การออกแบบเพื่อควบคุมการเคลื่อนไหวของหุ่นยนต์โดยใช้ PC

DESIGNING FOR CONTROL MOBILE OF ROBOT WITH PC

โดย

นาย ชัยณรงค์ นุชน้อมบุญ

นาย วัฒนา มาบรรดิษ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

หัวข้อปริญญานิพนธ์	การออกแบบเพื่อควบคุมการเคลื่อนไหวของหุนยนต์โดยใช้ PC
ชื่อนักศึกษา	นาย ชัยณรงค์ นุชน้อมบุญ
	นาย วัฒนา มาบรรดิษ
อาจารย์ที่ปรึกษา	อาจารย์ มนชนก ศรีเสือขาม
ภาควิชา	เทคนิคอุตสาหกรรม
คณะ	วิศวกรรมศาสตร์
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา	2538

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การออกแบบเพื่อควบคุมการเคลื่อนไหวของหุ่นยนต์โดยใช้ PC

โดย นาย ชัยณรงค์ นุชน้อมบุญ รหัส 37012004
นาย วัฒนา มาบรรดิษ รหัส 37012027

อาจารย์ที่ปรึกษา อาจารย์ มนชนก ศรีเสือขาม

ปีการศึกษา 2538

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้อธิบายถึง ทฤษฎีหุ่นยนต์เบื้องต้นและการใช้งานหุ่นยนต์ ET-18 และการพัฒนาหุ่นยนต์ทั้งทางด้านฮาร์ดแวร์และซอฟท์แวร์ โดยการนำเอาคอมพิวเตอร์ส่วนบุคคลเข้ามาควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์ เพื่อวัตถุประสงค์ที่จะควบคุมส่วนต่างๆของหุ่นยนต์ให้ทำงานตามคำสั่งจากคอมพิวเตอร์โดยใช้โปรแกรมภาษาปาสคาลซึ่งข้อมูลจากคอมพิวเตอร์ถูกส่งผ่านมายังหุ่นยนต์โดยใช้สายสัญญาณ ความสามารถของโครงงานนี้คือควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์โดยใช้คีย์บอร์ดของคอมพิวเตอร์และควบคุมให้ส่วนต่างๆของหุ่นยนต์ทำงานแบบต่อเนื่องได้ ผู้ใช้งานสามารถพัฒนาทางด้าน โปรแกรมและฮาร์ดแวร์ ขึ้นไปอีกได้ตามความต้องการ

DESIGNING FOR CONTROL MOBILE OF ROBOT WITH PC

BY MR. CHAINARONG NOOTHNOMEBOON NO. 37012004

MR. WATTANA MABUNDIT NO. 37012027

ADVISER MISS. MONCHARNOK SRISEARKARM

YEAR 1996

ABSTRACT

This project describes the principle of mobile robot , and application ET-18 mobile robot , and development the mobile robot in hardware and solfware. The personal computer is used for control the part of robot . For the purpose to control operate the part of robot by command from PC with PASCAL program. Data from PC transmit to robot by data wire . The capability of project is control operate the part of robot by use keyboard and control the part of robot to operate continue. The user can developement software and hardware follow purpose.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ ได้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของ อาจารย์มนชนก ศรีเสือขาม ซึ่งเป็นที่ปรึกษาปริญญานิพนธ์นี้ ซึ่งท่านได้ให้คำแนะนำและข้อคิดเห็นต่างๆ ในการทำการวิจัย , ทดลอง , ค้นคว้าข้อมูล ตลอดจนเครื่องมือและอุปกรณ์ที่ใช้ในการทำปริญญานิพนธ์ ท้ายนี้ผู้วิจัยใคร่ขอกราบขอบพระคุณ บิดา-มารดา ซึ่งสนับสนุนในด้านการเงินและให้กำลังใจมาจนสำเร็จการศึกษา

นาย ชัยณรงค์ นุชน้อมบุญ

นาย วัฒนา มาบรรดิษ

สารบัญ

บทนำ

เนื่องจากปัจจุบันนี้ในประเทศที่พัฒนาแล้ว ได้มีการนำหุ่นยนต์มาใช้แทนมนุษย์อย่างมากมาย เช่น ในงานที่เสี่ยงอันตราย งานที่ต้องทำซ้ำ ๆ เป็นต้น เนื่องจากเทคโนโลยีทางด้านนี้แพงมาก จึงเป็นการยากที่จะสรรหามา โดยเฉพาะประเทศที่ที่กำลังพัฒนา ทางหนึ่งที่จะช่วยให้ก้าวทันเทคโนโลยี คือ การเร่งส่งเสริมการศึกษา การพัฒนาหุ่นยนต์ให้สามารถทำงานได้อย่างมีประสิทธิภาพตามนโยบายพึ่งตนเอง เพื่อเป็นแนวทางในการศึกษาและการพัฒนาขั้นต่อไป

1.1 ความสำคัญและที่มาของโครงงาน

ในปัจบันการใช้ไมโครคอมพิวเตอร์ไปประยุกต์ใช้งานนั้นมีอยู่แพร่หลาย และได้มีการนำเอาคอมพิวเตอร์เป็นตัวรับส่งข้อมูล และในโครงงานได้ใช้ 8255 CARD PORT เสียบเข้ากับ SLOT ของคอมพิวเตอร์ และต่อสายแพไปควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์ เพื่อประโยชน์ในการควบคุมส่วนต่างๆของหุ่นยนต์ให้ทำงานตามต้องการ

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาระบบต่างๆของหุ่นยนต์

1.2.2 เพื่อออกแบบการควบคุมการทำงานของหุ่นยนต์โดยใช้ Personal Computer (PC)

1.2.3 เพื่อพัฒนาทางด้านซอฟแวร์ และ ฮาร์ดแวร์ที่ใช้ในการควบคุมการทำงาน

1.3 ขอบเขตของการวิจัย

เนื้อหาของโครงงานวิจัย แบ่งออกเป็น 3 ส่วนใหญ่ๆคือ

1.3.1 ทฤษฎีและการใช้งานเบื้องต้นของหุ่นยนต์

1.3.2 ฮาร์ดแวร์ คือ ระบบต่างๆของหุ่นยนต์และการอินเตอร์เฟส

1.3.3 ซอร์ฟแวร์ คือโปรแกรมที่ใช้ควบคุมการทำงาน

1.4 การดำเนินงาน

เทอม 1

- ศึกษาทฤษฎีเบื้องต้นของหุ่นยนต์
- ศึกษาลักษณะโครงสร้างภายในของหุ่นยนต์
- ออกแบบวงจรควบตุมส่วนต่างๆของหุ่นยนต์
- ทดสอบการทำงานของวงจรควบคุมส่วนต่างๆ ของหุ่นยนต์

เทอม 2

- ศึกษาการอินเตอร์เฟสระหว่างคอมพิวเตอร์กับวงจรควบคุมการทำงานของหุ่นยนต์
- ศึกษาโปรแกรมที่ใช้ควบคุมการทำงานของหุ่นยนต์
- เขียนโปรแกรมควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์
- ทดสอบการทำงานของโปรแกรม
- หาแนวทางแก้ปัญหาที่เกิดขึ้นและแก้ไขปรับปรุงต่อไป

1.5 ปัญหาที่พบ

ปัญหาที่พบอันดับแรกคือความยุ่งยากในการออกแบบวงจรควบคุมส่วนต่างๆของหุ่นยนต์ซึ่งต้องศึกษาโครงสร้างภายในและส่วนประกอบต่างๆของหุ่นยนต์ และต้องมาศึกษาหาวิธีที่จะควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์ให้ทำงานได้ ประการที่สองคือการหาวงจรอินเตอร์เฟส (INTERFACE) ระหว่างคอมพิวเตอร์กับวงจรควบคุมการทำงานของหุ่นยนต์และประการที่สามคือการเขียนโปรแกรมควบคุมการทำงานส่วนต่างๆของหุ่นยนต์ซึ่งต้องอาศัยเวลาในการศึกษาและทำความเข้าใจ

1.6 ประโยชน์ที่ได้รับ

ประโยชน์ที่ได้รับคือสามารถใช้คอมพิวเตอร์ร่วกับ 8255 CARD PORT และวงจร DRIVER ควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์ได้ สามารถเข้าใจการทำงานของสเต็ปปิ้งมอเตอร์และ ดี ซี มอเตอร์ และสามารถเขียนโปรแกรมควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์ได้ซึ่งจะสามารถโปรแกรมให้ส่วนต่างๆ ของหุ่นยนต์ ทำงานตามที่ต้องการได้

บทที่ 1
การทำงานของสเต็ปเปอร์มอเตอร์

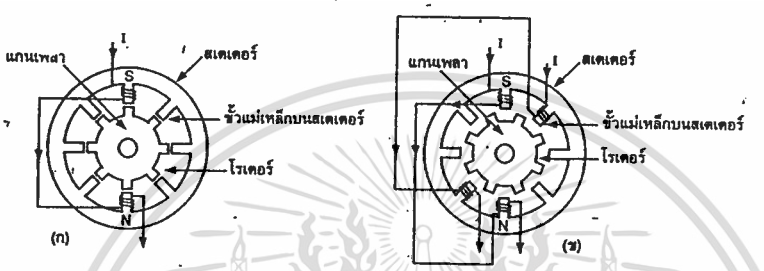
สเต็ปเปอร์มอเตอร์มีความแตกต่างจากมอเตอร์ทั่วๆไป โดยเมื่อป้อนกำลังไฟฟ้าให้กับมัน มันจะหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งแตกต่างจากมอเตอร์ทั่วๆไปซึ่งจะหมุนทันที และตลอดเวลา เมื่อป้อนแรงดันไฟฟ้า สเต็ปเปอร์มอเตอร์สามารถกำหนดตำแหน่งของการหมุนด้วยตัวเลขได้อย่างละเอียดโดยการใช้คอมพิวเตอร์เป็นตัวกำหนดและจัดเก็บตัวเลขเหล่านั้นไว้

สเต็ปเปอร์มอเตอร์สามารถใช้งานในระบบเปิด (open loop system) นั่นก็คือมันทำงานได้โดยไม่ต้องการการป้อนกลับ (feedback) แต่ทุกวิธีที่ต้องการกำหนดตำแหน่งได้อย่างถูกต้องจำเป็นที่ต้องการ การป้อนกลับไปยังระบบให้รับรู้ และตัวบอกตำแหน่งว่าถูกต้องหรือเกิดการผิดพลาด

วิธีหนึ่งที่ใช้กันโดยทั่วไปกับสเต็ปเปอร์มอเตอร์ คือ การใช้สวิทช์ติดตั้งไว้ที่ตำแหน่งที่ต้องการตรวจจับ (limit switch) เมื่อสเต็ปเปอร์มอเตอร์เริ่มหมุนและหมุนจนกระทั่งถึงตำแหน่งของสวิทช์ตรวจจับสัญญาณ ก็จะถูกป้อนกลับสู่ระบบและทราบการทำงานของสเต็ปเปอร์มอเตอร์ได้ตลอดเวลาซึ่งโดยปกติในวงจรคอนโทรลเลอร์จะมีการกำหนดจุดอ้างอิง (reference point) ไว้ด้วย เพื่อให้เริ่มต้นทำงานและอ้างอิงตำแหน่งได้อย่างถูกต้อง ตัวอย่างเช่น ถ้าเริ่มจ่ายกำลังไฟฟ้าให้กับฟล็อปปี้ดิสก์ไดรฟ์ จะได้ยินมันกำลังเคลื่อนที่เพื่อหาจุดอ้างอิงที่กำหนด หลังจากนั้นวงจรไดรฟ์คอนโทรลเลอร์จะเริ่มทำงานได้ โดยมันจะทราบถึงทุกๆสเต็ปที่กำลังขับเคลื่อนหัวอ่าน/ เขียนไปยังแต่ละแทร็คบนดิสก์ เช่นเดียวกับมอเตอร์ทั่วไปการที่จะทำให้เกิดการหมุนของโรเตอร์ (rotor) ได้ต้องมีการกระทำของสนามแม่เหล็กที่เกิดขึ้นระหว่างโรเตอร์และสเตเตอร์ (stator) ซึ่งขึ้นอยู่กับการจัดวางขั้วแม่เหล็ก (pole) การหมุนทำได้ทั้งแบบต่อเนื่องและกลับทิศทางไปมา โดยกระบวนการทางไฟสลับหรือการจัดวางแปรงถ่าน และการจัดแยกคอมมิวเตอร์ และทำการสวิทชิ่งกำลังไฟฟ้าทำให้เกิดแรงดึงดูดของแม่เหล็ก (magnetic attraction) ที่ขั้วแม่เหล็กสร้างและหยุดสลับกัน ผลก็คือเกิดสนามแม่เหล็กหมุนขึ้นบนสเตเตอร์โดยการจ่ายกำลังไฟฟ้าทีละคู่ของขั้วแม่เหล็กในทิศทางตรงกันข้ามไปตลอดเวลา และเมื่อต้องการหยุดหมุนทำได้โดยหยุดการเกิดขั้วแม่เหล็กที่จุดหนึ่งโดยหยุดการสวิชชิ่งในลำดับต่อไปเสีย การหมุนกลับทิศทางก็ทำได้เช่นเดียวกับที่กล่าวมาแล้วเพียงแต่ทำการสวิชชิ่งกำลังไฟฟ้าให้เกิดสนามแม่เหล็กหมุนในทิศทางกลับกันหรือกลับลำดับการสวิทช์ของมัน

โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ประกอบขึ้นจากแผ่นเหล็กวงแหวนที่มีซี่ยื่นออกมาแต่ละซี่เหล่านั้นจะมีคอยล์พันสวมอยู่ ดังนั้นเมื่อป้อนกระแสผ่านคอยล์ทำให้เกิดสนามแม่เหล็กไฟฟ้า (electromagnetic) ขึ้นด้านตรงข้ามของแต่ละขั้วแม่เหล็กจะได้รับกระแสไฟฟ้าในขณะเดียวกันแต่ว่าจะไหลวนในทิศทางตรงกันข้ามทำให้เกิดสนามแม่เหล็กไฟฟ้าในทิศตรงข้ามขึ้นดังแสดงในรูป

ที่ 1 (ก) ดังนั้นถ้าเพิ่มจำนวนของขั้วแม่เหล็กให้มากขึ้นจะเพิ่มจำนวนของสเต็ปต่อวงรอบมากขึ้นตามไปด้วย อย่างไรก็ตามผู้ใช้สามารถเพิ่มจำนวนของสเต็ปได้อีกวิธีหนึ่งโดยไม่ต้องปรับเปลี่ยนโครงสร้างภายในโดยทำการจ่ายกำลังไฟฟ้าไปยังขั้วแม่เหล็ก 2 ขั้วที่อยู่ใกล้กันในเวลาเดียวกันซึ่งจะทำให้โรเตอร์หยุดหมุนอยู่ระหว่างกลางของ 2 ขั้วแม่เหล็กนั้นหรือเคลื่อนที่ไปครึ่งสเต็ปเท่านั้น และวิธีการนี้ยังช่วยให้เกิดแรงบิด (torque) มากขึ้นด้วยดังแสดงในรูปที่ 1(ข)



รูปที่ 1 (ก) แสดงสเต็ปเปอร์มอเตอร์ที่มีการต่อวงจรขดลวดภายในเพื่อกระตุ้นให้เกิดขั้วแม่เหล็กขึ้น 1ขั้วในทิศทางตรงกันข้ามส่วนขดอื่นๆจะไม่ถูกกระตุ้นเลย

(ข)แสดงการต่อวงจรขดลวดแบบกระตุ้นให้เกิดขั้วแม่เหล็กพร้อมกัน 2ขั้วที่อยู่ใกล้กันทำให้โรเตอร์เคลื่อนที่มาหยุดอยู่ระหว่างขั้วแม่เหล็กทั้งสอง

สเต็ปเปอร์มอเตอร์โดยทั่วไปมีจำนวนของขั้วแม่เหล็กหรือจำนวนสเต็ปต่อรอบเป็นจำนวนมากปกติอยู่ที่ประมาณ 100 - 400 สเต็ปต่อรอบการมีจำนวนสเต็ปมากๆนี้ไม่ได้เพิ่มจำนวนขั้วแม่เหล็กไฟฟ้าที่สเตเตอร์ แต่ทำได้โดยเพิ่มจำนวนซี่ขั้วแม่เหล็กที่โรเตอร์ จำนวนสเต็ปต่อรอบทั้งหมดจะได้จากการคูณจำนวนขั้วแม่เหล็กบนสเตเตอร์และจำนวนซี่ที่โรเตอร์ ดังเช่น ถ้ามีขั้วแม่เหล็ก 3ขั้วบนสเตเตอร์ และ 8 ซี่ขั้วแม่เหล็กบนโรเตอร์ สเต็ปเปอร์มอเตอร์นี้จะทำงานที่ 24 สเต็ปต่อรอบหรือหมุนไปเป็นมุม 15 องศาต่อสเต็ป

การใช้วงจรดิจิตอลคอนโทรลเลอร์กำหนดการจ่ายกำลังไฟฟ้าเข้าสู่ขดลวดบนสเตเตอร์แบบซีเควนเชียลทำให้สามารถควบคุมการเคลื่อนที่ทุกสเต็ปได้เช่นเดียวกับการควบคุมในวงจรดีซีเซอร์โว (DC servo) แต่การควบคุมด้วยดิจิตอลไม่จำเป็นต้องมีการป้อนกลับ การเคลื่อนที่ทุกสเต็ปได้จากการคำนวณจำนวนรอบหรือมุมในการหมุนที่ต้องการ แล้วจึงส่งข้อมูลที่ได้ไปควบคุมการ

หมุนของมอเตอร์ พิกัดในการทำงานอาทิตความเร็ว , มุมในการเคลื่อนที่,ตำแหน่งของเพลาถูกกำหนดจากข้อมูลที่ส่งมาควบคุม

<u>ชนิดของสเต็ปเปอร์มอเตอร์</u>

สเต็ปเปอร์มอเตอร์แบ่งตามพื้นฐานได้เป็น 3 ชนิดคือ วาริเอเบิลรีลักแตนซ์ (variable reluctance:VR) , เพอร์มาเนนต์แมกเน็ต (permanent magnet:PM) และแบไฮบริดจ์ (hybrid)

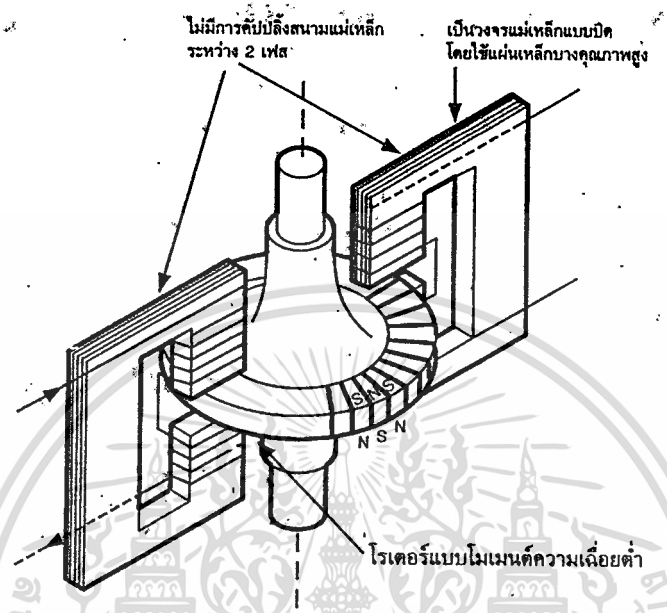
ชนิดวาริเอเบิลรีลักแตนซ์มีโครงสร้างของโรเตอร์แบบมัลติทูธ (multi-tooth) ทำจากเหล็กอ่อนเราจะทราบได้ว่าเป็นมอเตอร์ชนิดนี้โดยการทดสอบได้ง่ายมากคือใช้นิ้วหมุนเพลาของมอเตอร์และสังเกตมอเตอร์ชนิดนี้ที่โรเตอร์จะไม่เกิดปรากฏการณ์ทางแม่เหล็ก(magnetism)มันจึงหมุนได้ตลอดโดยไม่ติดขัดแตกต่างจากชนิด PM และชนิดไฮบริดซึ่งมีสนามแม่เหล็กที่โรเตอร์เมื่อหมุนจะรู้สึกขัดๆเหมือนเป็นฟันเฟืองสเต็ปเปอร์มอเตอร์ชนิดนี้มีจุดด้อยในเรื่องของความถูกต้องของตำแหน่งและทำงานได้ไม่ดีนักเมื่อมีสเต็ปในการหมุนสูง

ชนิดเพอร์มาเนนต์แมกเน็ตมีโครงสร้างของโรเตอร์แบบเรียบไม่มีขั้วแม่เหล็กและบนโรเตอร์จะเป็นแบบแม่เหล็กถาวร การควบคุมทำได้โดยป้อนกระกระตุ้นที่ขดลวดบนสเตเตอร์เช่นถ้าเป็นสแตอร์แบบ 4 เฟสจะมีขั้วแม่เหล็กอยู่ 4 ขั้วซึ่งมีคอยล์พันอยู่แยกจากกัน ขั้วแม่เหล็กถาวรบนโรเตอร์จะถูกดึงดูดจากขั้วแม่เหล็กบนสเตเตอร์เมื่อป้อนกระแสไฟฟ้าเข้าสู่ขดลวด และโรเตอร์จะอยู่คงที่ที่ขั้วแม่เหล็กบนสเตเตอร์นั้นถึงแม้ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป ทำให้เกิดเป็นแรงยึดหน่วงขึ้น สเต็ปเปอร์มอเตอร์ชนิดนี้มีข้อดีในเรื่องของความถูกต้องของตำแหน่ง และความเร็วมากขึ้นเมื่อเปรียบเทียบกับชนิดอื่น

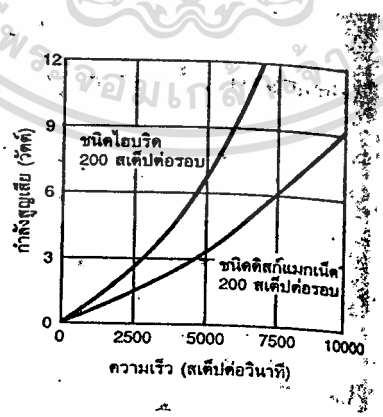
ชนิดไฮบริดเป็นชนิดที่นิยมใช้งานกันมากที่สุด โดยเฉพาะนำมาใช้งานอย่างมากในอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ ชนิดไฮบริดมีโครงสร้างภายในซึ่งได้จากการรวมเอาโครงสร้างของสเตเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ และโครงสร้างของโรเตอร์จากชนิดเพอร์มาเนนต์แมกเน็ตมาประกอบเข้าด้วยกันจึงทำให้เป็นมอเตอร์ชนิดที่มีแรงยึดหน่วงสูง,มีแรงบิดดึงและผลักได้ดีซึ่งมีความคงที่และทำงานได้ดีถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง

สเต็ปเปอร์มอเตอร์แบบใหม่อีกชนิดหนึ่งเป็นชนิดที่ปรับปรุงมาจากชนิดเพอมาเนนต์แมกเน็ตนั่นคือชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต (rare earth permanent magnet) ดังแสดงโครงสร้างภายในในรูปที่ 2 หรือที่เรียกกันว่าชนิดดิสก์แมกเน็ตสเต็ปเปอร์มอเตอร์ (disk magnet steppers) โครงสร้างของโรเตอร์ของมอเตอร์ชนิดนี้มีลักษณะเป็นแผ่นซึ่งยึดกับเพลาของมอเตอร์ การทำงานของมอเตอร์ยังคงเป็นเช่นเดิม แต่ด้วยโครงสร้างแบบใหม่นี้ช่วยทำให้เกิดโมเมนต์ของความเฉื่อยต่ำมาก , มีอัตราเร่งสูง มอเตอร์ชนิดนี้จึงจัดเป็นอีกชนิดหนึ่งที่มันก็มีประสิทธิภาพสูงอีกหลายด้าน

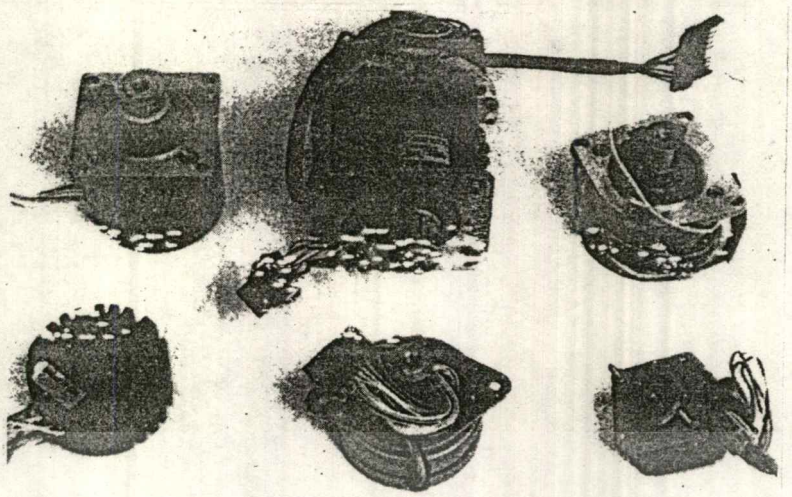
เช่น แรงบิดดึง , กำลังทางกลที่ได้ของมอเตอร์,ความถูกต้องของตำแหน่งสูงมาก และความเร็วในการเริ่มหมุนและหยุดหมุนสูง อีกทั้งยังมีความสูญเสียของกำลังงานต่ำดังแสดงในรูปที่ 3



รูปที่ 2 แสดงโครงสร้างภายในพื้นฐานของชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ตสเต็ปเปอร์มอเตอร์

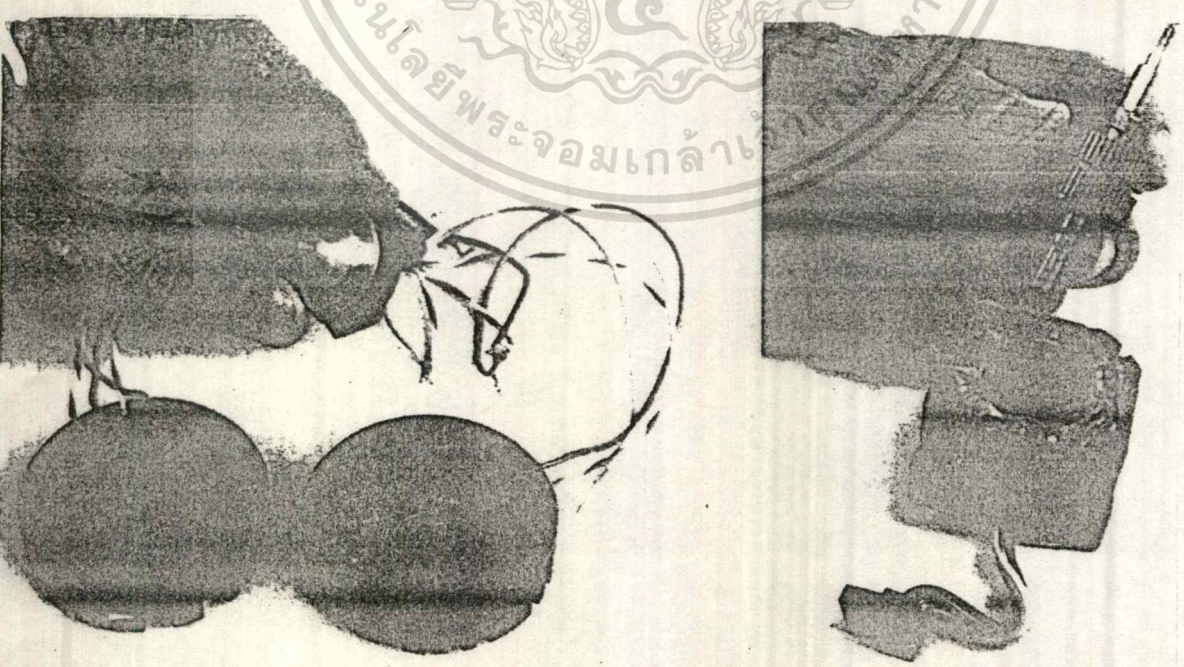


รูปที่ 3 แสดงกราฟข้อมูลการสูญเสียกำลังไฟฟ้า,ความเร็วในการหมุนโดยเปรียบเทียบระหว่างสเต็ปเปอร์มอเตอร์ชนิดไฮบริดและชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต



รูปที่ 4 แสดงรูปร่างหน้าตาตัวจริงของสเต็ปเปอร์มอเตอร์ชนิดต่างๆมอเตอร์เป็นชนิดวาริเอเบิลรีลักแตนซ์ซึ่งมีตัวเข้ารหัสต่ออยู่ด้านท้ายของเพลา

สำหรับรูปที่ 4 แสดงให้เห็นรูปร่างหน้าตาตัวจริงของสเต็ปเปอร์มอเตอร์แต่ละชนิดและรูปที่ 5 แสดงให้เห็นโครงสร้างภายในจริงเปรียบเทียบกันระหว่างสเต็ปเปอร์มอเตอร์ชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ตและชนิดไฮบริด ในรูปที่ 5(ก) จะเห็นได้ว่าที่ถืออยู่ในมือคือตัวโรเตอร์ซึ่งมีลักษณะเป็นแผ่นกลมติดกับเพลาและด้านล่างคือสเตเตอร์ซึ่งต่างจากชนิดไฮบริดในรูปที่ 5(ข) เห็นได้ว่าทั้งโรเตอร์และสเตเตอร์จะมีขี้ยื่นออกมาและในรูปจะเห็นขดลวดที่พันอยู่บนสเตเตอร์ได้อย่างชัดเจน



(ก) (ข)

รูปที่ 5 แสดงโครงสร้างภายในเปรียบเทียบระหว่างชนิดแรเอิร์ธเพอมาเนนต์แมกเน็ตและชนิดไฮ บริด

(ก) โครงสร้างภายในของชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต

(ข) โครงสร้างภายในของชนิดไฮบริด

การจำแนกชนิดของสเต็ปเปอร์มอเตอร์ด้วยการพันคอยล์

การพันขดลวดหรือคอยล์บนสเต็ปเปอร์มอเตอร์มีอยู่ 2 วิธีคือ แบบไบโพลาร์ (bipolar) และแบบยูนิโพลาร์ (unipolar) ดังแสดงในรูปที่ 6

รูปที่ 6 แสดงการพันขดลวดบนสเตเตอร์ของสเต็ปเปอร์มอเตอร์ ด้านซ้ายเป็นแบบไบโพลาร์ และที่เหลือเป็นแบบยูนิโพลาร์ซึ่งมีทั้งแบบ 5 สาย และ 6 สาย 4 เฟส

สเต็ปเปอร์มอเตอร์แบบไบโพลาร์มีการพันขดลวด 1ขดบนแต่ละขั้วแม่เหล็กของสเตตอร์ ขั้วแม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้าและสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามได้โดยการกลับทิศทางการไหลของกระแสไฟฟ้าซึ่งการกำหนดทิศการไหลและการการกลับทิศทางของกระแสไฟฟ้าทำได้โดยการใช้วงจรสวิทชิงกลับขั้วไฟฟ้า

สำหรับยูนิโพลาร์จะมีการพันขดลวด 2 ขด บนแต่ละขั้วแม่เหล็กของสเตเตอร์ ซึ่งแต่ละขดจะทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามกัน การกลับขั้วแม่เหล็กเปลี่ยนไปมาทำได้โดยการสวิทชิงกระแสไฟฟ้าจากขดลวดขดหนึ่งไปยังอีกขดหนึ่งแทนเท่านั้น โดยปกติขดลวดทั้งสองจะมีการเชื่อมต่อกันหรือมีจุดร่วมเพื่อลดจำนวนของสายไฟที่ต่อจากมอเตอร์ วงจรจ่ายกำลังไฟฟ้าของมอเตอร์แบบยูนิโพลาร์ทำได้ง่ายกว่าชนิดไบโพลาร์ เพราะมันต้องการเพียงสวิทช์ธรรมดาในการเปิดและปิดกำลังไฟฟ้าให้กับขดลวดบนสเตเตอร์ในทิศทางที่ต้องการให้หมุนได้ทันที รูปที่ 7 แสดง

วงจรจ่ายกำลังไฟฟ้าซึ่งใช้ทรานซิสเตอร์ทำหน้าที่เป็นตัวสวิทชิงให้กับสเต็ปเปอร์มอเตอร์ ที่ มีการพันขดลวดทั้ง 2 แบบ จะเห็นได้ว่าในแบบของยูนิโพลาร์ เป็นวงจรที่ง่ายและไม่มีความซับซ้อนเลย

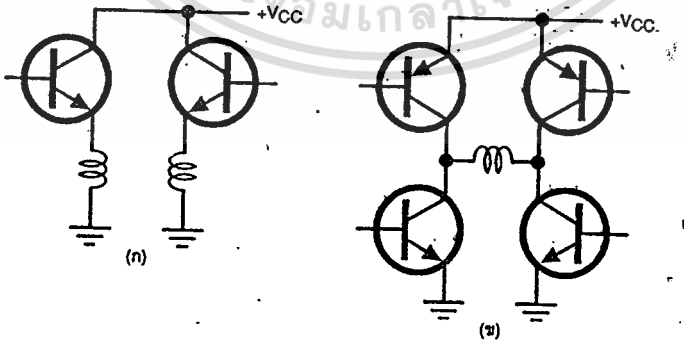
อย่างไรก็ตามการพันขดลวดแบบยูนิโพลาร์ก็มีจุดด้อยตรงที่การพันแบบนี้จะทำให้เกิดแรงบิดน้อยกว่าแบบไบโพลาร์เพราะจะมีเพียงครึ่งหนึ่งของขดลวดที่ถูกกระตุ้นให้ทำงานเท่านั้นในระยะเวลาหนึ่ง

การพิจารณาว่าสเต็ปเปอร์มอเตอร์ตัวใดมีการพันขดลวดแบบใดสังเกตได้ง่ายโดยถ้าเป็นแบบไบโพลาร์จะมีสายไฟต่อออกจากมอเตอร์เพียง 4 สาย และถ้าเป็นแบบยูนิโพลาร์จะมี 5 สาย หรือ 6 สายหรือทราบได้โดยการอ่านจากป้าย (name plate) ที่ติดอยู่กับมอเตอร์ได้

<u>การกระตุ้นและควบคุมการหมุนของสเต็ปเปอร์มอเตอร์</u>

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบชีเควนเชียลในรูปแบบที่ถูกต้องแบ่งได้เป็น 3 รูปแบบ คือ แบบเวฟ (wave) , แบบ 2 เฟส (two phase) และแบบครึ่งสเต็ป (half step) ทั้ง 3 แบบต่างก็มีข้อดีและข้อเสียแตกต่างกันออกไป

แบบเวฟเป็นการกระตุ้นรูปแบบที่ง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งและเรียงถัดกันไป ดังเช่นขดที่ 1,2,3,4,1 หรือ 1,4,3,2,1 ขึ้นอยู่กับทิศทางที่ต้องการให้หมุน ดังนั้นจึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆแสดงดังในตารางที่ 1



