise noted)

Characteristic		Symbol	Min	Max	Unit
OFF CHARACTERISTICS					
Collector-Emitter Sustaining Voltage (1) (I_C = 200 mAdc, I_B = 0)	2N5875, 2N5877	$V_{CEO(sus)}$	60	–	Vdc
	2N5876, 2N5878		80	–	
Collector Cutoff Current (V_{CE} = 30 Vdc, I_B = 0)	2N5875, 2N5877	I_{CEO}	–	1.0	mAdc
(V_{CE} = 40 Vdc, I_B = 0)	2N5876, 2N5878		–	1.0	
Collector Cutoff Current (V_{CE} = 60 Vdc, $V_{BE(off)}$ = 1.5 Vdc)	2N5875, 2N5877	I_{CEX}	–	0.5	mAdc
(V_{CE} = 80 Vdc, $V_{BE(off)}$ = 1.5 Vdc)	2N5876, 2N5878		–	0.5	
(V_{CE} = 60 Vdc, $V_{BE(off)}$ = 1.5 Vdc, T_C = 150°C)	2N5875, 2N5877		–	5.0	
(V_{CE} = 80 Vdc, $V_{BE(off)}$ = 1.5 Vdc, T_C = 150°C)	2N5876, 2N5878		–	5.0	
Collector Cutoff Current (V_{CB} = 60 Vdc, I_E = 0)	2N5875, 2N5877	I_{CBO}	–	0.5	mAdc
(V_{CB} = 80 Vdc, I_E = 0)	2N5876, 2N5878		–	0.5	
Emitter Cutoff Current (V_{EB} = 5.0 Vdc, I_E = 0)		I_{EBO}	–	1.0	mAdc
ON CHARACTERISTICS					
DC Current Gain (1) (I_C = 1.0 Adc, V_{CE} = 4.0 Vdc)		h_{FE}	35	–	–
(I_C = 4.0 Adc, V_{CE} = 4.0 Vdc)			20	100	
(I_C = 10 Adc, V_{CE} = 4.0 Vdc)			4.0	–	
Collector-Emitter Saturation Voltage (1) (I_C = 5.0 Adc, I_B = 0.5 Adc)		$V_{CE(sat)}$	–	1.0	Vdc
(I_C = 10 Adc, I_B = 2.5 Adc)			–	3.0	
Base-Emitter Saturation Voltage (1) (I_C = 10 Adc, I_B = 2.5 Adc)		$V_{BE(sat)}$	–	2.5	Vdc
Base-Emitter On Voltage (1) (I_C = 4.0 Adc, V_{CE} = 4.0 Vdc)		$V_{BE(on)}$	–	1.5	Vdc
DYNAMIC CHARACTERISTICS					
Current-Gain – Bandwidth Product (2) (I_C = 0.5 Adc, V_{CE} = 10 Vdc, f_{test} = 1.0 MHz)		f_T	4.0	–	MHz
Output Capacitance (V_{CB} = 10 Vdc, I_E = 0, f = 1.0 MHz)	2N5875, 2N5876	C_{ob}	–	500	pF
	2N5877, 2N5878		–	300	
Small-Signal Current Gain (I_C = 1.0 Adc, V_{CE} = 4.0 Vdc, f = 1.0 kHz)		h_{fe}	20	–	–
SWITCHING CHARACTERISTICS					
Rise Time	(V_{CC} = 30 Vdc, I_C = 4.0 Adc, I_{B1} = I_{B2} = 0.4 Adc, See Figure 2)	t_r	–	0.7	μs
Storage Time		t_s	–	1.0	μs
Fall Time		t_f	–	0.8	μs

*Indicates JEDEC Registered Data.
(1) Pulse Test: Pulse Width ≤ 300 μs, Duty Cycle ≤ 2.0%.
(2) f_T = $|h_{fe}| \cdot f_{test}$

FIGURE 2 – SWITCHING TIME TEST CIRCUIT



t_r, t_f ≤ 10 ns
DUTY CYCLE = 1.0%

FOR CURVES OF FIGURES 3 and 6, R_B and R_C ARE VARIED TO OBTAIN DESIRED CURRENT LEVELS

For NPN test circuit, reverse all polarities.

D_1 MUST BE FAST RECOVERY TYPE, eg:
1N5825 USED ABOVE I_B = 100 mA
MSD6100 USED BELOW I_B = 100 mA

FIGURE 3 – TURN-ON TIME

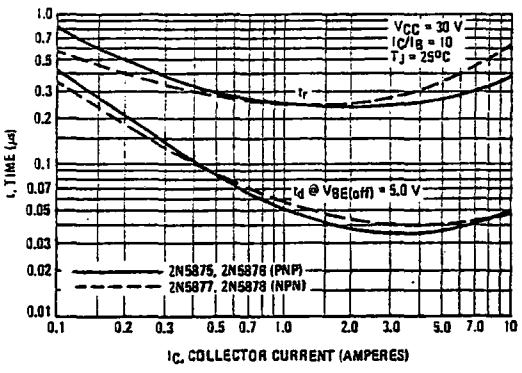


FIGURE 4 – THERMAL RESPONSE

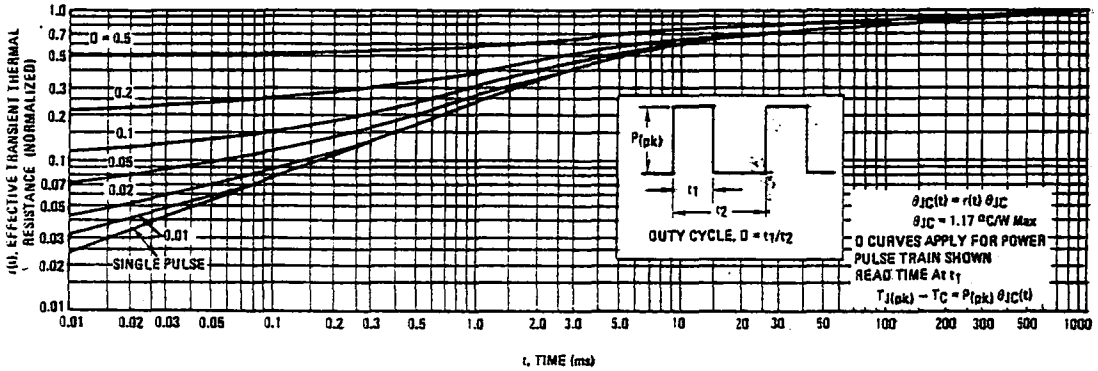
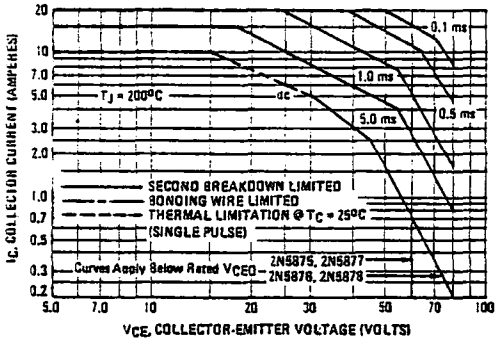


FIGURE 5 – ACTIVE REGION SAFE OPERATING AREA



There are two limitations on the power handling ability of a transistor: average junction temperature and second breakdown. Safe operating area curves indicate $I_C - V_{CE}$ limits of the transistor that must be observed for reliable operation, i.e., the transistor must not be subjected to greater dissipation than the curves indicate.

The data of Figure 5 is based on $T_{J(pk)} = 200^oC$; T_C is variable depending on conditions. Second breakdown pulse limits are valid for duty cycles to 10% provided $T_{J(pk)} \leq 200^oC$. $T_{J(pk)}$ may be calculated from the data in Figure 4. At high case temperatures, thermal limitations will reduce the power that can be handled to values less than the limitations imposed by second breakdown.

FIGURE 6 – TURN-OFF TIME

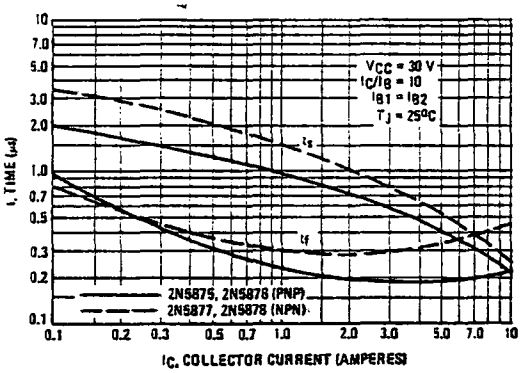


FIGURE 7 – CAPACITANCE

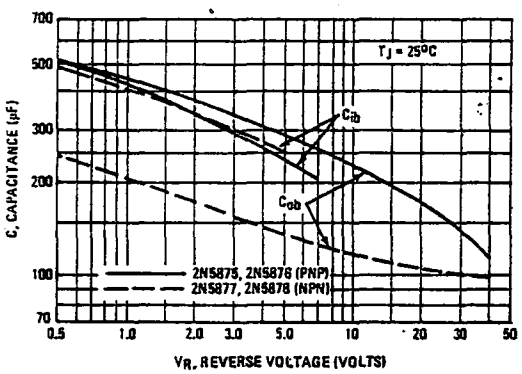


FIGURE 4 — THERMAL RESPONSE

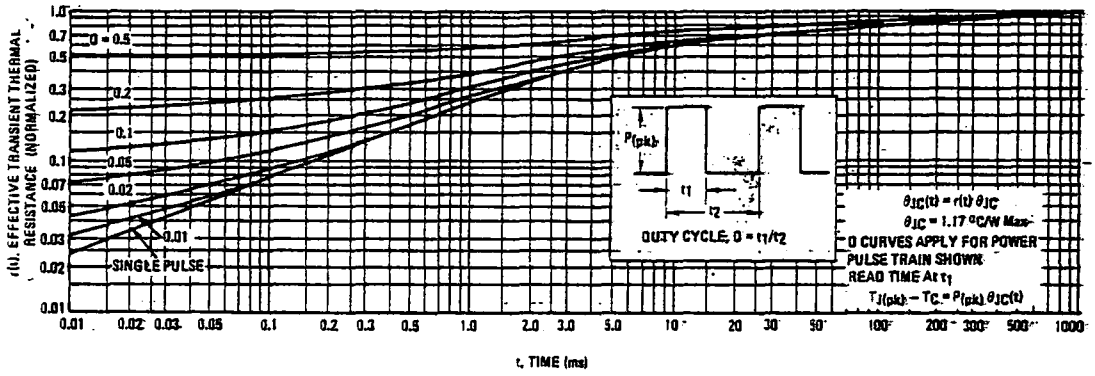
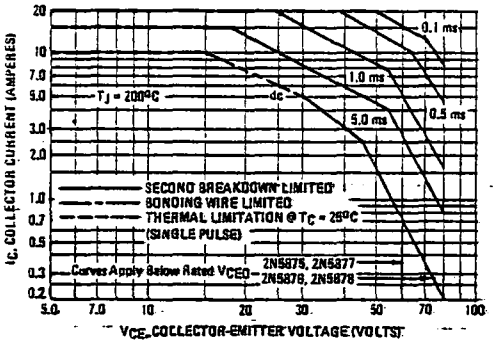


FIGURE 5 — ACTIVE REGION SAFE OPERATING AREA



There are two limitations on the power handling ability of a transistor: average junction temperature and second breakdown. Safe operating area curves indicate $I_C - V_{CE}$ limits of the transistor that must be observed for reliable operation, i.e., the transistor must not be subjected to greater dissipation than the curves indicate.

The data of Figure 5 is based on $T_{J(pk)} = 200^{\circ}C$; T_C is variable depending on conditions. Second breakdown pulse limits are valid for duty cycles to 10% provided $T_{J(pk)} \leq 200^{\circ}C$. $T_{J(pk)}$ may be calculated from the data in Figure 4. At high case temperatures, thermal limitations will reduce the power that can be handled to values less than the limitations imposed by second breakdown.

FIGURE 6 — TURN-OFF TIME

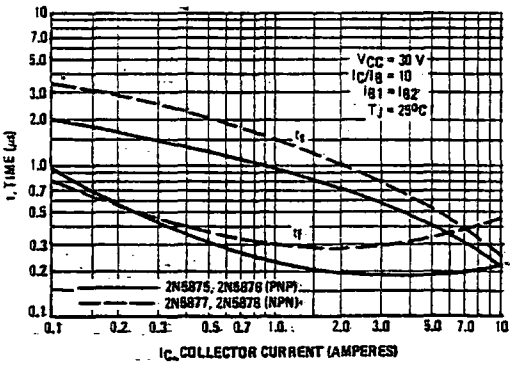
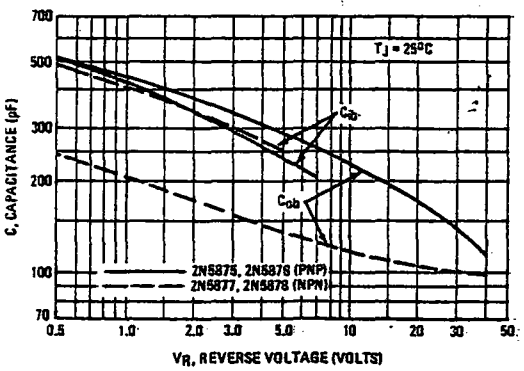
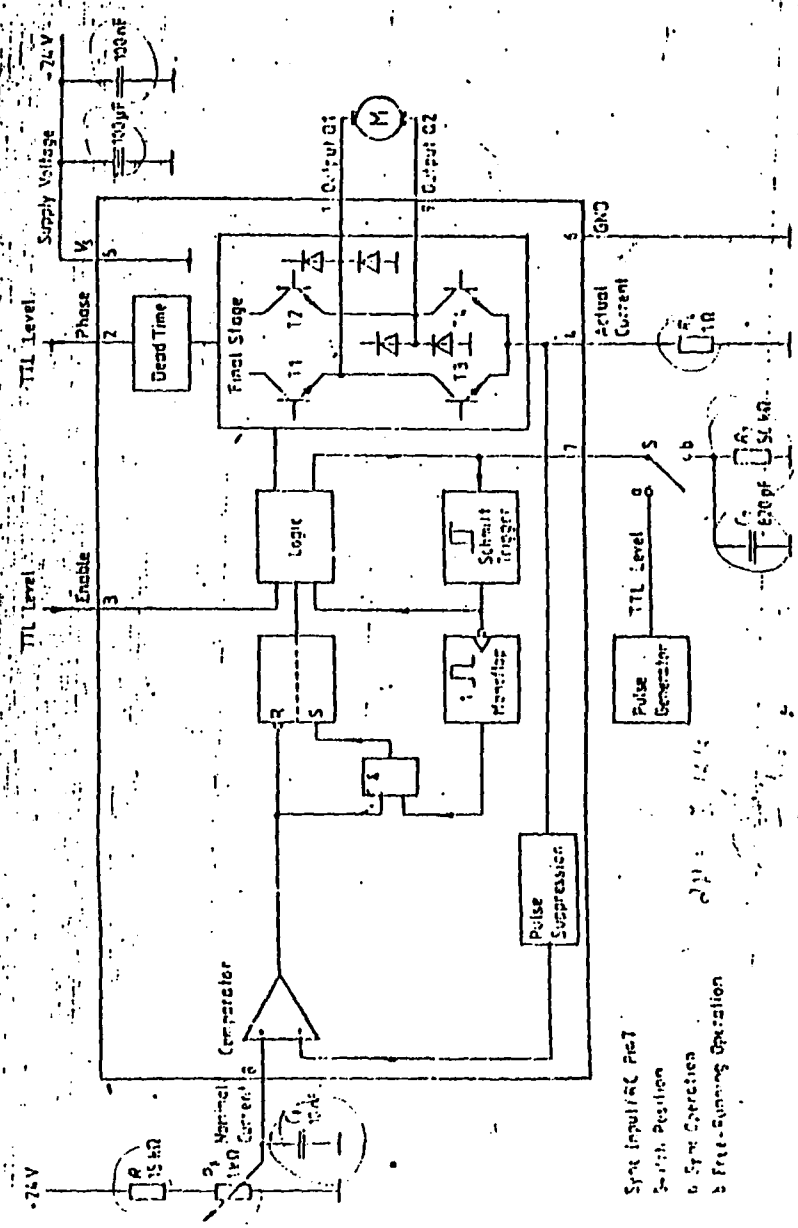


FIGURE 7 — CAPACITANCE



Block Diagram



Circuit Description

Outputs

Outputs Q1, Q2 (pins 1, 9) are fed by push-pull output stages. The two integrated free-wheel diodes, referred to ground or supply voltage respectively, protect the IC against flyback voltages from an inductive load.