รูปที่ 7 แสดงวงจรจ่ายกำลังไฟฟ้าให้กับสเต็ปเปอร์ทั้ง 2 แบบ

(ก) สำหรับชนิดยูนิโพลาร์ ซึ่งใช้ทรานซิสเตอร์สวิทช์เพียงตัวเดียวต่อ 1 คอยล์

(ข) สำหรับชนิดไบโพลาร์ ซึ่งต้องใช้ทรานซิสเตอร์สวิทช์ 4 ตัวต่อ 1 คอยล์

ตารางที่ 1 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบเวฟ

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

แบบ 2 เฟสเป็นการกระตุ้นอีกรูปแบบหนึ่งซึ่งคล้ายกับแบบเวฟแต่การกระตุ้นแบบนี้จะทำการโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกันและเรียงถัดกันไปเช่นเดียวกับแบบเวฟคือขดลวดที่ถูกกระตุ้น 12,23,34,41,12 หรือ 14,43,32,21,14 ขึ้นอยู่กับทิศทางการหมุน การเพิ่มจำนวนของขดลวดที่ถูกกระตุ้นนี้ทำให้เพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ได้ด้วยแรงดึงแบบเต็มแรง จาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกันและต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือ การกระตุ้นแบบนี้ต้องใช้แหล่งจ่ายกำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆแสดงดังในตารางที่ 2

ตารางที่ 2 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบ 2 เฟส

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

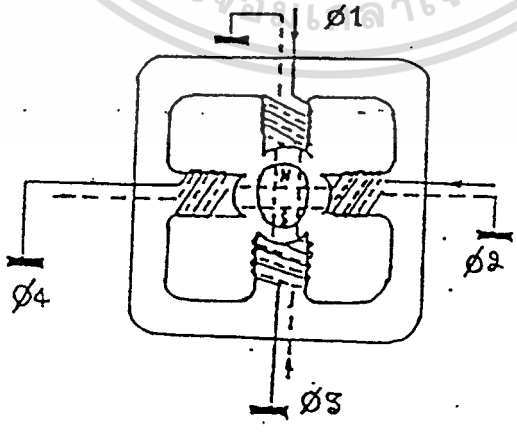
แบบครึ่งสเต็ปเป็นรูปแบบที่เกิดจากการผสมผสานระหว่างการกระตุ้นแบบเวฟและแบบ 2 เฟสเพื่อเพิ่มจำนวนจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ ขดลวดที่ถูกกระตุ้น 1,12,2,23,3,34,4,41,1 หรือในการหมุนอีกทิศทางหนึ่งจะได้เป็น 1,14,4,43,3,32,2,21,1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลงและแต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้นแต่ต้องพึงระวังไว้อีกประการหนึ่งว่าเมื่อถูกกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2 สเต็ปจึงจะได้เท่ากับ 1 สเต็ปเต็มเหมือนกับในการควบคุมแบบ 2 แบบแรก

สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้เทียบเท่ากับแบบ 2 เฟสจึงจะเพียงพอ ขั้นตอนการทำงานต่างๆดังแสดงในตารางที่ 3

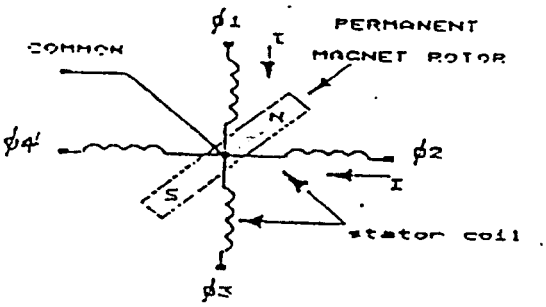
ตารางที่ 3 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบครึ่งสเต็ป

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	-
8	ทำงาน	-	-	ทำงาน

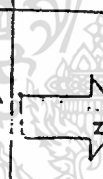
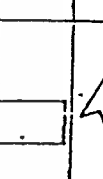
รูปแสดงโครงสร้างของ STEPPING MOTOR แบบหลายขั้ว



รูปแสดงโครงสร้างจำลองของ STEPPING MOTOR ขนิด 4 เฟสและตำแหน่งของโรเตอร์ขณะจ่ายกระแสไฟฟ้าแก่ เฟส 1 และ เฟส 2



รูปแสดงมุมของโรเตอร์เทียบกับกระแสไฟฟ้าที่จ่ายแก่ เฟสต่างๆ 8 ตำแหน่ง

เฟสที่จ่ายกระแสไฟฟ้า	ϕ1	ϕ1ϕ2	ϕ2	ϕ2ϕ3	ϕ3	ϕ3ϕ4	ϕ4	ϕ4ϕ1
ตำแหน่งโรเตอร์								

จากลักษณะของมุมโรเตอร์หมุนกับกระแสไฟที่ป้อนแก่เฟสต่างๆเราจะสามารถสั่งงานให้ STEPPING MOTOR หมุนได้ 3 อย่างคือ

1) แบบจ่ายกระแสไฟให้เฟสเดียววนเวียนกันไปเรียก ONE-EXCITATION หรือ HALF DRIVE คือ 01,02,03,04 การ OUT EXITATION แบบนี้แรงบิดจะน้อย

2)แบบจ่ายกระแสไฟให้พร้อมกันทีดียว 2 เฟส เรียก TWO-EXITTATION หรือ FULL STEP 0102,0203,0304,0401 หมุนเวียนกันไปแบบนี้แรงบิดจะมาก

3)แบบจ่ายกระแสไฟให้ทีละ 1 เฟสสลับกัน 2 เฟส เรียก ONE-TWO EXCITATION หรือ HALF STEPเหมือนรูปแสดงมุมของโรเตอร์ แต่แบบนี้จำนวน STEP จะเพิ่มขึ้นเป็น 2 เท่าของ 2 แบบแรก แต่แรงบิดเฉลี่ยจะน้อย

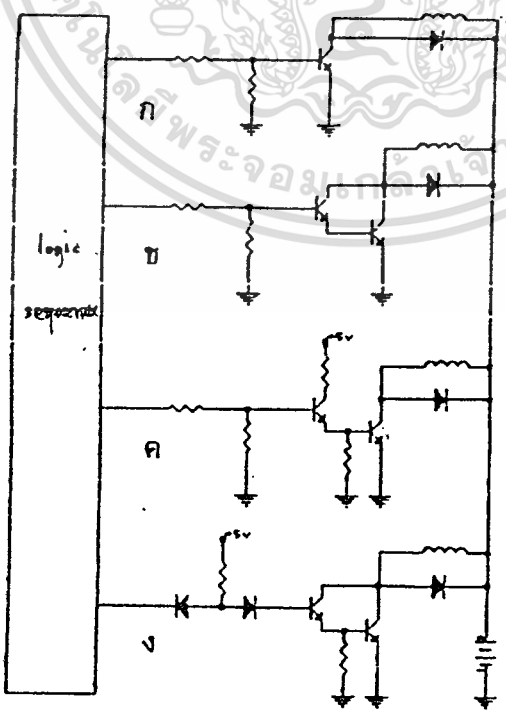
จากการจ่ายกระแสให้เฟสทั้ง 3 อย่าง เราก็สามารถสั่งให้ STEPPING MOTOR หมุนทวนเข็มได้โดยมองการจ่ายกระแสให้เฟสย้อนกลับ เช่น ตามเข็มนาฬิกาแบบ 1 0 เป็น

04	03	02	01	
0	0	0	1	01
0	0	1	0	02
0	1	0	0	04
1	0	0	0	08

ทวนเข็มจะเป็น

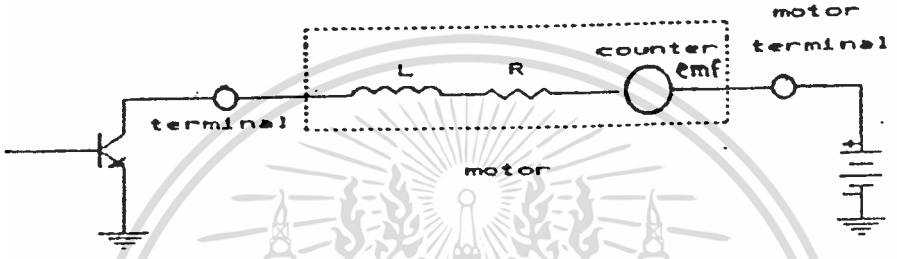
08 04 02 01

เมื่อรู้ซีเควนส์ของมันแล้ว ต่อไปเราก็ต้องมีวงจร DRIVER ให้แก่ STEPPING MOTOR วิธีง่ายที่สุดในการต่อวงจรซีเควนซ์เข้ากับวงจร DRIVER คือ การต่อโดยตรง ดังรูป ก และ ข แต่ถ้ากระแส OUTPUT ของวงจรซีเควนส์ไม่เพียงพอเราต้องต่อ BUFFER เพื่อขยายกระแสดังรูป ค และ ง



ปัญหาเกี่ยวกับวงจร DRIVER

ขดลวดของ STEPPING MOTOR เป็น INDUCTIVE และมีค่าเปรียบเสมือนผลรวมของอินดัคแตนซ์อนุกรมกับความต้านทานดังรูป

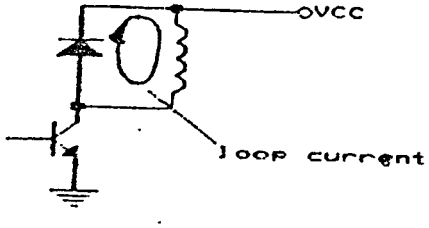


วงจรสมบูรณ์ของขดลวด STEPPING MOTOR

ซัพเพรสเซอร์

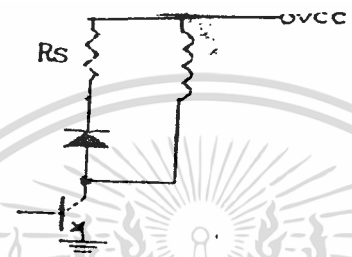
เมื่อ TRANSISTOR หยุดนำกระแส จะทำให้เกิด VOLTAGE ค่าสูงจำนวนหนึ่ง เนื่องจากผลของการเปลี่ยนแปลงของกระแสในอินดัคแตนซ์ และ VOLTAGE นี้อาจจะเป็นอันตรายแก่ TRANSISTOR ได้ วิธีป้องกัน

1)ไดโอดซัพเพรสเซอร์ ดังรูป



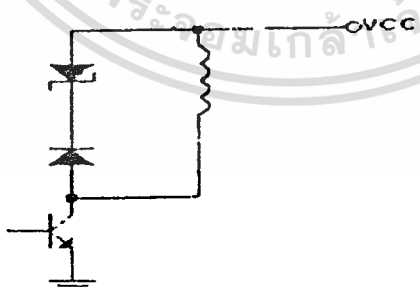
กระแสหมุนเวียน CIRCULATING CURRENT จะเริ่มไหลหลังจาก TRANSISTOR หยุดกระแสและศักดา COLLECTOR จะเท่ากับศักดาของแหล่งจ่าย ข้อเสียคือ กระแสจะหมุนเวียนอยู่นานและจะทำให้เกิดแรงบิดห้ามล้อ (BREAKING TORQUE) พลังงานส่วนใหญ่สูญเสียในความต้านทานของขดลวด มีปัญเรื่องทำความเย็น

2)ไดโอดและรีจิสเตอร์ชัพเพรสเซอร์ ดังรูป



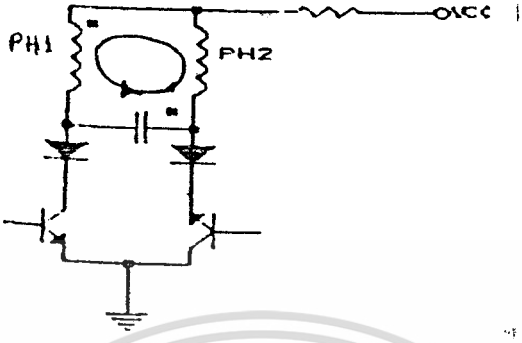
ถ้า RS ยิ่งมากกระแสหมุนเวียนก็จะลดลงเร็วขึ้น แต่ศักดาของ COLLECTOR จะมีค่าสูงขึ้นพลังงานส่วนใหญ่สูญเสียใน RS

3)ซีเนอร์ไดโอดชัพเพรสเซอร์ ดังรูป



เมื่อ TRANSISTOR CUTOFF กระแสจะลดลงได้เร็วกว่า 2 แบบแรก และศักดาที่ COLLECTOR จะเท่ากับ ศักดาของซีเนอร์บวกกับศักดาของแหล่งจ่าย ซึ่งเป็นอิสระต่อกระแสพลังงานส่วนใหญ่สูญเสียในซีเนอร์

4)คอนเดนเซอร์ชัพเพรสเซอร์ ดังรูป

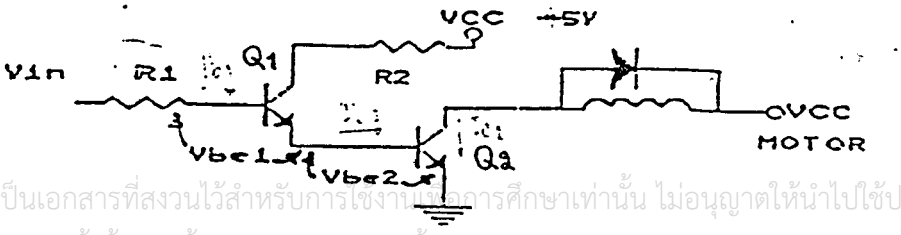


จะใส่ CONDENSER 01 กับ 03และ 02 กับ 04 เมื่อ TRASISTOR หยุดนำกระแส CONDENSER จะต่อกับ TRANSISTOR โดยผ่านทาง DIODE และจะดูดกลืน กระแสที่ค่อยๆลดลงจากขดลวดของ มอเตอร์เพื่อป้องกันทรานซิสเตอร์เสียและยังช่วยลดความร้อนที่เกิดขึ้นในขดลวดสเตเตอร์เนื่องจากการ ออสซิลเลท (OSCILLATE) ของโรเตอร์ (ROTOR)

<u>การแก้ไขการเพิ่มขึ้นของกระแส</u>

เมื่อทรานซิสเตอร์นำกระแสเพื่อกระตุ้นเฟส แหล่งจ่ายไฟ (POWER SUPPLY) จะต้องชนะผลของอินดัคแตนซ์ของขดลวดก่อนที่กระแสจะเพิ่มขึ้นได้ถึงค่าที่กำหนด คือ อินดัคแตนซ์มี คุณสมบัติที่จะต้านการเพิ่มขึ้นของกระแสต่อไซเคิล จะมีค่ามากขึ้น และเป็นผลให้แรงบิด (TORQUE) ลดลง และผลตอบสนองลดลง วิธีการลดเวลาการเพิ่มขึ้นของกระแส และปรับปรุงคุณลักษณะของแรงบิดที่มีความเร็วสูงให้ดีขึ้น โดยต่อความต้านทาน R อนุกรมกับขดลวดของ มอเตอร์ซึ่งต่อกับแหล่งจ่ายไฟตรง จะต้องเลือกค่าความต้านที่เหมาะสม เพื่อให้ได้กระแสที่ต้องการไหลผ่านขดลวดที่มีสภาวะคงที่ TIME CONSTANCE ของวงจรจะลดลงเนื่องจากผลรวมของความต้านทานในขดลวดเพิ่มขึ้น

การไบแอส(BIAS)วงจรภาคขยายกระแสไฟให้ได้กระแสแต่ละเฟสตามต้องการ



สมมุติให้ B2 คือค่าขยายกระแสต่ำสุดของ Q2 และ IC2 คือกระแสขับสูงสุดต่อฟส จะได้

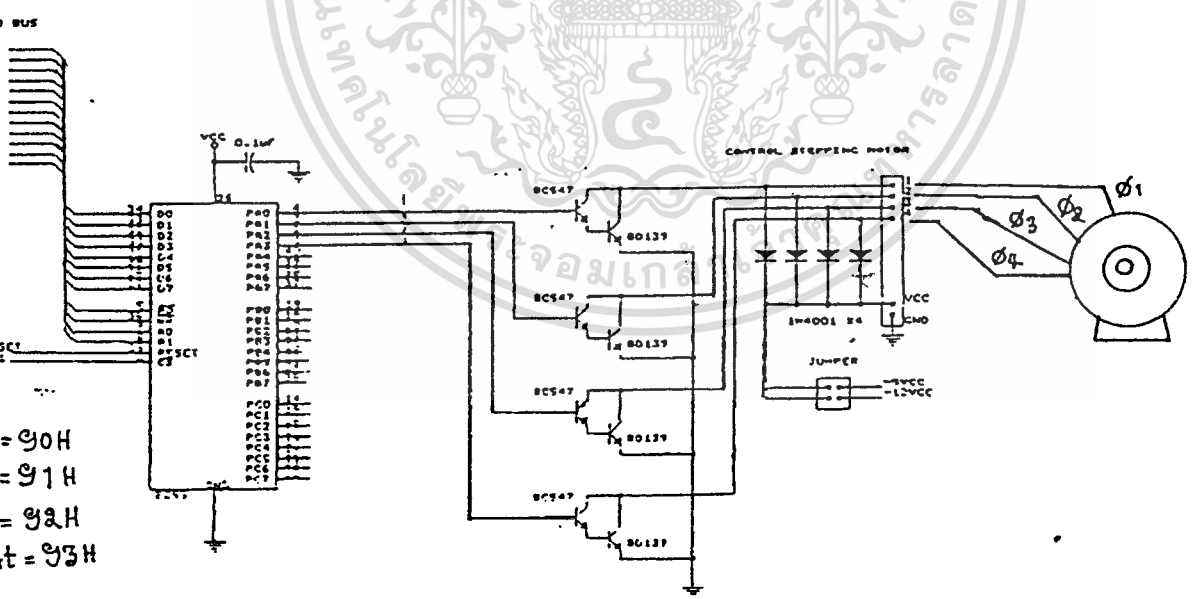
$$R2 = \frac{(5 - Vbe2)}{Ib2} = \frac{(5 - Vbe2)^* B}{Ic2}$$

สำหรับ R1 สมมุติให้ B คือ ค่าขยายกระแสต่ำสุดของ Q1 และ VIN มี ค่าประมาณ 4 โวลท์ สำหรับ ลอจิก(LOGIC) "1" จะได้

$$R1 = \frac{(4 - Vbe1 - Vbe2)}{Ib1} = \frac{(4 - Vbe1 - Vbe2)\ ^*B1}{Ic1}$$

$$R1 = \frac{2.8\ B1}{Ib2}$$

รูปแสดงการต่อวงจรภาคขับสเต็ปปิ้งมอเตอร์ (STEPPING MOTOR DRIVER)



บทที่ 2

CARD PORT

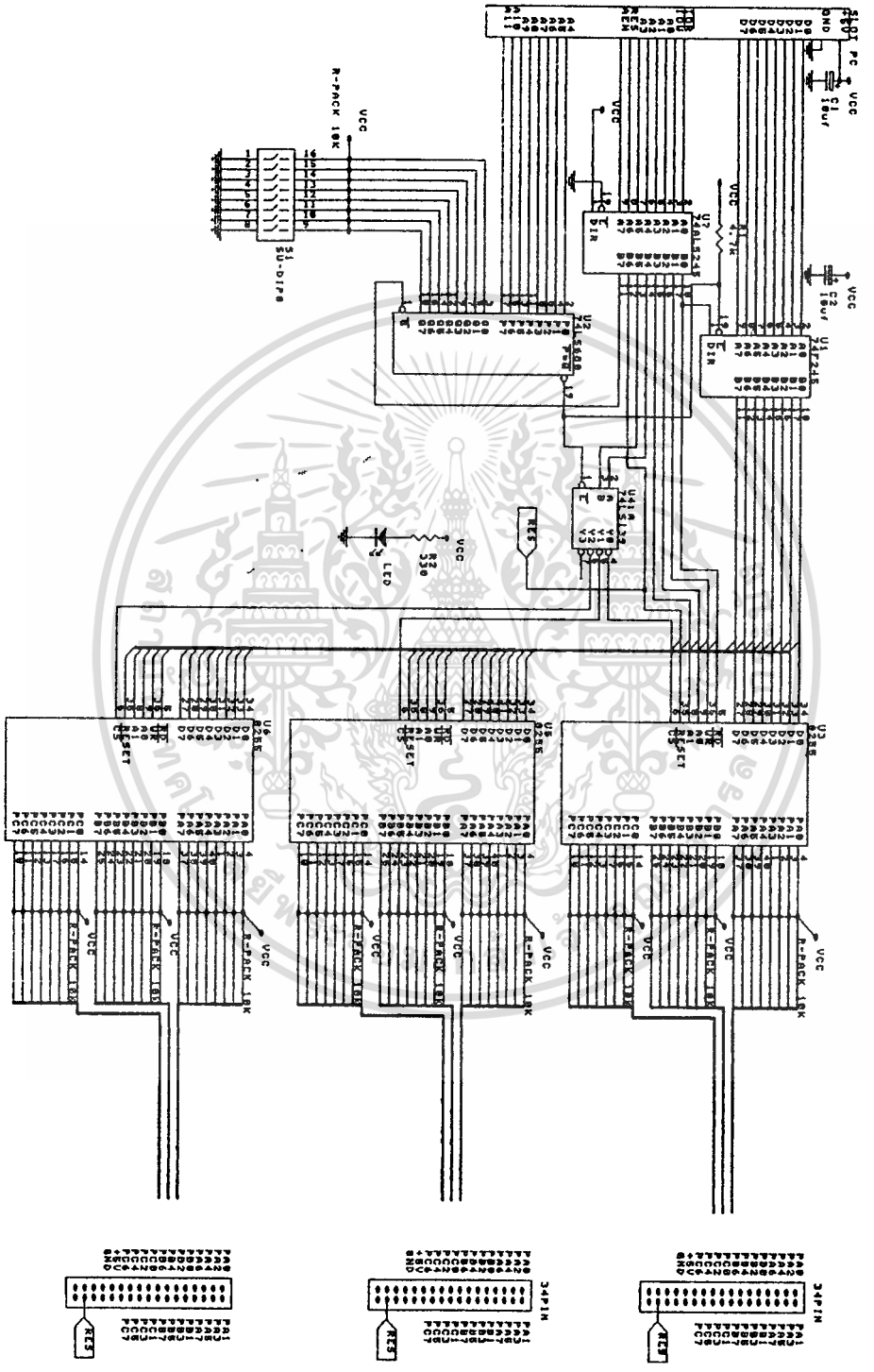
การควบคุมอุปกรณ์ไฟฟ้า โดยใช้เครื่องคอมพิวเตอร์นั้น จะต้องทำให้คอมพิวเตอร์ติดต่อกับอุปกรณ์ภายนอกให้ได้เสียก่อน แต่การที่จะทำเช่นนั้นได้ต้องผ่านอุปกรณ์ตัวหนึ่งเสียก่อน ซึ่งเรียกว่า พอร์ท (PORT) ซึ่งมีหลายลักษณะด้วยกัน แต่ในที่นี้จะใช้ IC#8255 ซึ่งเป็นพอร์ทขนาน ซึ่งทำมาในรูปของ CARD ประกอบด้วยส่วนประกอบต่างๆ รวมอยู่ใน CARD แผ่นเดียว ซึ่งจะกล่าวในรายละเอียดต่อไป

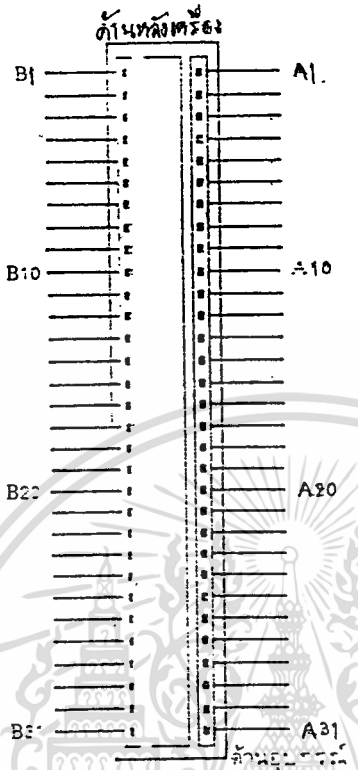
อุปกรณ์บน CARD

IC#8255	3	ตัว
IC#74LS139	1	ตัว
IC#74F245	1	ตัว
IC#74LS245	1	ตัว
IC#74LS688	1	ตัว
CONNECTOR 34 PIN	3	ตัว
DIP SW. 8 จุด	1	ตัว
RESISTOR-PACK 10 K	11	ตัว
RESIRTOR 330	1	ตัว
REISITOR 4.7 K	1	ตัว
CAPASISTOR 0.1 uF	1	ตัว
CAPASISTOR 33 uF 16 V	2	ตัว

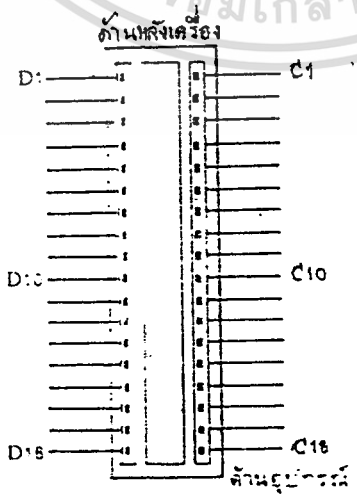
ใน CARD นี้จะมี 2 ส่วนใหญ่ด้วยกัน ซึ่งจะเป็นส่วนของวงจร DECODE และส่วนของ PORT 8255 ซึ่งจะอธิบายในรายละเอียดต่อไป

ที่เครื่องคอมพิวเตอร์นั้นจะมีช่องเสียบเรียกว่า SLOT PC โดยการเอา CARD มาเสียบที่ SLOT นี้ เพื่อเป็นการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์ภายนอก





การนับขาของสล็อตแบบ 62 ขา



การนับขาของสล็อตแบบ 36 ขา

ชื่อของสัญญาณขาต่างๆของ SLOT

ขา	อินพุต / เอาท์พุท	ชื่อสัญญาณ	อินพุต / เอาท์พุต
A1		- I/O CH CK	I
A2		SD7	I/O
A3		SD6	I/O
A4		SD5	I/O
A5		SD4	I/O
A6		SD3	I/O
A7		SD2	I/O
A8		SD1	I/O
A9		SD0	I/O
A10		- I/O CH RDY	I
A11		AEN	O
A12		SA19	I/O
A13		SA18	I/O
A14		SA17	I/O
A15		SA16	I/O
A16		SA15	I/O
A17		SA14	I/O
A18		SA13	I/O
A19		SA12	I/O
A20		SA11	I/O
A21		SA10	I/O
A22		SA9	I/O
A23		SA8	I/O

ขาอินพุท / เอาท์พุต	ชื่อสัญญาณ	อินพุต / เอาท์
A24	SA7	I/O
A25	SA6	I/O
A26	SA5	I/O
A27	SA4	I/O
A28	SA3	I/O
A29	SA2	I/O
A30	SA1	I/O
A31	SA0	I/O
B1	GND	กราวน์
B2	RESET DRV	O
B3	+ 5 VDC	แหล่งจ่ายไฟเลี้ยง
B4	IRQ9	I
B5	- 5 VDC	แหล่งจ่ายไฟเลี้ยง
B6	DRQ2	I
B7	- 12 VDC	แหล่งจ่ายไฟเลี้ยง
B8	OWS	I
B9	+ 12 VDC	แหล่งจ่ายไฟเลี้ยง
B10	GND	กราวน์
B11	- SMEMW	O
B12	- SEMER	O
B13	- IOW	I/O
B14	- IOR	I/O
B15	- DACK3	O
B16	DRQ3	I
B17	- DACK1	O

ขาอินพุต / เอาท์พุต	ชื่อสัญญาณ	อินพุต / เอาท์พุต
B18	DRQ1	I
B19	- REFRESH	I/O
B20	CLK	O
B21	IRQ7	I
B22	IRQ6	I
B23	IRQ5	I
B24	IRQ4	I
B25	IRQ3	I
B26	- DACK2	O
B27	T/C	O
B28	BALE	O
B29	+ 5 VDC	แหล่งจ่ายไฟเลี้ยง
B30	OSC	O
B31	GND	กราวน์
C1	SBHE	I/O
C2	LA23	I/O
C3	LA22	I/O
C4	LA21	I/O
C5	LA20	I/O
C6	LA19	I/O
C7	LA18	I/O
C8	LA17	I/O
C9	- MEMR	I/O
C10	- MEMW	I/O
C11	SD08	I/O

ขาอินพุต / เอาท์พุต	ชื่อสัญญาณ	อินพุต / เอาท์พุต
C12	SD09	I/O
C13	SD10	I/O
C14	SD11	I/O
C15	SD12	I/O
C16	SD13	I/O
C17	SD14	I/O
C18	SD15	I/O
D1	- MEM CS16	I
D2	- I/O CS16	I
D3	IRQ10	I
D4	IRQ11	I
D5	IRQ12	I
D6	IRQ15	I
D7	IRQ14	I
D8	- DACK0	O
D9	DRQ0	I
D10	- DACK5	O
D11	DRQ5	I
D12	- DACK6	O
D13	DRQ6	I
D14	- DACK7	O
D15	DRQ7	I
D16	+ 5 VDC	แหล่งจ่ายไฟเลี้ยง
D17	- MASTER	I
D18	GND	กราวน์

สัญลักษณ์ในตาราง

I คือ INPUT

O คือ OUTPUT

I/O คือ INPUT/OUTPUT

สัญญาณที่ออกจาก SLOT PC ที่ใช้กับ CARD มีดังนี้

A0 ~ A11 : เป็นแอดเดรสของระบบที่ใช้ติดต่อกับหน่วยความจำและอุปกรณ์อินพุท/เอาท์พุท

D0 ~ D7 : เป็นสัญญาณข้อมูลขนาด 8 บิท ที่ใช้ติดต่อกับหน่วยความจำของไมโครโปรเซสเซอร์

$\overline{IORQ}$: เป็นสัญญาณอ่านอินพุท/เอาท์พุท เป็นสัญญาณที่ส่งมาจาก CPU จะแอกทีฟที่ "0"

$\overline{IOW}$: เป็นสัญญาณเขียนข้อมูลลงบนอุปกรณ์อินพุท/เอาท์พุทสัญญาณนี้ควบคุมจากไมโครโปรเซสเซอร์ แอกทีฟที่ "0"

RES : สัญญาณนี้ใช้สำหรับรีเซ็ทระบบในขณะเปิดเครื่องหรือขณะที่แหล่งจ่ายไฟเลี้ยงขาด

AEN : อีนาเบิลแอดเดรส (เป็นเอาท์พุท)

เมื่อรู้สัญญาณต่างๆที่ออกมาจากเครื่องคอมพิวเตอร์แล้ว ก็มาดูรายละเอียดของวงจรบน CARD บ้าง ซึ่งมี 2 ส่วนดังที่กล่าวมาแล้วในตอนต้นแต่ทั้งสองส่วนจะทำงานร่วมกัน ส่วนของวงจร DECODE ซึ่งใน CARD นี้ประกอบด้วย IC#74LS688 74LS139 และ DIP SW. 3 PIN เพื่อที่สามารถปรับ SET DIP SW. ตั้งตำแหน่งเบอร์พอร์ท PORT ของ CARD ได้โดยเพียงแค่ปรับที่ DIP SW. เท่านั้น สิ่งที่ต้องระวังคือ ในการปรับ DIP SW. นั้นจะต้องให้ตรงกับตำแหน่ง PORT ที่เครื่องคอมพิงเตอร์ใช้อยู่แล้ว ส่วน IC#74F254 ทำหน้าที่เป็น BUFFER 2 ทาง คือ อยู่ในสถานะรับข้อมูลหรือส่งข้อมูลโดยการกำหนดที่เครื่องคอมพิวเตอร์ ส่วน 74LS245 ก็เป็นบัฟเฟอร์แต่ถูกเซ็ทให้ข้อมูลจาก A ไป B เท่านั้น

การทำงานของวงจร

เริ่มต้นที่ SLOT PC โดยจากวงจรจะเห็นว่า D0~D7 จะต่อเข้าที่ขา A1~A8 ของ IC#74F245 และที่ขา B1~B8 ของ IC#74F245 จะต่อเข้าขาที่ D0~D7 ของ PORT 8255 ทั้ง 3 ตัว โดยที่ขา DIR ของ 74F245 จะต่อเข้ากับขา B0 ของ 74LS245 เหตุที่ต่อเช่นนี้ เพราะขา $\overline{\text{IOR}}$ ของ SLOT PC ต่อกับขา A0 ของ 74LS245 ซึ่งจะเป็นตัวควบคุมว่าตอนนี้อยู่ในสภาวะรับหรือส่งข้อมูล ส่วนขา $\overline{\text{IOW}}$ ของ SLOT PC จะต่อกับขา A1 ของ IC 74 LS245 ซึ่งขา B0,B1 ของ 74LS245 ก็จะไปต่อกับขา $\overline{\text{IOR}}$, $\overline{\text{IOW}}$ ของพอร์ท 8255 ทุกตัว เช่นเดียวกัน เพื่อเป็นการบอกสถานะให้รู้ว่าจะรับหรือส่งข้อมูล ส่วนขา RESET A0,A1 ของ SLOT PC ก็จะต่อเข้าที่ขา A2,A3,A6 ของ 74LS245 โดยขา B7 จะต่อเข้าขา RESET ทุกตัวของพอร์ท 8255 ส่วนขา B2,B3 จะต่อเข้าที่ขา A0,A1 ของ 8255 ทุกตัวเพื่อเป็นการเช็ทว่าจะใช้งานพอร์ทไหน และให้เป็นอินพุทหรือเอาท์พุท โดยเช็ททที่ A0,A1 ที่มาจาก SLOT PC จะเห็นว่าขา $\overline{\text{E}}$ ของ 74F245 จะมีไฟเลี้ยงตลอดทำให้ 74F245 ไม่ทำงานจนกว่า จะมีลอจิก "0" จากเอาท์พุทของ 74LS688 มาเท่านั้น ส่วน 74LS245 จะทำงานตลอดเพราะต่อขา $\overline{\text{E}}$ ลงกราวน์และที่ขา DIR ก็มีไฟเลี้ยงตลอดเวลา ซึ่งเป็นการเช็ทให้ข้อมูลส่งจาก A ไป B เท่านั้น ส่วนขา A2,A3 ของ SLOT PC จะเข้าที่ขา A4,A5 ของ 74LS245 และออกที่ขา B4,B5 ไปเข้าที่ขา A,B ของ 74LS139 ซึ่งเป็นวงจร DECODE 2 LINE TO 4 LINEโดยขา A2,A3 ของ SLOT PC จะทำหน้าที่เลือกว่าจะให้ 8255 ตัวไหนทำงานซึ่งมี 3 ตัวด้วยกัน การที่จะส่งข้อมูลหรือรับข้อมูลนั้น เราต้องอ้าพอร์ทของ CARD ก่อน จากวงจรจะใช้ 74 LS688 ซึ่งเป็นวงจร COMPARATOR โดยขา A4~A11 ของ SLOT จะต่อเข้ากับ P0~P7 และที่ DIP SW. ขา 1-8 จะต่อเข้ากับขา Q0~Q7 ของ 74LS688 ซึ่งถ้า ADDRESS ที่ส่งมาตรงกับที่ตั้ง DIP SW. ไว้ ก็จะทำให้ขา $\overline{\text{P=Q}}$ แอกทีฟ "0" ซึ่งทำให้ขา $\overline{\text{E}}$ ของ 74F245 แอคทีฟทำให้รู้ว่า 8255 ตัวไหนทำงานที่ตำแหน่งพอร์ทเท่าไร และสใช้พอร์ทไหนในการทำงานและเป็นการรับหรือส่งข้อมูล

ตัวอย่างเช่น

เราตั้ง DIP SW. ไว้ที่ตำแหน่ง 300 (00110000) เมื่อเราป้อน A0~A11 มาเป็น 300H คือ

A0~A1 เป็น 00 จะทำให้รู้ว่าเราใช้ PORT A

A2~A3 เป็น 00 จะทำให้ 74LS139 DECODE ออกที่ Y0 ที่ให้ 8255 ตัวที่ 1 ทำงาน

A4~A11 เป็น 00001100 ก็จะตรงกับที่ DIP SW. ตั้งไว้ก็ทำให้วงจรทำงานสมบูรณ์

ในการอ้างพอร์ท (กรณี A4~A11 = 30XH)

8255 ตัวที่ 1
PORT A = 300H
PORT B = 301H
PORT C = 302H
CONTROL PORT1 = 303H

8255 ตัวที่ 2
PORT A = 304H
PORT B = 305H
PORT C = 306H
CONTROL PORT2 = 307H

8255 ตัวที่ 3
PORT A = 308H
PORT B = 309H
PORT C = 30AH
CONTROL PORT = 30BH

หมายเลข PORT สามารถเปลี่ยนแปลงได้แต่ห้ามไม่ให้ตรงกับที่ใช้อยู่ในเครื่องคอมพิวเตอร์อยู่แล้ว ดูได้จากตาราง

ตารางหมายเลขพอร์ท

หมายเลขพอร์ทรหัสฐานสิบหก	ชื่ออุปกรณ์
000-01F	DMA คอนโทรลเลอร์หมายเลข 1 8237A-5
020-03F	อินเตอร์รัพต์คอนโทรลเลอร์หมายเลข 1 8259A ตัวหลัก
040-05F	ไทเมอร์ 8254-2
060-06F	8042 คีบอร์ด
070-07F	นาฬิกาและ NMI และซีมอสแรม
080-09F	DMA เพจรีจิสเตอร์
0A0-0BF	อินเตอร์รัพต์คอนโทรลเลอร์หมายเลข 2 8237A-5
0C0-0DF	DMA คอลโทรลเลอร์หมายเลข 2 8237A-5
0F0	เคลียร์โปรเซสเซอร์คณิตศาสตร์
0F1	รีเซสโปรเซสเซอร์คณิตศาสตร์
0F8-0FF	โปรเซสเซอร์คณิตศาสตร์
1F0-1F8	ฮาร์ดดิสก์
200-207	เกมไอโอ
278-27F	พอร์ทเครื่องพิมพ์หมายเลข 2
2F8-2FF	พอร์ทอนุกรมหมายเลข 2
300-31F	โปรโตไทป์การ์ด
360-36F	สำรอง
378-37F	พอร์ทเครื่องพิมพ์หมายเลข 1
380-38F	SDLC, ไบซิงค์ 2
3A0-3AF	ไบซิงค์ 1
3B0-3BF	โมโนโครมและเครื่องพิมพ์

****

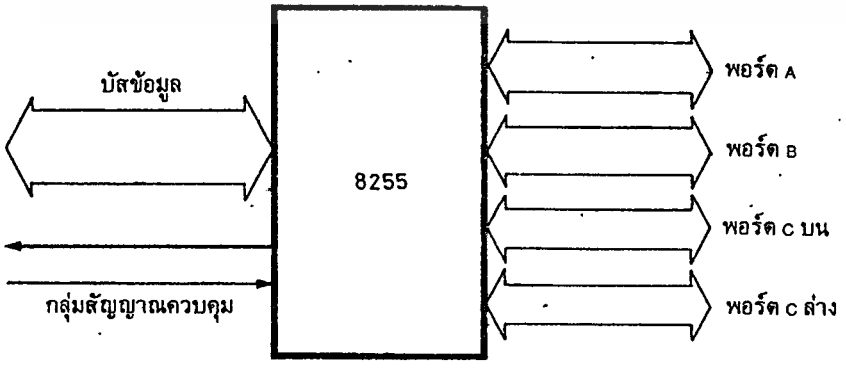
หมายเลขพอร์ทรหัสฐานสิบหก	ชื่ออุปกรณ์
3C0-3CF	สำรอง
3D0-3DF	จอภาพสี
3F0-3F7	ควบคุมดิสเก็ต
3F8-3FF	พอร์ทอนุกรมหมายเลข 1

****

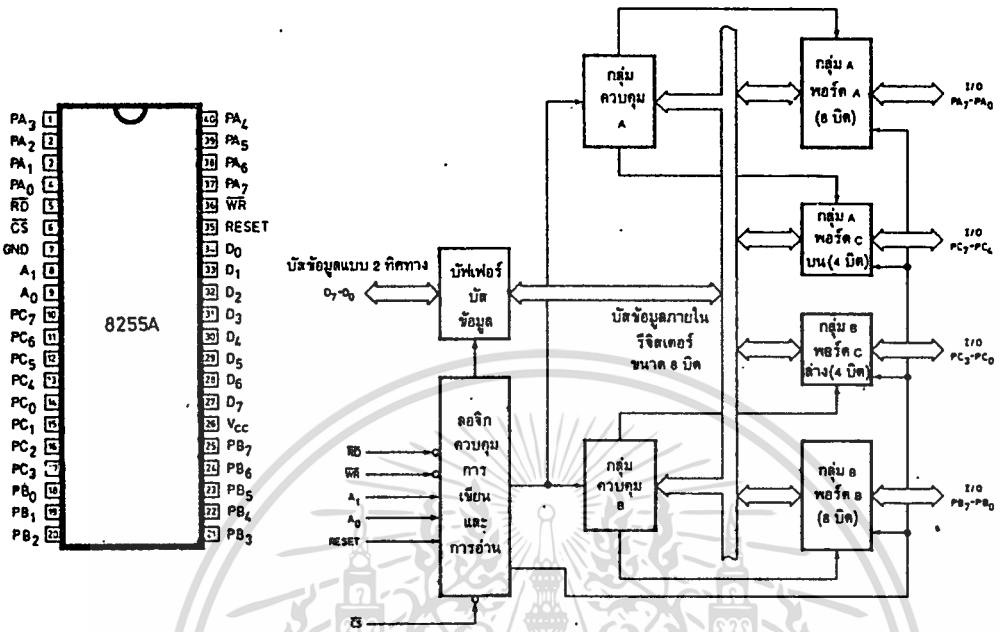
ลักษณะทั่วไปของ 8255

เป็นไอซีขนาด 40 ขา มีพอร์ตให้ใช้งานถึง 3 พอร์ต(เป็นขนาด 8 บิท) พอร์ท A, พอร์ท B, พอร์ท C. โดยพอร์ท C นี้สามารถแยกได้เป็น 2 ส่วนคือ พอร์ท C บนตั้งแต่ PC4-PC7 จำนวน 4 บิทและพอร์ท C ล่างตั้งแต่ PC0-PC3 โดยพอร์ททุกพอร์ท (A,B,C) สามารถโปรแกรมได้ให้เป็นอินพุทหรือเอาท์พุท ซึ่งจะได้กล่างถึงการโปรแกรมในรายละเอียดต่อไป การนำเอาไมโครโปรเซสเซอร์ไปใช้งานนั้น จำเป็นต้องให้ไมโครโปรเซสเซอร์ติดต่อกับโลกภายนอกซึ่งก็คือให้มันสามารถส่งสัญญาณมาควบคุมอุปกรณ์ต่างๆได้

ส่วนที่ทำให้ไมโครโปรเซสเซอร์ติดต่อกับโลกภายนอกได้ที่เรารู้จักกันคือ พอร์ท(PORT) ซึ่งมีอยู่หลายลักษณะด้วยกัน ดังรูป



แผนผังโครงสร้างของไอซี 8255



แผนผังวงจรภายในและการจัดขาของไอซี 8255

จากรูปจะเห็นโครงสร้างภายในที่แสดงถึงกลุ่มควบคุมที่มีอยู่ 3 กลุ่มคือ

- กลุ่มควบคุมชุด A จะควบคุมพอร์ท A และพอร์ท C บน
- กลุ่มควบคุมชุด B จะควบคุมพอร์ท B และพอร์ท C ล่าง
- กลุ่มควบคุมลอจิกการเขียนและอ่าน

การทำงานของ 8255 จะใช้สัญญาณควบคุมจากตัวไมโครโปรเซสเซอร์ มาควบคุโดยจะมีการส่งคำสั่ง (Control word) มาที่กลุ่มควบคุมชุด A,B แล้วกลุ่มควบคุมชุดนี้ก็จะสั่งต่อไปที่พอร์ทเพื่อให้เป็นไปตามข้อกำหนดของคำสั่งนั้นๆ เช่น ให้พอร์ท A เป็นอินพุทพอร์ท B เป็นเอาท์พุทพอร์ทเหล่านี้เป็นต้น ส่วนกรณีเมื่อมีการอ่านเขียนพอร์ทจาก CPU นั้น กลุ่มควบคุมลอจิกการเขียน อ่านจะเป็นตัวที่ส่งสัญญาณไปบอกแก่กลุ่มควบคุมชุดในแต่ละชุดอีกที ทั้งนี้แล้วแต่ว่า CPU จะมีการอ่านเขียนพอร์ทของกลุ่มควบคุมชุดใด

ขาต่างๆ ของไอซี 8255

D0~D7 เป็นขาข้อมูลของ 8255 ที่ใช้ติดต่อกับตัวไมโคโปรเซสเซอร์ซึ่งข้อมูลที่จะเข้าออกสู่พอร์ทต่างๆ ของ 8255 จะต้องผ่านขาข้อมูลนี้

$\overline{CS}$ เป็นขาอินพุทที่รับสัญญาณลอจิก "0" จากภายนอกเพื่อแสดงว่าต้องการเลือกใช้ไอซีเบอร์นี้ หากได้รับลอจิก "1" ก็จะทำให้ไอซีตัวนี้ไม่ทำงานคือไม่รับรู้สัญญาณใดๆ ทั้งสิ้น

$\overline{RD}$ เป็นขาอินพุทที่รับสัญญาณจากตัวไมโครโปรเซสเซอร์ โดยหากมีลอจิกเป็น "0" จะเป็นการแสดงว่า CPU ต้องการที่จะอ่านข้อมูลจากตัว 8255

$\overline{WR}$ เป็นขาอินพุทที่รับสัญญาณจากตัวไมโครโปรเซสเซอร์ โดยหากมีลอจิกเป็น "0" ก็จะเป็นการแสดงว่า CPU ต้องการที่จะเขียนข้อมูลจากตัว 8255

A0~A1 เป็นอินพุทที่รับแอดเดรสจากตัวไมโครโปรเซสเซอร์ที่ถอดรหัสตำแหน่งของ 8255 เรียบร้อยแล้วโดยจะมีตำแหน่งใช้งาน 4 ตำแหน่ง เพื่ออ่านเขียนรีจิสเตอร์ (พอร์ท) ของ 8255 ที่อยู่ด้วยกัน 4 ตัว

RESET เป็นขาอินพุทที่นับสัญญาณจากภายนอกเข้ามาทำการรีเซทตัว 8255 โดยหากได้รับลอจิก "1" จะทำให้พอร์ททุกพอร์ทเป็นอินพุทพอร์ททั้งหมด เพื่อไม่ต้องการให้มีสัญญาณออกไปรบกวนต่อระบบภายนอกเพื่อ 8255 ได้รับสัญญาณรีเซท

PA0~PA7 เป็นขาสัญญาณพอร์ท A ที่ใช้ติดต่อกับอุปกรณ์ภายนอก

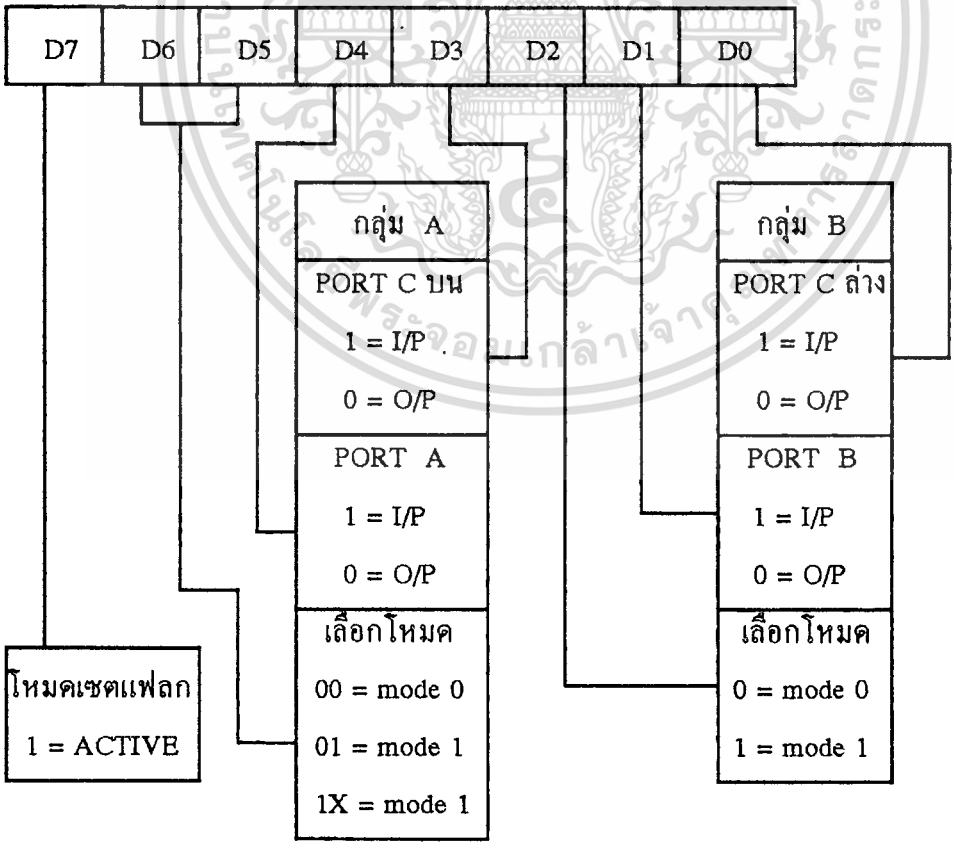
PB0~PB7 เป็นขาสัญญาณพอร์ท B ที่ใช้ติดต่อกับโลกภายนอก

PC0~PC7 เป็นขาสัญญาณพอร์ท C ที่ใช้ติดต่อกับอุปกรณ์ภายนอกซึ่งพอร์ทนี้จะแบ่งออกเป็น 2 กลุ่มคือ PC0 ~ PC7 และ PC4 ~ PC7 ซึ่งสามารถโปรแกรมแยกกันได้อีกต่างหาก

การโปรแกรม 8255

เราได้ทราบแล้วในรูปที่ 2 ว่าโครงสร้างภายในของ 8255 มีกลุ่มควบคุมชุดอยู่ 3 กลุ่มซึ่งทั้ง3 กลุ่มนี้จะทำงานร่วมกันดังที่กล่าวมาและเราสามารถจะควบคุมการทำงานของพอร์ทจาก CPU โดยสั่งงานมาที่กลุ่มควบคุมดังกล่าว แต่ตัว CPU จะมองเห็น 8255 เป็น 4 พอร์ทด้วยกันโดยแต่ละพอร์ทเสมือนเป็นรีจิสเตอร์ที่ CPU สามารถจะทำการอ่าน / เขียนได้ซึ่งหากมีการอ่านเขียนไปยังพอร์ทดังกล่าวก็จะใช้ร่วมกับสัญญาณ $\overline{RD}$ โดยสัญญาณ $\overline{WR}$ หมายถึง เอาท์พุทข้อมูล และ RD แอคทีฟ หมายถึง อินพุทข้อมูล

การใช้งานเราจะต้องส่งรหัสควบคุม (Control Code) เข้าไปยังพอร์ทควบคุม (หรือเรียกอีกอย่างว่ารีจิสเตอร์ควบคุม) ซึ่งจะเป็นข้อมูลขนาด 1 ไบท์ส่งไปที่ แอดเดรส 13H (กรณีนี้เราถอดรหัสไว้ที่ 13H) โดยความหมายของแต่ละบิทที่เราไปโปรแกรม การทำงานเป็นดังนี้



ความหมายของบิทต่างๆ ในรหัสควบคุม

การทำงานในโหมด 0

จัดว่าเป็นโหมดพื้นฐานที่นิยมใช้กันมากที่สุด ทั้งสามพอร์ทเราสามารถจะใช้ให้พอร์ทใดเป็นอินพุท เอาท์พุทได้ โดยเฉพาะพอร์ท C ยังแยกให้เป็น 2 ชุดๆ ละ 4 บิต ซึ่งในแต่ละชุดนี้ก็สามารถจะโปรแกรมให้ชุดใดชุดหนึ่งเป็นอินพุทหรือเอาท์พุทพอร์ทได้อีก ฉะนั้นโดยสรุปแล้วจะมีพอร์ทที่จะโปรแกรมให้เป็นอินพุทหรือเอาท์พุทได้ 4 พอร์ท คือ พอร์ท A พอร์ท B พอร์ท C บนและพอร์ท C ล่าง (แต่โปรแกรมแบกเฉพาะบิตในแต่ละพอร์ทไม่ได้)

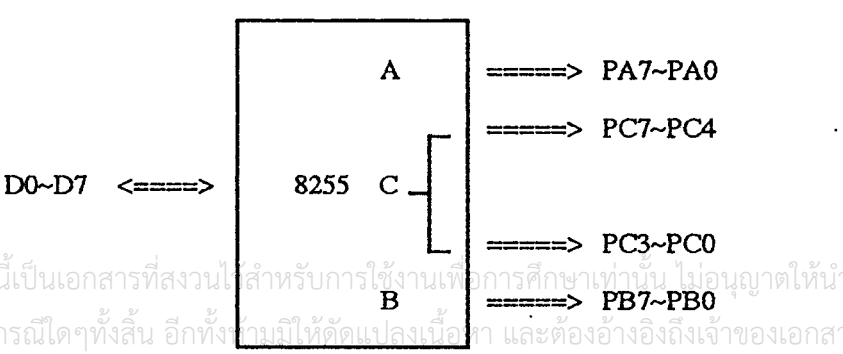
1	0	0	X	X	0	X	X
BIT 7	6	5	4	3	2	1	0

รูปแสดงรหัสคำสั่งของโหมด

ซึ่งหากเราดูที่รหัสคำสั่งแล้วจะเห็นว่ามีอยู่ 4 บิต ที่ถูกกำหนดตายตัวคือบิต 7, บิต 6, บิต 5, และบิต 2, ส่วนที่เหลืออีก 4 บิต ก็คือข้อกำหนดว่าจะให้พอร์ทใดเป็นพอร์ท อินพุท/เอาท์พุท นั้นเอง ซึ่งหากเราให้พอร์ทใดเป็นอินพุทเราก็ใส่ลอจิก "1" ที่บิตนั้นหรือหาต้องการให้พอร์ทใดเป็นเอาท์พุทก็ใส่ "0" ที่บิตนั้น จะเห็นได้ว่าจะมีความเป็นไปได้ในการกำหนดลักษณะของพอร์ทในโหมดนี้อยู่ 16 อย่างด้วยกัน ยกตัวอย่างเช่น

ตัวอย่าง 1. ต้องการให้ทุกพอร์ทเป็นเอาท์พุทหมด เราจะได้รหัสคำสั่งเป็น

1	0	0	0	0	0	0	0



ตัวอย่างที่ 2. ต้องการให้ พอร์ท A ====> เอาท์พุท

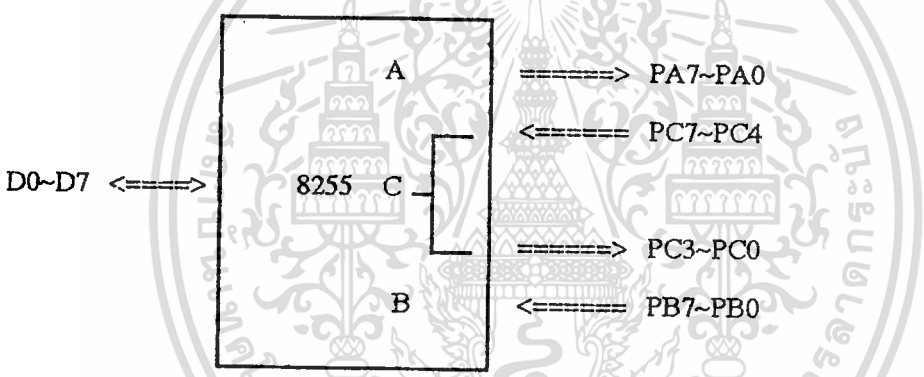
พอร์ท B ====> อินพุท

พอร์ท C บน ====> อินพุท

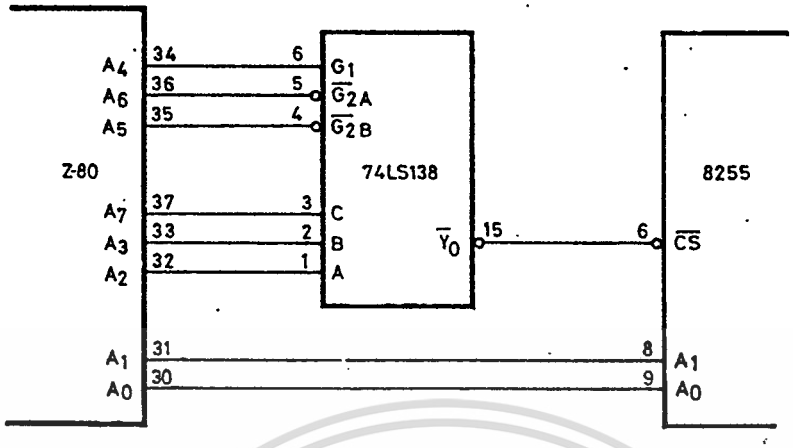
พอร์ท C ล่าง ====> เอาท์พุท

จะได้รหัสคำสั่งเป็น

1	0	0	0	1	0	1	0

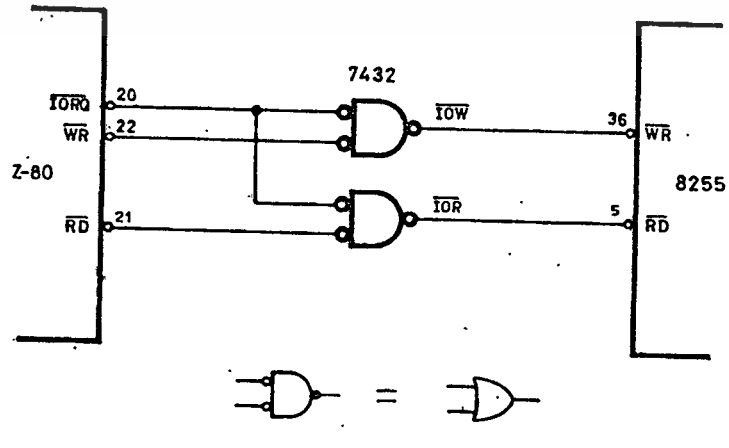


แสดงตัวอย่างพอร์ทที่ถูกโปรแกรมในโหมด 0



การกำหนดแอดเดรสให้กับ 8255

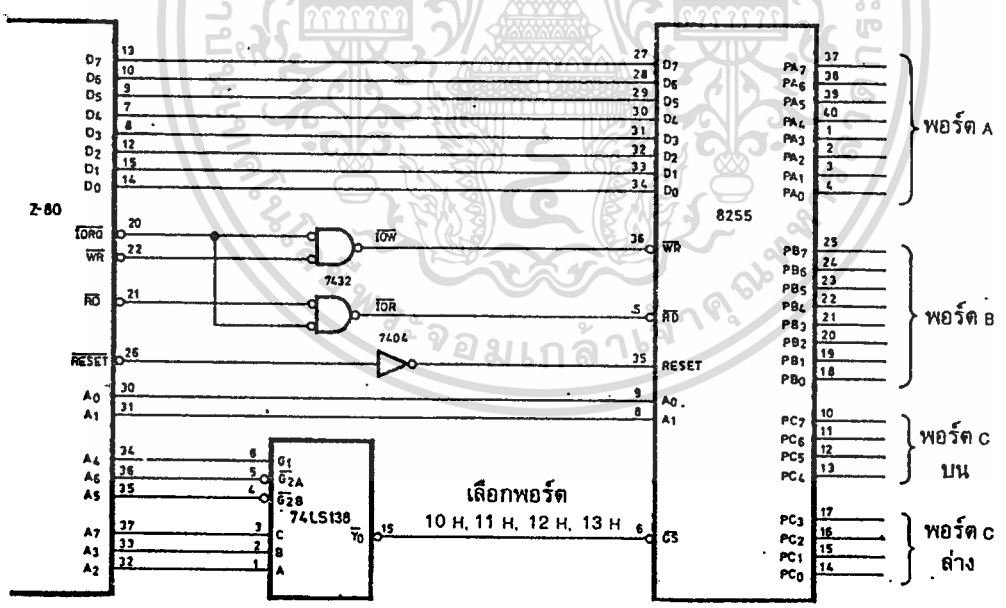
สังเกตว่าขณะสัญญาณ $\overline{CS}$ แอคตีฟสัญญาณแอดเดรส A_7,A_6,A_5,A_4,A_3,A_2 จะต้องมีข้อมูล 00100 และเมื่อรวมกับ A_1,A_0 จะเป็น 000100XX พอร์ตที่เกิดขึ้นเมื่อ A_0,A_1 เป็น00คือพอร์ต 10H และถ้า A_1,A_0 เป็น 11 พอร์ตจะเป็น 13H การกำหนดพอร์ตของ Z-80 จะใช้ข้มูลบนแอดเดรส 8เส้นคือ A_0-A_7 เท่านั้น สัญญาณที่จะควบคุม 8255 อีกชุดหนึ่งคือ สัญญาณควบคุมการเขียนและการอ่าน หากสัญญาณ $\overline{WR}$ แอคตีฟเป็น "0" จะหมายถึง การเขียนพอร์ตหรือส่งข้อมูลให้พอร์ตเอาต์พุต แต่ถ้าสัญญาณ $\overline{RD}$ แอคตีฟเป็น "0" จะหมายถึง การอ่านพอร์ตหรือรับข้อมูลอินพุต



วงจรการเชื่อมต่อสายสัญญาณควบคุมการเขียนและการอ่าน 8255

เพื่อให้แยกกันระหว่างการเขียนและการอ่านหน่วยความจำกับการเขียนและการอ่านพอร์ตอินพุตเอาต์พุต จึงต้องใช้สัญญาณ IORQ ร่วมด้วย กล่าวคือถ้าสัญญาณ WR เกิดขึ้นพร้อมกับสัญญาณ IORQ จะหมายถึง สัญญาณ IOW หรือสัญญาณเขียนพอร์ต และถ้าให้สัญญาณ IORQ แอคตีฟพร้อมกับสัญญาณ RD จะหมายถึงสัญญาณ IOR หรือสัญญาณอ่านพอร์ต ซึ่งการเชื่อมต่อสายสัญญาณควบคุมการเขียนและการอ่านหน่วยความจำแสดงดูดังรูป

เมื่อเชื่อมต่อระบบ จะต้องมีการเชื่อมสัญญาณ RESET ของ Z -80 มายังขา RESET ของ 8255 การรีเซตของ 8255 ใช้ "1" ซึ่งตรงข้ามกับ Z-80 ดังนั้นจำเป็นต้องมี อินเวอร์เตอร์เปลี่ยนลอจิกก่อน การที่ต้องรีเซต 8255 พร้อมกับ Z -80 ก็เนื่องจากว่า ขณะที่ Z-80 รีเซตเราจะเริ่มจากให้พอร์ตทุกพอร์ตของ 8255 เป็นอินพุตเพื่อว่าอาจจะมีข้อมูลบางส่วน ไปออกที่พอร์ตเอาท์พุตในขณะที่เรายังไม่ต้องการ ซึ่งอาจจะมีระบบอินเตอร์เฟสภายนอกมีปัญหาได้ เพราะเราไม่รู้สถานะที่แน่นอนของ 8255 ก่อนการโปรมแกรมโหมดการทำงาน ระบบการเชื่อมต่อของ 8255 กับ Z-80 ทั้งระบบแสดงดังรูป



การเชื่อมต่อ 8255 กับ Z-80 ทั้งระบบ

เมื่อต่อ 8255 เข้ากับ Z-80 ได้แล้ว สิ่งที่ผู้ใช้จะต้องทำก็คือ การโปรมแกรมให้ 8255 ทำงานตามที่ต้องการ จากการที่ 8255 มีพอร์ตที่ Z-80 มองเห็น 4 พอร์ต แต่ละ พอร์ตจะเสมือนเป็นรีจิสเตอร์ที่สามารถเขียนและอ่านได้ รีเจิสเตอร์แต่ละตัวนี้จึงถูกกำหนดด้วย แอดเดรสตามที่ตั้ง

ไว้ เช่น ในกรณีที่เป็นแอดเดรส 10H,11H,12H, และ 13H รีเจสเตอร์ แต่ละตัวจะได้รับการกำหนดการควบคู่กับสัญญาณ $\overline{RD}$ และ $\overline{WR}$ เพื่อแสดงความหมาย ตัวอย่าง เช่น พอร์ต 10H เป็นพอร์ต A ซึ่งเมื่อได้เขียนที่พอร์ตนี้ จะเป็นการส่งข้อมูลเอาต์พุต และ ถ้าอ่านพอร์ตนี้ก็จะเป็นการอินพุตข้อมูลจากพอร์ตดังนั้นสัญญาณของขาควบคุมที่ประกอบกันจะแสดง ความหมายดังตาราง

สัญญาณควบคุมการกระทำของ 8255

RD	WR	A1	A0	ความหมาย
1	0	0	0	เขียนพอร์ต A ซึ่งเป็นข้อมูล
0	1	0	0	อ่านพอร์ต A ซึ่งเป็นข้อมูล
1	0	0	1	เขียนพอร์ต B ซึ่งเป็นข้อมูล
0	1	0	1	อ่านพอร์ต B ซึ่งเป็นข้อมูล
1	0	1	0	เขียนพอร์ต C ซึ่งเป็นข้อมูล
0	1	1	0	อ่านพอร์ต C ซึ่งเป็นข้อมูล
1	0	1	1	เขียนข้อมูลซึ่งเป็นรหัสควบคุม
0	1	1	1	อ่านเข้ามาซึ่งไม่มีความหมายใด

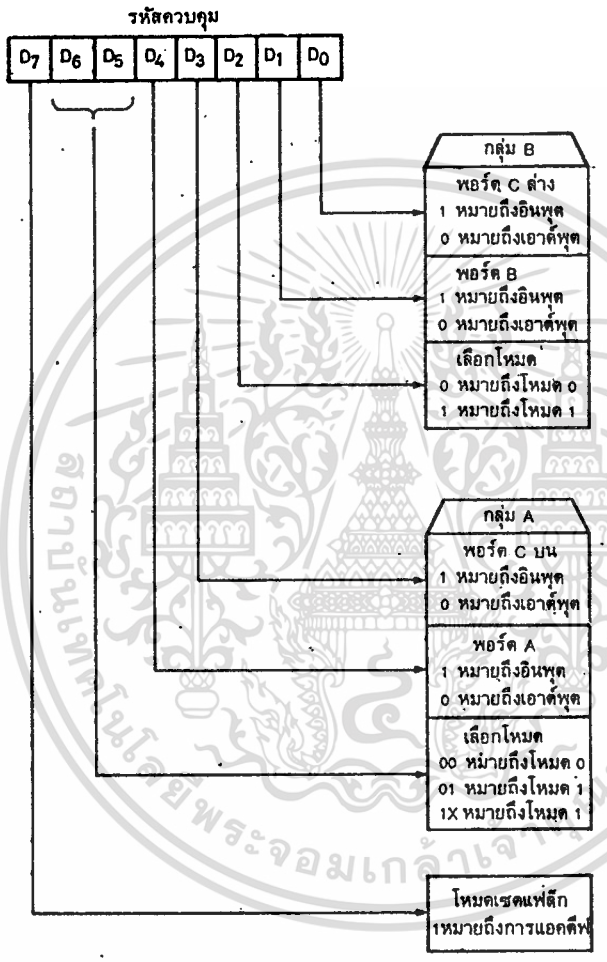
การใช้งาน 8255 จะต้องส่งรหัสควบคุม [contorl code] เข้าไปยังพอร์ตข้อมูลควบคุมเพื่อคุมการทำงานของ 8255 โดยใช้สัญญาณการควบคุมหมายเลข 13H การควบคุมการทำงานของ 8255 มีหลายโหมด แต่ละโหมดจะแตกต่างกันออกไป การโปรแกรมให้ 8255 ทำงานจะทำได้ 3 โหมดคือ โหมด 0 โหมด 1 และโหมด 2

<u>โหมด 0 หรืออินพุทเอาท์พุทแบบพื้นฐาน</u>

การกำหนดโหมดการทำงาน จะต้องส่งขอ้มูลคำสั่งเข้าโปรแกรมในโหมดควบคุมของ 8255 ซึ่งในที่นี้โหมดหมายเลข 13H แต่ละบิตของข้อมูลที่ส่งไปจะมีความหมายในตัวเอง ลักษณะความหมายแต่ละบิตลักษณะควบคุมแสดงดังรูป การโปรมแกรม 8255 คือ การให้ค่ารหัสบิตต่าง ๆ เข้าไปในรหัสควบคุมแล้วส่งไปยังรีจิสเตอร์ของพอร์ตควบคุม ความหมายของบิตต่าง ๆ มีดังนี้

บิต D_7 เป็นบิตที่แสดงรหัสคำสั่งควบคุม ถ้าบิตนี้เป็น"1"หมายถึงรหัสควบคุมนี้จะมีผลต่อการเปลี่ยนแปลงการเซตโหมดต่าง ๆ ของ 8255

บิต D_6 และ D_5 เป็นการเลือกโหมดของพอร์ตA ซึ่งมี 3 โหมดคือโหมด 0 โหมด 1 และโหมด 2 ดังรูป