Enable

Outputs Q1 and Q2 are turned off when voltage $V_3 \leq 0.8$ V is applied to pin 3. The supply current then decreases maximally to 1 mA. The same occurs if pin 3 is open. The sink transistors are turned on when $V_3 \geq 2$ V.

Phase

The voltage at pin 2 determines the phase position of the output current. Output Q1 acts as sink for $V_2 \leq 0.8$ V and as source for $V_2 \geq 2$ V.

Similarly output Q2 acts as

sink when $V_2 \geq 2$ V and
source when $V_2 \leq 0.8$ V

The sink transistors are current-chopped. An internal circuit avoids undesired cross-over currents at phase change.

Nominal Current Input

The peak current in the motor winding is determined by the voltage at pin 6. A comparator compares this with the voltage drop at the actual current sensor at pin 4. If the nominal current is exceeded, the output sink transistors are turned off by a logic circuit.

Sync Input/RC

Outputs are turned on by a signal at pin 7. Two operation modes are possible: Synchronizing by a fed-in TTL signal or free running with the external RC combination.

Free-Running Operation

When the supply voltage is applied, capacitor C_7 at pin 7 charges to a limiting voltage, typically 2.4 V. With increasing current in the motor winding, the voltage rises at the actual current sensor R_4 (pin 4). After exceeding the predetermined value at the nominal current input (pin 6) the comparator, in conjunction with pulse suppression, resets an RS flipflop. The logic turns off sink transistors T3 and T4. C_7 ceases charging and the parallel resistance R_7 then discharges C_7. The sink transistors remain turned off until the lower threshold voltage of the Schmitt trigger is reached. This off period is thus controlled by the time constant $t_a = R_7 \times C_7$. After the lower trigger threshold has been passed, the monoflop is triggered by the falling edge of the Schmitt trigger output and, provided the voltage at the actual current sensor (pin 4) is lower than the nominal value at pin 6, the RS flipflop is reset. The logic circuit then turns on the sink transistors T3 or T4 and recharges capacitor C_7. If the voltage at pin 4 rises above the comparator value at pin 6, the sink transistors T3 and T4 are turned off again. Turn-on cannot be repeated until capacitor C_7 has discharged to the lower trigger threshold, the discharge time being a function of R_7 and C_7.

Operation

...vel sync signal is fed to pin 7, the negative edge sets the RS flipflop, via the ...trigger/monoflop combination, provided that the voltage at pin 4 is below the ...minal value at pin 8. As in the free-running operation mode, the relevant output transistors become conducting. Similarly they are cut off by resetting the RS flipflop once the voltage at pin 4 is higher than the nominal value at pin 8.

Pulse Suppression

In all cases the pulse suppression circuit eliminates positive pulses, typically of 0.5 µs duration, at pin 4. These can result from cross-over currents in chopper operation through the integrated free-wheel diodes. As a result, the voltage at pin 4 rises well above the nominal value, and without pulse suppression this would lead to dynamic current limiting. The duration of these basically unavoidable cross-over currents is of the same order of magnitude as the reverse-recovery time of the free-wheel diodes.

Temperature Safeguard

If the temperature of the IC rises to approx. 150 °C, the final stages are turned off. At approx. 130 °C they are turned on again.

Logic Table

Enable		L	L	H	H	
Phase		L	H	L	H	
Output	Q1	/	/	L	H	
Output	Q2	/	/	H	L	
Transistor	T1	X	X	X	·	at:
Transistor	T2	X	X	·	X	$V_4 > 10$ mV
Transistor	T3	X	X	··	X	$R_4 > 0\ \Omega$
Transistor	T4	X	X	X	··	

L = Low voltage level, input open
H = High voltage level
X = Transistor turned off
· = Transistor conducting
·· = Transistor conducting with current limiting turned on
/ = Output high-impedance

Maximum Ratings

$T_C = -25\,°C$ to $+85\,°C$

Description	Symbol	min	max	Unit
Supply voltage, pin 5	V_S	−0.3	45	V
Supply current, pin 5	I_S	0	2.5	A
Peak current in output transistors, pin 1, 9	I_O	−2.5	2.5	A
Diode currents				
Diode to $+V_S$	I_{FH}		2.5	A
Diode to ground	I_{FL}		2.5	A
Input voltage, pins 2, 3, 7, 8	V_I	−0.3	6	V
Output current, pin 4	I_4	−2.5		A
Voltage, pin 4	V_4	−0.3	5	V
Ground current, pin 6	I_6		2.5	A
Junction temperature	T_j		150	°C
Storage temperature	T_{stg}	−40	125	°C
Thermal resistance system – ambient	$R_{th\,SA}$		70	K/W
system – case	$R_{th\,SC}$		8	K/W

Operating Range

Supply voltage, pin 5	V_S	8	40	V
Package temperature	T_C	−25	85	°C
Input voltage, pins 2, 3, 7	V_I		5	V
Output current	I_O	−2	2	A

Maximum Ratings
T_C = 25 °C bis +85 °C

Description	Symbol	min	max	Unit
Supply voltage, pin 5	V_S	−0.3	45	V
Supply current, pin 5	I_S	0	1.25	A
Peak current in output transistors, pins 1, 9	I_Q	−1.25	1.25	A
Diode currents, pins 1,9				
Diode against + V_S	I_{1H}		1.25	A
Diode against ground	I_{1L}		1.25	A
Input voltage, pins 2, 3, 7, 8	V_i	−0.3	6	V
Output current, pin 4	I_4	−1.25		A
Voltage, pin 4	V_4	−0.3	6	V
Ground current, pin 6	I_6		1.25	A
Junction temperature	T_j		150	°C
Storage temperature	T_{stg}	−40	125	°C
Thermal resistance system – ambient	$R_{th\,SA}$		70	K/W
system – case (measured at pin 14)	$R_{th\,SC}$		15	K/W

Operating Range

Supply voltage, pin 5	V_S	8	40	V
Package temperature measured at pin 14	T_C	−25	85	°C
Input voltage, pins 2, 3, 7	V_i		5	V
Output current, pins 1, 9	I_Q	−1	1	A

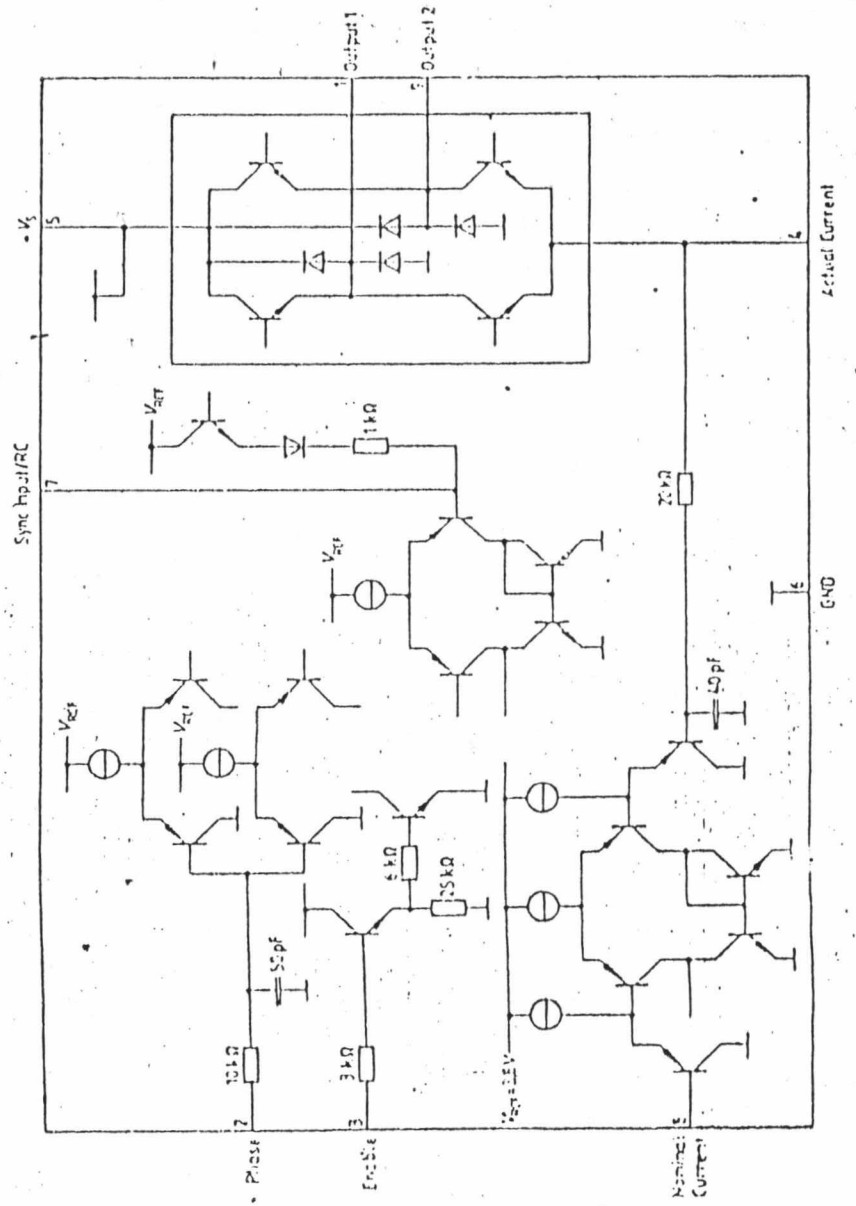
Characteristics
V_S = 14 V; T_C = 25 °C

...tion	Symbol	Test conditions	min	typ	max	Unit
...urrent, pin 5	I_S	$V_{I3} = V_{IH}$		18	30	mA
...urrent, pin 5	I_S	$V_{I3} = V_{IL}$		0.5	1	mA
...pins 1, 9						
...oltage; source	V_{OH}	$\|I_Q\| = 1$ A		1.7	1.9	V
...oltage; source	V_{OH}	$\|I_Q\| = 1.5$ A		1.9	2.1	V
...oltage; sink	V_{OL}	$\|I_Q\| = 1$ A		1.2	1.4	V
...oltage; sink	V_{OL}	$\|I_Q\| = 1.5$ A		1.5	1.7	V
...current	$\|I_{QS}\|$				300	µA
...ad time	t_T	Figure 1	0.1	0.3	1.0	µs
...voltage of diodes ...V_S	V_{1H}	$I_{1H} = 1$ A		1.0	1.2	V
	V_{1H}	$I_{1H} = 1.5$ A		1.1	1.3	V
...voltage of diodes ...ound	V_{1L}	$I_{1L} = 1$ A		1.1	1.3	V
	V_{1L}	$I_{1L} = 1.5$ A		1.3	1.5	V
...nable, pin 8 ...so, pin 2						
...oltage	$V_{...}$		2			V
...oltage	$V_{...}$				0.8	V
...urrent	$I_{...}$	$V_{...} = 5$ V			100	µA
...rrent	$-I_{...}$	$V_{...} = 0$ V			100	µA
...all time	t_f				2	µs
...current, pin 8						
...nge	$V_{...}$		0		2	V
...ent	$-I_{...}$	$V_{...} = 0$ V			5	µA
...t voltage	$V_{...}$	Figure 5		0		mV
...rrent, pin 4						
...nge	V_4	Figure 5	0		2	V
...elay	t_D	Figure 3		2	3	µs
...t/RC, pin 7						
...ency	f	Duty cycle 0.5	1		100	kHz
	D	f = 40 kHz	0.1		0.9	
...all time	t_f				2	µs
...rrent pin 7	$-I_{...}$		1.2	1.6	2.0	mA
...stage, pin 7	$V_{...}$	Figure 2		0.6	0.8	V
...pin C	$V_{...}$		2.2	2.4		V
	$t_{...}$	Figure 4		0.4		µs
...put resistance	R_i	$V_i = 1.5$ V		1		kΩ

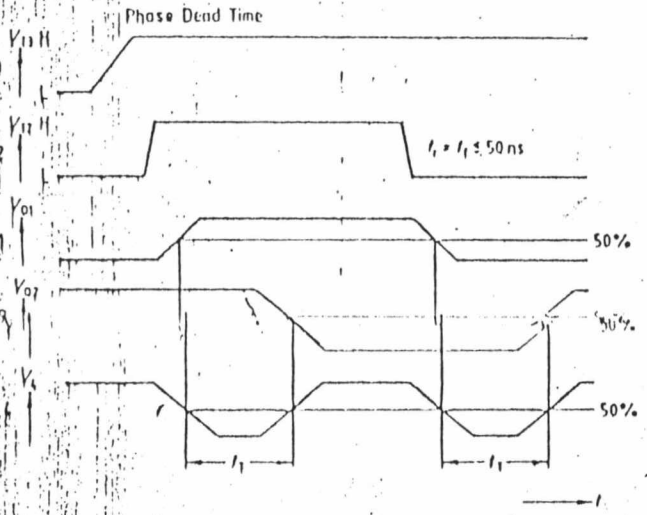
ristics
V; T_C = 25 °C

Description	Symbol	Test conditions	min	typ	max	Unit
Supply current, pin 5	I_S	$V_{ID} = V_{IH}$		18	30	mA
Supply current, pin 5	I_S	$V_{ID} = V_{IL}$		0.5	1	mA
Output, pins 1, 9						
Output voltage: source	V_{OH}	$\|I_Q\| = 0.5$ A		1.6	1.8	V
Output voltage: source	V_{OH}	$\|I_Q\| = 0.75$ A		1.65	1.90	V
Output voltage: sink	V_{OL}	$\|I_Q\| = 0.5$ A		1.0	1.2	V
Output voltage: sink	V_{OL}	$\|I_Q\| = 0.75$ A		1.1	1.4	V
Reverse current	$\|I_{QS}\|$				300	µA
Phase dead time	t_t	Figure 1	0.1	0.3	1.0	µs
Forward voltage of diodes to $+V_S$	V_{FH}	$I_{FH} = 0.5$ A		0.9	1.1	V
	V_{FH}	$I_{FH} = 0.75$ A		0.95	1.15	V
Forward voltage of diodes to ground	V_{FL}	$I_{FL} = 0.5$ A		0.95	1.15	V
	V_{FL}	$I_{FL} = 0.75$ A		1.0	1.2	V
Inputs: enable, pin 3 and phase, pin 2						
H input voltage	V_{IH}		2			V
L input voltage	V_{IL}				0.8	V
H input current	I_{IH}	$V_{IH} = 5$ V		50	100	µA
L input current	$-I_{IL}$	$V_{IL} = 0$ V			100	µA
Rise and fall time	t_r, t_f				2	µs
Nominal current, pin 8						
Control range	V_{I8}		0		2	V
Input current	$-I_{I8}$	$V_{I8} = 0$ V			5	µA
Input offset voltage	$V_{I(8-4)}$	Figure 5		0		mV
Actual current, pin 4						
Regulating range	V_{I4}	Figure 5	0		2	V
Turn-off delay	t_D	Figure 3		2	3	µs
Sync input/RC, pin 7						
Sync frequency	f	Duty cycle: 0.5	1		100	kHz
Duty cycle	D	f = 40 kHz	0.1		0.9	
Rise and fall time	t_r, t_f				2	µs
Output current, pin 7	$-I_{Q7}$		1.2	1.6	2.0	mA
Trigger threshold, pin 7	V_{L7}	Figure 2		0.6	0.8	V
Charging limit C_7	V_{G7}		2.2	2.4		V
Off period	t_{OFF}	Figure 4		64		µs
Dynamic input resistance pin 7	R_{i7}	V_i = 1.5 V		1		kΩ