ความหมายของบิตต่างๆในรหัสควบคุม

บิต D_4 ถ้ามีค่าเป็น"0"หมายถึงการกำหนดพอร์ตA เป็นเอาต์พุต ถ้ามีค่าเป็น "1" จะหมายถึงการกำหนดให้พอร์ต A เป็นอินพุต

บิต D_3 เป็นบิตที่บอกถึงการเซตของพอร์ต C บน ถ้าเป็น"0"จะทำให้พอรต์ C บนเป็นเอาท์พุต

บิต D_1 เป็นการกำหนดอินพุตเอาท์พุตของพอร์ต B ถ้าเป็น "0" หมายถึง เอาท์พุตถ้าเป็น "1" หมายถึงอินพุต

บิต D_0 เป็นการกำหนดอินพุตเอาท์พุตของพอร์ต C ล่างถ้าเป็น "0" หมายถึง เอาต์พุตถ้าเป็น "1" หมายถึงอินพุต

การโปรแกรม 8255 จะเริ่มจากเซตค่าที่ต้องการแล้วเอาต์พุตไปยังพอร์ตควบคุม เช่น ถ้าต้องการโปรแกรมให้ทั้งพอร์ต A , B และC เป็นพอร์ตเอาต์พุตหมด เราจะเลือก 8255 ให้อยู่ในโหมด 0 โดยมีรหัสควบคุมเป็น 10000000 หรือ 80H ดังนั้นจึงเขียนคำสั่งได้เป็น

```
LD   A,80H      หมายถึงการกำหนดรหัสควบคุม
OUT (13H), A    หมายถึงส่งไปยังพอร์ตควบคุม
```

หลังจากที่กระทำคำสั่ง OUT ที่ผ่านไปแล้วพอร์ต A , B และ C จะเป็นพอร์ตเอาต์พุตหมด ซึ่งก็จะส่งข้อมูลจากซีพียูไปยังพอร์ตต่าง ๆ ได้ เช่น ถ้าต้องการส่งข้อมูล 8AH ไปยังพอร์ต A ข้อมูล 41H ไปยังพอร์ต B และข้อมูล 25H ไปยังพอร์ต C คำสั่งที่ใช้คือ

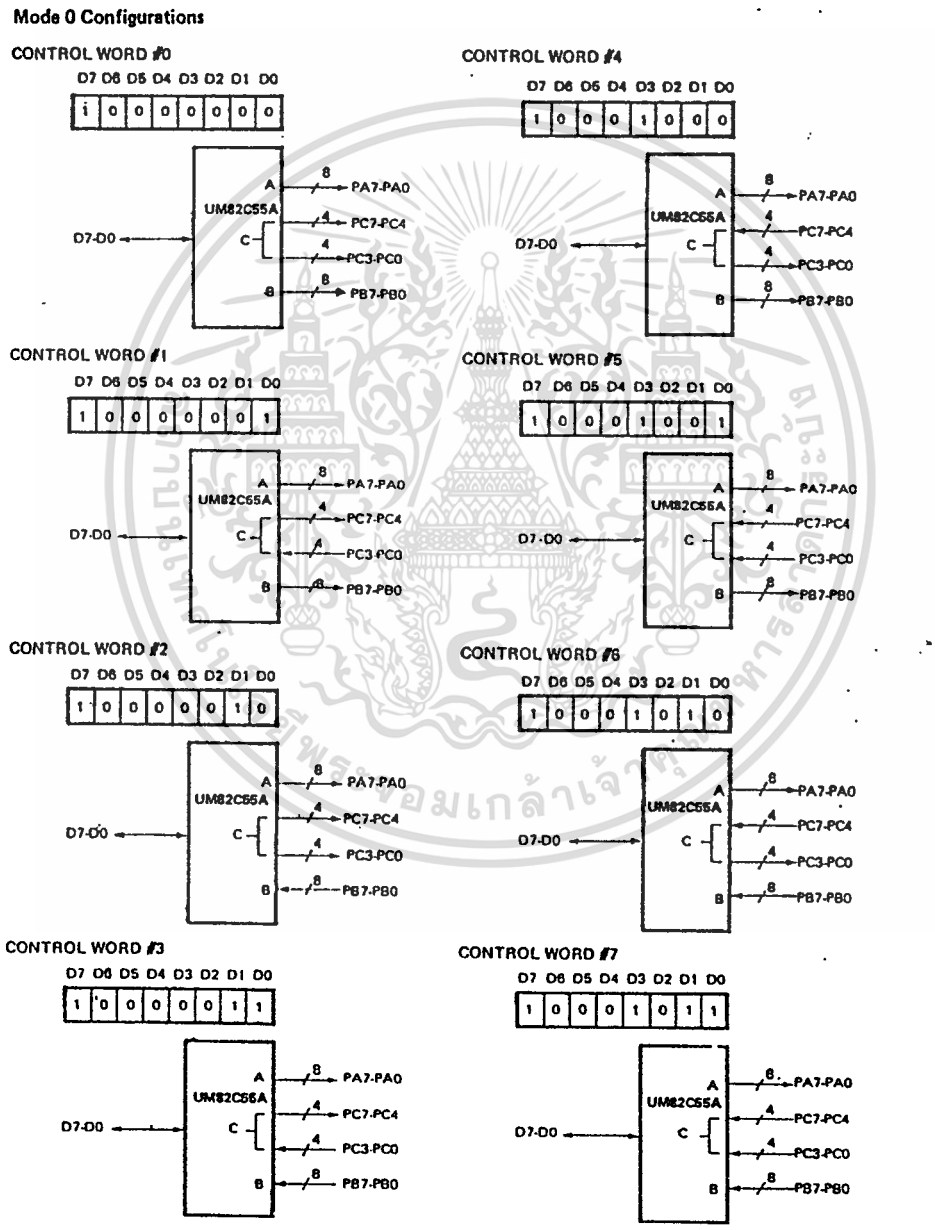
```
LD  A,8AH     หมายถึงเลือกค่า 8AH
OUT [10H],A   หมายถึงส่งให้พอร์ต A
LD  A,41H     หมายถึงเลือกค่า 41H
OUT [11H],A   หมายถึงส่งให้พอร์ต B
LD  A ,25H    หมายถึงส่งให้พอร์ต 25H
OUT [12H],A   หมายถึงส่งให้พอร์ต C
```

เนื่องจากมีพอร์ตสำหรับส่งข้อมูล 3 พอร์ต คือ พอร์ต A พอร์ต B และพอร์ต C ซึ่งพอร์ต C จะแยกออกเป็น 2 ส่วน คือ พอร์ต C ล่าง และพอร์ต C บน เราสามารถโปรแกรมให้ทั้ง 4 พอร์ตนี้เป็นอินพุตหรือเอาต์พุตก็ได้เช่นถ้าให้รหัสควบคุมเป็น 82H จะทำให้พอร์ต B เป็นอินพุตพอร์ตและพอร์ต C เป็นเอาต์พุต

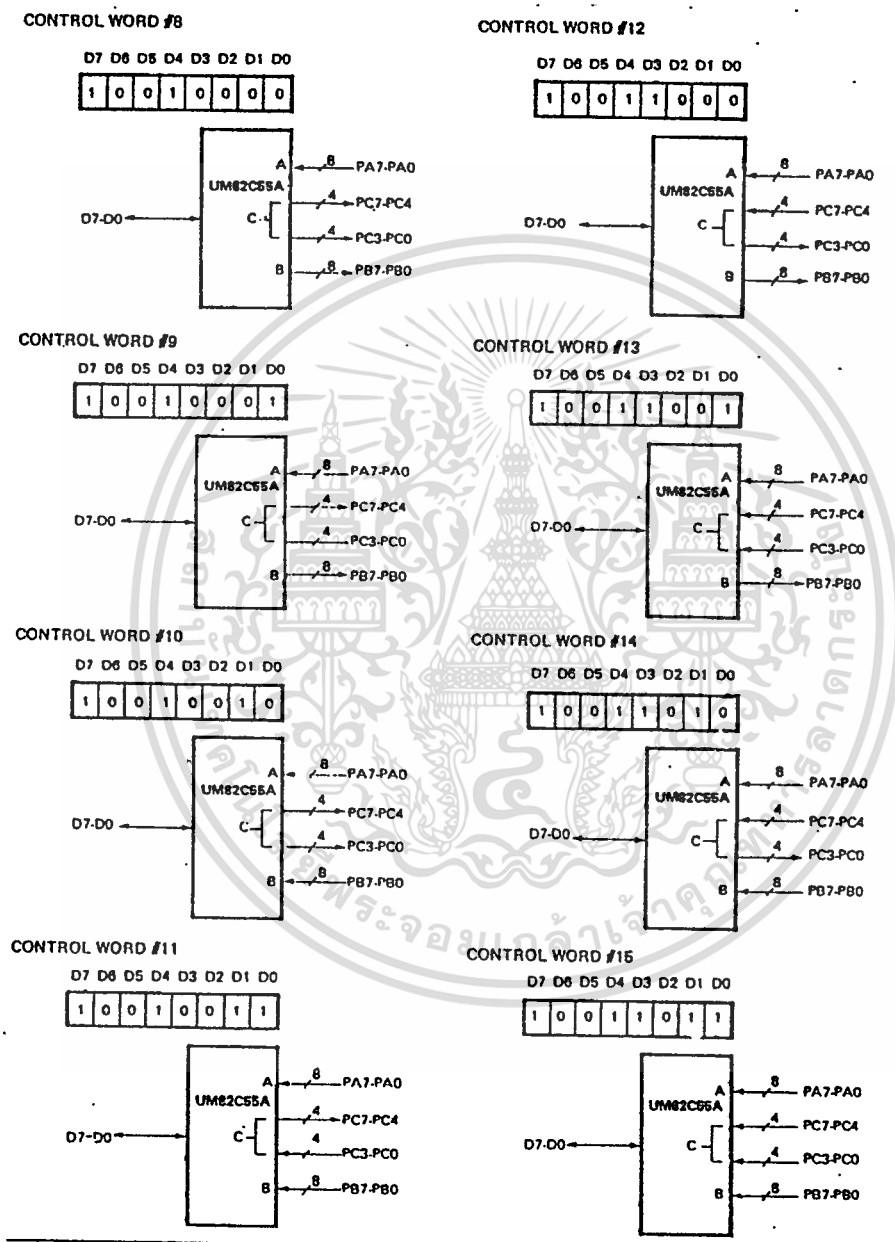
<u>การทำงานในโหมด</u> 0 โหมด 0 เป็นโหมดให้พอร์ตทุกพอร์ตบนตัว 8255 เป็นพอร์ตอินพุตเอาต์พุตแบบพื้นฐานรูปแบบความเป็นไปได้จึงมีทั้งสิ้น 16 รูปแบบตามลักษณะของพอร์ต A พอร์ต B พอร์ต C บนและ พอร์ต C ล่าง ลักษณะการควบคุมแต่ละแบบเป็นดังรูป

การทำงานในโหมด 0

โหมด 0 เป็นโหมดให้พอร์ตทุกพอร์ตบนตัว 8255 เป็นพอร์ตอินพุตเอาต์พุตแบบพื้นฐาน รูปแบบความเป็นไปได้จึงมีทั้งสิ้น 16 รูปแบบตามลักษณะของพอร์ต A พอร์ต B พอร์ต C บน และ พอร์ต C ล่าง ลักษณะการควบคุมแต่ละแบบเป็นดังรูป

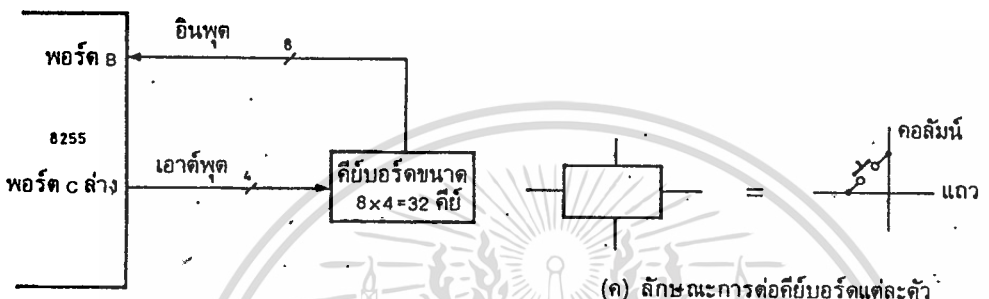


ลักษณะการควบคุมแบบต่าง ๆ ในโหมด 0



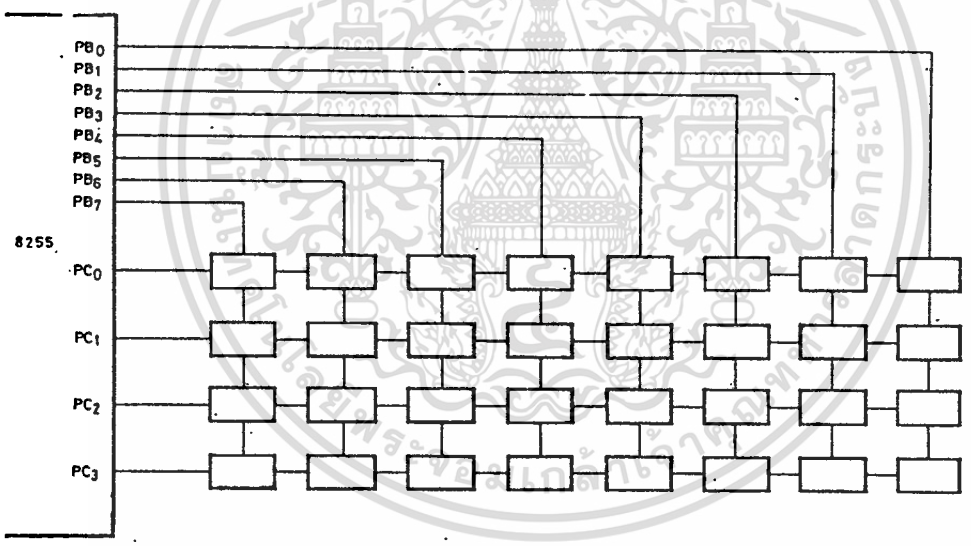
ลักษณะของรหัสควบคุมแบบต่าง ๆ ในโหมด 0

ตัวอย่างการใช้งาน 8255ในโหมด 0 นี้ เช่น เมื่อต้องการให้พอร์ต B เป็นอินพุต และพอร์ตล่างเป็นเอาต์พุต เพื่อรับรู้การกดคีย์บอร์ดและหาดูว่ามีคีย์ใดกด ซึ่งเราสามารถใช้การ จัดคีย์บอร์และหาดูว่ามีคีย์ใดกด ซึ่งเราสามารถใช้การจัดคีย์บอร์ดในรูปแบบเมตริกซ์ได้ดังรูป



(ก) กำหนดให้พอร์ต B และพอร์ต C ต่อกับคีย์บอร์ด

(ค) ลักษณะการต่อคีย์บอร์ดแต่ละตัว



(ข) การจัดวางตำแหน่งของคีย์บอร์ด

วงจรการต่อคีย์บอร์ดขนาด 32 คีย์แบบเมตริกซ์

สาเหตุที่เรานิยมต่อคีย์จำนวนมากแบบเมตริกซ์เนื่องจากเป็นโครงสร้างทางฮาร์ดแวร์แบบประหยัดและช่วยลดข้อยุ่งยากต่างๆ ได้ แต่ก็ยังต้องใช้ซอฟต์แวร์ในการควบคุมหรือตรวจสอบว่า คีย์ใดกดในรูปที่ 8 เป็นการต่อแบบเมตริกซ์ ซึ่งมีจำนวน 4 แถว และคอลัมน์ 8 คอลัมน์ทำให้ ได้จำนวนคีย์ทั้งสิ้น 32 คีย์

หลักการทำงานทั่วไปเราจะทำการแสกนดู กล่าวคือกำหนดให้แต่ละแถวซึ่งเป็นพอร์ต เอาต์พุตเป็น "0" หรือ "1" ในเวลาต่างกัน เช่น แถวแรกเป็น "0" แถวอื่นเป็น "1" หมดแล้วทำการอ่านข้อมูลที่พอร์ตอินพุตว่า มีบิตใดบิตหนึ่งทางคอลัมน์เป็น "0" หรือไม่ ถ้ามีก็ทราบได้ ว่าคีย์ในตำแหน่งแถวแรกและคอลัมน์ที่เท่าไรเป็นคีย์ที่ได้รับการกด แต่ถ้าไม่มี ซีพียู ก็จะแสกนไปยัง แถวถัดไปและวนรอบไปเรื่อยๆ ตลอดเวลา ดังนั้นการตรวจสอบคีย์กดจะทำให้ทราบว่าแถวหรือคอลัมน์ ที่เท่าไรซึ่งเป็นคีย์ที่ได้รับการกด ลองพิจารณารูปซึ่งต่อคีย์ในลักษณะ 4x8 เราจะแสกนทีละแถวโดยเริ่มจากแถว 0 โดยการให้เอาต์พุตที่บิต 1 เป็น "0" แล้ววนไปเรื่อย ๆ โดยใช้ค่าในรีจิสเตอร์ C เป็นค่า สำหรับกำหนดแถว โปรแกรมที่เขียนขึ้นจะได้ดังโปรแกรม

โปรแกรมส่วนตรวจสอบการปล่อยคีย์

```
RELEASE CHECK THAT KEY HAS BEEN RELEASE
   RELEASE LD C,00H
           CALL SCAN       :เรียกโปรแกรมสแกนคีย์
           IN A,11H
           XOR 0FF         :ไม่กด = FFH
           JR NZ,PRESS
           INC C
           LD A,40H
           CP C            : ครบ 4 แถว ?
           RET C           : ถ้าทุกคีย์ปล่อยหมด ให้ RET
           JR RELEASE + 2
    PRESS CALL DLY10
           JR RELEASE + 5
```

จากโปรแกรม เป็นการตรวจสอบคีย์ว่า ทุกคีย์อยู่ในสภาพปล่อยหมดหรือไม่ โดยมีการเรียกโปรแกรมย่อย SCAN เพื่อส่งค่าแสกนคีย์ทีละแถวให้ตรวจสอบ โดยมีโปรแกรมย่อย SCAN และถ้าพบว่ามีการกดคีย์ก็จะมีการหน่วงเวลา 10 มิลลิวินาที แล้วตรวจสอบใหม่ การใช้ คำสั่ง RELEASE + 5 หมายถึง กระโดดไปยังตำแหน่ง RELEASE บวกอีก 5 ไบต์ และคำสั่ง $+4 หมายถึงกระโดดจากตำแหน่งของตัวเองไปอีก 4 ไบต์ ซึ่งโปรแกรมย่อยของแสกนคีย์จะเป็นดังโปรแกรม

โปรแกรมส่วนสแกนคีย์

```
SCAN KEYBOARD BY OUTPUT TOROW SCAN
    SCAN LD HL, LINE
    LD A,(HL)
    RLCA            :เลื่อนไปแถวถัดไป
    CP 10H          : เกินแถวที่ 4 ?
    JR NZ,$+4
    LD A,01H        : เริ่มต้นแถวที่ 1ใหม่
    LD (HL),A
    OUT 10H,A
    OUT 12H,A
    RET
  LINE DB 01H
```

ส่วนโปรแกรมหน่วงเวลา 10 มิลลิวินาที ซึ่งเราใช้โปรแกรมมวนลูปจะมีลักษณะดัง โปรแกรมหน่วงเวลา

โปรแกรมส่วนหน่วงเวลา 10 มิลลิวินาที

```
;DELAY LOOP 10 MILLISECOND
DLY10 PUSH DE       : SAVE DE
    LD DE,1247      : จำนวนลูป
    DEC DE
    JR NZ,$-1       : ลูปให้วน
    POP DE
    RET
```

ผลจากโปรแกรม SCAN จะบอกสถานะการแสกนว่า แสกนมาถึงแนวใดซึ่งจะอยู่ในหน่วยความจำ LINE นั่นก็คือ เราจะทราบได้ว่า แถวใดกดโดยดูจากLINE และจะตรวจสอบการกดคีย์ ว่าเป็นคอลัมน์ใดได้ โดยทำโปรแกรมรับคีย์เข้าทางพอร์ตอินพุตและเก็บไว้ในรีจิสเตอร์A ดังตัวอย่างในโปรแกรมรับคีย์และดีบาวนซ์

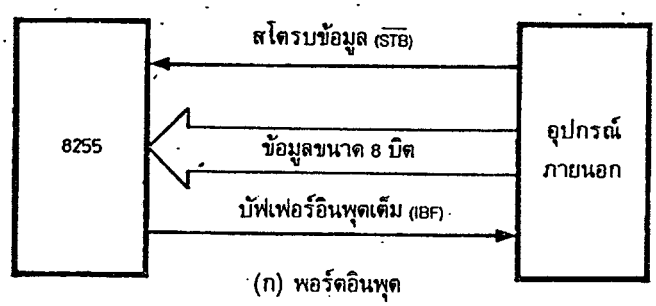
โปรแกรมส่วนการรับคีย์และดีบาวนซ์

```
;CHECKTHAT KEY WAS PRESSED AND DEBOUNCE IT
 KEYIN CALL SCAN
      .IN A,11H
       XOR 0FFH
       JR Z,KEYIN
DEBOUN LD B,A
       CALL DLY10          : หน่วงเวลา 100 mS
       IN A,11H            : อ่านค่าคอลัมน์
       XOR 0FFH            : กลับค่า
       CALL Z,DLY10
       JR Z,KEYIN+3
       CP B                : ยังกดตัวเดิมอยู่ ?
       JR NZ,DEBOUN        : ให้วนอีกครั้ง
       RET
```

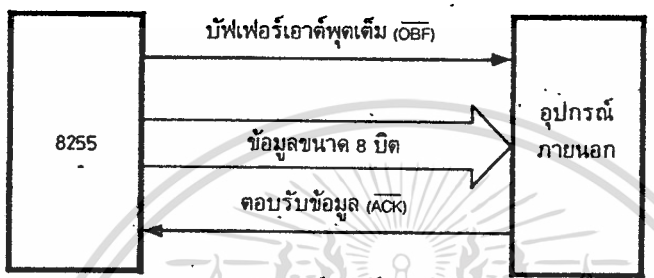
จากโปรแกรมนี้ จะได้ค่าคีย์ตามแนวคอลัมน์อยู่ในรีจิสเตอร์ A และแถวอยู่ในตัวแปร LINE ซึ่งเราจะนำไปใช้ในการถอดรหัสว่าเป็นคีย์อะไรได้ สังเกตว่าเรามีการตรวจสอบเพื่อแก้การกระเด้ง หรือดีบาวนซ์ (debounce) คีย์ด้วย และมีการหน่วงเวลา 10 มิลลิวินาทีเพื่อป้องกันการอ่านค่าคีย์ ผิดพลาดได้ และสิ่งที่สำคัญคือการทำงานของซีพียูเร็วมาก ดังนั้นจึงต้องมีการตรวจสอบการปล่อยคีย์ (release) ก่อนแล้วจึงแสกนหาคอลัมน์และแถวเพื่อกำหนดตำแหน่งคีย์

การทำงานของ 8255 ในโหมด 1

การทำงาน 8255 ในโหมด 1 เป็นโหมดที่ทำให้อินพุตเอาต์พุตมีการตรวจสอบสัญญาณ (handshaking) โดยใช้อินพุตเอาต์พุตของพอร์ต A และพอร์ต B เป็นหลัก และใช้พอร์ต C บนเป็นตัวตรวจสอบสัญญาณ (handshahake) ของพอร์ต A ส่วนพอร์ต C ล่างเป็นตัวตรวจสอบสัญญาณของพอร์ต B การจัดสัญญาณต่าง ๆ เหล่านี้แสดงดังรูปแสดงโครงสร้างตัวตรวจสอบสัญญาณของพอร์ตอินพุทและพอร์ตเอาต์พุต



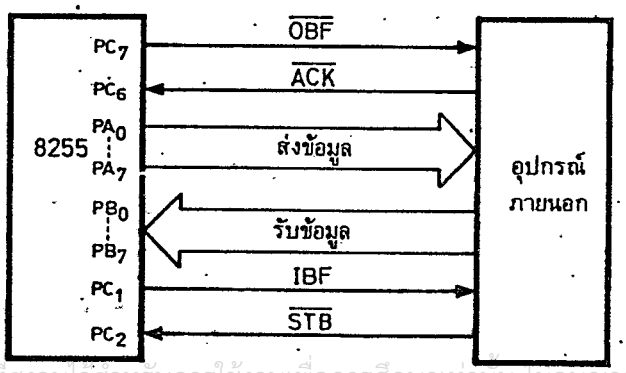
(ก) พอร์ตอินพุต



(ข) พอร์ตเอาต์พุต

โครงสร้างตัวตรวจสอบสัญญาณของพอร์ตอินพุตและพอร์ตเอาต์พุต

แนวความคิดของการใช้พอร์ตอินพุตเอาต์พุตโดยมีตัวตรวจสอบสัญญาณก็เพื่อให้มีการซิงโครไนซ์ระหว่างอุปกรณ์ภายนอกที่ทำงานได้ช้ากับการทำงานของคอมพิวเตอร์ที่ทำงานได้เร็ว เช่น เครื่องพิมพ์ทำงานได้ช้า เมื่อคอมพิวเตอร์ส่งตัวอักษรตัวแรกมาพิมพ์ เครื่องพิมพ์รับตัวอักษรและกำลังจะพิมพ์ คอมพิวเตอร์ก็ส่งตัวอักษรตัวที่2ตัวที่3 ตามมา ทำให้การประมวลผลของอุปกรณ์เครื่องพิมพ์ทำงานไม่ทัน ซึ่งอาจทำให้ข้อมูลสูญหาย ดังนั้นเครื่องพิมพ์จึงส่งสัญญาณบอกคอมพิวเตอร์ว่า "อย่าเพิ่งส่งมาเพราะยังไม่พร้อมที่จะรับ" ลักษณะของการรับส่งข้อมูลอินพุตเอาต์พุต แบบมีการตรวจสอบสัญญาณดังรูป จะใช้ PA_0-PA_7 เป็นเอาต์พุต และ PB_0-PB_7 เป็นอินพุต โดย มีพอร์ต C เป็นตัวตรวจสอบสัญญาณ ดังแผนผังในรูปวงจรการต่อ 8255 ในโหมด 1



วงจรการต่อ 8255 ในโหมด 1

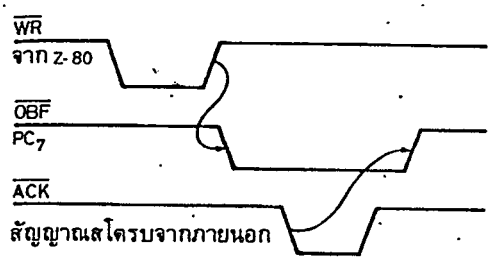
เมื่อโปรแกรม 8255 เป็นโหมด 1 แล้ว ตัว 8255 จะให้พอร์ต C เป็นสัญญาณควบคุม โดยแต่ละบิตของพอร์ต C เป็นไปตามที่กำหนดไว้ ดังตารางด้านล่าง

ตารางหน้าที่ของสัญญาณต่างๆของพอร์ตC ในการทำงานเป็นตัวตรวจสอบสัญญาณเมื่อ 8255 ทำงานในโหมด 1

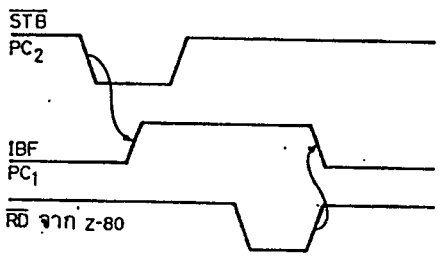
ขา	กรณีอินพุต	กรณีเอาท์พุต
PC_0	$INTR_B$	$INTR_B$
PC_1	IBF_B	OBF_B
PC_2	STB_B	ACK_B
PC_3	$INTR_A$	$INTR_A$
PC_4	STB	I/O
PC_5	IBF	I/O
PC_6	I/O	ACK_A
PC_7	I/O	OBF_A

โดยปกติ 8255 จะให้สัญญาณอินเตอร์รัพต์ไปบอกซีพียูด้วย สัญญาณอินเตอร์รัพต์ของ 8255 จะเกิดขึ้นที่ PC_0 และ PC_3 โดยที่บัฟเฟอร์พร้อมแล้วและต้องการให้ซีพียูส่งอินพุตหรือเอาต์พุต มาที่บัฟเฟอร์ สัญญาณอินเตอร์รัพต์ก็จะเกิดขึ้น สังเกตว่า สัญญาณอิเตอร์รัพต์เป็นสัญญาณแอคตีฟ "1" ซึ่งตรงกับของ 8080 แต่เมื่อใช้กับ Z-80 สัญญาณ INT ของ Z-80 จะรับด้วย "0"

โครงสร้างการตรวจสอบสัญญาณของ 8255 แสดงด้วยสัญญาณทางไฟฟ้าได้ดังรูป



(ก) เมื่อเป็นพอร์ตเอาต์พุต



(ข) เมื่อเป็นพอร์ตอินพุต

แผงผังเวลาการรับและการส่งข้อมูลโดยใช้ตัวตรวจสอบสัญญาณ

สังเกตว่า การทำงาน 8255 จะเกี่ยวข้องกับสัญญาณ $\overline{RD}$ และ $\overline{WR}$ ซึ่งจะทำให้สัญญาณ ควบคุมเปลี่ยนแปลงไป การตรวจสอบสัญญาณซึ่งกันและกันนี้ เป็นวิธีการรับส่งที่มีประสิทธิภาพ เช่น ในกรณีอินพุต เมื่ออุปกรณ์ภายนอกต้องการส่งข้อมูลให้ ซีพียู ก็ส่งข้อมูลแบบขนานเข้ามาพร้อมทั้ง สโตรบ [STB] บอก 8255 ซึ่ง 8255 จะนำข้อมูลนั้นไปเก็บไว้ในรีจิสเตอร์ภายในก่อนแล้วส่งสัญญาณตอบบอก ว่า "บัฟเฟอร์ยังเต็มอยู่นะ (IBF) อย่าเพิ่งส่งมาอีก" ครั้นเมื่อซีพียูอ่านข้อมูลจากรีจิสเตอร์ไปแล้ว ส่วนของสัญญาณบัฟเฟอร์อินพุต (IBF) ก็จะบอกว่า "ว่างแล้วส่งมาได้" อุปกรณ์ภายนอกก็จะส่งข้อมูลมาให้อีก

ทำนองเดียวกัน สำหรับพอร์ตเอาต์พุต เมื่อซีพียูส่งข้อมูลออกทางพอร์ตเอาต์พุตให้กับ 8255 ตัว 8255 ก็จะรับไว้ในรีจิสเตอร์ภายใน พร้อมทั้งส่งสัญญาณอออกไปบอกอุปกรณ์ภายนอก ว่า "เอาต์พุตบัฟเฟอร์ของฉันมีข้อมูลนะ ($\overline{OBF}$) มาอ่านเอาไปซิ" อุปกรณ์ภายนอกเมื่อทราบแล้วพร้อม จะอ่านก็จะส่งสัญญาณตอบรับ ($\overline{ACK}$) พร้อมกับอ่านข้อมูลไป โดยสัญญาณ $\overline{ACK}$ จะมีความหมายว่า "ฉันอ่านข้อมูลไปแล้วนะ" ตัว 8255 ก็จะตอบกลับว่า "บัฟเฟอร์ฉันว่างแล้ว เธอรอก่อนนะจะมีข้อมูล ส่งมาให้อีก"

ในการที่จะโปรแกรมโหมด 1 นี้ เราจะใช้รหัสควบคุมเป็น 101 (I/O) 01 (I/O) 0 ในส่วน I/0 หมายถึง ถ้าเป็นอินพุตก็คือ "1" ถ้าเป็นเอาต์พุตก็คือ "0" โดย I/0 ตัวแรกเป็นของพอร์ต A ตัวที่ 2 เป็นของพอร์ต B เช่น ถ้าต้องการให้พอร์ต A เป็นเอาต์พุต และพอร์ต B เป็นอินพุต เราจะใช้รหัสควบคุมเป็น 10100110 หรือ A6H

จากการพิจารณาการทำงานของซีพียูจะเห็นว่า ทำอย่างไรจึงจะเขียนหรืออ่านพอร์ตได้ ถูกต้องวิธีที่ง่ายวิธีหนึ่งคือ ซีพียูจะคอยตรวจสอบสัญญาณของ 8255 เช่นกรณีเอาต์พุต ซีพียูจะคอยอ่านพอร์ตC แล้วตรวจสอบบิต 7 (OBF) หลังจากที่ส่งข้อมูลไปแล้ว ถ้าบิต 7 ยังเป็น "0" แสดงว่ายังไม่ได้รับการสโตรบแต่ถ้าเป็น "1" แล้วแสดงว่าอุปกรณ์ภายนอกรับข้อมูลไปแล้ว สำหรับกรณี อินพุตก็คอยตรวจสอบสัญญาณ IBF ได้เช่นกันว่า มีข้อมูลใหม่เข้ามาหรือยัง คือตรวจสอบบิต PC_1 ของพอร์ต C

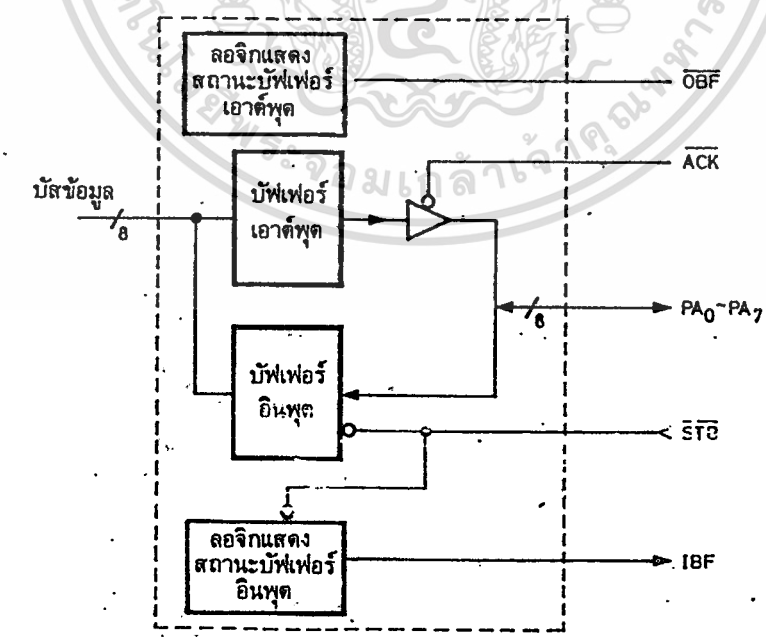
<u>การทำงานของ 8255 ในโหมด 2</u>

8255 ยังมีโหมดการทำงานอีกโหมดหนึ่งคือโหมด2ซึ่งทำได้เฉพาะพอร์ต Aในโหมดนี้ 8255 จะใช้พอร์ตAทำหน้าที่เป็นพอร์ตแบบ 2 ทิศทางคือสามารถเป็นได้ทั้งพอร์ตอินพุต และเอาต์พุตโดยโครงสร้างของพอร์ต A ทั้งอินพุตเอาต์พุตมีการตรวจสอบสัญญาณทั้งคู่ ส่วนพอร์ต C จะทำหน้าที่ เป็นสัญญาณตรวจสอบ โดยมีสัญญาณแต่ละขาดังตารางหน้าที่ของพอร์ต C ในโหมด 2

ตารางแสดงหน้าที่ของพอร์ต C ในโหมด 2

พอร์ต C	ความหมาย
PC_0	I/O
PC_1	I/O
PC_2	I/O
PC_3	$INTR_A$
PC_4	STB_A
PC_5	IBF_A
PC_6	ACK_A
PC_7	OBF_A

โครงสร้างของพอร์ต A ที่ทำงานแบบ 2 ทิศทาง แสดงได้ดังรูป



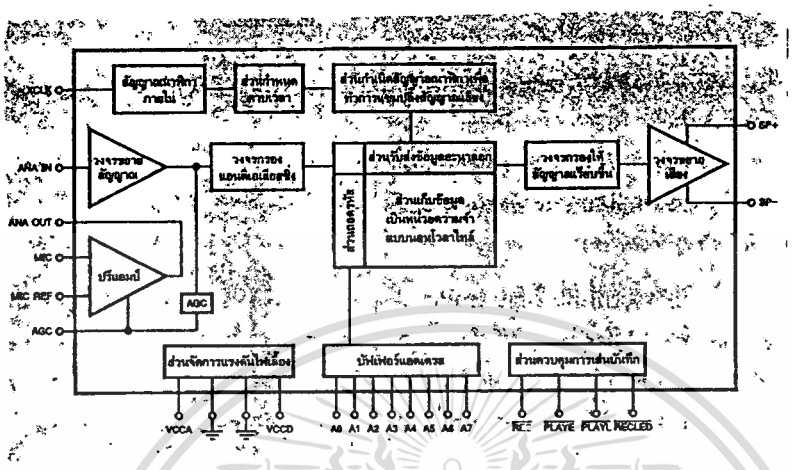
โครงสร้างของพอร์ต A ที่ทำงานแบบพอร์ต 2 ทิศทาง

สังเกตว่าเมื่อโปรแกรมพอร์ต A เป็นโหมด 2 แล้ว พอร์ต B จะต้องโปรแกรมเป็นโหมด 0 หรือโหมด 1ก็ได้ซึ่งก็ทำงานแบบแยกอิสระอีก ในการใช้งานพอร์ตแบบ 2 ทิศทางนี้ใช้ได้กับงานบาง ประเภท เช่น ใช้ในการรับส่งข้อมูลของพอร์ตมาตราฐานบางประเภทเช่น IEEE 488 หรือใช้เชื่อมโยง ระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ในการรับส่งข้อมูลสลับกันไปและกลับ

บทสรุป

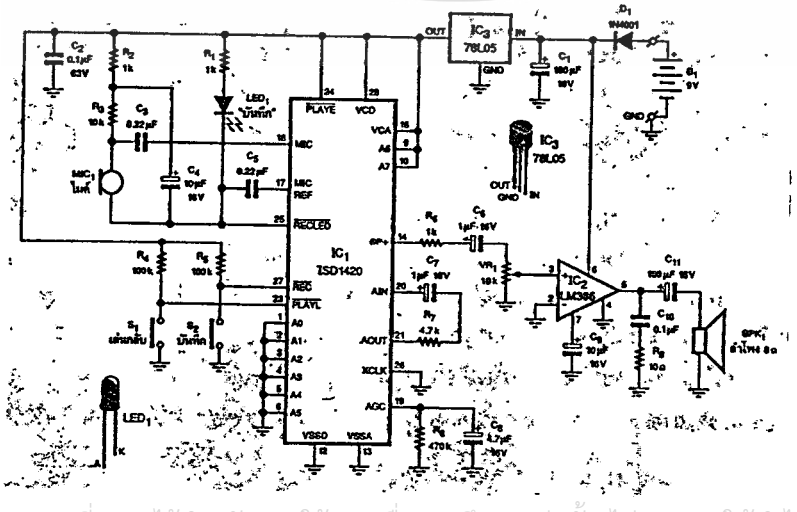
การใช้ 8255 เป็นการใช้ที่เราพบเห็นในไมโครโปรเซสเซอร์ทั่วไป เพราะใช้งานได้ง่ายและมีความคล่องตัว ผู้ออกแบบที่ต้องใช้พอร์ตขนานมักเลือกใช้ 8255 เป็นตัวอินเตอร์เฟสกับอุปกรณ์ภายนอก

วงจรเสียงพูดของหุ่นยนต์



รูปที่ 1 บล็อกไดอะแกรมภายในของไอซี ISD 1420

รูปที่ 1 เป็นบล็อกไดอะแกรมการทำงานภายในของ ISD 1420 จะเห็นส่วนประกอบภายในว่ามันสามารถบันทึกสัญญาณเสียงไปเก็บไว้ได้อย่างไร สัญญาณอนาลอกที่ทำการบันทึกจะถูกนำไปเก็บไว้ในหน่วยความจำความจุ 128,000 เซล การบันทึกไม่ได้ใช้หลักการทำงานแปลงสัญญาณจากสัญญาณอนาลอกเป็นดิจิตอล (A/D) หรือจากดิจิตอลเป็นอนาลอก (D/A) แต่อย่างใด สามารถบันทึกสัญญาณอนาลอกเข้าไปเก็บไว้ในหน่วยความจำได้โดยตรงเลย โดยสัญญาณจะถูกเก็บอยู่ในรูปของแรงดันระดับต่างๆ ภายในเซล สัญญาณเอาท์พุทสามารถที่จะขับออกลำโพงขนาดเล็กได้โดยตรง หรือต่อเข้ากับวงจรขยายสัญญาณภายนอกก็ได้



รูปที่ 2 วงจรสมบูรณ์ของเครื่องบันทึกเสียงไอซีตัวเดียว

การทำงาน

วงจรสมบูรณ์ของเครื่องบันทึกเสียงไร้เส้นเทปนี้แสดงดังรูปที่ 2 เป็นวงจรที่ว่างและใช้อุปกรณ์น้อยชิ้น การทำงานเริ่มต้นโดยเมื่อจ่ายไฟเลี้ยงจากแบตเตอรี่ 9 โวลท์ ให้วงจร IC1 ก็พร้อมที่จะรับข้อมูลเพื่อทำการบันทึก โดยเมื่อสวิตช์ S2 ถูกกดให้ต่อวงจร ทำให้ขาที่ควบคุมการบันทึก ($\overline{REC}$) มีสถานะเป็น " 0 " ในขณะนี้ก็จะเกิดการบันทึกเสียงเข้าไปใน IC1 โดยมีคอนเดนเซอร์ไมโครโฟน (MIC1) ทำหน้าที่รับสัญญาณเสียงและ R2, R3 เป็นตัวจัดไบอัสให้กับไมค์ สัญญาณจะถูกคัปปลิ้งผ่าน C3 มาเข้าที่ขา18 เพื่อทำการขยายสัญญาณให้แรงขึ้น สัญญาณที่ผ่านการขยายโดยวงจรปรีแอมป์จะออกมาทางขา 21 ซึ่งเป็นสัญญาณอนาลอกเอาท์พุทและคัปปลิ้งผ่าน R7 และ C7 เข้าขา 20 ซึ่งเป็นวงจรขยายภายในไอซีเช่นกัน สัญญาณที่ถูกขยายจะถูกบันทึกลงไว้ในหน่วยความจำภายในไอซีที่ขา 25 ($\overline{RECLED}$) เป็นขาขับชุดแสดงสภาวะขณะทำการบันทึกซึ่งแสดงผลโดย LED1 ที่ขา 19 มีตัวต้านทาน R8 และ C8 จัดเป็นวงจรรักษาระดับสัญญาณการบันทึกให้คงที่หรือ AGC (automatic gain control) เพื่อให้สัญญาณของการบันทึกมีเหมาะสมเมื่อเล่นกลับสัญญาณจะได้ไม่เกิดการผิดเพี้ยน

เมื่อทำการบันทึกไปจนครบเวลาที่กำหนดไว้คือ 20 วินาที วงจรบันทึกจะหยุดทำการบันทึก หากต้องการเล่นกลับก็ต้องควบคุมที่ขา 23 ($\overline{PLAYL}$) ด้วยระดับลอจิก " 0 " โดยการกด S1 กระบวนการเล่นกลับก็จะทำงานขึ้นภายในตัวไอซีและให้สัญญาณเอาท์พุทออกมาทางขา 14 ถึงแม้ว่าเอาท์พุทนี้จะสามารถขับลำโพงเล็กๆ ได้โดยตรง แต่อาจจะมีความดังของเสียงค่อยไป ดังนั้นจึงเพิ่มภาคขยายเสียงเข้าไปอีก โดยสัญญาณจะถูกคัปปลิ้งผ่าน R6 และ C6 มาเข้าโวลุ่ม VR1 สัญญาณจะถูกส่งเข้าขา 3 ของ IC2 เบอร์ LM 386 ซึ่งเป็นไอซีขยายเสียง สัญญาณที่ผ่านการขยายแล้วจะออกมาทางขา 5 ของ IC2 ผ่าน C11 ขับออกสู่ลำโพง SPK1 โดยมี C10 และ R9 ทำหน้าที่ชดเชยวงจรย่านความถี่สูงและป้องกันการออสซิเลทางเอาท์พุทของวงจรขยาย

แรงดันไฟเลี้ยงวงจรได้จากแบตเตอรี่ 9 โวลท์ ซึ่งแรงดันนี้สามารถจ่ายให้กับ IC2 ได้โดยตรงแต่สำหรับ IC1 ต้องการแรงดันไฟตรง +5 โวลท์ จึงต้องเพิ่มชุดเรกูเลเตอร์ IC3 เพื่อควบคุมแรงดันให้คงที่โดยมี D1 เป็นตัวป้องกันการต่อแรงดันแบตเตอรี่ผิดขั้ว

บทที่ 3

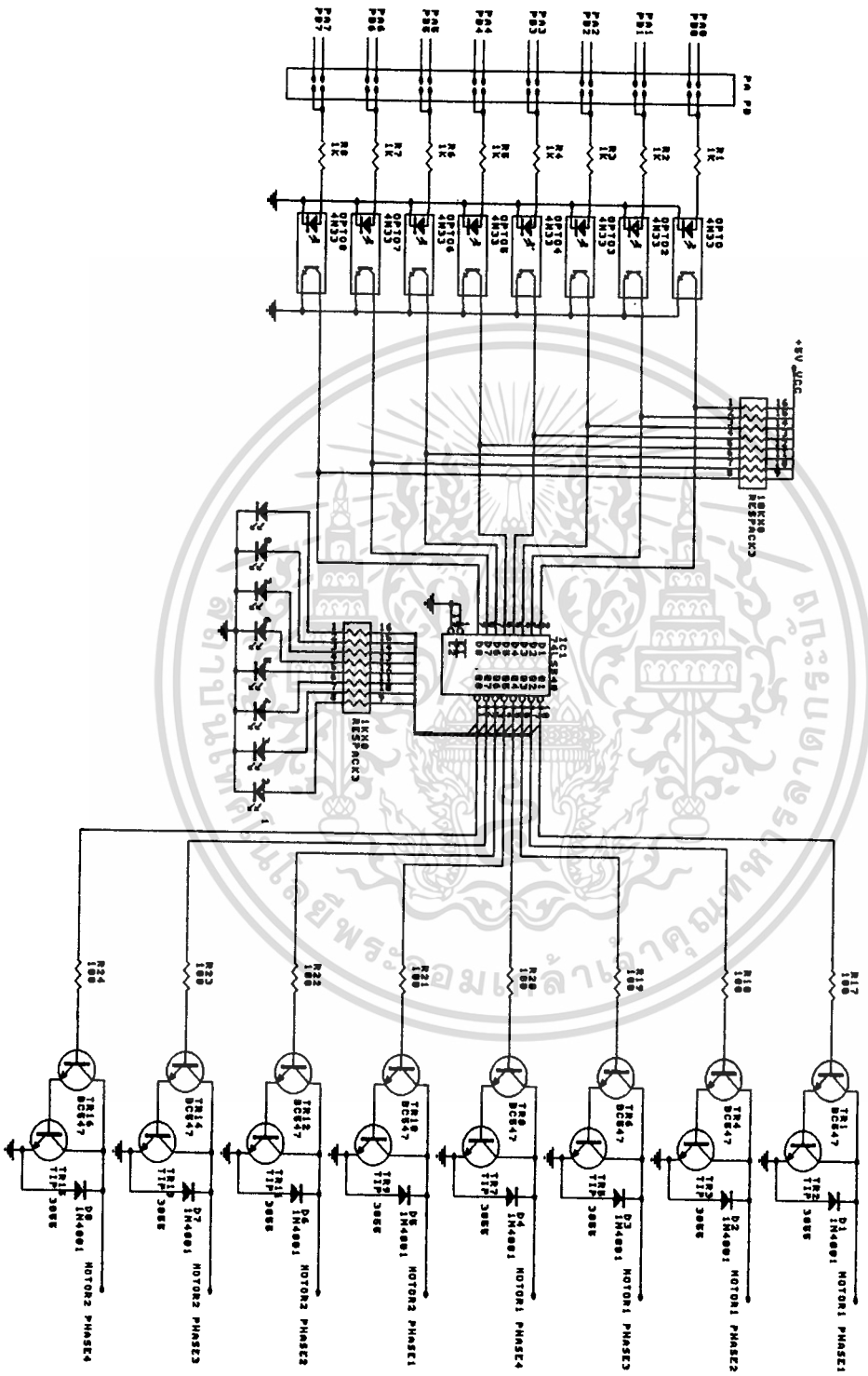
การทำงานของวงจรภาค DRIVER

การทำงานของวงจรควบคุม STEPPING MOTOR

การทำงานของวงจร DRIVER เริ่มด้วยสัญญาณจากการ์ดพอร์ต 8255 (8255 CARD PORT) ส่งผ่านตามสายสัญญาณมาที่วงจร DRIVER เข้ามาที่ออปโต้คัปเปอร์(OPTOCOUPLER) ซึ่งจะทำหน้าที่เป็นตัวดีเท็ค (DETECT) ตรวจสอบสัญญาณที่เข้ามา ซึ่งมีทั้งลอจิก "0" และลอจิก "1" ถ้ามีระดับแรงไฟเป็นลอจิกเป็น"1"จะทำให้เอาท์พุทของออปโต้คัปเปอร์ซึ่งเป็นทรานซิสเตอร์ทำงานเปรียบเสมือนสวิทช์ ON เอาท์พุตจึงเปรียบเสมือน ต่อลงกราวน์และมีระดับลอจิก (LOGIC) เป็น "0" จากนั้นสัญญาณจะถูกส่งให้ ไอซี 74LS540 ซึ่งทำหน้าที่เป็นตัว BUFFER และจะมี INVERTER อยู่ในตัวไอซี 74LS540 ด้วย ที่เอาท์พุตของไอซี 74LS540 จะมีตัวต้านทานต่ออนุกรมกับแอลอีดี (LED) เพื่อแสดงสัญญาณที่ออกจากเอาท์พุตของไอซี 74LS540 จากนั้นสัญญาณจะถูกส่งไปที่ทรานซิสเตอร์ไดรฟ์ ซึ่งภาคไดรฟ์จะใช้ทราซิสเตอร์ เบอร์ BC 547 ต่อแบบดาร์ลิงตันกับทรานซิสเตอร์ เบอร์ TIP 3055 ทรานซิสเตอร์จะต่อกันแบบดาร์ลิงตันและมีไดโอดเบอร์1N4001ต่อคร่อมขาคอลเลคเตอร์กับขาอีมิตเตอร์เพื่อให้ไฟลบในขดลวดถูกบายพาส (BYPASS) ลงกราวน์และเพื่อป้องกันแรงดันย้อนกลับในขดลวดของมอเตอร์ ซึ่งการกระตุ้นการทำงานของสเต็ปปิ้งมอเตอร์จะต้องเป็นไปตามลำดับ (SEQUENCE) ทั้งในทิศทางตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา คือ เราจะใช้โปรแกรม PASCAL เขียนโปรแกรมเพื่อติดต่อกับ 8255 CARD PORT ซึ่ง CARD PORT นี้จะต่ออยู่กับภาคขับเคลื่อน (DRIVER) โดยใช้สายแพเป็นสายนำสัญญาณข้อมูล(BUS) ซึ่งต้องใช้สายแพ 34 PIN จำนวน 3 เส้น เนื่องจาก 8255 CARD PORT ใช้ไอซี 8255 จำนวน 3 ตัว โดยแต่ละตัวทำงานแยกจากกันเป็น 3 ชุด แต่ละชุดจะมี พอร์ตข้อมูล (DATA PORT) 3 พอร์ต คือ พอร์ต A ,พอร์ต B และ พอร์ต C ซึ่งในการเขียนโปรแกรมควบคุมการทำงาน เราจะให้ พอร์ต A , พอร์ต B และ พอร์ต C เป็น OUTPUT PORT ทั้งหมด

ฮาร์ดแวร์ควบคุมการทำงานของ STEPPING MOTOR ประกอบด้วย 8255 CARD PORT ซึ่งเสียบอยู่กับ SLOT ของ คอมพิวเตอร์ และ ต่อสายแพ 34 PIN 3 เส้นมายังภาคขับเคลื่อน (DRIVER) โดยที่ ขา 1 ของ ออปโต้คัปเปอร์ เป็น อินพุทที่ต่อมาจาก 8255 CARD PORT ซึ่งออปโต้คัปเปอร์ 1ตัวจะต่อกับ DATA BUS ได้ 1 เส้น ในแผงวงจรภาค DRIVER 1 แผง จะมี ออปโต้คัปเปอร์ 8 ตัว จะต่อกับ DATA BUS ได้ 8 เส้น ขา 2 ของออปโต้คัปเปอร์จะต่อลงกราวน์ของ 8255 CARD PORT ซึ่งจะต่อร่วมกับกราวน์ของคอมพิวเตอร์ ที่เอาต์พุตของออปโต้คัปเปอร์ ขา 5 ต่อกับไฟเลี้ยง +5 โวลท์ และต่อเข้ากับอินพุทของไอซี 74LS540 ซึ่งเป็นไอซี BUFFER INVERTER โดยขาอินพุทของไอซี 74LS540 มี 8ขา คือ A1 - A8 และ มีเอาท์พุท 8 ขา คือ $\overline{Y1}$ - $\overline{Y8}$ ขา 20 เป็นไฟเลี้ยงไอซี ขา 10 เป็นขากราวน์ ขา 1 ของไอซี คือ $\overline{G1}$ และขา 2 ของไอซี คือ $\overline{G2}$ ขา 1 และ ขา 2 จะต่อลงกราวน์ เพื่อให้สัญญาณสามารถปรากฏที่เอาท์พุทของ

ไอซีได้ จากเอาท์พุทของไอซี 74LS540 ต่อขนานกับตัวความต้านทานที่ต่ออนุกรมกับ LED เพื่อแสดงผลเอาท์พุทของไอซี 74LS540 และต่ออนุกรมกับตัวความต้านทานไบแอสที่ขาเบสของทรานซิสเตอร์ BC 547 และ ทรานซิสเตอร์ TIP 3055 ที่ต่อกันแบบดาร์ลิงตัน ซึ่งมีทั้งหมด 8ชุด แต่ละชุดต่ออยู่กับเอาท์พุทของไอซีแต่ละขาแยกจากกัน ที่ขาคอลเลคเตอร์ของทรานซิสเตอร์ TIP 3055 แต่ละชุดจะต่ออยู่กับขดลวดของ สเต็ปปิ้งมอเตอร์ โดย 1 ชุดต่อกับขดลวด 1ขด ซึ่งการต่อจะต้องต่อตามลำดับการทำงานของขดลวดที่ถูกต้อง มิเช่นนั้นมอเตอร์จะหมุนไม่เป็นลำดับ ซึ่งถ้าเป็นสเต็ปปิ้งมอเตอร์แบบ 6 สาย จะมีสายไฟขดลวด 4 เส้นและ CENTER TAP 2 เส้นโดยจะใช้ CENTER TAP ต่อเข้ากับไฟเลี้ยงของมอเตอร์ และที่เหลืออีก 4 เส้น ต่อเข้ากับขาคอลเลคเตอร์ของของทรานซิสเตอร์ TIP 3055 ของแต่ละชุดโดย สเต็ปปิ้งมอเตอร์ 1 ตัว จะต้องใช้ชุดของทรานซิสเตอร์ไดรฟ์ 4 ชุด เพราะฉะนั้น แผงวงจร DRIVER 1 แผง สามารถใช้ได้กับสเต็ปปิ้งมอเตอร์ 2 ตัว

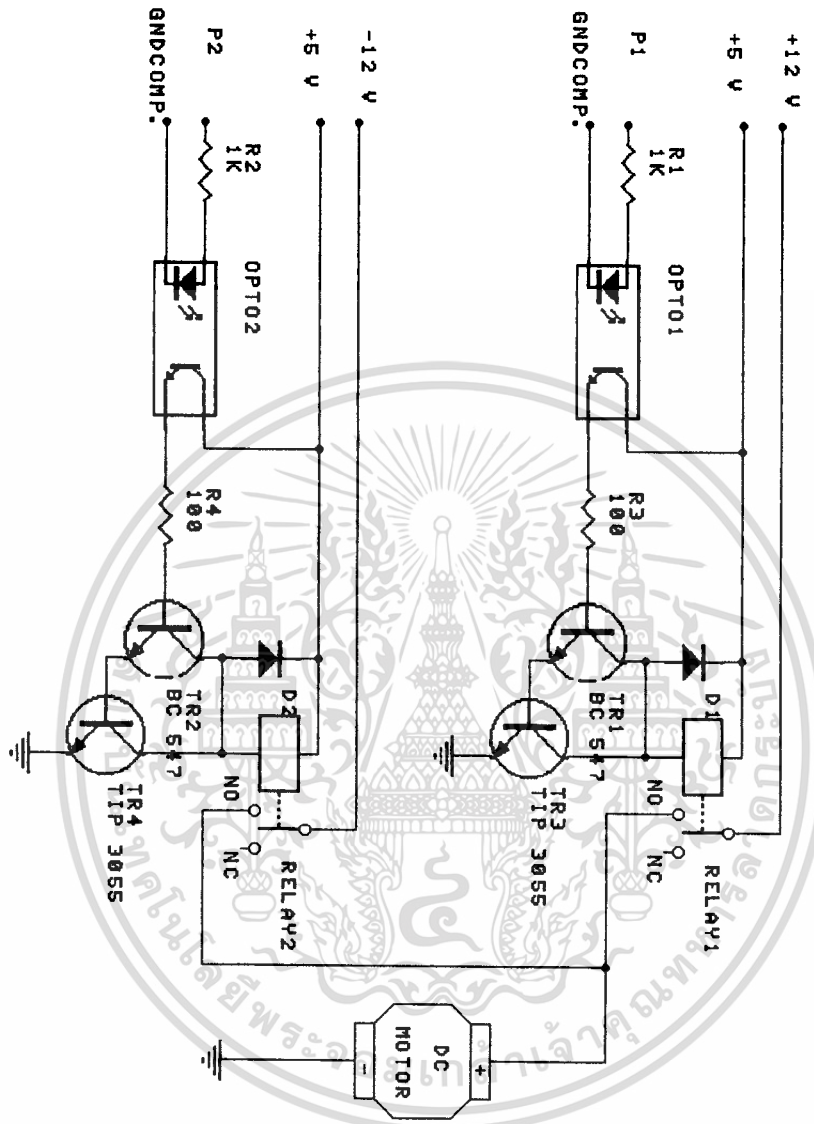


วงจร DRIVER ควบคุมการทำงานของ STEPPING MOTOR

วงจรควบคุมการทำงานของ DC MOTOR

วงจรควบคุมการทำงานของ ดี ซี มอเตอร์ จะประกอบไปด้วยออปโต้คัปเปอร์ 2 ตัว ซึ่งแต่ละตัวจะต่ออยู่กับสายสัญญาณข้อมูลจาก 8255 CARD PORT โดยออปโต้คัปเปอร์ 1 ตัวจะต่อกับสายสัญญาณข้อมูล 1 เส้น ซึ่งที่ขา 1 ของออปโต้คัปเปอร์จะต่อกับสายสัญญาณข้อมูล มี ตัวความต้านทาน 1 กิโลโอห์มต่ออนุกรมอยู่เพื่อลดทอนกระแสและแรงดัน ส่วนขา 2 ของออปโต้คัปเปอร์ต่อลงกราวน์ของ 8255 CARD PORT ขา 5 ต่อกับไฟเลี้ยง + 5 โวลท์ ขา 4 ต่อผ่านตัวความต้านทาน 100 โอห์มเข้าที่ขาเบสของทรานซิสเตอร์ BC 547 ซึ่งต่อแบบคาร์ลิงตันกับทรานซิสเตอร์ TIP 3055 และขาอิมิตเตอร์ของทรานซิสเตอร์ TIP 3055 ต่อลงกราวน์ ส่วนขาคอลเลคเตอร์ต่อเข้ากับขดลวดของรีเลย์ ที่ขดลวดรีเลย์มีไดโอดต่อขนานกับขดลวดและต่อเข้ากับไฟเลี้ยงขดลวดรีเลย์ +5 โวลท์ ชุดวงจรควบคุมรีเลย์มี 2 ชุด คือ ชุดที่ 1 จะแป็นไฟเลี้ยงมอเตอร์ +12 โวลท์ และชุดที่ 2 จะเป็นไฟเลี้ยงมอเตอร์ -12 โวลท์ สาเหตุที่ต้องใช้ไฟเลี้ยงมอเตอร์ 2 ชุดเพื่อให้ ดี ซี มอเตอร์สามารถหมุนได้ 2 ทิศทาง

ฮาร์ดแวร์ควบคุมการทำงานของ ดี ซี มอเตอร์ ซึ่งเป็นส่วนล้อสำหรับการเคลื่อนที่ของหุ่นยนต์ จะประกอบด้วย 8255 CARD PORT ซึ่งเสียบอยู่กับ SLOT ของคอมพิวเตอร์ต่อสายสัญญาณข้อมูลมายังวงจรควบคุมการทำงานของ ดี ซี มอเตอร์โดยใช้สายแพ ซึ่งวงจรควบคุมการทำงานของ ดี ซี มอเตอร์จะใช้สายสัญญาณข้อมูลเพียง 2 เส้น และสายกราวน์ของ 8255 CARD PORT อีก 1 เส้น การควบคุมการทำงานของ ดี ซี มอเตอร์จะใช้โปรแกรม PASCAL ควบคุมการทำงานของรีเลย์ ให้ทำงานทีละ 1 ตัว ซึ่งชุดรีเลย์ 1 ชุดจะควบคุมทิศทางการหมุนของ ดี ซี มอเตอร์ 1 ทิศทาง ใน 2 ทิศทาง คือหมุนเดินหน้าหรือถอยหลัง ข้อควรระวังคือชุดของรีเลย์ 2 ชุดจะทำงานพร้อมกันไม่ได้จะทำให้เกิดการลัดวงจรของแหล่งจ่ายไฟ



วงจรควบคุมการทำงานของ ดี ซี มอเตอร์

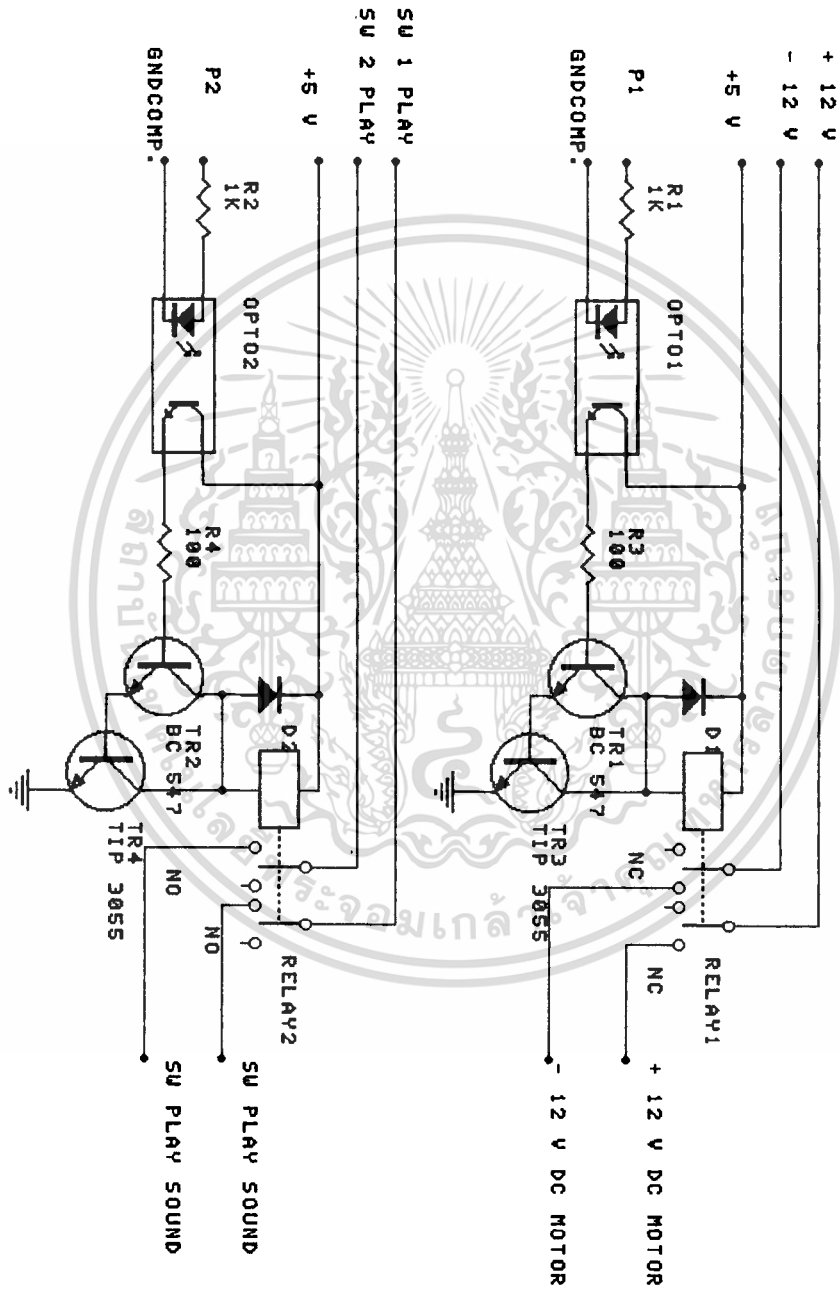
การทำงานของภาคควบคุมเสียงพูดและป้องกัน ดี ซี มอเตอร์

วงจรควบคุมเสียงพูดประกอบด้วยออปโต้คัปเปอร์ 1 ตัว สำหรับเชื่อมต่อสัญญาณข้อมูลจาก 8255 CARD PORT โดยขา 1 ต่อเข้ากับสายสัญญาณข้อมูลโดยมี ตัวความต้านทาน 1 กิโลโอห์มต่ออนุกรมอยู่ และขา 2 ต่อลงกราวน์ของ 8255 CARD PORT ส่วนขา 5 ต่อเข้ากับไฟ + 5 โวลท์และที่เอาท์พุทขา 4 ของออปโต้คัปเปอร์ต่อเข้าอนุกรมกับตัวความต้าททาน 100 โอห์มและต่อเข้าที่ขาเบสของทรานซิสเตอร์คู่ ดาร์ลิงตัน BC 547 และ TIP 3055 และที่ขาคอลเลคเตเอร์ของทราซิสเตอร์ TIP 3055 ต่อเข้ากับขดลวดของรีเลย์ DPDT โดยมีไดโอดต่อคร่อมขดลวด ซึ่งในการควบคุมเสียงพูดจะใช้คอนแทคของรีเลย์เป็นสวิทช์สำหรับการ PLAY เสียงที่บันทึกไว้ในไอซี ISD 1420 ซึ่งไอซีนี้บันทึกเสียงได้นานประมาณ 20 วินาที

การควบคุมเสียงพูดทำได้โดยการส่งสัญญาณข้อมูลจาก 8255 CARD PORT มายังวงจรควบคุมเสียงพูดเพื่อควบคุมการทำงานของรีเลย์ซึ่งทำหน้าที่เป็นสวิทช์ PLAY เสียงพูดที่ถูกบันทึกไว้ในไอซี ISD 1420 โดยจะต้องตั้งค่าหน่วงเวลาตามที่ต้องการให้ PLAY เสียงพูด

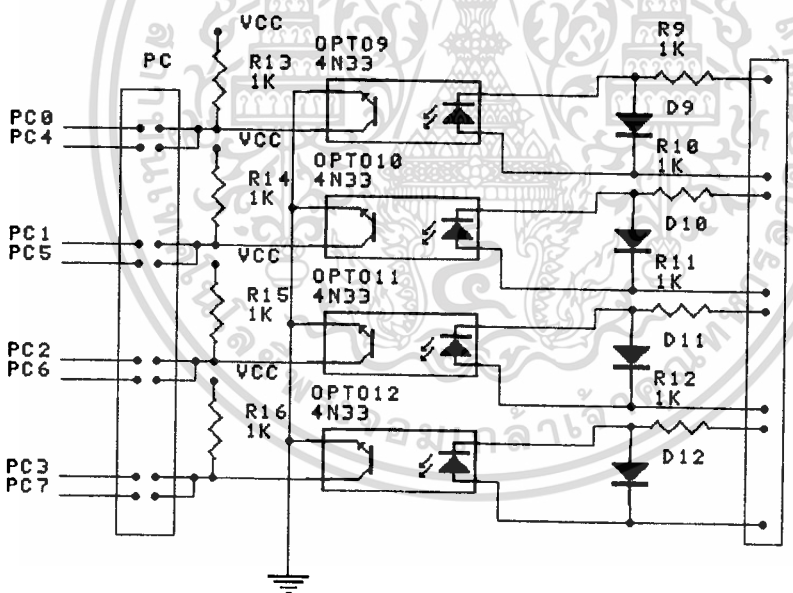
วงจรป้องกัน ดี ซี มอเตอร์มีวงจรเหมือนกับวงจรควบคุมเสียงพูด แต่วงจรป้องกัน ดี ซี มอเตอร์จะใช้คอนแทคของรีเลย์เป็นตัวป้องกันไฟบวก และไฟลบ ไหลเข้า ดี ซี มอเตอร์พร้อมกันในขณะเริ่มเปิดสวิทช์ใช้งานคอมพิวเตอร์ ซึ่งการเปิดสวิทช์ คอมพิวเตอร์ หรือ กดปุ่ม RESET ก็ตามจะทำให้เอาท์พุตของ 8255 CARD PORT เป็น "1" ตลอดทุกพอร์ตข้อมูลของไอซี 8255 แต่ละตัว ซึ่งจะทำให้ชุดควบคุม ดี ซี มอแตอร์และชุดควบคุมอื่นๆ ทำงานผิดพลาด แต่ที่เป็นอันตรายมากที่สุดคือชุดควบคุม ดี ซี มอเตอร์ ซึ่งจะทำให้เกิดการลัดวงจรของไฟ +12 โวลท์ และ ไฟ - 12 โวลท์ ทำให้เกิดการลัดวงจรซึ่งทำให้วงจรจ่ายไฟและตัว ดี ซี มอเตอร์ได้รับความเสียหาย