Internal Wiring of Pins



Phase Dead Time



gure 2
gger Threshold

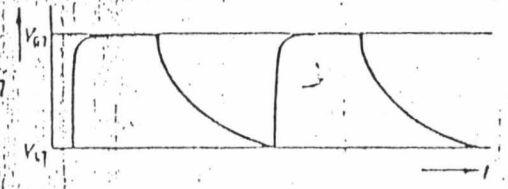


Figure 3
Turn-OFF Delay

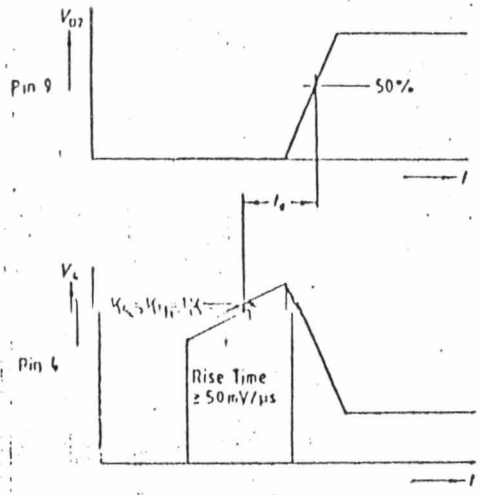
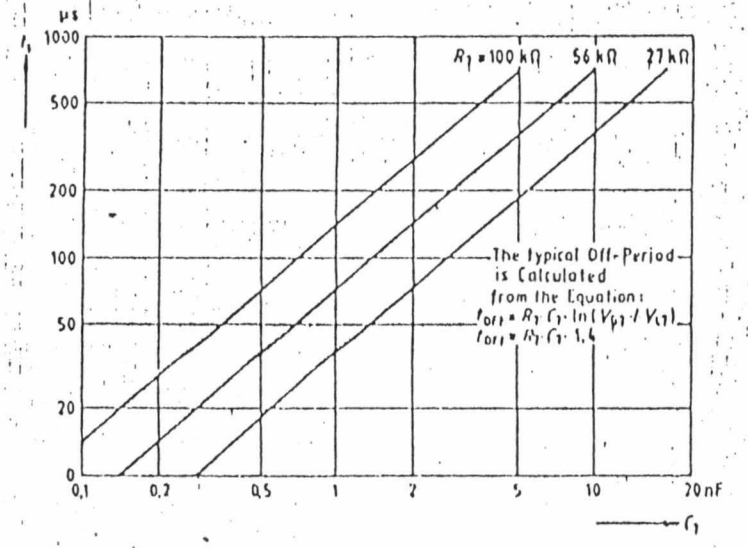
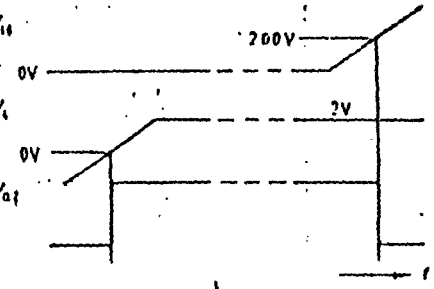


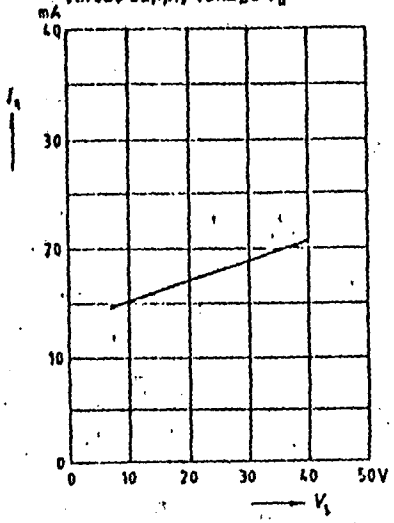
Figure 4
OFF Period versus Capacitance



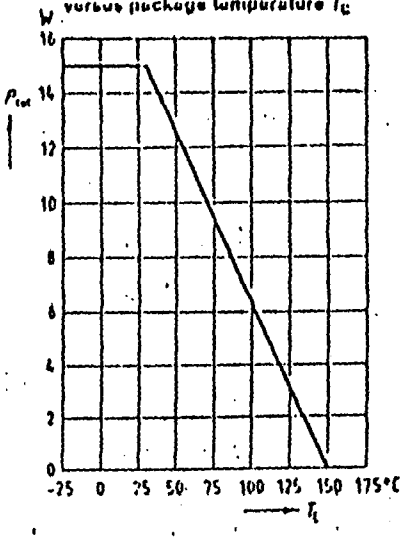
Range, Input Offset Voltage



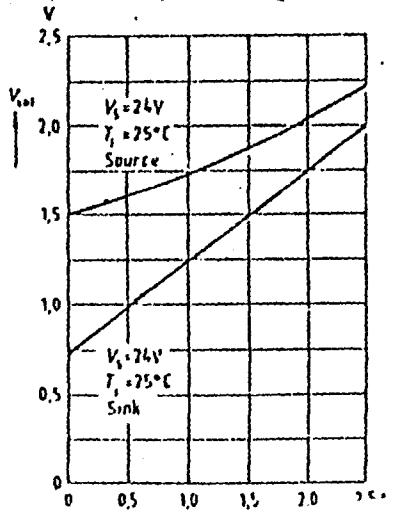
Quiescent current I_q
versus supply voltage V_s



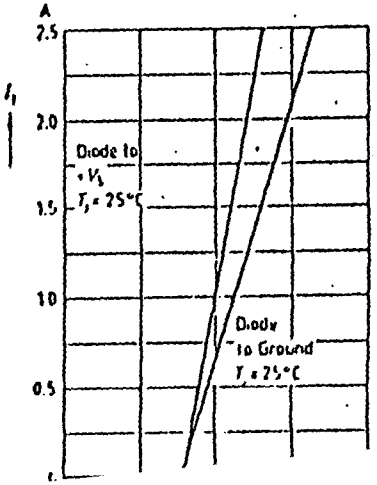
Permissible power dissipation P_{tot}
versus package temperature T_c



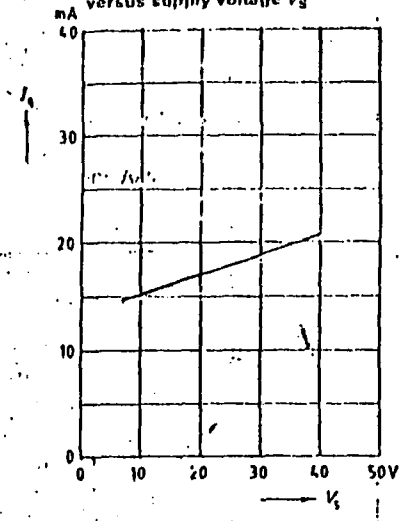
Output saturation voltages V_{sat}
versus output current I_Q



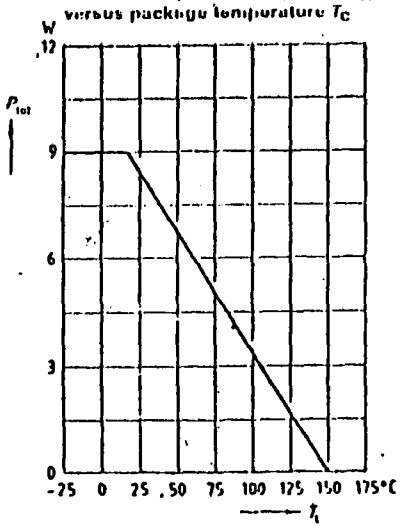
Forward current I_f of free-wheel
diodes versus forward voltages V_f



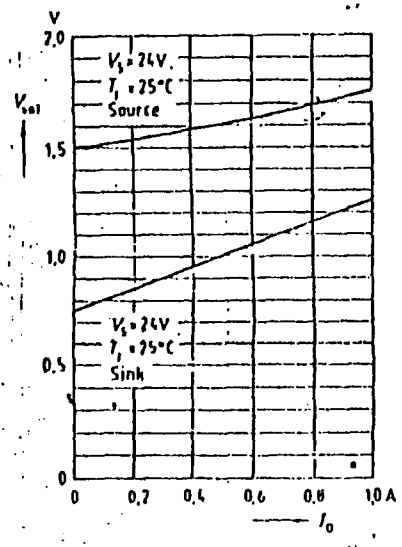
Quiescent current I_q versus supply voltage V_S



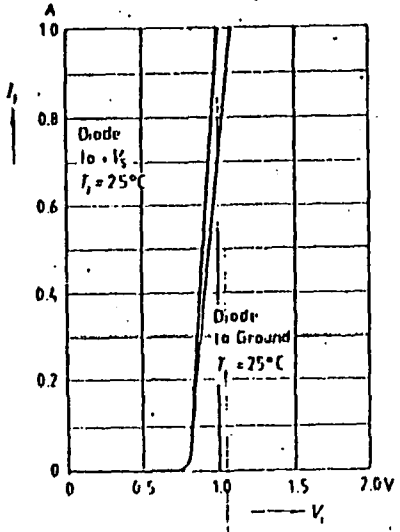
Permissible power dissipation P_{tot} versus package temperature T_C



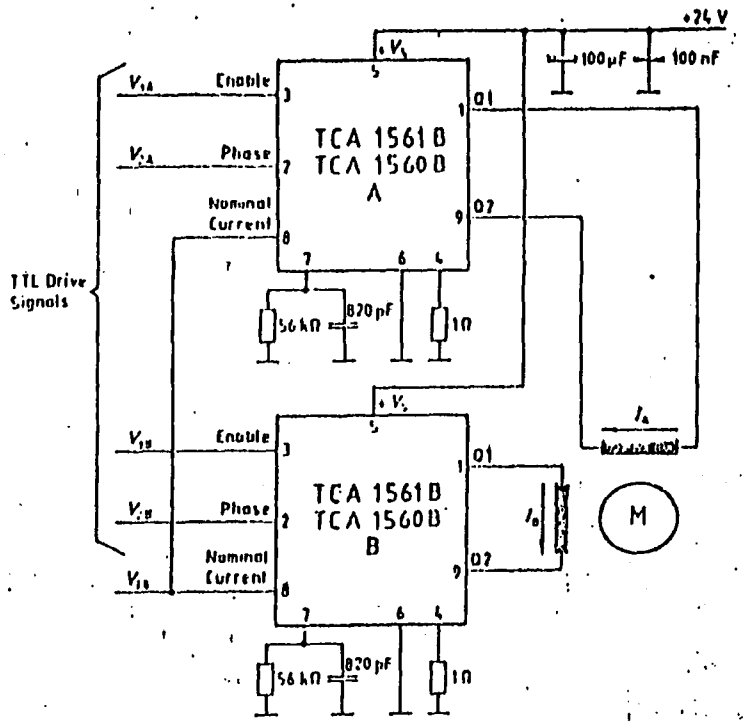
Output saturation voltages V_{sat} versus output current I_Q



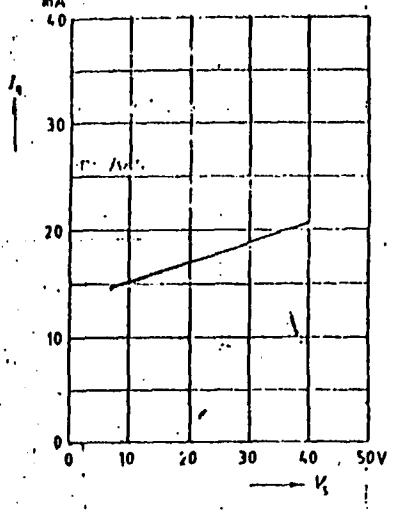
Forward current I_F of free-wheel diodes versus forward voltages V_F



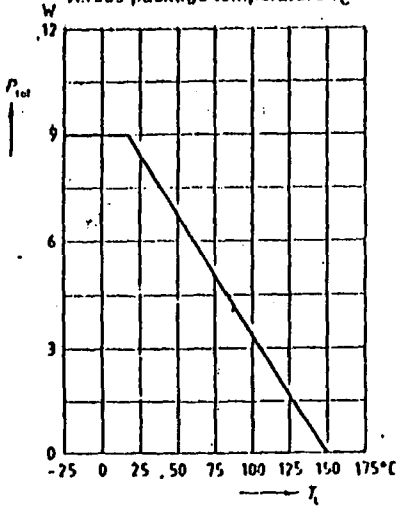
Application Circuit



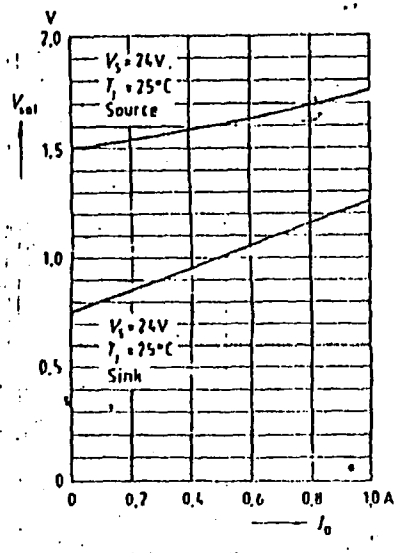
Quiescent current I_q versus supply voltage V_S



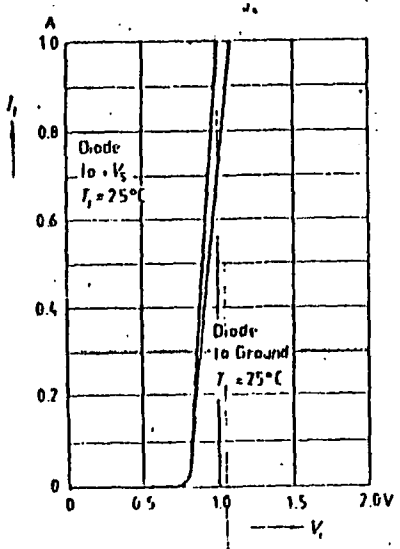
Permissible power dissipation P_{tot} versus package temperature T_C



Output saturation voltages V_{sat} versus output current I_Q



Forward current I_F of free-wheel diodes versus forward voltages V_F



Application Circuit

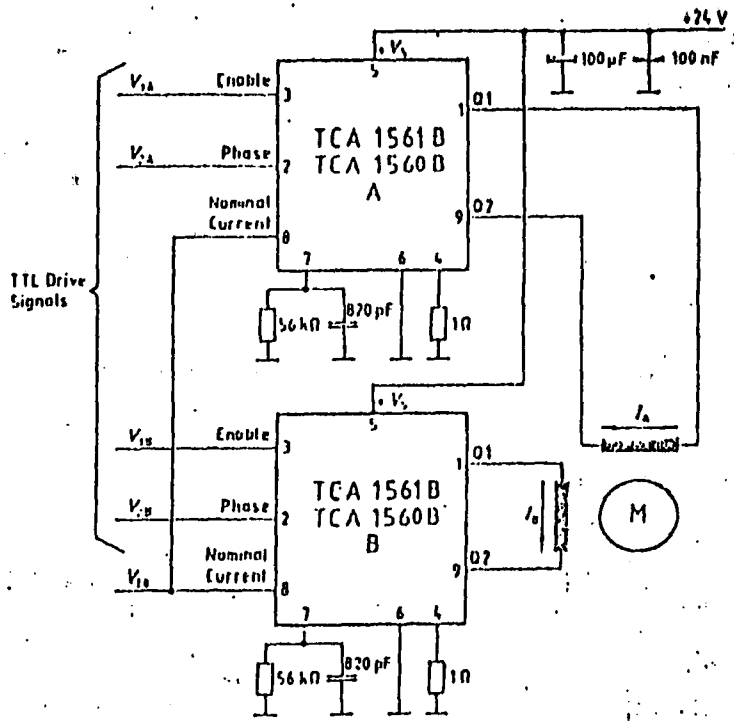
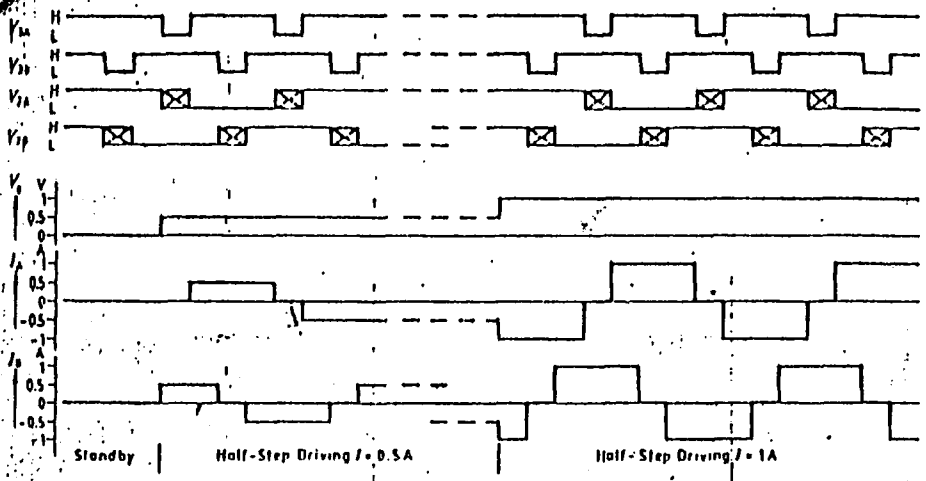