การป้องกันดี ซี มอเตอร์ทำได้โดย ต่อไฟ + 12 โวลท์ และ - 12 โวลท์ ที่จะป้อนเข้าสู่วงจรควบคุมการทำงานของ ดี ซี มอเตอร์ไว้กับคอนแทค NC แต่ละชุด ของรีเลย์ ซึ่ง ถ้าเปิดเครื่องคอมพิวเตอร์หรือกดปุ่ม รีเซ็ต รีเลย์จะทำงานและตำแหน่งของคอนแทคจะอยู่ที่ NO ทำให้ไฟ +12 โวลท์ และ - 12 โวลท์ ไม่สามารถไหลผ่านรีเลย์ไปได้ จึงป้องกันการเกิดการลัดวงจรได้



วงจรควบคุมเสียงพูดและป้องกัน ดี ซี มอเตอร์

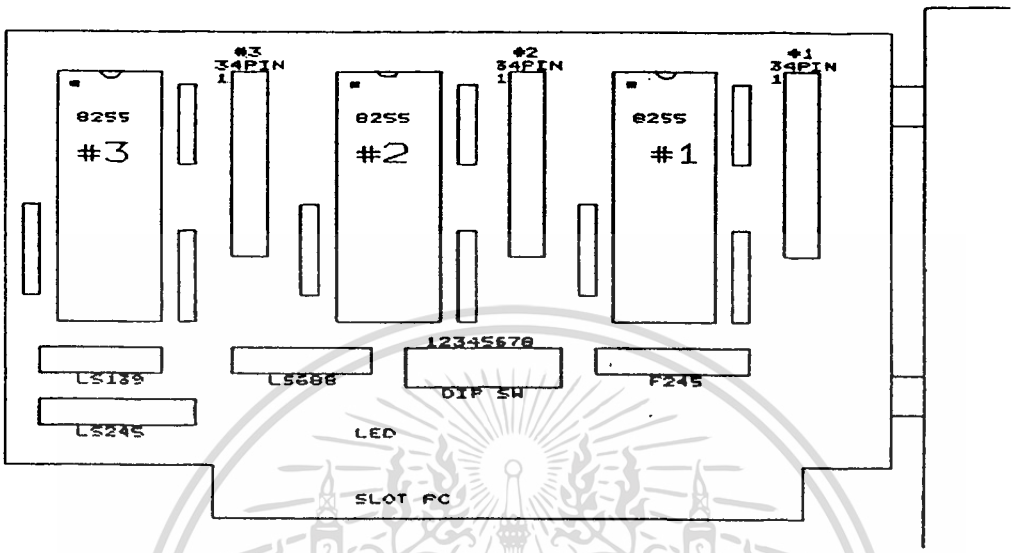
การใช้พอร์ต C ของ 8255 ทำงานเป็นอินพุตพอร์ตก็อาศัยออปโต้คัปเปอร์เป็นตัวดีเท็คสัญญาณที่เข้ามาทางอินพุท โดยสัญญาณที่เข้ามาจะมีระดับแรงดันไฟฟ้าเป็น 0 โวลท์ หรือ 5โวลท์ ผ่านตัวต้านทานลดทอนกระแสและผ่านไดโอดป้องกันแรงไฟขั้วที่กลับขั้ว จากนั้นผ่านเข้าตัวออปโต้คัปเปอร์ ซึ่งเป็นตัวตรวจสอบสัญญาณที่อินพุต คือ ถ้าอินพุทมีระดับแรงดัน เป็น 0โวลท์ เอาท์พุตจะไม่ต่อลงกราวน์มีระดับลอจิกเป็น “1” แต่ถ้าอินพุตมีระดับแรงดันเป็น 5 โวลท์ เอาท์พุทจะถูกต่อลงกราวน์คือทรานซิสเตอร์ทำงาน จึงมีระดับลอจิกเป็น “0” สัญญาณจะผ่านเข้าพอร์ตของ 8255 CARD PORT เมื่อกำหนดให้พอร์ต C ทำหน้าที่เป็นพอร์ตอินพุทและรับสัญญาณเข้ามา



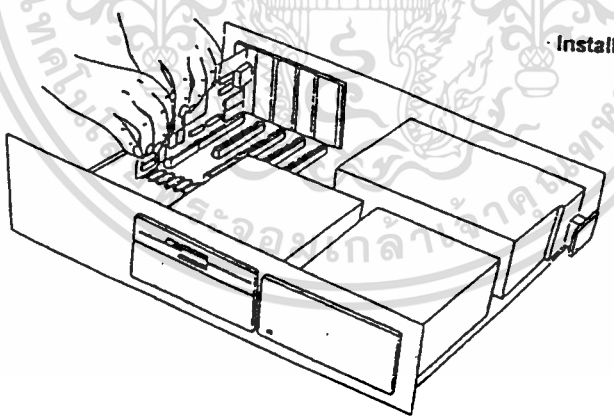
วงจรดีเทคสัญญาณอินพุท

<u>การเชื่อมต่อคอมพิวเตอร์กับฮาร์ดแวร์ควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์</u>

นำ 8255 CARD PORT เสียบที่ SLOT ของคอมพิวเตอร์และต่อสายแพมายังวงจรควบคุมการทำงานของหุ่นยนต์ (DRIVER)



ลักษณะของ 8255 CARD PORT



การติดตั้ง 8255 CARD PORT กับเครื่องคอมพิวเตอร์ PC

<u>การติดตั้ง 8255 CARD PORT</u>

1. ปิดสวิทช์ POWER ของเครื่องคอมพิวเตอร์ PC นั้นก่อน และเปิดฝาเครื่องออออก
2. SET DIP SW. ตำแหน่ง PORT ของ CARD ไม่ให้ตรงกับตำแหน่งของ CARD อื่นๆ
3. นำ CARD ใส่เข้าไปยังเครื่องคอมพิวเตอร์ PC ทาง SLOT PC 62 PIN
4. ต่อสายแพจาก 34 PIN ไปใช้งานให้เรียบร้อยก่อนและเปิดเครื่องคอมพิวเตอร์ใช้งาน

กำหนดการทำงานของ 8255 CARD PORT โดยกำหนดการ DECODE PORT ณ ตำแหน่งแอดเดรสที่ต้องการ

Hex Range	Usage
000-00F	DMA Chip 8237A-5
020-021	Interrupt 8259A
040-043	Timer 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Registers
0AX*	NMI Mask Register
0CX	Reserved
0EX	Reserved
200-20F	Game Control
210-217	Expansion Unit
220-24F	Reserved
278-27F	Reserved
2F0-2F7	Reserved
2F8-2FF	Asynchronous Communications (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Printer
380-38C**	SDLC Communications
380-389**	Binary Synchronous Communications (Secondary)
3A0-3A9	Binary Synchronous Communications (Primary)
3B0-3BF	IBM Monochrome Display/Printer
3C0-3CF	Reserved
3D0-3DF	Color/Graphics
3E0-3E7	Reserved
3F0-3F7	Diskette
3F8-3FF	Asynchronous Communications (Primary)

* At power-on time, the Non Mask Interrupt into the 8088 is masked off. This mask bit can be set and reset through system software as follows:

Set mask: Write hex 80 to I/O Address hex A0 (enable NMI)

Clear mask: Write hex 00 to I/O Address hex A0 (disable NMI)

** SDLC Communications and Secondary Binary Synchronous Communications cannot be used together because their hex addresses overlap.

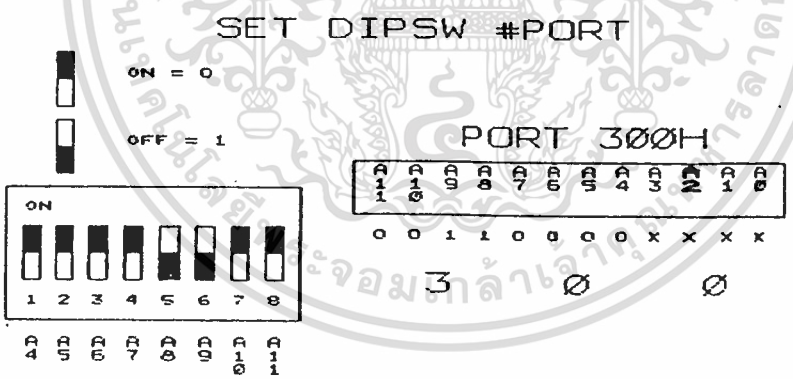
ตาราง I/O ADDRESS MAP

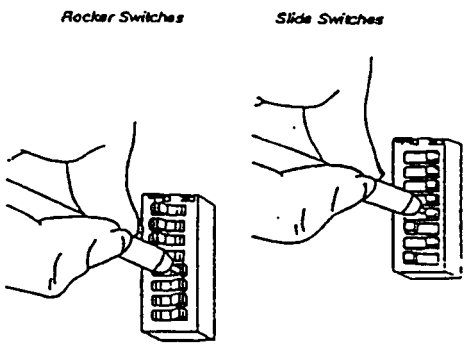
การ DECODE PORT 8255 บน CARD จะใช้ ไอซี ทีทีแอล 74LS 688 ,ไอซี ทีทีแอล 74LS 139 และ DIP SWITCH 8 PIN เป็นวงจร DECODE เพื่อให้สามารถปรับ SET DIP SW. ตั้งตำแหน่งเบอร์ PORT ของ CARD ได้ โดยในการปรับ DIP SW. นั้นจะต้องไม่ไปตรงกับตำแหน่งของ PORT ของ COMPUTER PC นั้นด้วย โดย 8255 CARD PORT จะใช้ตำแหน่งพอร์ต 12 PORT ต่อ 1 CARD

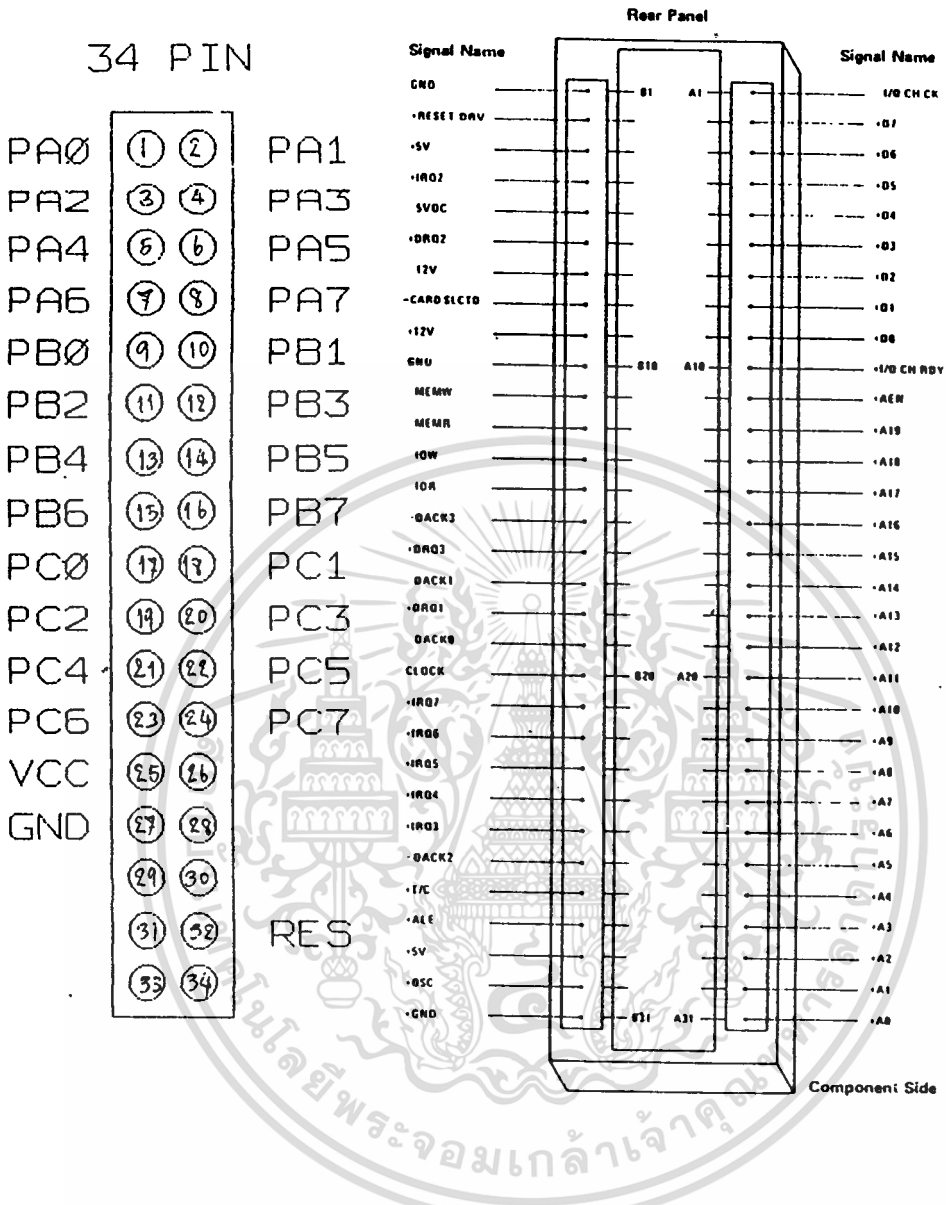
DECODE PORT

XX0H	PORT A (1)
XX1H	PORT B (1)
XX2H	PORT C (1)
XX3H	PORT CONTROL (1)
XX4H	PORT A (2)
XX5H	PORT B (2)
XX6H	PORT C (2)
XX7H	PORT CONTROL (2)
XX8H	PORT A (3)
XX9H	PORT B (3)
XXAH	PORT C (3)
XXBH	PORT CONTROL (3)

การตั้งเบอร์ DECODE PORT ได้โดยการปรับ DIP SW.ซึ่งก็มีค่าเท่ากับค่า ADDRESS นั้นๆ เช่นเราตั้งตำแหน่ง 300H จะ SET DIP SW. ดังนี้







ตำแหน่งเอาท์พุทขาต่างของ 8255 CARD PORT และ SLOT ของ COMPUTER

จากรูปข้างบนตำแหน่งของเอาท์พุทต่างๆของ 8255 CARD PORT เป็นดังนี้

PIN NUMBER 1 - 8 คือ PA0 - PA7

PIN NUMBER 9 - 16 คือ PB0 - PB7

PIN NUMBER 17 - 24 คือ PC0 - PC7

PIN NUMBER 25 คือ VCC

PIN NUMBER 27 คือ GND

PINNUMBER 32 คือ RESET

โดย 8255 CARD PORT จะมี SLOT 34 PIN 3 ชุด ใน 1 CARD

ต่อสายสัญญาณข้อมูลจาก 8255 CARD PORT ไปยังภาค DRIVER ดังนี้

PORT PA1 ของไอซี 8255 ตัวที่ 1 ให้ PA4 - PA7 ต่อสายแพไปยังวงจรควบคุมเสต็ปปิ้งมอเตอร์หมุนทิศทางของล้อหน้าของหุ่นยนต์ (FRONT WHEEL TURN) ตั้งชื่อสเต็ปปิ้งมอเตอร์หมุนทิศทางล้อหน้าของหุ่นยนต์ เป็น M7

PORT PB1 ของไอซี 8255 ตัวที่ 1 ให้ PB4 - PB7 ต่อสายแพไปยังวงจรควบคุมสเต็ปปิ้งมอเตอร์หมุนส่วนหัวของหุ่นยนต์ (HEAD TURN) ตั้งชื่อสเต็ปปิ้งมอเตอร์หมุนส่วนหัวของหุ่นยนต์เป็น M1

PORT PC1 ของ ไอซี 8255 ตัวที่ 1 ให้ PC4 - PC7 ต่อสายแพไปยังวงจรควบคุม ดีซี มอเตอร์ขับเคลื่อนให้ล้อหมุน (FRONT WHEEL MOVE) ตั้งชื่อ ดีซี มอเตอร์ขับเคลื่อนให้ล้อหมุน เป็น M8

PORT PC1 ของ ไอซี 8255 ตัวที่ 1 ให้ PC0 - PC3 ต่อสายแพไปยังวงจรควบคุมเสียงพูดของหุ่นยนต์ (SPEAK) ตั้งชื่อส่วนควบคุมเสียงพูดของหุ่นยนต์ เป็น SP

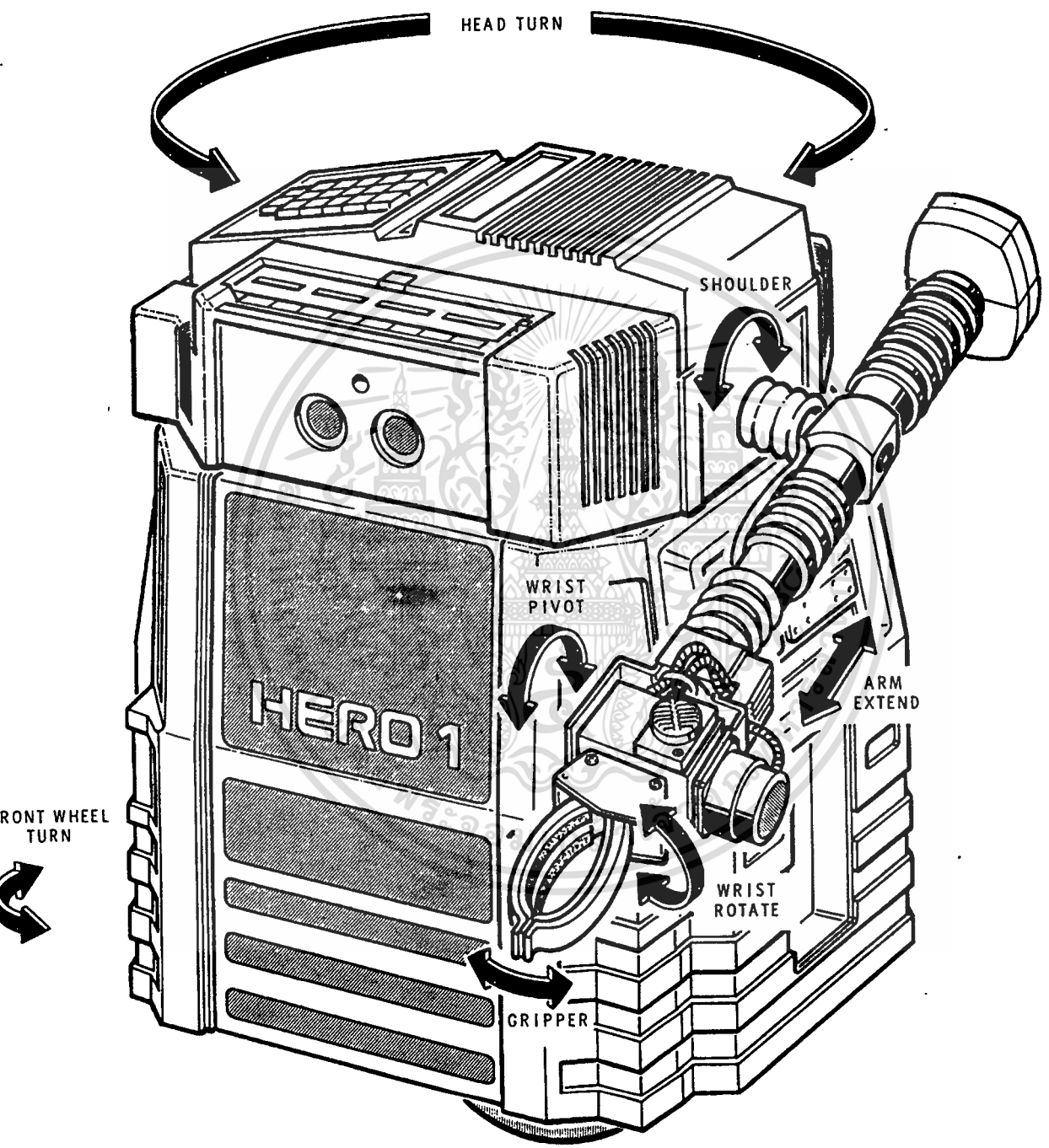
PORT PA2 ของไอซี 8255 ตัวที่ 2 ให้ PA4 - PA7 ต่อสายแพไปยังวงจรควบคุมสเต็ปปิ้งมอเตอร์ส่วนหมุนวงแขนของหุ่นยนต์ (SHOULDER) ตั้งชื่อสเต็ปปิ้งมอเตอร์ส่วนหมุนวงแขนของหุ่นยนต์ เป็น M2

PORT PB2 ของไอซี 8255 ตัวที่ 2 ให้ PB4 - PB7 ต่อสายแพไปยังวงจรควบคุมสเต็ปปิ้งมอเตอร์ส่วนยืดและหดแขนของหุ่นยนต์ (ARM EXTEND) ตั้งชื่อสเต็ปปิ้งมอเตอร์ส่วนยืดและหดแขนของหุ่นยนต์ เป็น M3

PORT PA3 ของไอซี 8255 ตัวที่ 3 ให้ PA4 - PA7 ต่อสายแพไปยังวงจรควบคุมสเต็ปปิ้งมอเตอร์ส่วนหมุนมือหนีบของหุ่นยนต์ (WRIST ROTATE) ตั้งชื่อสเต็ปปิ้งมอเตอร์ส่วนหมุนมือหนีบของหุ่นยนต์ เป็น M5

PORT PB3 ของไอซี 8255 ตัวที่ 3 ให้ PB4 - PB7 ต่อสายแพไปยังวงจรควบคุมสเต็ปปิ้งมอเตอร์ส่วนมือหนีบของหุ่นยนต์ (GRIPPER) ตั้งชื่อสเต็ปปิ้งมอเตอร์ส่วนมือหนีบของหุ่นยนต์ เป็น M6

PORT PC3 ของไอซี 8255 ตัวที่ 3 ให้ PC4 - PC7 ต่อสายแพไปยังวงจรควบคุมสเต็ปปิ้งมอเตอร์ส่วนหมุนข้อมือหนีบของหุ่นยนต์ (WRIST PIVOT) ตั้งชื่อสเต็ปปิ้งมอเตอร์ส่วนหมุนข้อมือหนีบของหุ่นยนต์ เป็น M4



รูปแสดงรูปร่างของหุ่นยนต์ ET - 18

บทที่ 4

การเขียนโปรแกรมเพื่อควบคุมการทำงานของหุ่นยนต์

การเขียนโปรแกรมเพื่อควบคุมการทำงานของหุ่นยนต์ในที่นี้จะใช้โปรแกรม PASCAL เป็นโปรแกรมติดต่อกับฮาร์ดแวร์ส่วนควบคุมต่างๆของหุ่นยนต์โดยใช้ 8255 CARD PORT

การกำหนด ADDRESS ของ PORT ต่างๆของ 8255 CARD PORT ให้เป็นดังนี้

```
PA1 = $300
PB1 = $301
PC1 = $302
PA2 = $304
PB2 = $305
PC2 = $306
PA3 = $308
PB3 = $309
PC3 = $30A
```

กำหนด ADDRESS ของ รหัส CONTROL PORT ต่างๆ ของ 8255 CARD PORT ให้เป็นดังนี้

```
PCONT_1 = $303
PCONT_2 = $307
PCONT_3 = $30B
```

กำหนดรหัสควบคุมการทำงานของ 8255 แต่ละตัวให้ทำงานในโหมด 0 และเป็น OUTPUT PORT ทั้งหมด

```
PORT[PCONT_1]  := $80   {PA1,PB1,PC1 = OUTPUT PORT}
PORT[PCONT_2]  := $80   {PA2,PB2,PC2 = OUTPUT PORT}
PORT{PCONT_3} := $80   {PA3,PB3,PC3 = OUTPUT PORT}
```

กำหนด DATA ต่างๆ โดยกำหนดในรูปของ ARRAY [.....] กำหนดขนาดและตั้งชื่อ ARRAY แต่ละ ARRAY ดังนี้

```
SP_DATA = ARRAY [1..4] OF BYTE  {เสียงพูด}
FR_DATA = ARRAY [1..4] OF BYTE {สเต็ปปิ้งมอเตอร์หมุนตามเข็มนาฬิกา}
RV_DATA = ARRAY [1..4] OF BYTE {สเต็ปปิ้งมอเตอร์หมุนทวนเข็มนาฬิกา}
MOVEFORW_DATA =  ARRAY [1..4] OF BYTE {ดี ซี มอเตอร์ ล้อหมุนเดินหน้า}
MOVEREV_DATA = ARRAY [1..4] OF BYTE  {ดี ซี มอเตอร์ ล้อหมุนถอยหลัง}
```

C_DATA = ARRAY [1..2] OF BYTE {เคลียร์ ข้อมูลที่เอาท์พุต}

กำหนดการส่งผ่านข้อมูล และกำหนดค่าของข้อมูลต่างๆ

DATA_OUT : SP_DATA = ($10,$10,$10,$10)

DATA_OUT1 : FR_DATA = ($01,$02,$04,$08)

DATA_OUT2 : RV_DATA = ($08,$04,$02,$01)

DATA_OUT3 : MOVFORW_DATA = ($03,$03,$03,$03)

DATA_OUT4 : MOVREV_DATA =($0A,$0A,$0A,$0A)

CLR_DATA : C_DATA = ($00,$00)

กำหนด รอบการหมุนของ สเต็ปปิ้งมอเตอร์ และ ดีซีมอเตอร์

FOR I := 1 TO 4 {จำนวนรอบการทำงานของมอเตอร์}

FOR X := 1 TO (จำนวนรอบการหมุนของมอเตอร์)

เช่น FOR X := 1 TO 100 {สเต็ปปิ้งมเตอร์จะหมุน = 100 x 4 STEP}

กำหนด PORT ที่จะนำข้อมูลออกที่ OUTPUT ของ 8255 ตัวใด และพอร์ตใดของ 8255 ตัวนั้น พร้อมทั้งกำหนดชื่อข้อมูลที่ต้องการให้ออก OUTPUT

PORT [(พอร์ตที่ต้องการจะนำข้อมูลออก OUTPUT)] := ชื่อข้อมูลที่ต้องการ

เช่น PORT [PA1] := DATA_OUT [I] {ข้อมูลชื่อ DATA_OUT ออกที่พอร์ต PA1}

กำหนดการหน่วงเวลาการทำงานของการส่งข้อมูลออก OUTPUT

DELAY({เวลาที่ต้องการหน่วงคิดเป็น 1 / 1000 วินาที(mS)})

เช่น DELAY(10) { เวลาที่หน่วง = 10 / 1000 วินาที }

<u>ตัวอย่างโปรแกรม</u>

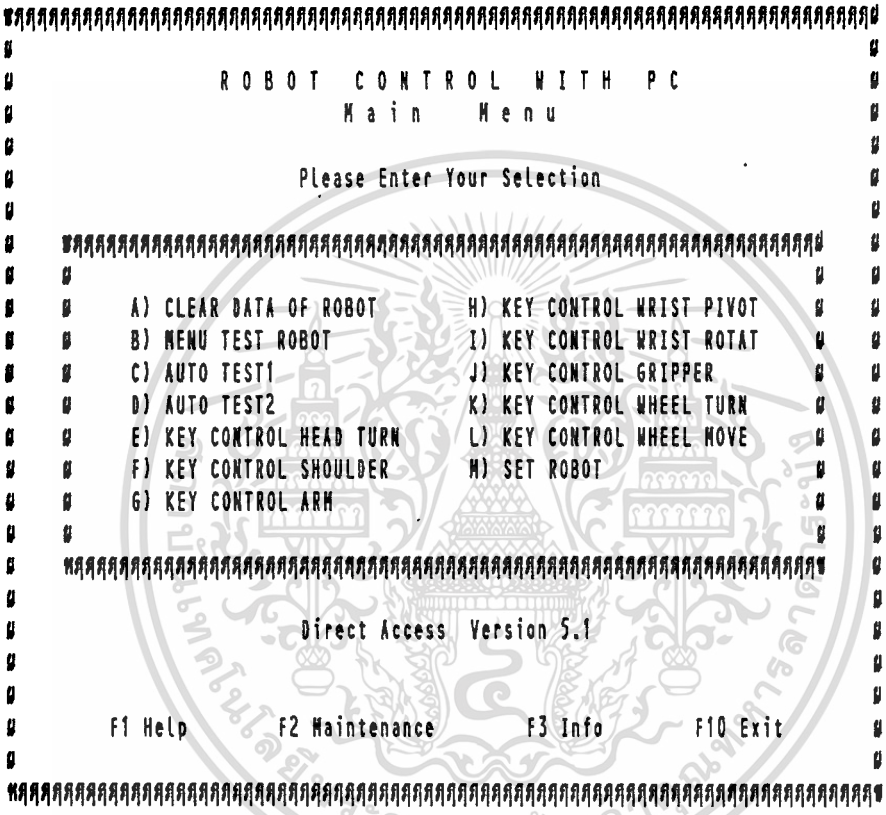
```
PROGRAM_CONTROL_ STEPPING_ MOTOR : { ชื่อของโปรแกรม}
USES CRT ;
VAR    I : BYTE ;        {กำหนดชนิดของตัวแปร;
       X · INTEGER ;
TYPE   FR_DATA = ARRAY [1..4] OF BYTE ; {กำหนดชื่อและชนิดของข้อมูล}
CONST  PA1 = $0300 ;   {กำหนดค่าตัวแปรคงที่ }
       PCONT_1 = $0303 ;
       DATA_OUT : FR_DATA = ($01,$02,$04,$08) ; {กำหนด  DATA}
BEGIN
      PORT [PCONT_1] := $80 ;  {PA1,PB1,PC1 = OUTPUT PORT }
          FOR  X :=  1 TO 100 DO {กำหนดรอบการหมุนของมอเตอร์}
           BEGIN
```

```
            FOR  I := 1 TO 4 DO {กำหนดรอบการทำงาน}
              BEGIN
            PORT [PA1] := DATA _OUT[I] ; {กำหนดพอร์ตที่ส่งข้อมูลออก}
               DELAY (50) ;  { กำหนดการหน่วงเวลา}
        END ;
      END;
END.
```

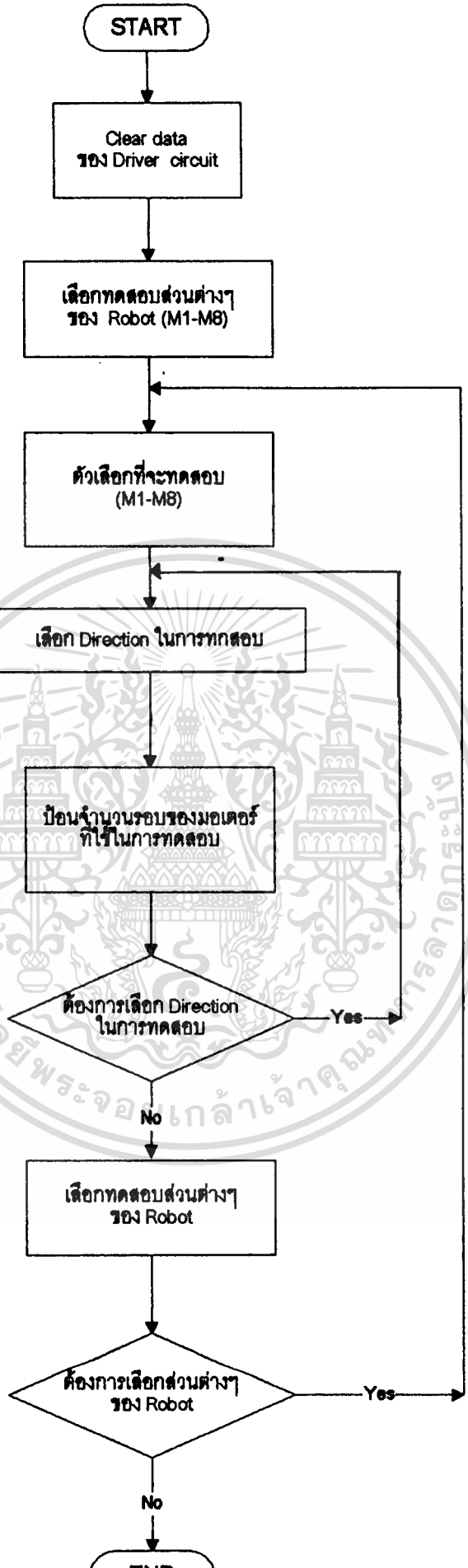
โปรแกรม PASCAL สามารถ COMPILE เป็นไฟล์ (FILE)ที่มีนามสกุลเป็น EXE เพื่อสะดวกในการเรียกใช้งาน เมื่อจะใช้งานโปรแกรมก็สามารถเรียก ชื่อไฟล์ นั้นแล้วกด ENTER เพื่อ RUN โปรแกรมนั้นได้

บทที่ 5

การใช้งานโปรแกรมควบคุมการทำงานของหุ่นยนต์



เมนูควบคุมการทำงานของส่วนต่างๆของหุ่นยนต์



FLOW CHARTทดสอบการทำงานของส่วนต่างๆของหุ่นยนต์

ลักษณะการทำงานของ FLOW CHART

จากรูปแบบของ FOLW CHART โปรแกรมทดสอบการทำงานของส่วนต่างๆ ของหุ่นยนต์จะมีขั้นตอนการทำงานดังนี้

ขั้นตอนที่ 1 เริ่มต้นด้วย START

ขั้นตอนที่ 2 ทำการ CLEAR DATA ของวงจร DRIVER ของสเต็ปปิ้งมอเตอร์เพื่อไม่ให้มีข้อมูลค้างอยู่

ขั้นตอนที่ 3 เลือกทดสอบส่วนต่างๆ ของหุ่นยนต์ โดยมีเมนู (M1 - M8) จะปรากฏที่หน้าจอคอมพิวเตอร์ ซึ่งเราสามารถเลือกที่จะทดสอนการทำงานแต่ละส่วนของหุ่นยนต์ได้

ขั้นตอนที่ 4 รับตัวเลือกที่จะทดสอบ (M1 - M8) จะมีเคอร์เซอร์กระพริบอยู่พร้อมที่จะรับตัวเลือก

ขั้นตอนที่ 5 เป็นการเลือกทิศทาง (DIRECTION) ในการทดสอบจะมีเมนูปรากฏอยู่เราสามารถเลือกทิศทางการทำงานของหุ่นยนต์ได้

ขั้นตอนที่ 6 ป้อนจำนวนรอบของมอเตอร์ที่ใช้ในการทดสอบ

ขั้นตอนที่ 7 เป็นการเปรียบเทียบเงื่อนไขที่ว่าถ้าต้องการเลือก DIRECTION ต่อไป ก็ให้กลับไปทำในขั้นตอนที่ 5 เพื่อเลือกทิศทางในการทดสอบ แต่ถ้าไม่ต้องการเลือกทิศทางก็ให้ทำในขั้นตอนต่อไป