Diagram for Application Circuit



Calculation of Power Dissipation

The total power dissipation P_{tot} comprises

Saturation losses P_{sat}	(transistor saturation voltage and diode forward voltages)
Quiescent current losses P_q	(quiescent current multiplied by supply voltage)
Switching losses P_s	(turn-on/turn-off operation)

The following equations give the power dissipation for chopper operation without phase reversal. This can be regarded as "worst case", as, in addition to the switching losses, full-load current flows for the entire time.

$$P_{tot} = P_{sat} + P_q + P_s$$

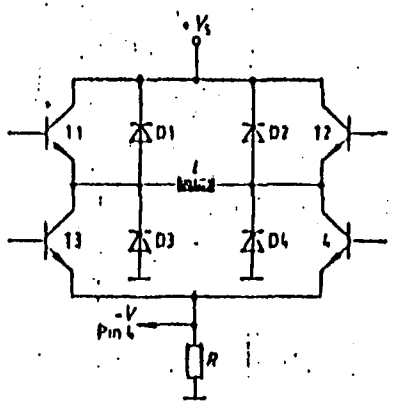
with

$$P_{sat} \approx I_R \left| V_{satu} \cdot D + V_{fb} (1 - D) + V_{satu} \right|$$

$$P_q = I_q \cdot V_S$$

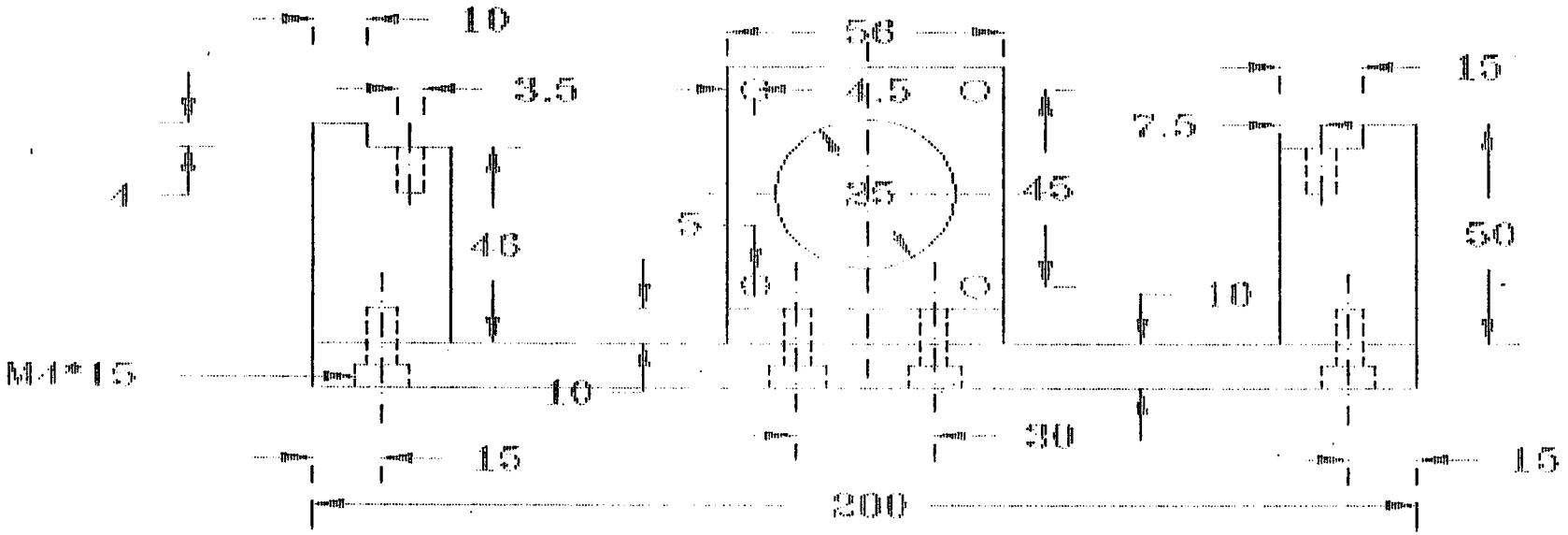
$$P_s \approx \frac{V_S}{T} \left\{ \frac{I_p \cdot t_{p\,on}}{2} + \frac{(I_p + I_i)\, t_{on}}{4} + \frac{I_R}{2} \left(t_{p\,on} + t_{off} \right) \right\} f_{\varphi}$$

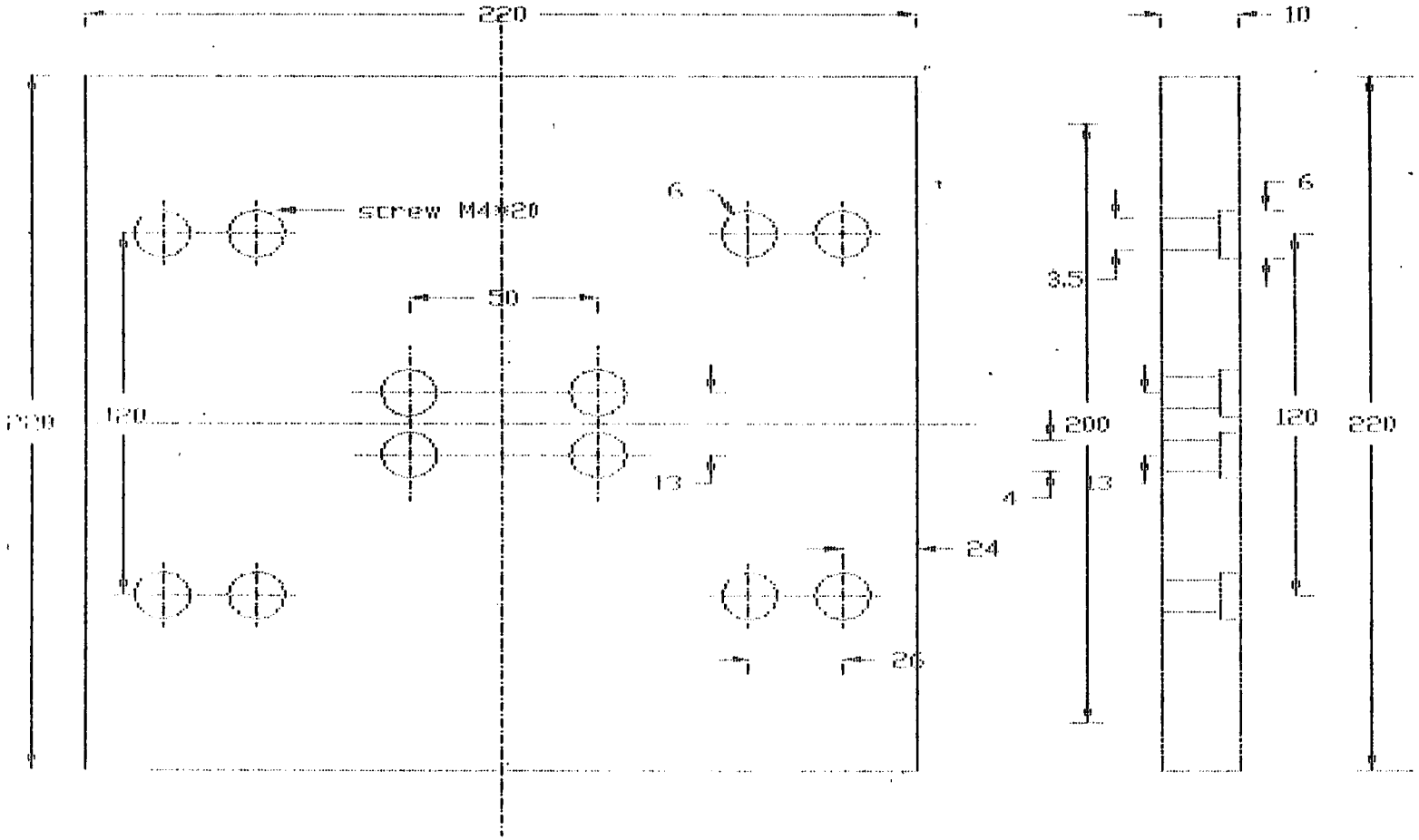
I_R = Rated current (mean value)
I_q = Quiescent current
i_p = Reverse current during turn-on delay time
I_i = Peak reverse current
t_p = Conducting time of chop transistor
t_{on} = Turn-on time
t_{off} = Turn-off time
$t_{p\,on}$ = Turn-on delay time
$t_{p\,off}$ = Turn-off delay time
T = Cycle duration
D = Duty cycle t_p/T
V_{satu} = Saturation voltage of sink transistor (T3, 4)
V_{sato} = Saturation voltage of source transistor (T1, 2)
V_{fb} = Forward voltage of free-wheel diode (D1, 2)
V_S = Supply voltage

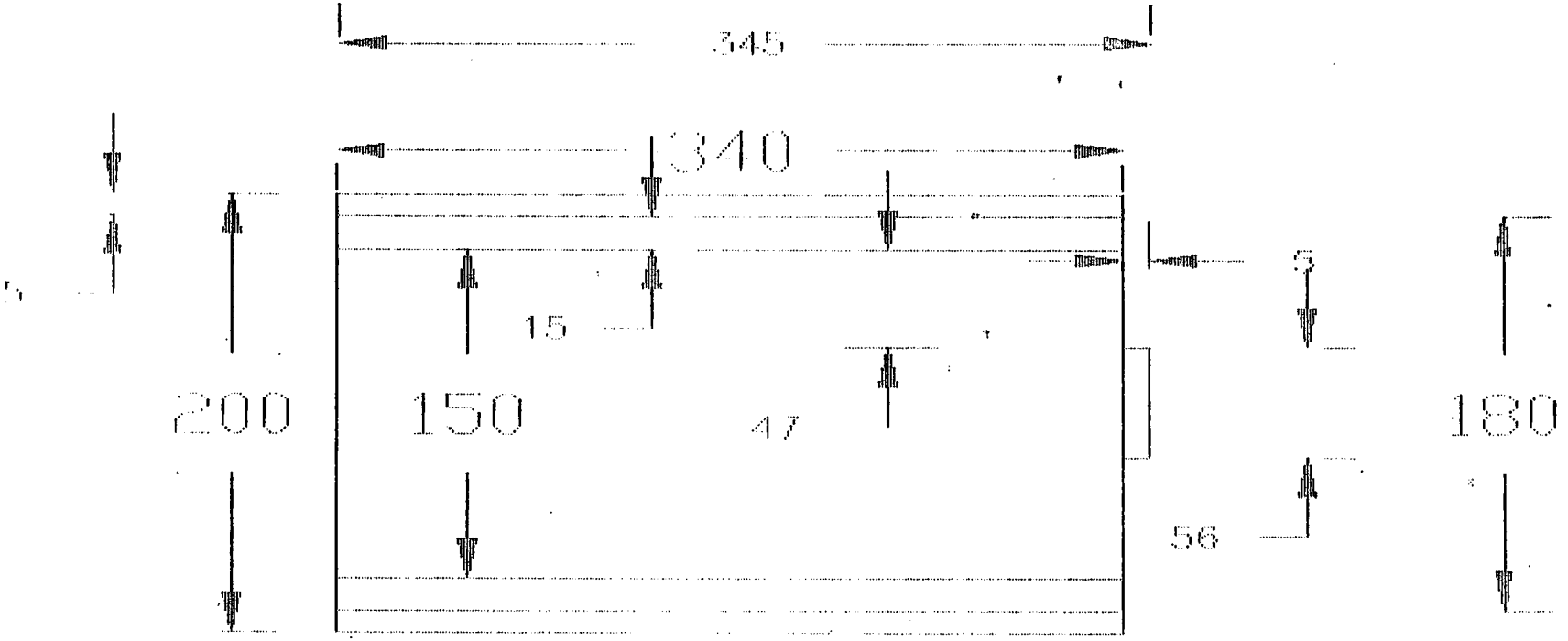


แบบฝึกหัด XYZ-TABLE

front view

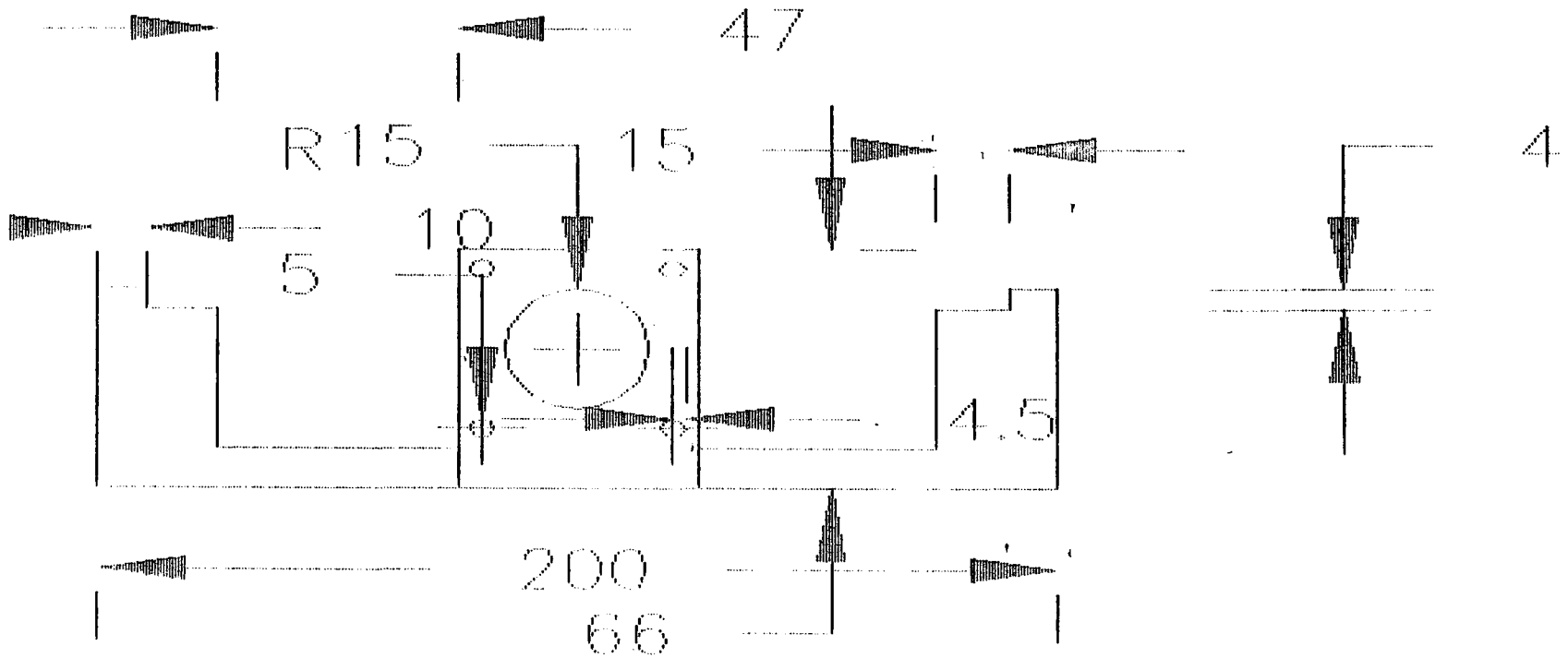



220
10
screw M4*20
6
50
120
13
24
26
3.5
200
4
13
6
120
220

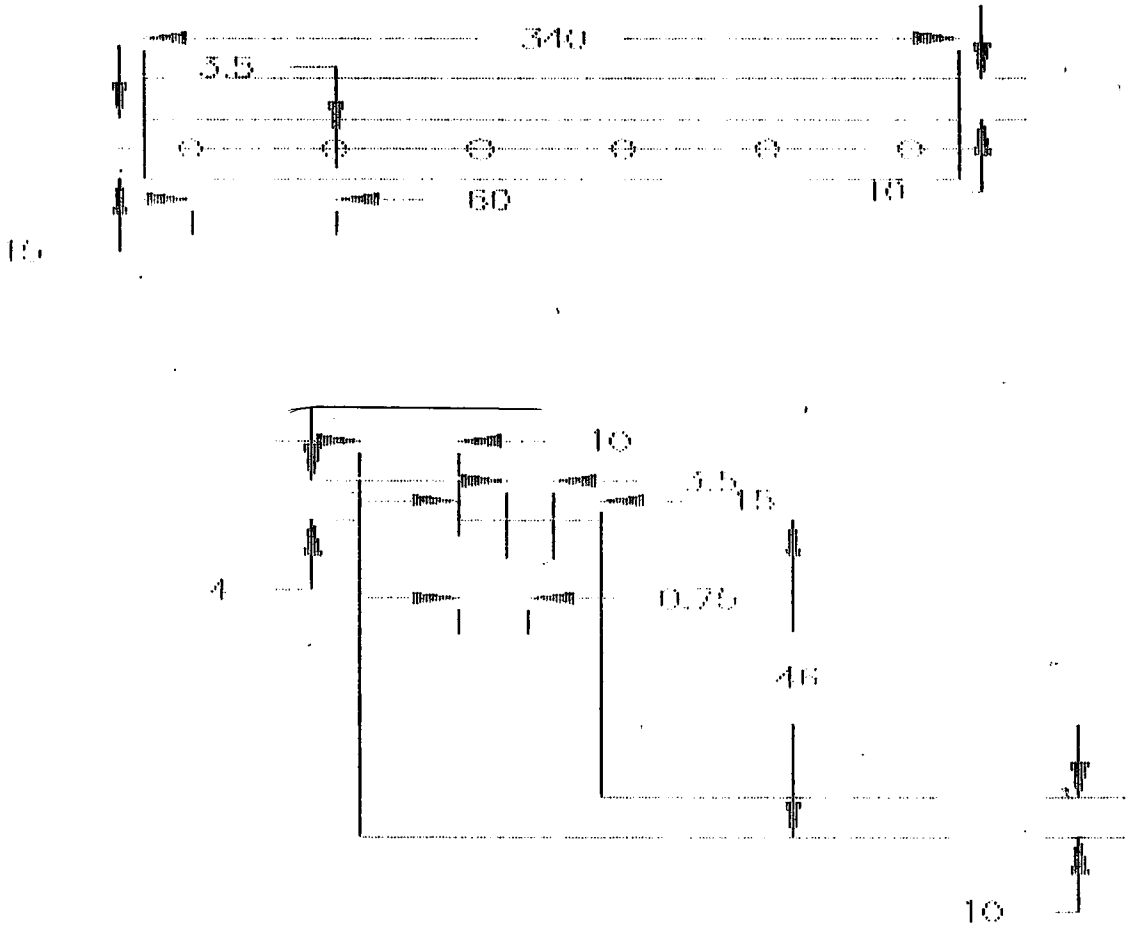


top view

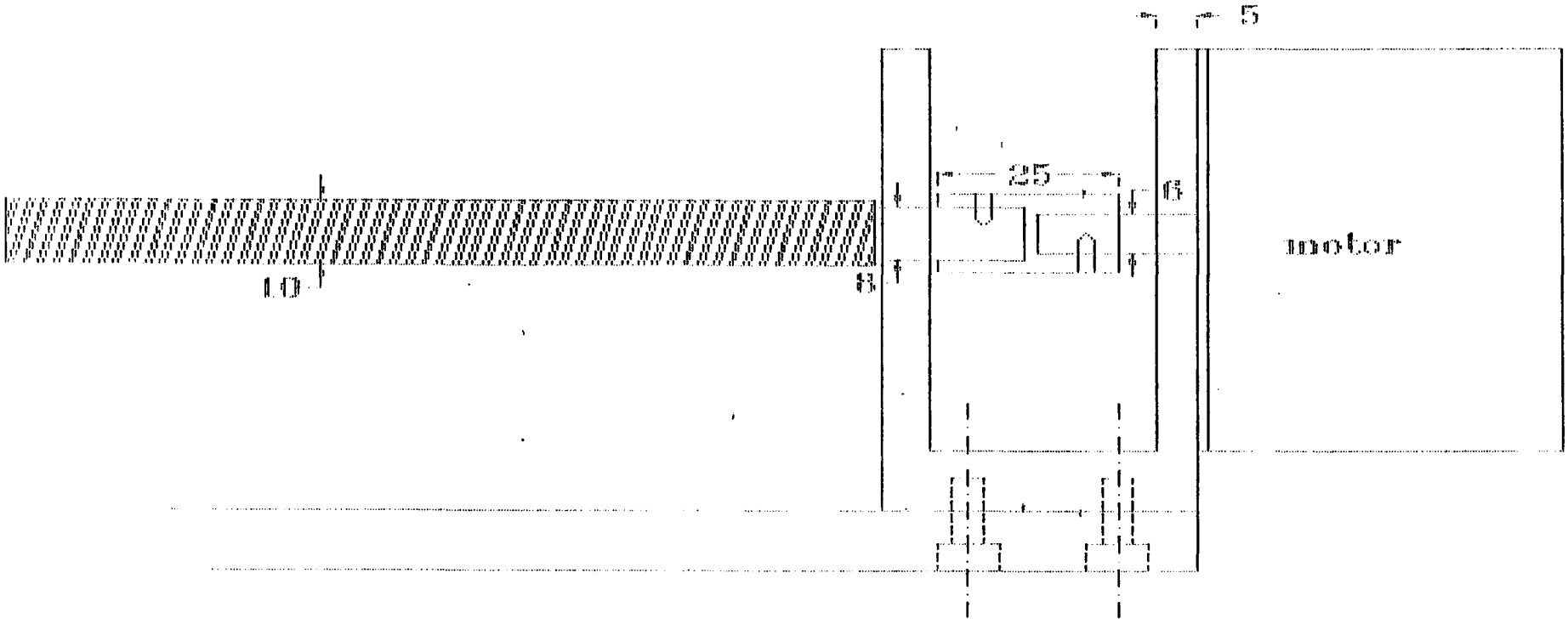

350
340
56
30
5

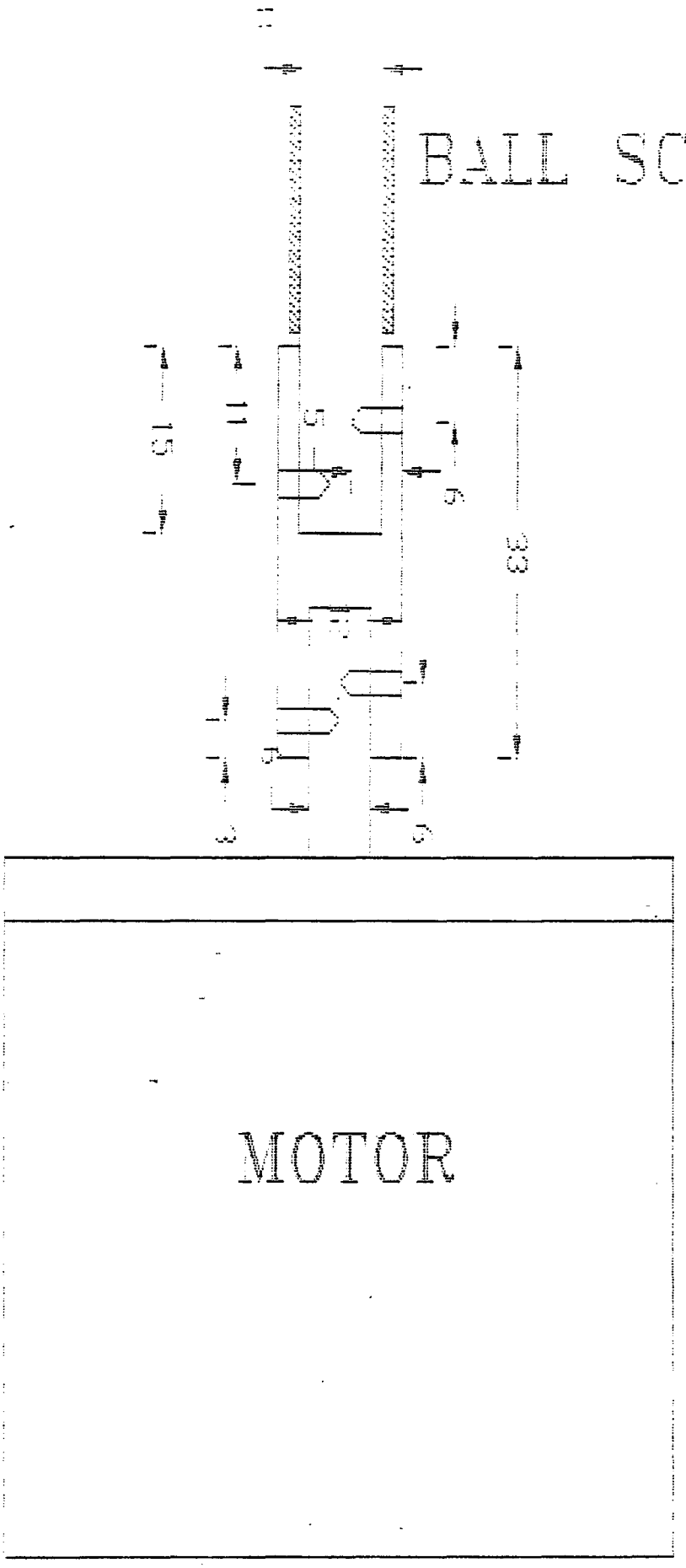


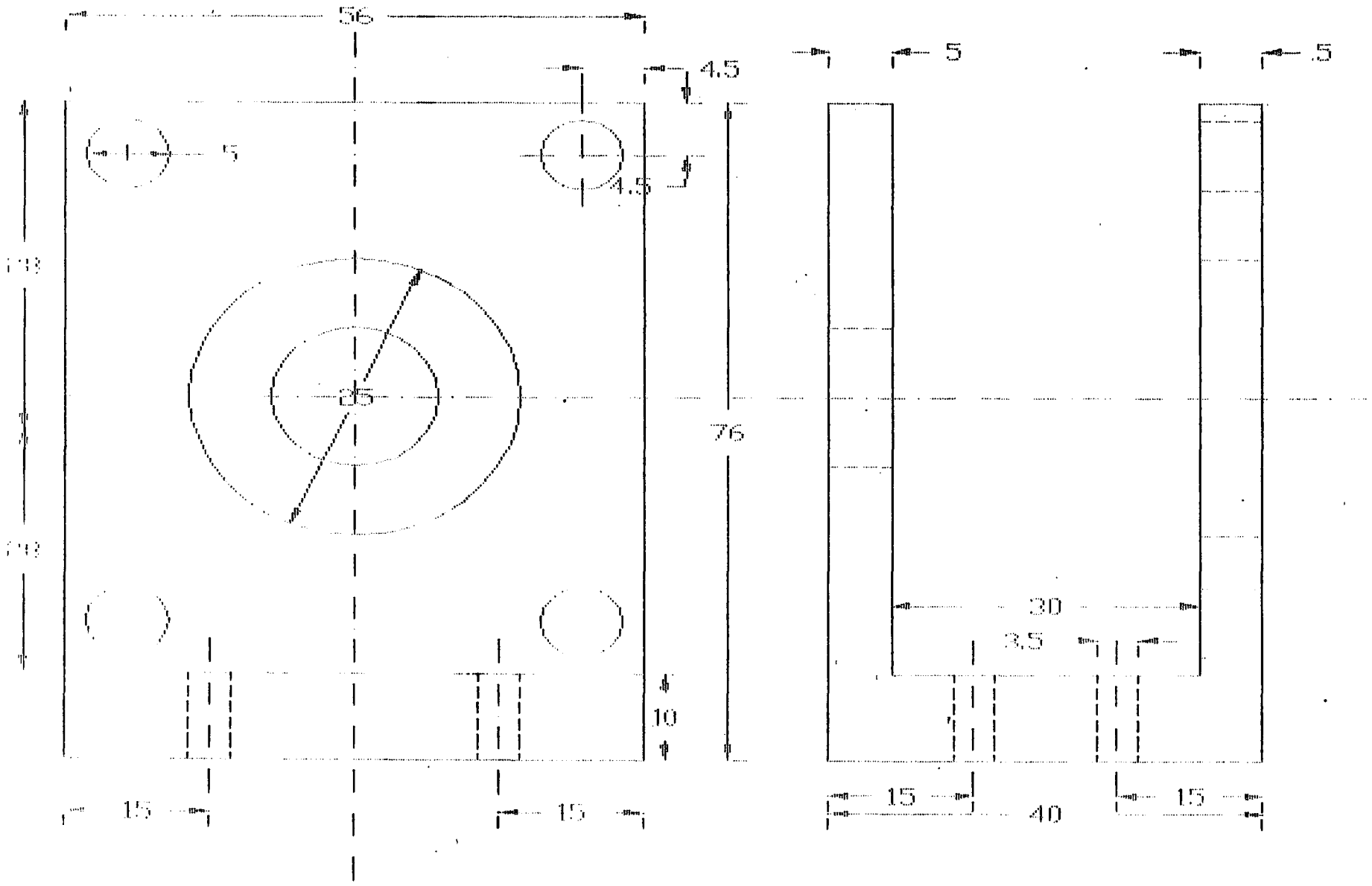
front view


340
3.5
80
10
10
3.5
15
4
0.75
40
10

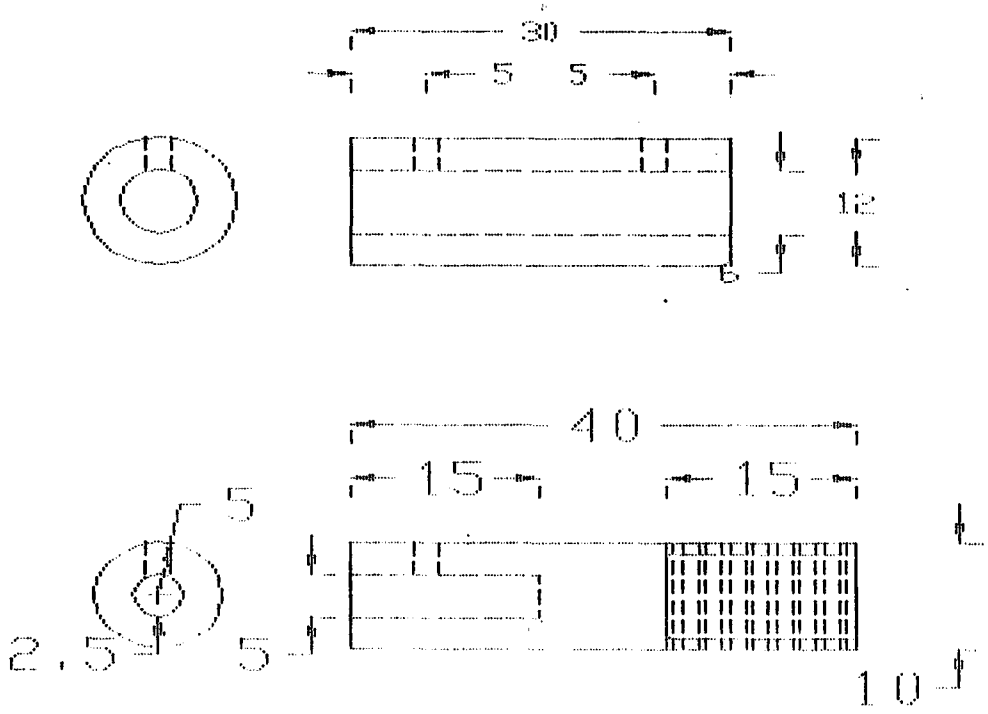
0.1
3.5
60
0.5
4
80
4
5.1
5.10
10
5.1
5.7
3.5