ขั้นตอนที่ 8 เลือกทดสอบการทำงานส่วนต่างๆ ของหุ่นยนต์

ขั้นตอนที่ 9 เป็นการเปรียบเทียบเงื่อนไขที่ว่าถ้าต้องการที่จะเลือกทดสอบการทำงานของส่วนต่างๆ ของหุ่นยนต์ ก็ให้กลับไปทำในขั้นตอน 4 แต่ถ้าไม่ต้องการที่จะทดสอบการทำงานของหุ่นยนต์ก็ให้ทำในขั้นตอนต่อไป

ขั้นตอนที่ 10 เป็นการจบการทำงานของ FLOW CHART

โปรแกรมทดสอบการทำงาน

เป็นโปรแกรมที่ใช้ทดสอบการทำงานของหุ่นยนต์ โดยจะทดสอบการทำงานของมอเตอร์แต่ละตัวซึ่งมีทั้งหมด 8 ตัว และทดสอบการทำงานของวงจรเสียงพูดของหุ่นยนต์ โดยการทดสอบการทำงานของหุ่นยนต์นี้ สามารถเรียกได้จากโปรแกรมที่ชื่อว่า KEY1 โปรมแกรมจะเข้าสู่การทำงานโดยอัตโนมัติดังนี้

C:\ KEY1

กดปุ่ม ENTER

จะแสดงผลที่หน้าจอดังนี้

```
THE NUMBER TO CONTROL OPERATION OF ROBOT
0 : CLEAR OUTPUT OF DRIVER CIRCUIT ( CLR )
1 : CONTROL HEAD TURN OF ROBOT ( M1 )
2 : CONTROL SHOULER OF ROBOT ( M2 )
3 : CONTROL ARM EXTEND OF ROBOT ( M3 )
4 : CONTROL WRIST PIVOT OF ROBOT ( M4 )
5 : CONTROL WRIST ROTATE OF ROBOT ( M5 )
6 : CONTROL GRIPPER OF ROBOT ( M6 )
7 : CONTROL FRONT WHEEL TURN OF ROBOT ( M7 )
8 : CONTROL FRONT WHEEL MOVE OF ROBOT ( M8 )
9 : CONTROL SPEAK OF ROBOT ( SP )
10 : EXIT PROGRAM
SELECT THE NUMBER TO CONTROL OPERATION OF ROBOT
_
```

เมื่อ RUN โปรแกรม KEY1 จะแสดงผลที่หน้าจอดังแสดงตามข้อความข้างบน และจะมีเคอร์เซอร์กระพริบอยู่เพื่อรับค่าของโปรแกรมย่อยต่างๆ

โปรแกรม 0

โปรแกรมนี้เป็นโปรแกรมที่ใช้สำหรับ CLEAR ข้อมูลที่ OUTPUT ของ DRIVER CIRCUIT ทุกตัว เพื่อไม่ให้มีข้อมูลค้าง

0

ENTER

โดยโปรแกรมนี้จะไม่แสดงการทำงานให้เห็น

โปรแกรม 1

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการหมุนหัวของหุ่นยนต์ (HEAD TURN) จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL MOTOR

0: HEAD TURN LEFT

1: HEAD TURN RIGHT

2: EXIT

_

และจะมีเคอร์เซอร์กระพริบอยู่ เพื่อทำการรอรับคีย์ 0 , 1 , 2 ซึ่งจะหมายความว่า

กด 0 เป็นการหมุนหัวหุ่นยนต์ไปทางซ้าย

กด 1 เป็นการหมุนหัวหุ่นยนต์ไปทางขวา

จะแสดงผลดังนี้

ENTER NUMBER OF STEP TO MOVE

_

(ป้อนจำนวนรอบในการหมุน แล้วกด ENTER)

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT TO PROGRAM

โปรแกรม 2

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการหมุนไหล่ (SHOULDER) จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL MOTOR

0: SHOULDER TURN ANTICLOCKWISE

1: SHOULDER TURN CLOCKWISE

2: EXIT

_

กด 0 เป็นการหมุนไหล่ของหุ่นยนต์ให้เคลื่อนที่ลง

กด 1 เป็นการหมุนไหล่ของหุ่นยนต์ให้เคลื่อนที่ขึ้น

จะปรากฏข้อความดังนี้

ENTER NUMBER OF STEP TO MOVE

_

(ใส่จำนวนรอบในการหมุนแล้วกด ENTER)

กด 2 ออกจากโปรแกรมย่อย

EXIT PROGRAM

โปรแกรม 3

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการยืดและหดแขนของหุ่นยนต์ (ARM EXTEND) จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL MOTOR

0 : ARM PULLING

1 : ARM EXTEND

2 : EXIT

_

กด 0 เป็นการหดแขนของหุ่นยนต์

กด 1 เป็นการยืดแขนของหุ่นยนต์

จะแสดงผลดังนี้

ENTER NUMBER OF STEP TO MOVE

_

(ใส่จำนวนรอบในการหมุนแล้วกด ENTER)

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT PROGRAM

โปรแกรม 4

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการหมุน WRIST PIVOT จะแสดงข้อความดังนี้

ENTER NUMBER OF STEP TO MOVE

0 : WRIST PIVOT MOVE ANTICLOCKWISE

1 : WRIST PIVOT MOVE CLOCKWISE

2 : EXIT

_

กด 0 เป็นการหมุน WRIST PIVOT ให้เคลื่อนที่ลง

กด 1 เป็นการหมุน WRIST PIVOT ให้เคลื่อนที่ขึ้น

จะแสดงผลดังนี้

ENTER NUMBER OF STEP TO MOVE

_

(ใส่จำนวนรอบในการหมุนแล้วกด ENTER)

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT PROGRAM

โปรแกรม 5

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการหมุน WRIST ROTATE จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL MOTOR

0 : WRIST ROTATE ANTICLOCKWISE

1 : WRIST ROTATE CLOCKWISE

2 : EXIT

_

กด 0 เป็นการหมุน WRIST ROTAE ทวนเข็มนาฬิกา

กด 1 เป็นการหมุน WRIST ROTAE ตามเข็มนาฬิกา

จะแสดงผลดังนี้

ENTER NUMBER OF STEP TO MOVE

_

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT PROGRAM

โปรแกรม 6

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการหนีบ (GRIPPER) จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL MOTOR

0 : GRIPPER MOVE EXPAND

1 : GRIPPER MOVE GRIP

2 : EXIT

_

กด 0 เป็นการอ้าออก (EXPAND)

กด 1 เป็นการหนีบเข้า (GRIP)

จะแสดงผลดังนี้

ENTER NUMBER OF STEP TO MOVE

_

(ใส่จำนวนรอบในการหมุนแล้วกด ENTER)

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT PROGRAM

โปรแกรม 7

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการหมุนล้อ (FRONT WHEEL TURN) จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL MOTOR

0 : FRONT WHEEL TURN LEFT

1 : FRONT WHEEL TURN RIGHT

2 : EXIT

_

กด 0 เป็นการหมุนล้อให้ไปทางซ้าย

กด 1 เป็นการหมุนล้อให้ไปทางขวา

จะแสดงผลดังนี้

ENTER NUMBER OF STEP TO MOVE

_

(ใส่จำนวนรอบในการหมุนแล้วกด ENTER)

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT PROGRAM

โปรแกรม 8

โปรแกรมนี้เป็นโปรแกรมที่ใช้ทดสอบการหมุนล้อให้เดินหน้าและถอยหลัง จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL MOTOR

0 : FRONT WHEEL MOVE FORWARD

1 : FRONT WHEEL MOVE BACKWARD

2 : EXIT

_

กด 0 เป็นการหมุนล้อให้เคลื่อนที่ไปข้างหน้า

กด 1 เป็นการหมุนล้อให้เคลื่อนที่ไปข้างหลัง

จะแสดงผลดังนี้

ENTER NUMBER OF STEP TO MOVE

_

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT PROGRAM

โปรแกรม 9

เป็นโปรแกรมที่ใช้ทดสอบการทำงานของวงจรเสียงพูด จะแสดงผลดังนี้

ENTER THE DIRECTION TO CONTROL SPEAK OF ROBOT

1 : SPEAK OF ROBOT

2 : EXIT

_

กด 1 เป็นการทดสอบวงจรเสียงพูด

กด 2 เป็นการออกจากโปรแกรมย่อย

EXIT PROGRAM

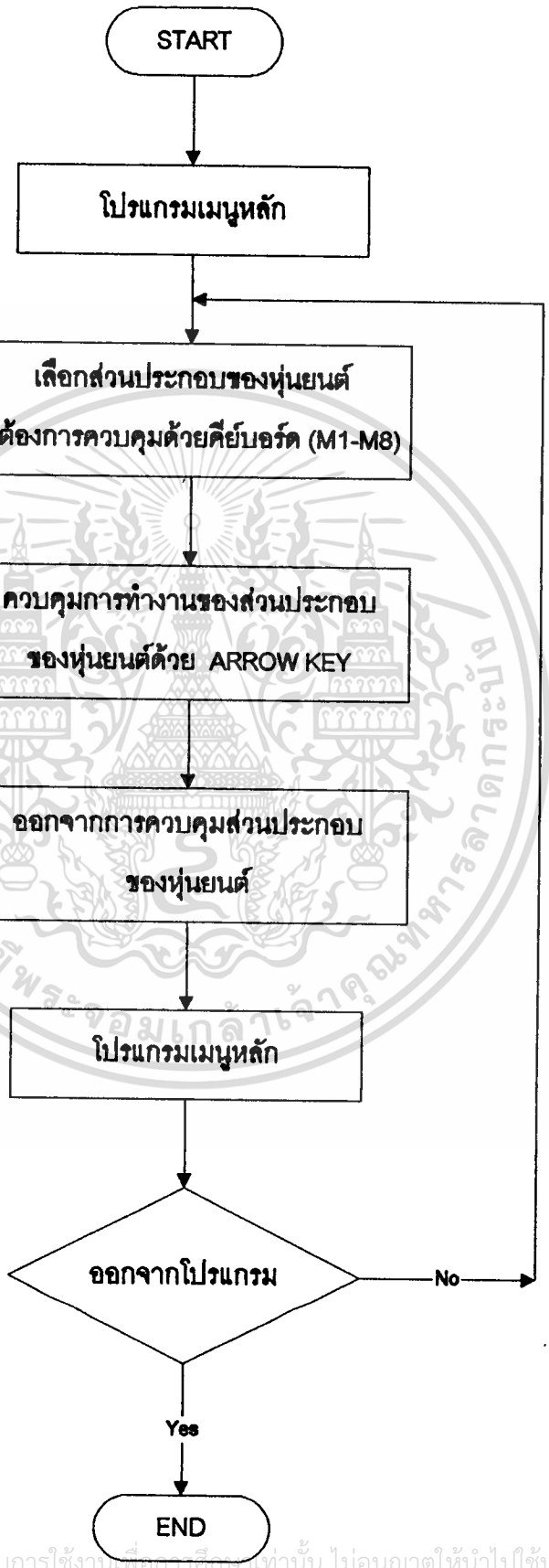
โปรแกรม 10

เป็นโปรแกรมที่ใช้ออกจากโปรแกรมหลักหรือโปรแกรมเมนู (EXIT PROGRAM)

****

FLOW CHART

CONTROL ROBOT BY KEYBOARD



การทำงานของ FLOW CHART

การทำงานของ FLOW CHART ของโปรแกรม CONTROL ROBOT BY KEYBOARD นี้สามารถที่จะเลือกควบคุมส่วนต่างๆ ของหุ่นยนต์ได้ง่าย โดยการเลือกจากโปรแกรมเมนูหลัก ซึ่งส่วนประกอบของหุ่นยนต์ที่ใช้ในการควบคุมมีทั้งหมด 8 ส่วน ในการควบคุมส่วนประกอบของหุ่นยนต์สามารถควบคุมได้โดยการใช้คีย์บอร์ด(ARROW KEY) ซึ่งการทำงานของ FLOW CHART สามารถอธิบายเป็นขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 เริ่มต้นการทำงาน

ขั้นตอนที่ 2 เข้าสู่โปรแกรมเมนูหลัก โดยจะแสดงส่วนประกอบของต่างๆ ของหุ่นยนต์

ขั้นตอนที่ 3 เลือกส่วนประกอบของหุ่นยนต์ที่ต้องการควบคุม

ขั้นตอรที่ 4 ควบคุมการทำงานของส่วนประกอบของหุ่นยนต์ด้วย ARROW KEY ซึ่งในการควบคุมแต่ละส่วนของหุ่นยนต์ เราจะใช้ฟังค์ชันคีย์ใดๆ ในการควบคุมแต่ละส่วน ซึ่งจะอธิบายในส่วนของการทำงานของโปรแกรม

ขั้นตอนที่ 5 เป็นการออกจากการควบคุมส่วนประกอบของหุ่นยนต์

ขั้นตอนที่ 6 กลับเข้าสู่เมนูหลัก

ขั้นตอนที่ 7 ในขั้นตอนนี้จะเป็นการกำหนดเงื่อนไขถ้าไม่ต้องการออกจากโปรแกรมก็ให้กลับไปทำในขั้นตอนที่ 3 แต่ถ้าต้องการออกจากโปรแกรมก็ลงมาทำในขั้นตอนต่อไป

ขั้นตอนที่ 8 เป็นการจบการทำงานของโปรแกรม

****

การทำงานของโปรแกรม CONTROL ROBOT BY KEYBOARD

การทำงานของโปรแกรม CONTROL ROBOT BY KEYBOARD นี้ สามารถที่จะควบคุมการทำงานของหุ่นยนต์ได้โดยการใช้ปุ่ม ARROW KEY จากแป้นพิมพ์ ซึ่งการเลือกที่จะควบคุมส่วนประกอบต่างๆ ของหุ่นยนต์นั้น ซึ่งจะแแบ่งส่วนประกอบของหุ่นยนต์ได้เป็น 8 ส่วน และสามารถที่จะควบคุมส่วนต่างๆ ของหุ่นยนต์ได้ ในการเลือกที่จะควบคุมหุ่นยนต์โปรแกรมแต่ละส่วนจะปรากฏอยู่บนโปรแกรมเมนูหลักแล้ว หรือถ้าไม่ได้อยู่ในโปรแกรมเมนูหลัก ให้เรียกโปรแกรมที่ชื่อว่า MENU ส่วนการทำงานของโปรแกรมจะอธิบายได้ดังนี้

โปรแกรมส่วนที่ 1 (HEAD OF ROBOT)

จะเป็นโปรแกรมควบคุมการทำงานของส่วนหัวของหุ่นยนต์ ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEY BOARD "

" USE LEFT ARROW KEY TO CONTROL HEAD ROBOT MOVE LEFT "

" USE RIGHT ARROW KEY TO CONTROL HEAD ROBOT MOVE RIGHT "

" ESC TO EXIT "

เราสามารถควบคุมได้โดยการกด LEFT ARROW KEY และ RIGHT ARROW KEY ซึ่งในขณะที่กดคีย์ลูกศรที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE HEAD OF ROBOT IS OPERATING "

ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

โปรแกรมส่วนที่ 2 (SHOULDER OF ROBOT)

เป็นโปรแกรมควบคุมการทำงานส่วนไหล่ (SHOULDER) ของหุ่นยนต์ ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEYBOARD "

" USE UP ARROW KEY TO CONTROL SHOULDER MOVE UP "

" USE DOWN ARROW KEY TO CONTROL SHOULSER MOVE DOWN "

" ESC TO EXIT "

เราสามารถที่จะควบคุมได้โดยการกด UP ARROW KEY และ DOWN ARROW KEY ซึ่งในขณะที่กดคียที่หน้าจอจะปรากฏดังนี้

" NOW ! THE SHOULDER OF ROBOT IS OPERATION "

ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

โปรแกรมส่วนที่ 3 (ARM OF ROBOT)

เป็นโปรแกรมควบคุมการทำงานในส่วนของการยืดและหดแขน ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEYBOARD "

" USE UP ARROW KEY TO CONTROL ARM MOVE PULL "

" USE DOWN ARROW KEY TO CONTROL ARM MOVE EXTEND "

" ESC TO EXIT "

เราสามารถที่จะควบคุมได้โดยการกด UP ARROW KEY และ DOWN ARROW KEY ในขณะที่กดคีย์ที่หน้าจอจะปรากฏดังนี้

" NOW ! THE ARM OF ROBOT IS OPERATING "

ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

โปรแกรมส่วนที่ 4 (WRIST PIVOT OF ROBOT)

เป็นโปรแกรมควบคุมการทำงานในส่วนหมุนข้อมือของหุ่นยนต์ ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEYBOARD "

" USE UP ARROW KEY TO CONTROL WRIST PIVOT MOVE CLOCKWISE "

" USE DOWN ARROW KEY TO CONTROL WRIST PIVOT MOVE ANTICLOCKWISE "

" ESC TO EXiT "

เราสามารถที่จะควบคุมการทำงานได้โดยกด UP ARROW KEY และ DOWN ARROW KEY ซึ่งในขณะกดคีย์ที่หน้าจอจะจะปรากฏข้อความดังนี้

' NOW ! THE WRIST PiVOT OF ROBOT IS OPERATING "

ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

โปรแกรมส่วนที่ 5 (WRIST ROTATE OF ROBOT)

เป็นโปรแกรมควบคุมการทำงานในส่วนหมุนมือหนีบของหุ่นยนต์ ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEYBOARD "

" USE RIGHT ARROW KEY TO CONTROL WRIST ROTATE MOVE RIGHT "

" USE LEFT ARROW KEY TO CONTROL WRIST ROTATE MOVE LEFT "

" ESC TO EXIT "

เราสามารถที่จะควบคุมการทำงานได้โดยกด RIGHT ARROW KEY และ LEFT ARROW KEY ซึ่งในขณะกดคีย์ที่หน้าจอจะ ปรากฏข้อความดังนี้

" NOW ! THE WRIST ROTATE OF ROBOT IS OPERATING " .

ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

โปรแกรมส่วนที่ 6 (GRIPPER OF ROBOT)

เป็นโปรแกรมที่ใช้ในการควบคุมส่วนมือหนีบของหุ่นยนต์ ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEYBOARD "

" USE RIGHT ARROW KEY TO CONTROL GRIPPER MOVE GRIP "

" USE LEFT ARROW KEY TO CONTROL GRIPPER MOVE EXPAND "

" ESC TO EXIT "

เราสามารถที่จะควบคุมการทำงานได้โดยกด RIGHT ARROW KEY และ LEFT ARROW KEY ซึ่งในขณะที่กดคีย์ ที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE GRIPPER OF ROBOT IS OPERATING "

ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

โปรแกรมส่วนที่ 7 (FRONT WHEEL TURN)

เป็นโปรแกรมที่ใช้ในการควบคุมการทำงานในส่วนหมุนล้อของหุ่นยนต์ ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEYBOARD "

" USE RIGHT ARROW KEY TO CONTROL FRONT WHEEL TURN RIGHT "

" USE LEFT ARROW KEY TO CONTROL FRONT WHEKK TURN LEFT "

" ESC TO EXIT "

เราสามารถที่จะควบคุมการทำงานได้โดยกด RIGHT ARROW KEY และ LEFT ARROW KEY ซึ่งในขณะกดคีย์จะปรากฏข้อความดังนี้

" NOW ! THE FRONT WHEEL TURN OF ROBOT IS OPERATING "

ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

โปรแกรมส่วนที่ 8 (FRONT WHEEL MOVE)

เป็นโปรแกรมที่ใช้ในการควบคุมการทำงานในส่วนล้อของหุ่นยนต์ ที่หน้าจอจะปรากฏข้อความดังนี้

" MENU CONTROL ROBOT WITH KEYBOARD "

" USE UP ARROW KEY TO CONTROL FRONT WHEEL MOVE FORWARD "

" USE DOWN KEY ARROW TO CONTROL FRONT WHEEL MOVE BACKWARD "

" ESC TO EXIT "

เราสามารถที่จะควบคุมการทำงานได้โดยกด UP ARROW KEY และ DOWN ARROW KEY ซึ่งในขณะที่กดคีย์จะปรมกฏข้อความที่หน้าจอดังนี้

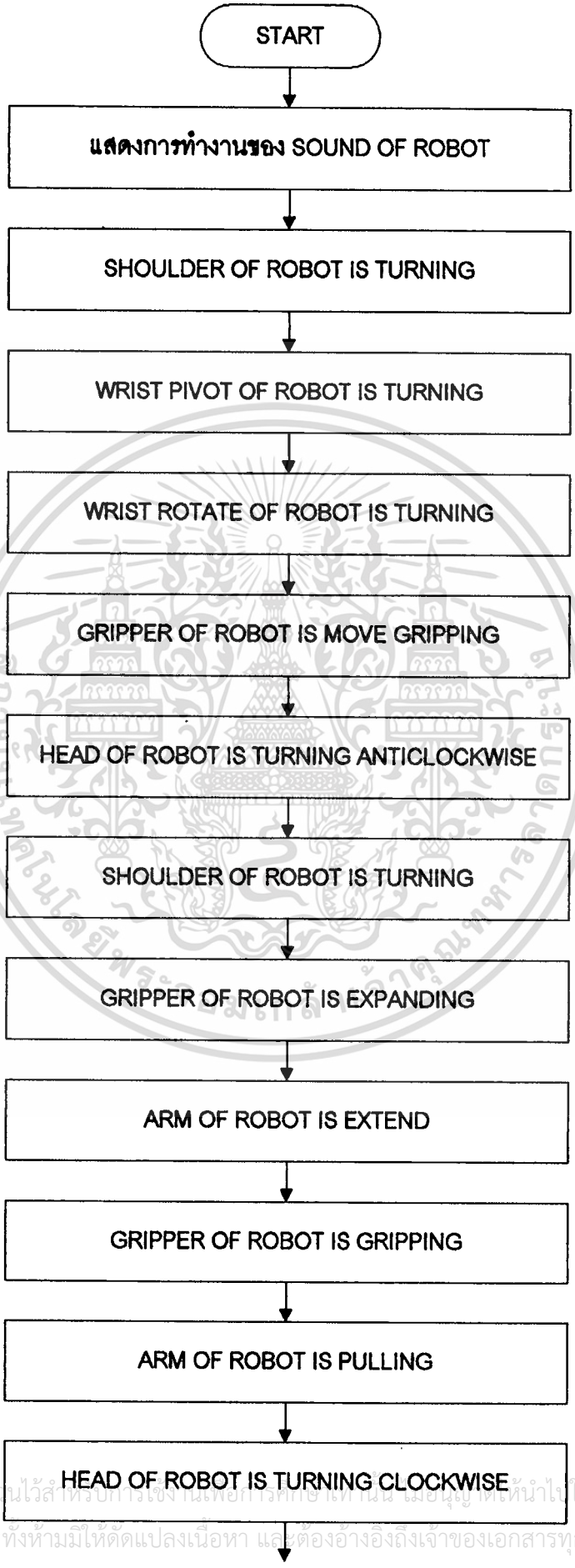
" NOW ! THE FRONT WHEEL MOVE OF ROBOT IS OPERATING "

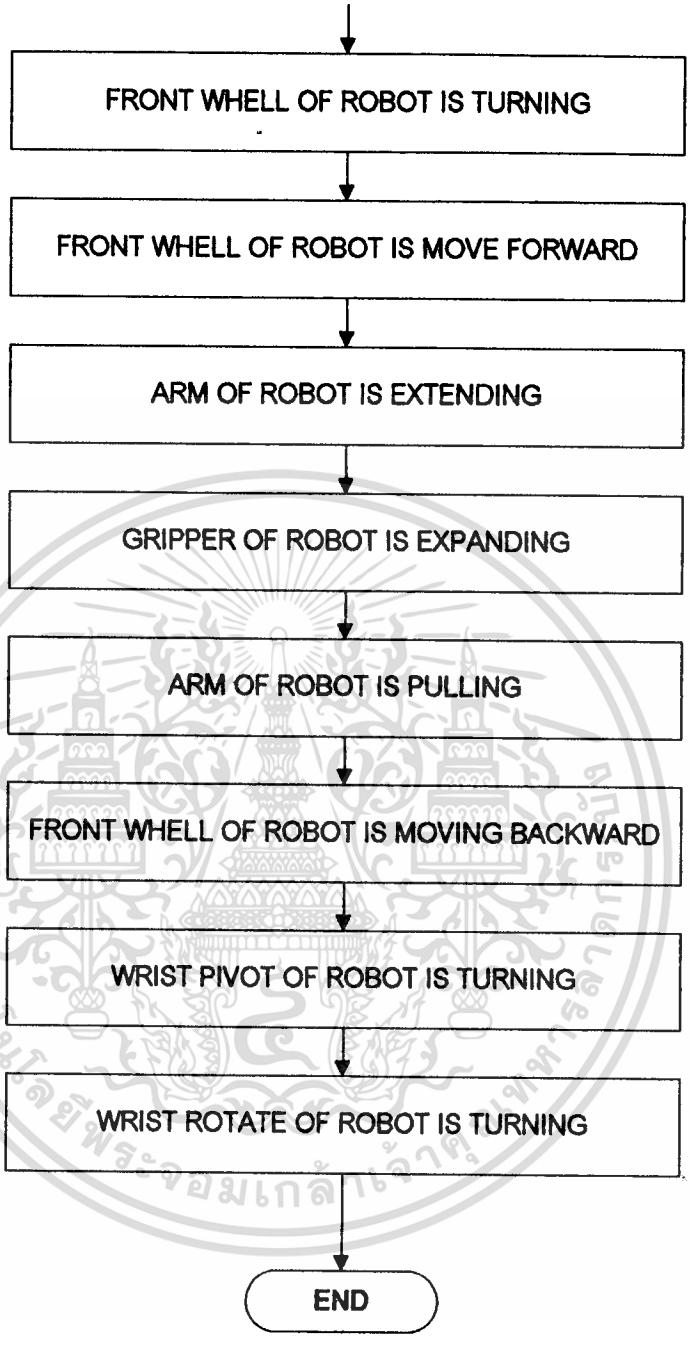
ถ้ากดคีย์ ESC จะออกไปยังโปรแกรมเมนูหลัก

****

FLOW CHART

ROBOT CONTROL BY COMPUTER (AUTO TEST2)





FLOW CHART

ROBOT CONTROL BY COMPUTER (AUTO TEST2)

FLOW CHART

ROBOT CONTROL BY COMPUTER (AUTO TEST2)

จาก FLOW CHART จะแสดงการทำงานของโปรแกรม AUTO TEST2 ซึ่งโปรแกรมนี้เป็นโปรแกรมทดสอบการทำงานของหุ่นยนต์แบบอัตโนมัติ โดยจะมีการสั่งให้หุ่นยนต์หยิบของจากจุดหนึ่งไปยังอีกจุดหนึ่ง ก่อนที่จะเรียกโปรแกรมนี้จะต้องทำให้หุ่นยนต์อยู่ในตำแหน่งจุดเริ่มต้นก่อน (ซึ่งตำแหน่งจุดเริ่มต้นนี้เราสามารถกำหนดเองได้ ในการกำหนดตำแหน่งจะสัมพันธ์กับโปรแกรม AUTO TEST2 ด้วย) ซึ่งการทำงานของ FLOW CHART ของโปรแกรม AUTO TEST2 จะอธิบายเป็นขั้นตอนได้ดังนี้

เมื่อ RUN โปรแกรม AUTO TEST2 จากโปรแกรม MENU

1. หุ่นยนต์จะพูดออกมาและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! IT IS THE SOUND OF ROBOT "

2. แขนของหุ่นยนต์จะเคลื่อนที่ขึ้นจำนวน 500 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE SHOULDER OF ROBOT IS TURNING "

3. ข้อมือของหุ่นยนต์จะเคลื่อนที่ขึ้นจำนวน 100 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE WRIST PIVOT OF ROBOT IS TURNING "

4. ข้อมือหนีบของหุ่นยนต์หมุนตามเข็มนาฬิกาจำนวน 100 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE WRIST ROTATE OF ROBOT IS TURNING "

5. มือหนีบของหุ่นยนต์จะหนีบเข้าจำนวน 100 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE GRIPPER OF ROBOT IS MOVE GRIPPING "

6. ส่วนหัวของหุ่นยนต์จะเคลื่อนที่ทวนเข็มนาฬิกาจำนวน 400 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE HEAD OF ROBOT IS TURNING ANTICLOCKWISE "

7. แขนของหุ่นยนต์จะเคลื่อนที่ลงมาตำแหน่งเดิมจำนวน 500 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE SHOULDER OF ROBOT IS TURNING "

8. มือหนีบของหุ่นยนต์จะขยายออกจำนวน 100 สเต็ป (เพื่อที่จะหยิบสิ่งของ) และที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE GRIPPER OF ROBOT IS EXPANDING "

9. ข่วงแขนของหุ่นยนต์จะยืดออกจำนวน 600 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE ARM OF ROBOT IS EXTENDING "

10. มือหนีบของหุ่นยนต์จะหนีบเข้ามาจำนวน 100 สเต็ป (ทำการหยิบสิ่งของ) และที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE GRiPPER OF ROBOT IS GRIPPING "

11. ข่วงแขนของหุ่นยนต์จะหดเข้าจำนวน 600 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE ARM OF ROBOT IS PULLING "

12. ส่วนหัวของหุ่นยนต์จะหมุนตามเข็มจำนวน 400 สเต็ปกลับมาตำแหน่งเดิมและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE HEAD OF ROBOT IS TURNING CLOCKWISE "

13. ล้อของหุ่นยนต์จะหมุนเลี้ยวซ้ายและขวาแล้วกลับมาตำแหน่งเดิมและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE FRONT WHEEL OF ROBOT IS TURNING "

14. หุ่นยนต์จะเคลื่อนที่ไปข้างหน้าจำนวน 100 สเต็ปและหยุดและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE FRONT WHEEL OF ROBOT IS MOVE FORWARD "

15. ข่วงแขนของหุ่นยนต์จะยืดออกจำนวน 600 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE ARM OF ROBOT IS EXTENDING "

16. มือหนีบของหุ่นยนต์จะขยายออกจำนวน 100 สเต็ป (เพื่อทำการปล่อยสิ่งของ) และที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE GRIPPER OF ROBOT IS EXPENDING "

17. ข่วงแขนของหุ่นยนต์จะยืดเข้าจำนวน 600 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE ARM OF ROBOT IS PULLING "

18. หุ่นยนต์จะเคลื่อนที่ไปข้างหลังจำนวน 100 สเต็ปแล้วหยุดและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE FRONT WHEEL OF ROBOT IS MOVING BACKWARD "

19. ข้อมือของหุ่นยนต์จะบิดลงจำนวน 100 สเต็ปและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE WRIST PIVOT OF ROBOT IS TURNING "

20. ข้อมือหนีบของหุ่นยนต์จะหมุนทวนเข็มจำนวน 100 สเต็ปกลับมาตำแหน่งเดิมและที่หน้าจอจะปรากฏข้อความดังนี้

" NOW ! THE WRIST ROTATE OF ROBOT IS TURNING "

เมื่อสิ้นสุดการทำงานของโปรแกรม AUTO TEST2 แล้วก็จะกลับไปยังโปรแกรมเมนูหลัก

****

สรุปการวิจัยและข้อเสนอแนะ

สรุป

1. การควบคุมหุ่นยนต์ทำได้เฉพาะ MANUAL เท่านั้น ส่วนการทำให้หุ่นยนต์สามารถจำได้ยังไม่สามารถทำได้

2. แบตเตอรี่หากไม่ได้รับการบำรุงรักษาที่ดีพอจะทำให้อายุการใช้งานสั้น หรืออาจเสียหายจนใช้งานไม่ได้

3. การใช้แหล่งจ่ายไฟของหุ่นยนต์ จะต้องเลือกใช้เพียง 1 อย่าง คือจะใช้ไฟเลี้ยงวงจรจากแบตเตอรี่หรือจาก POWER SUPPLY ถ้าใช้แบตเตอรี่ก็ต้องปิดแหล่งจ่ายไฟ POWER SUPPLY ถ้าใช้ POWER SUPPLY ก็ต้องถอดแบตเตอรี่ออกทั้งหมดเพื่อป้องกันการลัดวงจร

4. เมื่อเริ่มเปิดเครื่องคอมพิวเตอร์ จะมีสัญญาณข้อมูล เป็น "1" ออกที่เอาท์พุตของ 8255 CARD PORT ทำให้วงจร DRIVER ทำงานผิดพลาด ต้องใช้โปรแกรมเคลียร์ข้อมูลให้เอาท์พุททุกตัวเป็นศูนย์ก่อน

5. ส่วนหัวของหุ่นยนต์หมุนได้ไม่ครบรอบเนื่องจากมีสายไฟโผล่ออกมาที่ส่วนหัว

6. มอเตอร์มีหลายตัวและต้องการแรงดันไฟฟ้าที่เรียบและกินกระแสสูง

7. การออกแบบ SENSOR ไม่ได้ทำขึ้นเนื่องจากเป็นการควบคุมแบบ OPEN LOOP

แนวทางในการพัฒนาต่อไป

การพัฒนาคือการก้าวทันเทคโนโลยี ในอนาคตหุ่นยนต์ย่อมมีประสิทธิภาพสูงกว่าที่เป็นอยู่อย่างแน่นอน จึงขอเสนอแนวทางในการพัฒนาต่อไปดังนี้