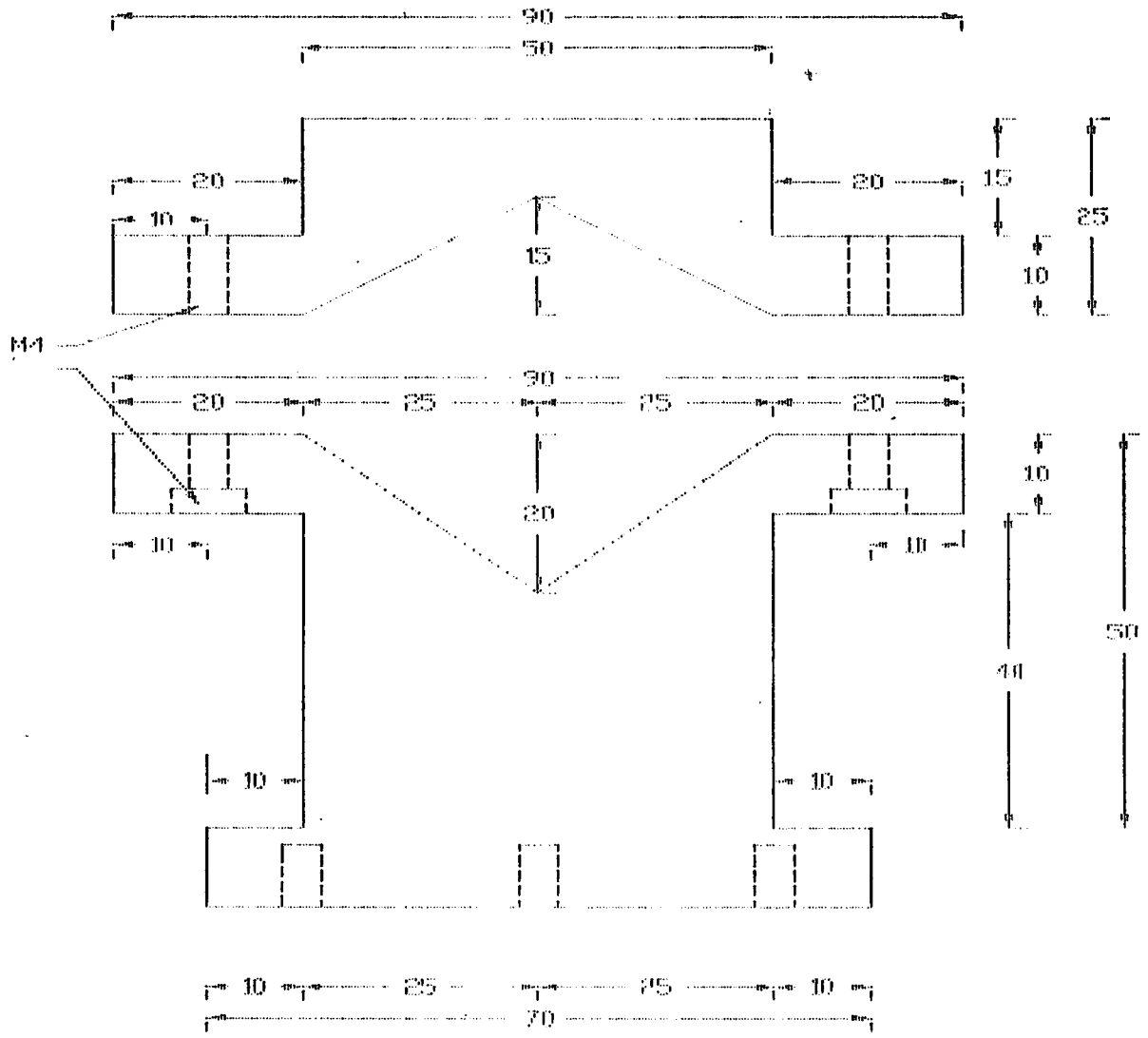

5
25
motor

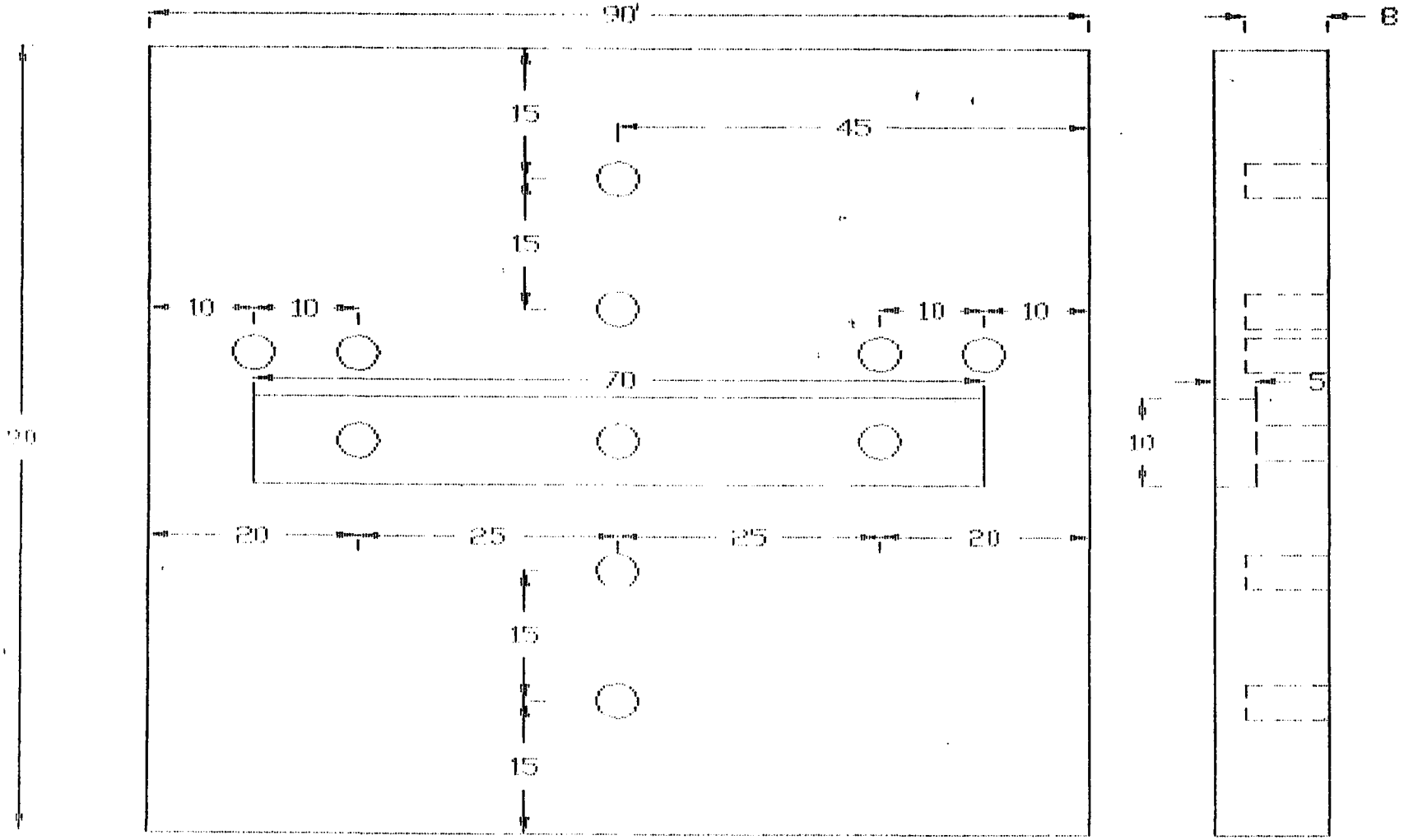

BALL SCREW
MOTOR

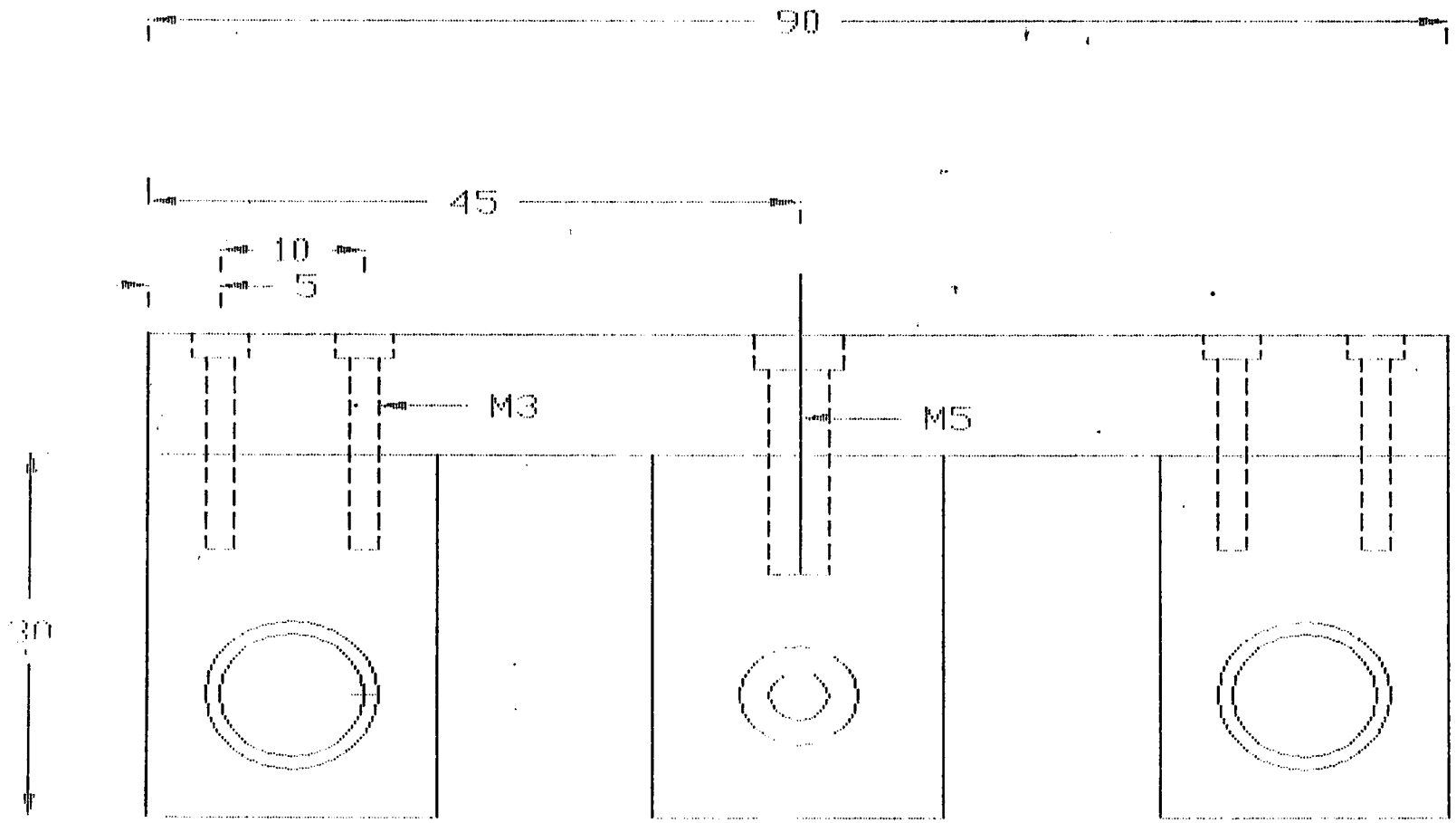

56
4.5
5
4.5
133
25
76
10
15
15
5
5
30
3.5
15
40
15

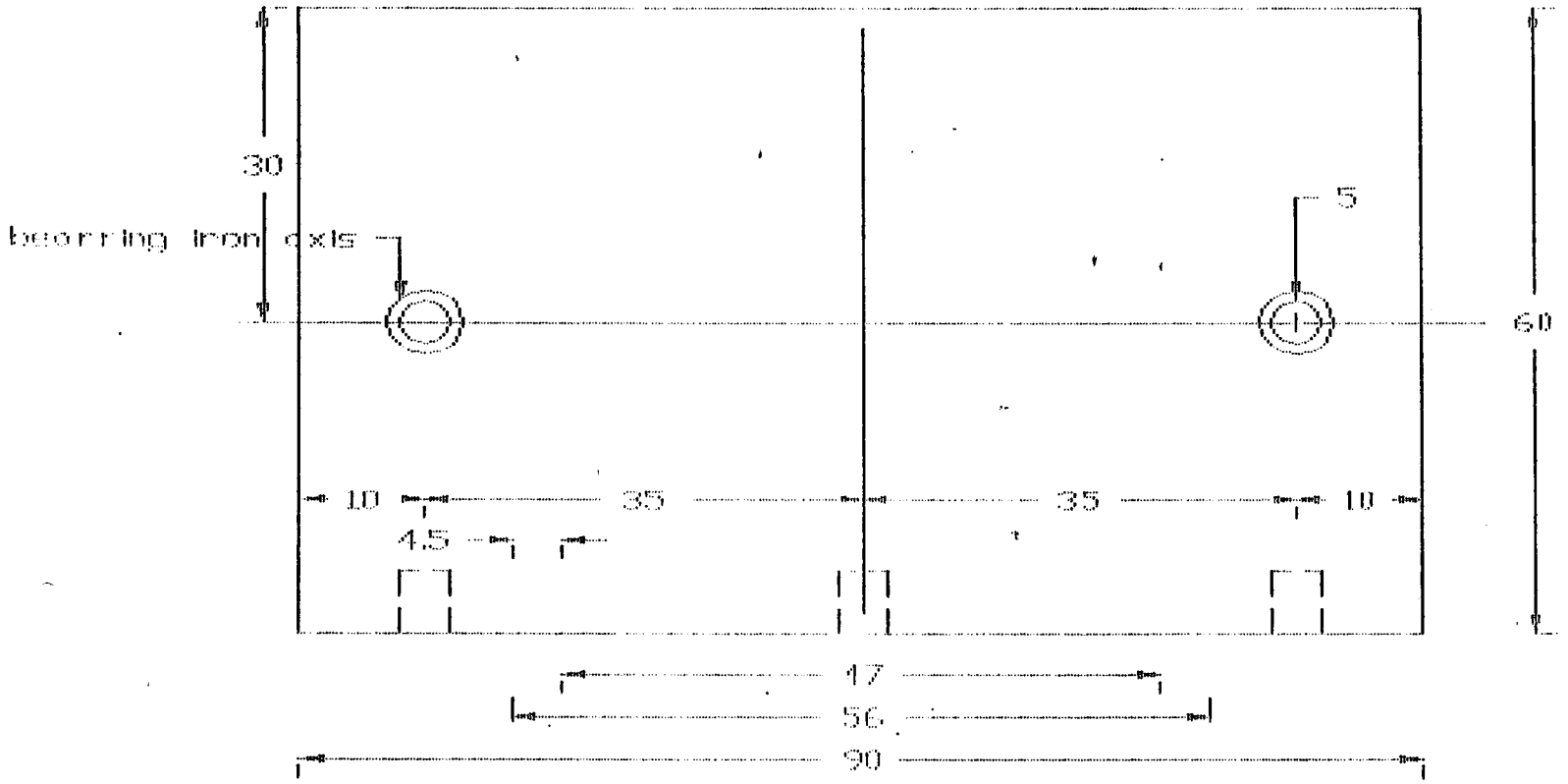
coupling motor

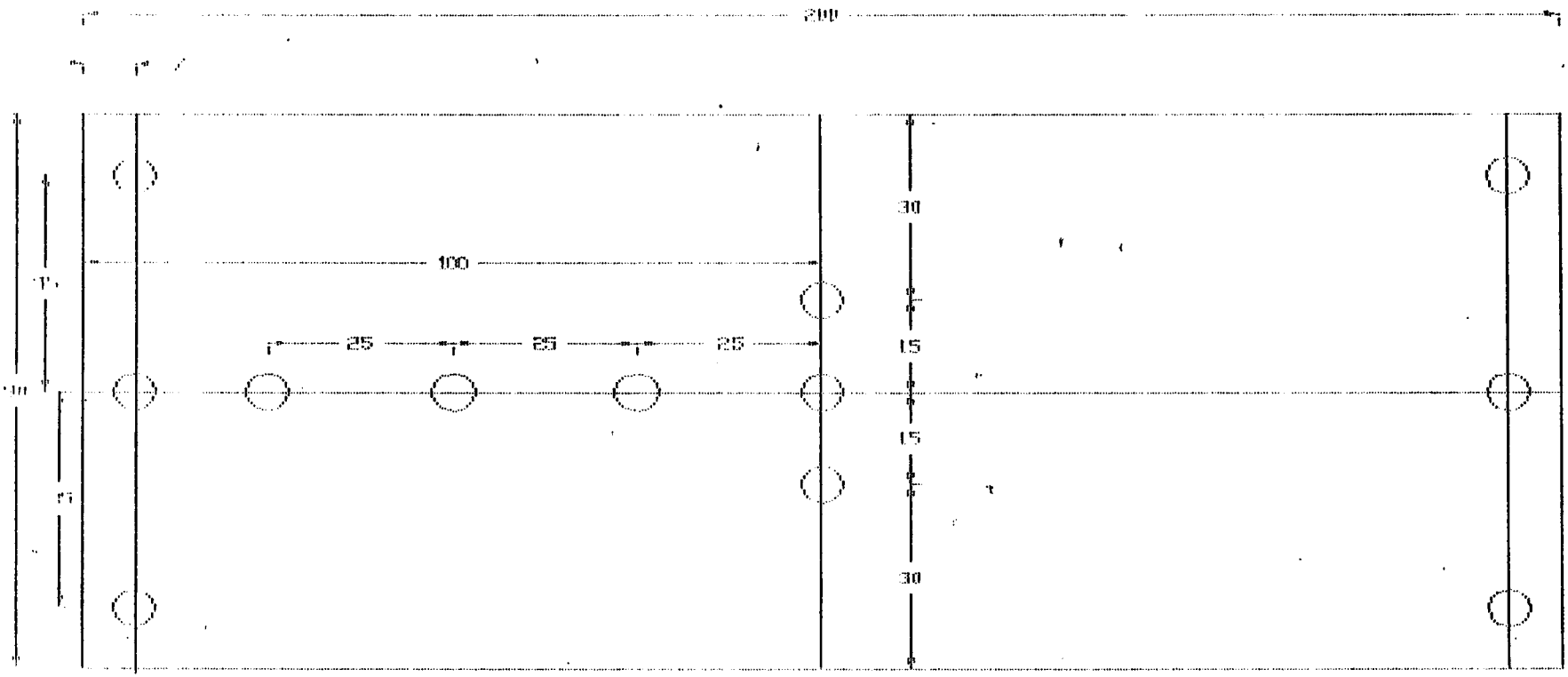


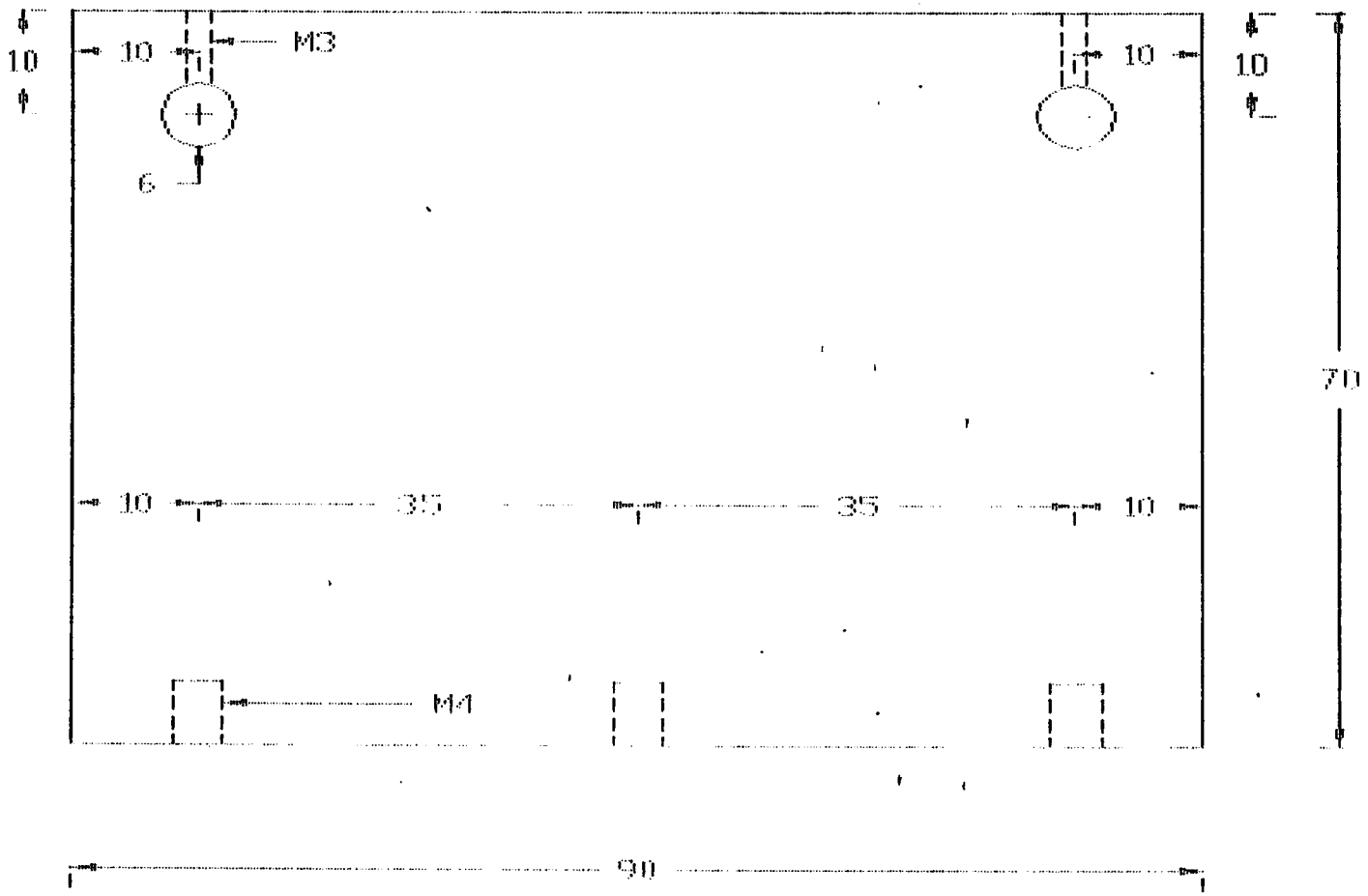

90
50
20
20
10
15
15
25
10
M4
90
20
25
75
20
10
20
10
10
50
40
10
10
10
25
75
10
70

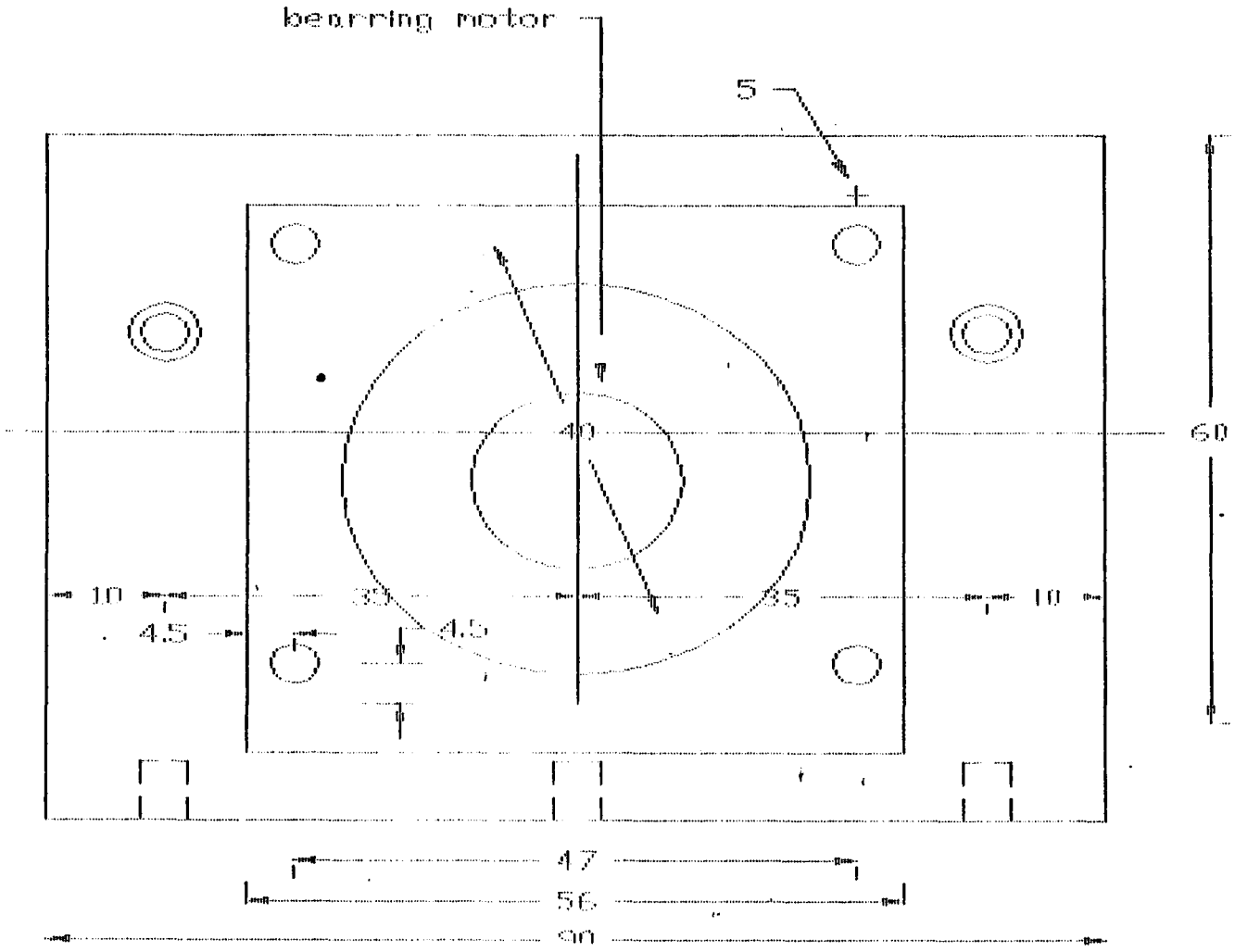

90'
8
15
45
15
10
10
10
10
70
5
10
20
25
25
20
15
15

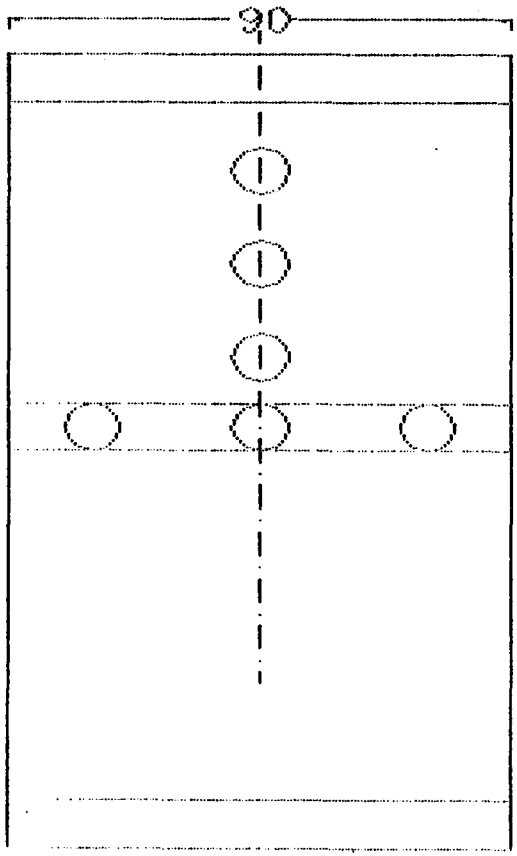

90
45
10
5
M3
M5
30

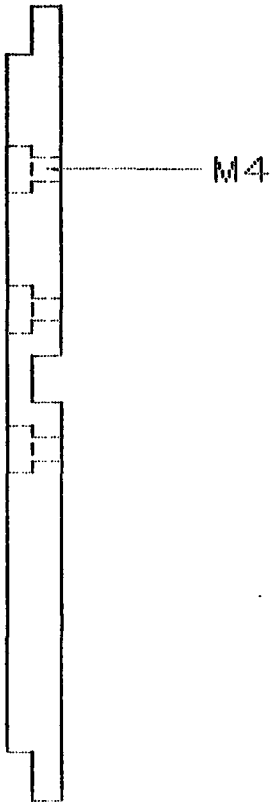

30
bearing iron axis
5
60
10
35
35
10
4.5
17
56
90


200
100
25
25
25
30
15
15
30

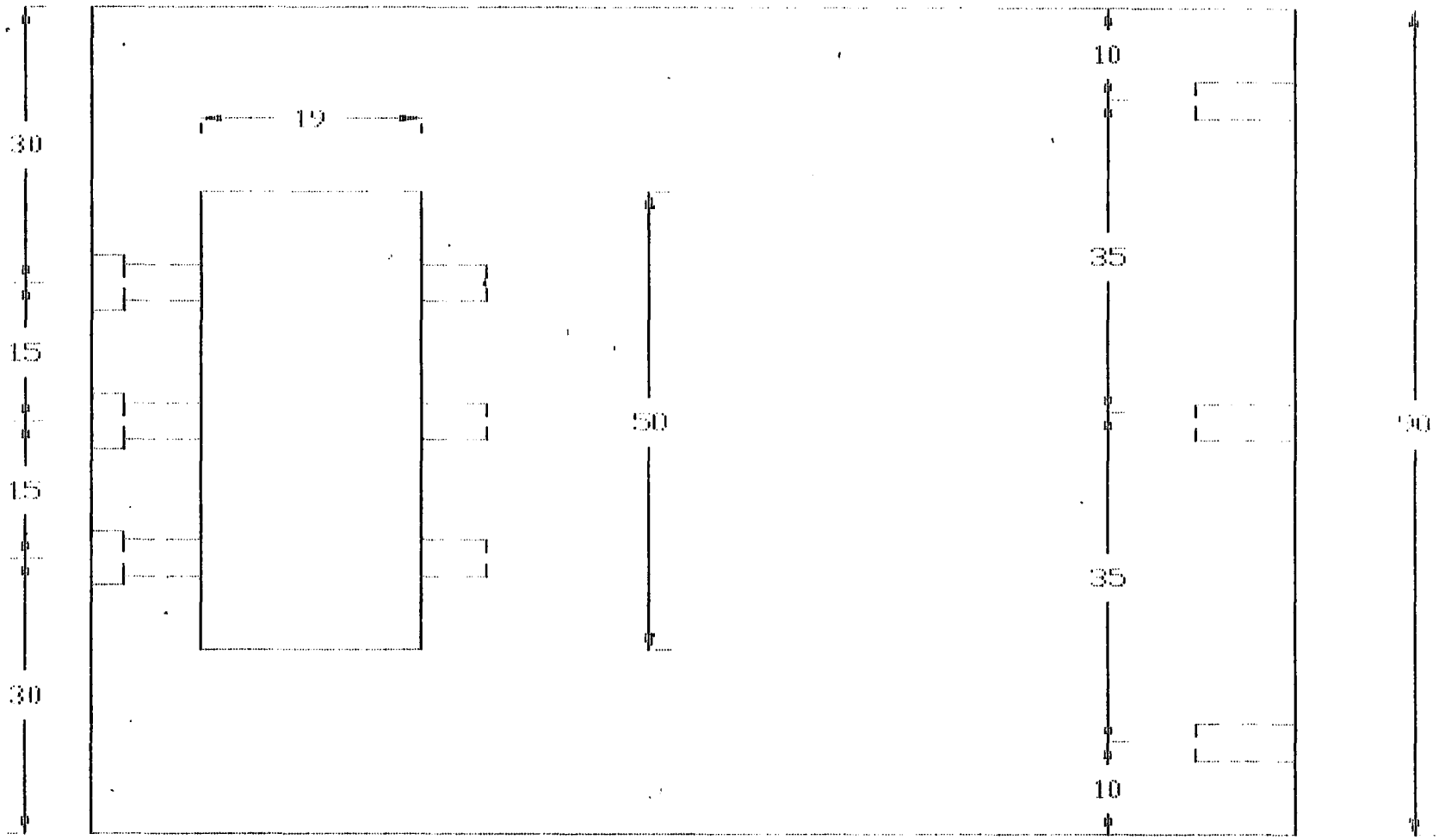

10
10
M3
6
10
10
70
10
35
35
10
M4
90


bearring motor
5
60
40
10
35
35
10
4.5
4.5
47
56
90

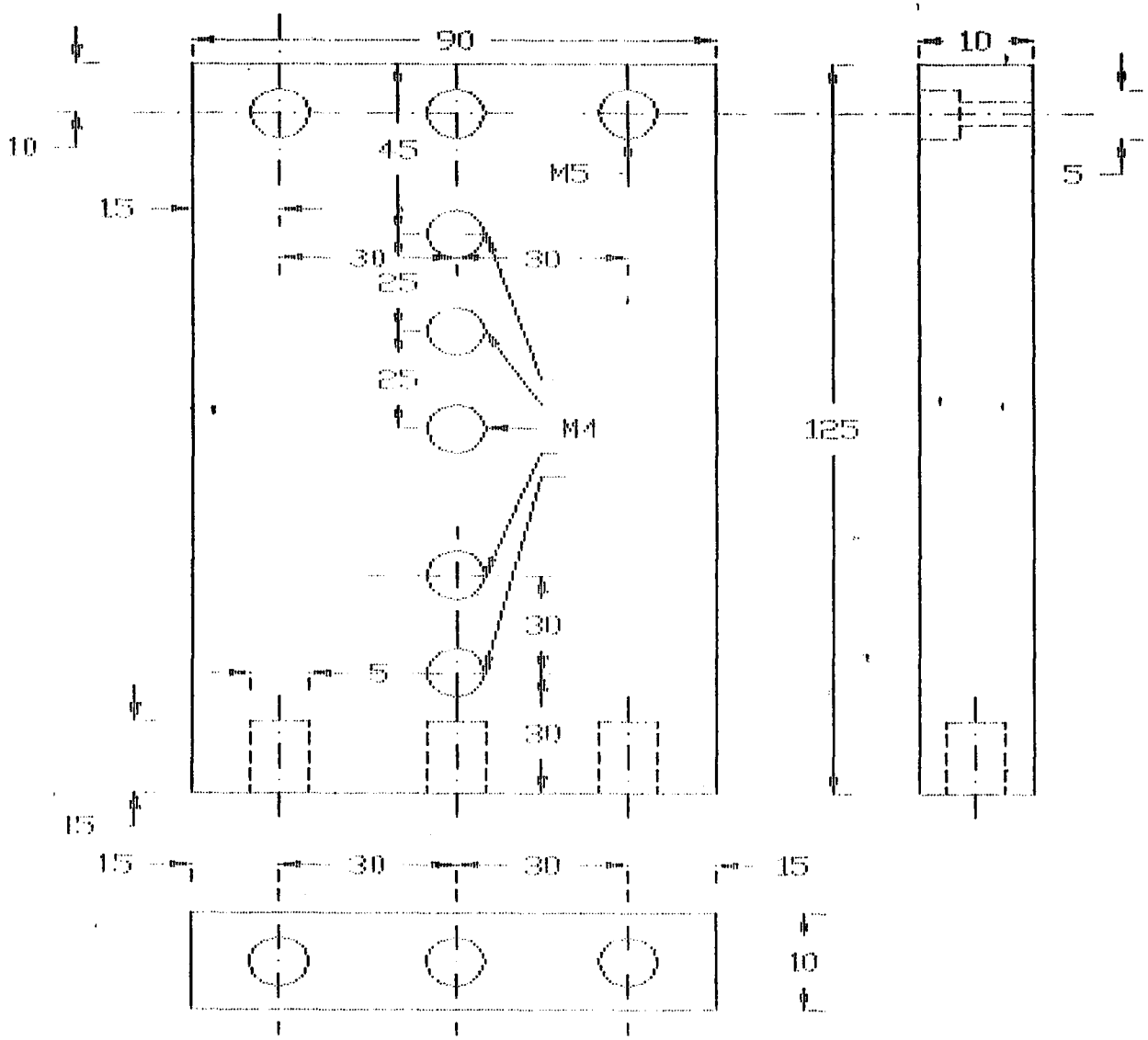


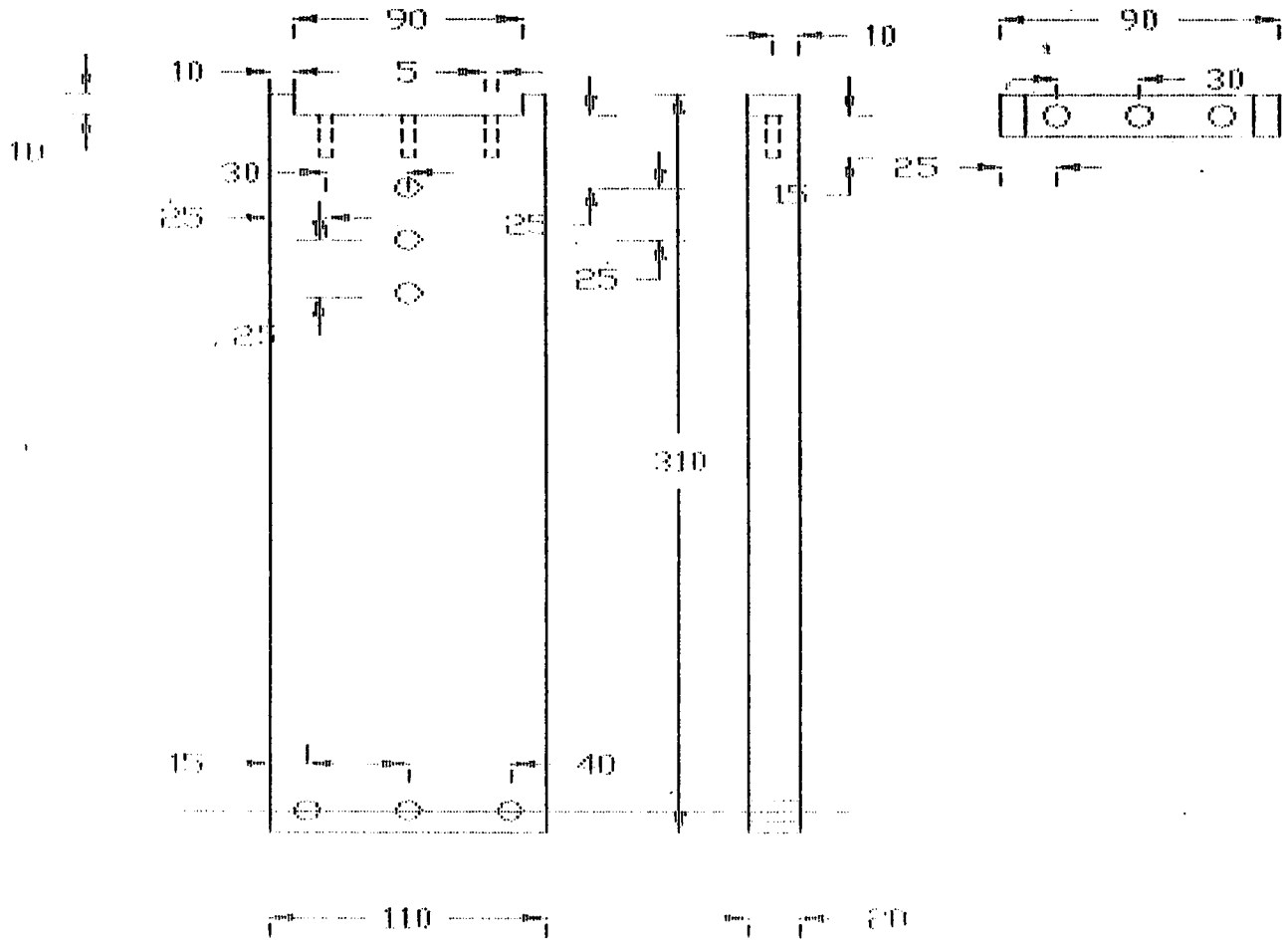


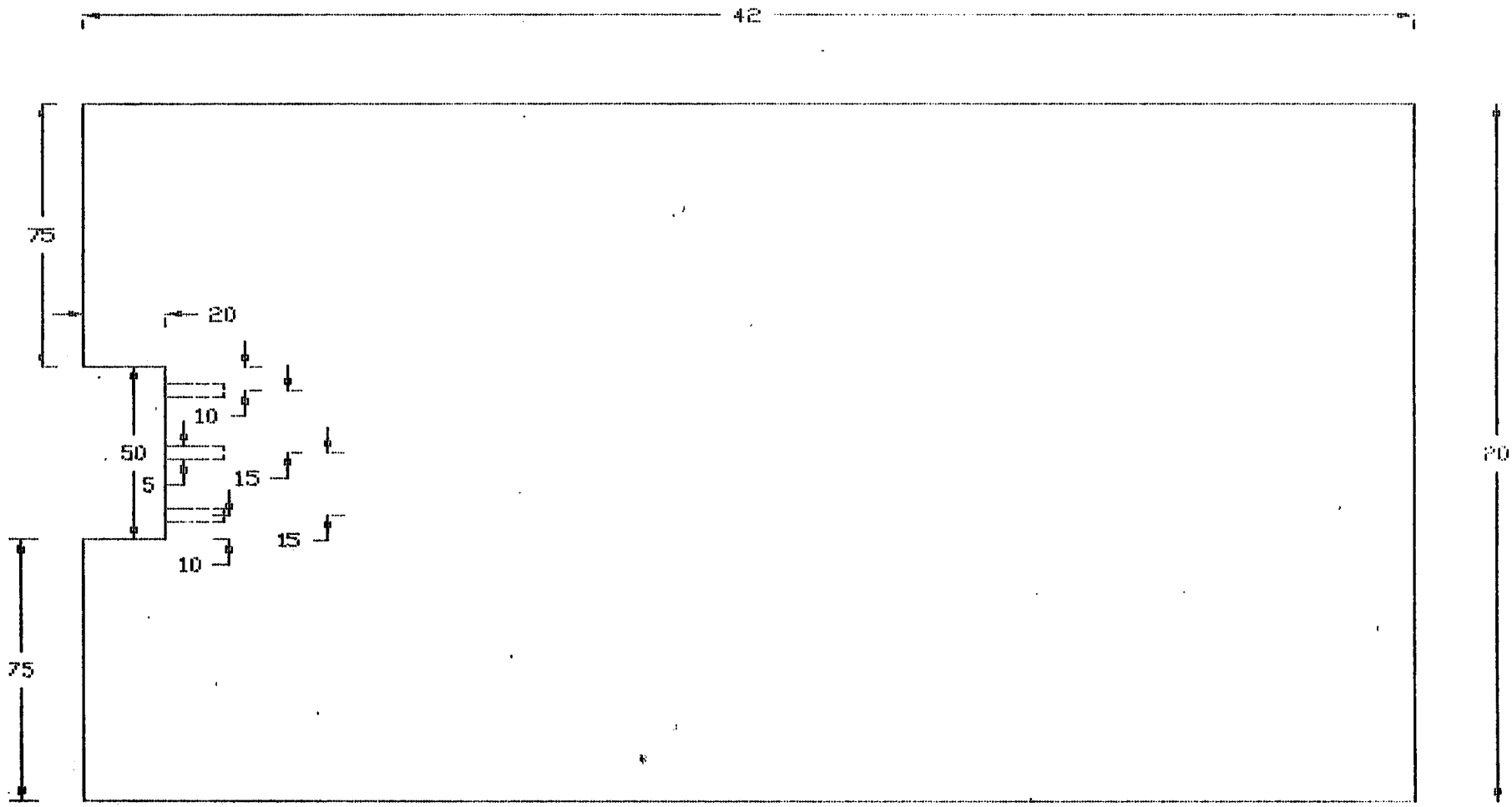
10

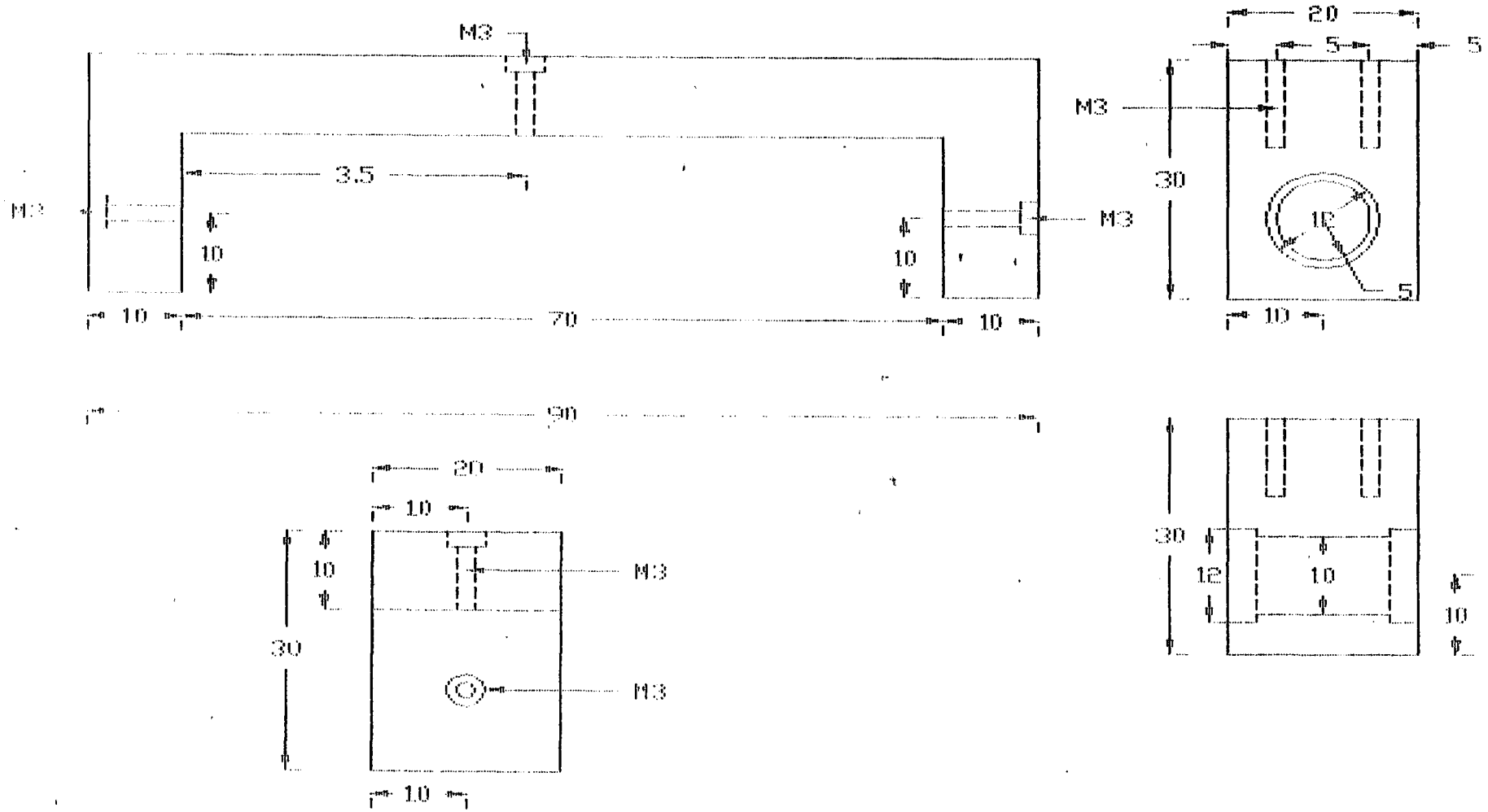

30
15
15
30
19
50
10
35
35
10
90


M5
P3
P5


90
10
10
45
M5
15
30
30
25
25
M4
125
5
30
5
30
15
15
30
30
15
10


90
10
5
10
30
25
25
25
310
15
40
110
10
90
30
25
15


42
75
20
10
50
5
15
15
10
75
70


M3
3.5
M3
10
10
M3
10
70
10
90
20
5
5
M3
30
10
5
10
20
10
10
M3
30
M3
10
30
12
10
10

เอกสารอ้างอิง

1.โยธิน เปรมปราณีรัชต์,ระบบเซอร์โว และ อิเล็กทรอนิกส์คอนโทรลมอเตอร์, ตำราชุดวิศวกรรมศาสตร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ,2533

2.จิติ หนูแก้ว,เทคนิคการเชื่อมต่อ IBM PC กับอุปกรณ์ภายนอกเพื่อประยุกต์ใช้งานต่าง ๆ,ซีเอ็ดยูเคชั่น,2533

3.นุกูล กระจาย,การเขียนโปรแกรมและประมวลผลข้อมูลด้วยเทอร์โบปาสคาล, ซีเอ็ดยูเคชั่น,2535

4.วัฒนา พลพละ,"กราฟิกพล็อตเตอร์ ขนาด A2/A3 ทำเองกับมือ", เซมิคอนดัคเตอร์ อิเล็คทรอนิกส์ ฉบับที่ 127(2536):25-36

5.OSG CORPORATION,Products Information Vol.54,Japan,1990

6.THK CO.,LTD.,Ball Screw and Suport Unit for Small OA Equipment,Japan 1992

7.THK CO.,LTD.,Ball Screws,Japan 1992