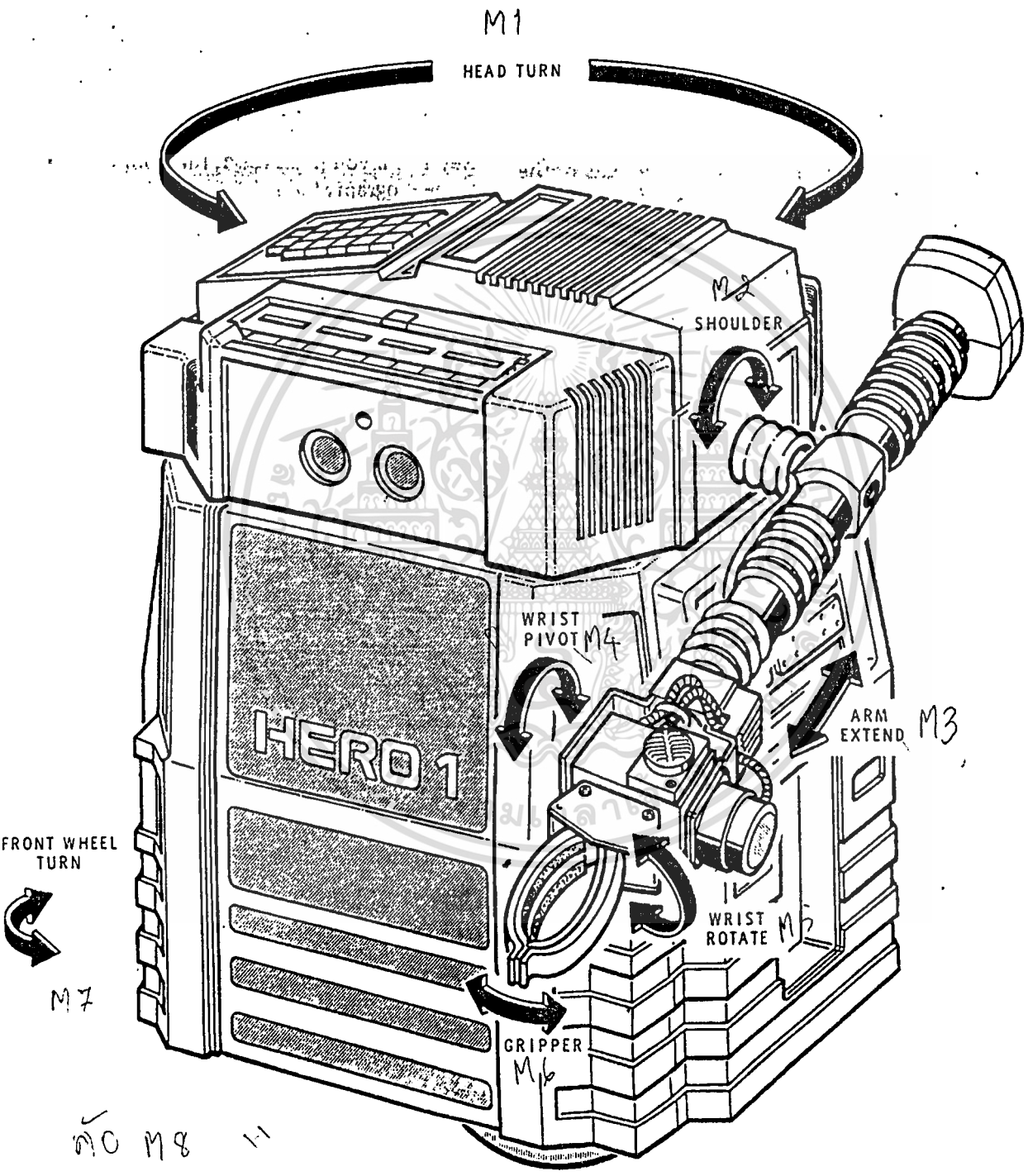
1. พัฒนาหุ่นยนต์ให้มีความจำสามารถทำงานที่เคยทำมาแล้วได้และสามารถทำได้หลายๆครั้ง

2. พัฒนาหุ่นยนต์ให้มีตาซึ่งเป็นพวกกล้องโทรทัศน์ทำให้ผู้ควบคุมหุ่นยนต์สามารถมองเห็นภาพ ณจุดที่หุ่นยนต์อยู่ได้

3. พัฒนาหุ่นยนต์ให้มีการควบคุมโดยใช้คลื่นวิทยุโดยไม่ต้องใช้สายต่อจากคคอมพิวเตอร์

4. พัฒนาให้มี SENSOR การทำงานให้หลบหลีกสิ่งกีดขวางได้

ภาคผนวก



```
PROGRAM CLEAR DATA OF ROBOT_CONTROL;
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 VAR      I:BYTE;
          X:INTEGER;
 TYPE     C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          CLR_DATA  :C_DATA=($00,$00);

   BEGIN
        PORT[PCONT_1]:=$80;  {PA1,PB1,PC1=OUTPUT PORT}
        PORT[PCONT_2]:=$80;  {PA2,PB2,PC2=OUTPUT PORT}
        PORT[PCONT_3]:=$80;  {PA3,PB3,PC3=OUTPUT PORT}
        BEGIN
        FOR X := 1 TO 10 DO
            FOR I:= 1 TO 2 DO
             PORT[PA1]:=CLR_DATA[I];
             PORT[PB1]:=CLR_DATA[I];
             PORT[PC1]:=CLR_DATA[I];
             PORT[PA2]:=CLR_DATA[I];
             PORT[PB2]:=CLR_DATA[I];
             PORT[PC2]:=CLR_DATA[I];
             PORT[PA3]:=CLR_DATA[I];
             PORT[PB3]:=CLR_DATA[I];
             PORT[PC3]:=CLR_DATA[I];
        END;
   END.
```

```
PROGRAM ROBOT_CONTROL_BY_COMPUTER;
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 VAR      I:BYTE;
          M1,M2,M3,M4,M5,M6,M7,M8,X : INTEGER;
 TYPE
          SP_DATA = ARRAY [1..4] OF BYTE;
          FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
          C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT  :SP_DATA=($10,$10,$10,$10);
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);

   BEGIN
        PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
        PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
        PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
        BEGIN
            FOR X := 1 TO 2 DO
              BEGIN
                FOR I:= 1 TO 2 DO
             PORT[PA1]:=CLR_DATA[I];
             PORT[PB1]:=CLR_DATA[I];
             PORT[PC1]:=CLR_DATA[I];
             PORT[PA2]:=CLR_DATA[I];
             PORT[PB2]:=CLR_DATA[I];
             PORT[PC2]:=CLR_DATA[I];
             PORT[PA3]:=CLR_DATA[I];
             PORT[PB3]:=CLR_DATA[I];
             PORT[PC3]:=CLR_DATA[I];
                END;
        END;

        CLRSCR;
        TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(5);
        GOTOXY(25,12);
         WRITELN('NOW ! IT IS THE SOUND OF ROBOT ');
```

```
    FOR M8 := 1 TO 200 DO
      BEGIN
        FOR I := 1 TO 4 DO
          BEGIN
          PORT[PC1]:=DATA_OUT[I];
          DELAY(5);
          END;
          IF M8 = 200 THEN
          PORT[PC1]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE HEAD OF ROBOT IS TURNING'):
    FOR M1:=1 TO 300 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PB1]:=DATA_OUT1[I];
          DELAY(5);
          END;
          IF M1 = 300 THEN
          PORT[PB1]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE  WRIST PIVOT OF ROBOT IS TURNING');
    FOR M4:=1 TO 100 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PC3]:=DATA_OUT1[I];
          DELAY(5);
          END;
          IF M4 = 100 THEN
          PORT[PC3]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE WRIST ROTATE  OF ROBOT IS TURNING');
    FOR M5:=1 TO 100 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PA3]:=DATA_OUT1[I];
          DELAY(5);
          END;
          IF M5 = 100 THEN
          PORT[PA3]:=CLR_DATA[I];
     END;


CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE GRIPPER OF ROBOT IS MOVING');
```

```
 FOR M6:=1 TO 100 DO
   BEGIN
     FOR I:= 1 TO 4 DO
       BEGIN
       PORT[PB3]:=DATA_OUT1[I];
       DELAY(5);
       END;
       IF M6 = 100 THEN
       PORT[PB3]:=CLR_DATA[I];
  END:


 CLRSCR;
 GOTOXY(25,12);
 WRITELN('NOW ! THE SHOULDER OF ROBOT IS TURNING');
 FOR M2 := 1 TO 300 DO
     BEGIN
       FOR I:= 1 TO 4 DO
         BEGIN
         PORT[PA2]:= DATA_OUT2[I];
         DELAY(5);
         END;
         IF M2 = 300 THEN
         PORT[PA2]:=CLR_DATA[I];
     END;


CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE ARM OF ROBOT IS EXTENDING');
FOR M3 := 1 TO 300 DO
    BEGIN
      FOR I:=1 TO 4 DO
        BEGIN
        PORT[PB2]:=DATA_OUT2[I];
        DELAY(4);
        END;
        IF M3 = 300 THEN
        PORT[PB2]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE GRIPPER OF ROBOT IS GRIPING');
FOR M6 := 1 TO 100 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PB3]:=DATA_OUT2[I];
        DELAY(5);
        END;
        IF M6 = 100 THEN
        PORT[PB3]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
```

```
WRITE('NOW ! THE ARM OF ROBOT IS PULLING ');
FOR M3:= 1 TO 300 DO
    BEGIN
      FOR I:=1 TO 4  DO
        BEGIN
        PORT[PB2]:=DATA_OUT1[I];
        DELAY(4);
        END;
        IF M3 = 300  THEN
        PORT[PB2]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE HEAD OF ROBOT IS TURNING ');
FOR M1:= 1 TO 300 DO
    BEGIN
      FOR I:=1 TO 4  DO
        BEGIN
        PORT[PB1]:=DATA_OUT2[I];
        DELAY(5);
        END;
        IF M1 = 300  THEN
        PORT[PB1]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE FRONT WHEEL OF ROBOT IS TURNING ');
FOR M7:= 1 TO  50 DO
    BEGIN
      FOR I:=1 TO 4 DO
        BEGIN
        PORT[PA1]:=DATA_OUT2[I];
        DELAY(5);
        END;
        IF M7  = 50 THEN
        PORT[PA1]:=CLR_DATA[I];
    END;
FOR M7:= 1 TO  50 DO
  .  BEGIN
      FOR I:=1 TO 4 DO
        BEGIN
        PORT[PA1]:=DATA_OUT1[I];
        DELAY(5);
        END;
        IF M7  = 50 THEN
        PORT[PA1]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE  FRONT WHEEL OF ROBOT IS  MOVE FORWARD');
FOR M8:= 1 TO 50 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
```

```
        PORT[PC1]:=DATA_OUT3[I];
        DELAY(10);
        END;
        IF M8= 50 THEN
        PORT[PC1]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE ARM OF ROBOT  IS EXTENDING');
FOR M3:= 1 TO 300 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PB2]:=DATA_OUT2[I];
        DELAY(4);
        END;
        IF M3 = 300 THEN
        PORT[PB2]:=CLR_DATA[I]:
    END;
    CLRSCR;


CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE GRIPPER OF ROBOT  IS EXPANDING');
FOR M6:= 1 TO 100 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PB3]:=DATA_OUT1[I];
        DELAY(5);
        END;
        IF M6 = 100 THEN
        PORT[PB3]:=CLR_DATA[I];
    END;


CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE ARM OF ROBOT  IS PULLING');
FOR M3:= 1 TO 300 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PB2]:=DATA_OUT1[I];
        DELAY(4);
        END;
        IF M3 = 300 THEN
        PORT[PB2]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE FRONT WHEEL OF ROBOT  IS MOVING BACKWARD');
FOR M8:= 1 TO 50 DO
    BEGIN
```

```
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PC1]:=DATA_OUT4[I];
        DELAY(5);
        END;
        IF M8 = 50 THEN
        PORT[PC1]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE SHOULDER OF ROBOT  IS TURNING');
FOR M2:= 1 TO 300 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PA2]:=DATA_OUT1[I];
        DELAY(5);
        END;
        IF M2 = 300 THEN
        PORT[PA2]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE WRIST PIVOT OF ROBOT  IS TURNNING');
FOR M4:= 1 TO 100 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PC3]:=DATA_OUT2[I];
        DELAY(5);
        END;
        IF M4 = 100 THEN
        PORT[PC3]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE WRIST ROTATE OF ROBOT  IS TURNING');
FOR M5:= 1 TO 100 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PA3]:=DATA_OUT2[I];
        DELAY(5);
        END;
        IF M5 = 100 THEN
        PORT[PA3]:=CLR_DATA[I];
    END;


CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE GRIPPER OF ROBOT IS GRIPPING');
FOR M6:= 1 TO 100 DO
    BEGIN
```

```
            FOR I:= 1 TO 4 DO
              BEGIN
              PORT[PB3]:=DATA_OUT2[I];
              DELAY(5);
              END;
              IF M6 = 100 THEN
              PORT[PB3]:=CLR_DATA[I];
          END;

      CLRSCR;
END.
```

```
PROGRAM CONTROL_STEPPING_MOTOR;
USES  CRT;
LABEL RETURN;
VAR   I:BYTE;
      CHOICE,X,Y,Z:INTEGER;
TYPE
      FOR_DATA = ARRAY [1..4] OF BYTE;
      REV_DATA = ARRAY [1..4] OF BYTE;
      MOVE_FORW_DATA= ARRAY [1..4] OF BYTE;
      MOVE_REV_DATA=ARRAY [1..4] OF BYTE;
      SPEAK_DATA = ARRAY [1..4] OF BYTE;
      C_DATA = ARRAY [1..2] OF BYTE;

CONST
      PA1 = $300;
      PB1 = $301;
      PC1 = $302;
      PA2 = $304;
      PB2 = $305;
      PC2 = $306;
      PA3 = $308;
      PB3 = $309;
      PC3 = $30A;
      PCONT_1 = $303;
      PCONT_2 = $307;
      PCONT_3 = $30B;
      DATA_OUT1 : FOR_DATA=($01,$02,$04,$08);
      DATA_OUT2 : REV_DATA=($08,$04,$02,$01);
      DATA_OUT3 : MOVE_FORW_DATA=($03,$03,$03,$03);
      DATA_OUT4 : MOVE_REV_DATA=($0A,$0A,$0A,$0A);
      SP_DATA   : SPEAK_DATA=($10,$10,$10,$10);
      DATA_CLR :  C_DATA=($00,$00);
BEGIN
  PORT[PCONT_1] := $80; {PA1,PB1,PC1 = OUTPUT PORT}
  PORT[PCONT_2] := $80; {PA2,PB2,PC2 = OUTPUT PORT}
  PORT[PCONT_3] := $80; {PA3,PB3,PC3 = OUTPUT PORT}
BEGIN
  CLRSCR;
  REPEAT
RETURN:  BEGIN
         CLRSCR;
         GOTOXY(15,5);
         TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
         WRITELN('THE NUMBER TO CONTROL OPERATION OF ROBOT');
         TEXTCOLOR(LIGHTGREEN);  TEXTBACKGROUND(3);
         GOTOXY(15,6);
         WRITELN('0 : CLEAR OUTPUT OF DRIVER CIRCUIT   (CLR)');
         GOTOXY(15,7);
         WRITELN('1 : CONTROL HEAD TURN OF ROBOT        (M1)');
         GOTOXY(15,8);
         WRITELN('2 : CONTROL SHOULER OF ROBOT          (M2)');
         GOTOXY(15,9);
         WRITELN('3 : CONTROL ARM EXTEND OF ROBOT       (M3)');
         GOTOXY(15,10);
         WRITELN('4 : CONTROL WRIST PIVOT OF ROBOT      (M4)');
         GOTOXY(15,11);
```

```
        WRITELN('5 : CONTROL WRIST ROTATE OF ROBOT     (M5)');
        GOTOXY(15,12);
        WRITELN('6 : CONTROL GRIPPER OF ROBOT          (M6)');
        GOTOXY(15,13);
        WRITELN('7 : CONTROL FRONT WHEEL TURN OF ROBOT (M7)');
        GOTOXY(15,14);
        WRITELN('8 : CONTROL FRONT WHEEL MOVE OF ROBOT (M8)');
        GOTOXY(15,15);
        WRITELN('9 : CONTROL SPEAK OF ROBOT            (SP)');
        GOTOXY(15,16);
        WRITELN('10 :      EXIT PROGRAM');
        END;
        TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
        GOTOXY(13,17);
        WRITELN('SELECT THE NUMBER TO CONTROL OPERATION OF ROBOT');
        TEXTBACKGROUND(3);
        GOTOXY(20,18);
        READLN(CHOICE);
        IF CHOICE > 10 THEN
        GOTO RETURN;
        IF CHOICE < 0 THEN
        GOTO RETURN;

 CASE CHOICE OF

0:BEGIN
  CLRSCR;
  FOR X := 1 TO 2 DO
   BEGIN
    FOR I :=1 TO 2 DO
    BEGIN
    PORT[PA1] := DATA_CLR[I];
    PORT[PB1] := DATA_CLR[I];
    PORT[PC1] := DATA_CLR[I];
    PORT[PA2] := DATA_CLR[I];
    PORT[PB2] := DATA_CLR[I];
    PORT[PC2] := DATA_CLR[I];
    PORT[PA3] := DATA_CLR[I];
    PORT[PB3] := DATA_CLR[I];
    PORT[PC3] := DATA_CLR[I];
    END;
  END;
 END;

1:BEGIN
  REPEAT
  CLRSCR;
  GOTOXY(17,5);
  TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
  WRITELN('THE DIRECTION TO CONTROL MOTOR (M1)');
  GOTOXY(20,6);
  TEXTBACKGROUND(3);
  WRITELN(' 0 : HEAD TURN RIGHT');
  GOTOXY(20,7);
  WRITELN(' 1 : HEAD TURN LEFT');
  GOTOXY(20,8);
  WRITELN(' 2 : EXIT');
```

```
GOTOXY(17,9);
TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
WRITE('SELECT NUMER TO CONTROL HEAD OF ROBOT');
GOTOXY(25,10);
TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
READLN(Y);
BEGIN
IF Y = 2 THEN
GOTO RETURN;
IF Y>1  THEN
GOTO RETURN;
IF Y<0 THEN
GOTO RETURN;
END;
GOTOXY(20,12);
WRITELN('ENTER NUMBER OF STEPS TO MOVE');
GOTOXY(25,13);
READLN(Z);
IF Z>1000 THEN
GOTO RETURN;
CLRSCR;
 BEGIN
 GOTOXY(25,12);
 TEXTCOLOR(LIGHTGREEN);
 WRITELN('NOW ! THE HEAD TURN IS OPERATING');
 PORT[PCONT_1]:= $80; {PA1,PB1,PC1=OUTPUT PORT}
  FOR X:= 1 TO Z DO
   BEGIN
     FOR I:= 1 TO 4 DO
      BEGIN
        IF Y = 0 THEN
        BEGIN
        GOTOXY(25,13);
        WRITELN('THE HEAD TURN RIGHT');
        PORT[PB1]:= DATA_OUT1[I];
        END;
        DELAY(3);
        IF  Y = 1 THEN
        BEGIN
        GOTOXY(25,14);
        WRITELN('THE HEAD TURN LEFT');
        PORT[PB1]:= DATA_OUT2[I];
        END;
        DELAY(3);
      END;
     END;
    END;
   IF X = Z
       THEN
        PORT[PB1] :=DATA_CLR[I];
   CLRSCR;
  UNTIL Y = 2;
END;

2:BEGIN
 REPEAT
 CLRSCR;
```

```
GOTOXY(17,5);
TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
WRITELN('THE DIRECTION TO CONTROL MOTOR (M2)');
GOTOXY(20,6);
TEXTBACKGROUND(3);
WRITELN(' 0 : SHOULDER TURN CLOCKWISE');
GOTOXY(20,7);
WRITELN(' 1 : SHOULER TURN ANTICLOCKWISE');
GOTOXY(20,8);
WRITELN(' 2 : EXIT');
GOTOXY(15,9);
TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
WRITE('SELECT NUMER TO CONTROL SHOULDER OF ROBOT');
GOTOXY(25,10);
TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
READLN(Y);
IF Y = 2 THEN
GOTO RETURN;
IF Y>2  THEN
GOTO RETURN;
IF Y<0 THEN
GOTO RETURN;
GOTOXY(20,12);
WRITELN('ENTER NUMBER OF STEPS TO MOVE');
GOTOXY(25,13);
READLN(Z);
IF Z>1000 THEN
GOTO RETURN;
CLRSCR;
 BEGIN
 GOTOXY(25,12);
 TEXTCOLOR(LIGHTGREEN);
 WRITELN('NOW ! THE SHOULDER OF ROBOT IS OPERATING');
 PORT[PCONT_2]:= $80; {PA2,PB2,PC2=OUTPUT PORT}
  FOR X:= 1 TO Z DO
   BEGIN
     FOR I:= 1 TO 4 DO
      BEGIN
        IF Y = 0 THEN
        BEGIN
        GOTOXY(25,13);
        WRITELN('THE SHOULDER TURN CLOCKWISE');
        PORT[PA2]:= DATA_OUT1[I];
        END;
        DELAY(3);
        IF  Y = 1 THEN
        BEGIN
        GOTOXY(25,14);
        WRITELN('THE SHOULDER TURN ANTICLOCKWISE');
        PORT[PA2]:= DATA_OUT2[I];
        END;
        DELAY(3);
      END;
     END;
   END;
  IF X = Z
      THEN
      PORT[PA2]:=DATA_CLR[I];
```

```
      CLRSCR;
     UNTIL Y = 2;
   END;

3:BEGIN
   REPEAT
   CLRSCR;
   GOTOXY(17,5);
   TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
   WRITELN('THE DIRECTION TO CONTROL MOTOR (M3)');
   GOTOXY(20,6);
   TEXTBACKGROUND(3);
   WRITELN(' 0 : ARM OF ROBOT IS PULL');
   GOTOXY(20,7);
   WRITELN(' 1 : ARM OF ROBOT IS EXTEND');
   GOTOXY(20,8);
   WRITELN(' 2 : EXIT');
   GOTOXY(17,9);
   TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
   WRITE('SELECT NUMER TO CONTROL ARM OF ROBOT');
   GOTOXY(25,10);
   TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
   READLN(Y);
   IF Y = 2 THEN
   GOTO RETURN;
   IF Y>2  THEN
   GOTO RETURN;
   IF Y<0 THEN
   GOTO RETURN;
   GOTOXY(20,12);
   WRITELN('ENTER NUMBER OF STEPS TO MOVE');
   GOTOXY(25,13);
   READLN(Z);
   IF Z>1000 THEN
   GOTO RETURN;
   CLRSCR;
    BEGIN
    GOTOXY(25,12);
    TEXTCOLOR(LIGHTGREEN);
    WRITELN('NOW ! THE ARM OF ROBOT IS OPERATING');
    PORT[PCONT_2]:= $80; {PA2,PB2,PC2=OUTPUT PORT}
     FOR X:= 1 TO Z DO
      BEGIN
        FOR I:= 1 TO 4 DO
         BEGIN
           IF Y = 0 THEN
           BEGIN
           GOTOXY(25,13);
           WRITELN('THE ARM IS PULLING ');
           PORT[PB2]:= DATA_OUT1[I];
           END;
           DELAY(3);
           IF  Y = 1 THEN
           BEGIN
           GOTOXY(25,14);
           WRITELN('THE ARM IS EXTENDING');
           PORT[PB2]:= DATA_OUT2[I];
```

```
          END;
          DELAY(3);
        END;
       END;
      END;
     IF X = Z
         THEN
          PORT[PB2]:=DATA_CLR[I];
    CLRSCR;
   UNTIL Y = 2;
  END;

4:BEGIN
  REPEAT
  CLRSCR;
  GOTOXY(17,5);
  TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
  WRITELN('THE DIRECTION TO CONTROL MOTOR (M4)');
  GOTOXY(20,6);
  TEXTBACKGROUND(3);
  WRITELN(' 0 : WRIST PIVOT TURN CLOCKWISE');
  GOTOXY(20,7);
  WRITELN(' 1 : WRIST PIVOT TURN ANTICLOCKWISE');
  GOTOXY(20,8);
  WRITELN(' 2 : EXIT');
  GOTOXY(15,9);
  TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
  WRITE('SELECT NUMER TO CONTROL WRIST PIVOT OF ROBOT');
  GOTOXY(25,10);
  TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
  READLN(Y);
  IF Y = 2 THEN
  GOTO RETURN;
  IF Y>2  THEN
  GOTO RETURN;
  IF Y<0 THEN
  GOTO RETURN;
  GOTOXY(20,12);
  WRITELN('ENTER NUMBER OF STEPS TO MOVE');
  GOTOXY(25,13);
  READLN(Z);
  IF Z>300 THEN
  GOTO RETURN;
  CLRSCR;
   BEGIN
   GOTOXY(25,12);
   TEXTCOLOR(LIGHTGREEN);
   WRITELN('NOW ! THE WRIST PIVOT IS OPERATING');
   PORT[PCONT_3]:= $80; {PA3,PB3,PC3=OUTPUT PORT}
    FOR X:= 1 TO Z DO
     BEGIN
       FOR I:= 1 TO 4 DO
        BEGIN
          IF Y = 0 THEN
          BEGIN
          GOTOXY(25,13);
```

```
          WRITELN('THE WRIST PIVOT TURN CLOCKWISE');
          PORT[PC3]:= DATA_OUT2[I];
          END;
          DELAY(3);
          IF  Y = 1 THEN
          BEGIN
          GOTOXY(25,14);
          WRITELN('THE WRIST PIVOT TURN ANTICLOCK');
          PORT[PC3]:= DATA_OUT1[I];
          END;
          DELAY(3);
        END;
       END;
      END;
     IF X = Z
         THEN
          PORT[PC3]:=DATA_CLR[I];
    CLRSCR;
   UNTIL Y = 2;
  END;

5:BEGIN
  REPEAT
  CLRSCR;
  GOTOXY(17,5);
  TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
  WRITELN('THE DIRECTION TO CONTROL MOTOR (M5)');
  GOTOXY(20,6);
  TEXTBACKGROUND(3);
  WRITELN(' 0 : WRIST ROTATE TURN CLOCKWISE');
  GOTOXY(20,7);
  WRITELN(' 1 : WRIST ROTATE TURN ANTICLOCKWISE');
  GOTOXY(20,8);
  WRITELN(' 2 : EXIT');
  GOTOXY(15,9);
  TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
  WRITE('SELECT NUMER TO CONTROL WRIST ROTATE OF ROBOT');
  GOTOXY(25,10);
  TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
  READLN(Y);
  IF Y = 2 THEN
  GOTO RETURN;
  IF Y>2  THEN
  GOTO RETURN;
  IF Y<0 THEN
  GOTO RETURN;
  GOTOXY(20,12);
  WRITELN('ENTER NUMBER OF STEPS TO MOVE');
  GOTOXY(25,13);
  READLN(Z);
  IF Z>300 THEN
  GOTO RETURN;
  CLRSCR;
   BEGIN
   GOTOXY(25,12);
   TEXTCOLOR(LIGHTGREEN);
   WRITELN('NOW ! THE WRIST ROTATE IS OPERATING');
```

```
   PORT[PCONT_3]:= $80; {PA3,PB3,PC3=OUTPUT PORT}
    FOR X:= 1 TO Z DO
     BEGIN
       FOR I:= 1 TO 4 DO
        BEGIN
          IF Y = 0 THEN
          BEGIN
          GOTOXY(25,13);
          WRITELN('THE WRIST ROTATE TURN CLOCKWISE');
          PORT[PA3]:= DATA_OUT2[I];
          END;
          DELAY(3);
          IF  Y = 1 THEN
          BEGIN
          GOTOXY(25,14);
          WRITELN('THE WRIST ROTATE TURN ANTICLOCKWISE');
          PORT[PA3]:= DATA_OUT1[I];
          END;
          DELAY(3);
        END;
       END;
      END;
     IF X = Z
         THEN
          PORT[PA3]:=DATA_CLR[I];
    CLRSCR;
   UNTIL Y = 2;
  END;


6:BEGIN
  REPEAT
  CLRSCR;
  GOTOXY(17,5);
  TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
  WRITELN('THE DIRECTION TO CONTROL MOTOR (M6)');
  GOTOXY(20,6);
  TEXTBACKGROUND(3);
  WRITELN(' 0 : GRIPPER MOVE EXPAND');
  GOTOXY(20,7);
  WRITELN(' 1 : GRIPPER MOVE GRIP');
  GOTOXY(20,8);
  WRITELN(' 2 : EXIT');
  GOTOXY(15,9);
  TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
  WRITE('SELECT NUMER TO CONTROL GRIPPER OF ROBOT');
  GOTOXY(25,10);
  TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
  READLN(Y);
  IF Y = 2 THEN
  GOTO RETURN;
  IF Y>2  THEN
  GOTO RETURN;
  IF Y<0 THEN
  GOTO RETURN;
  GOTOXY(20,12);
  WRITELN('ENTER NUMBER OF STEPS TO MOVE');
```

```
  GOTOXY(25,13);
  READLN(Z);
  IF Z>150 THEN
  GOTO RETURN;
  CLRSCR;
   BEGIN
   GOTOXY(25,12);
   TEXTCOLOR(LIGHTGREEN);
   WRITELN('NOW ! THE GRIPPER IS OPERATING');
   PORT[PCONT_3]:= $80; {PA3,PB3,PC3=OUTPUT PORT}
    FOR X:= 1 TO Z DO
     BEGIN
       FOR I:= 1 TO 4 DO
        BEGIN
          IF Y = 0 THEN
          BEGIN
          GOTOXY(25,13);
          WRITELN('THE GRIPPER MOVE EXPAND');
          PORT[PB3]:= DATA_OUT1[I];
          END;
          DELAY(3);
          IF  Y = 1 THEN
          BEGIN
          GOTOXY(25,14);
          WRITELN('THE GRIPPER MOVE GRIP');
          PORT[PB3]:= DATA_OUT2[I];
          END;
          DELAY(3);
        END;
       END;
      END;
     IF X = Z
         THEN
          PORT[PB3]:=DATA_CLR[I];
     CLRSCR;
    UNTIL Y = 2;
  END;

7:BEGIN
  REPEAT
  CLRSCR;
  GOTOXY(17,5);
  TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
  WRITELN('THE DIRECTION TO CONTROL MOTOR (M7)');
  GOTOXY(20,6);
  TEXTBACKGROUND(3);
  WRITELN(' 0 : FRONT WHEEL TURN RIGHT');
  GOTOXY(20,7);
  WRITELN(' 1 : FRONT WHEEL TURN LEFT');
  GOTOXY(20,8);
  WRITELN(' 2 : EXIT');
  GOTOXY(12,9);
  TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
  WRITE('SELECT NUMER TO CONTROL FRONT WHEEL TURN OF ROBOT');
  GOTOXY(25,10);
  TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
  READLN(Y);
```

```
  IF Y = 2 THEN
  GOTO RETURN;
  IF Y>2  THEN
  GOTO RETURN;
  IF Y<0 THEN
  GOTO RETURN;
  GOTOXY(20,12);
  WRITELN('ENTER NUMBER OF STEPS TO MOVE');
  GOTOXY(25,13);
  READLN(Z);
  IF Z>100 THEN
  GOTO RETURN;
  CLRSCR;
   BEGIN
   GOTOXY(25,12);
   TEXTCOLOR(LIGHTGREEN);
   WRITELN('NOW ! THE FRONT WHEEL TURN IS OPERATING');
   PORT[PCONT_1]:= $80; {PA1,PB1,PC1=OUTPUT PORT}
    FOR X:= 1 TO Z DO
     BEGIN
       FOR I:= 1 TO 4 DO
        BEGIN
          IF Y = 0 THEN
          BEGIN
          GOTOXY(25,13);
          WRITELN('THE FRONT WHEEL TURN RIGHT');
          PORT[PA1]:= DATA_OUT1[I];
          END;
          DELAY(3);
          IF  Y = 1 THEN
          BEGIN
          GOTOXY(25,14);
          WRITELN('THE FRONT WHEEL TURN LEFT');
          PORT[PA1]:= DATA_OUT2[I];
          END;
          DELAY(3);
        END;
       END;
      END;
     IF X = Z
         THEN
          PORT[PA1]:=DATA_CLR[I];
     CLRSCR;
    UNTIL Y = 2;
  END;

8:BEGIN
  REPEAT
  CLRSCR;
  GOTOXY(17,5);
  TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
  WRITELN('THE DIRECTION TO CONTROL MOTOR (M8)');
  GOTOXY(20,6);
  TEXTBACKGROUND(3);
  WRITELN(' 0 : FRONT WHEEL MOVE FORWARD');
  GOTOXY(20,7);
  WRITELN(' 1 : FRONT WHEEL MOVE BACKWARD');
```

```
  GOTOXY(20,8);
  WRITELN(' 2 : EXIT');
  GOTOXY(12,9);
  TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
  WRITE('SELECT NUMER TO CONTROL FRONT WHEEL MOVE OF ROBOT');
  GOTOXY(25,10);
  TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
  READLN(Y);
  IF Y = 2 THEN
  GOTO RETURN;
  IF Y>2  THEN
  GOTO RETURN;
  IF Y<0 THEN
  GOTO RETURN;
  GOTOXY(20,12);
  WRITELN('ENTER NUMBER OF TIME (mS) TO CONTROL MOVE');
  GOTOXY(25,13);
  READLN(Z);
  IF Z>300 THEN
  GOTO RETURN;
  CLRSCR;
   BEGIN
   GOTOXY(25,12);
   TEXTCOLOR(LIGHTGREEN);
   WRITELN('NOW ! THE FRONT WHEEL IS OPERATING');
   PORT[PCONT_1]:= $80; {PA1,PB1,PC1=OUTPUT PORT}
    FOR X:= 1 TO Z DO
     BEGIN
       FOR I:= 1 TO 4 DO
        BEGIN
          IF Y = 0 THEN
          BEGIN
          GOTOXY(25,13);
          WRITELN('THE FRONT WHEEL MOVE FORWARD');
          PORT[PC1]:= DATA_OUT3[I];
          END;
          DELAY(5);
          IF  Y = 1 THEN
          BEGIN
          GOTOXY(25,14);
          WRITELN('THE FRONT WHEEEL MOVE BACKWARD');
          PORT[PC1]:= DATA_OUT4[I];
          END;
          DELAY(5);
        END;
       END;
      END;
     IF X = Z
         THEN
          PORT[PC1]:=DATA_CLR[I];
     CLRSCR;
    UNTIL Y = 2;
  END;

9:BEGIN
  REPEAT
  CLRSCR;
```

```
    GOTOXY(17,5);
    TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
    WRITELN('THE CHOICE TO CONTROL SPEAK OF ROBOT (SP)');
    GOTOXY(20,6);
    TEXTBACKGROUND(3);
    WRITELN(' 1 : ROBOT SPEAK');
    GOTOXY(20,7);
    WRITELN(' 2 : EXIT');
    GOTOXY(17,9);
    TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(6);
    WRITE('SELECT NUMER TO CONTROL SPEAK OF ROBOT');
    GOTOXY(25,10);
    TEXTCOLOR(WHITE); TEXTBACKGROUND(3);
    READLN(Y);
    IF Y = 2 THEN
    GOTO RETURN;
    IF Y>2  THEN
    GOTO RETURN;
    IF Y<0 THEN
    GOTO RETURN;
    GOTOXY(20,12);
    WRITELN('ENTER NUMBER OF TIME TO SPEAK (mS)');
    GOTOXY(25,13);
    READLN(Z);
    IF Z>1000 THEN
    GOTO RETURN;
    CLRSCR;
    BEGIN
     GOTOXY(25,12);
     TEXTCOLOR(LIGHTGREEN);
     WRITELN('NOW ! THE ROBOT SPEAK OPERATING ');
     PORT[PCONT_1]:= $80; {PA1,PB1,PC1=OUTPUT PORT}
      FOR X:= 1 TO Z DO
       BEGIN
         FOR I:= 1 TO 4 DO
          BEGIN
            IF Y = 1 THEN
            BEGIN
            GOTOXY(25,13);
            WRITELN('IT IS SPEAK OF ROBOT');
            PORT[PC1]:= SP_DATA[I];
            END;
            DELAY(5);
          END;
        END;
       END;
      IF X = Z
          THEN
           PORT[PC1]:=DATA_CLR[I];
      CLRSCR;
     UNTIL Y = 2;
    END;

10:BEGIN
    CLRSCR;
    GOTOXY(25,5);
    TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(5);
```

```
    WRITELN('EXIT PROGRAM');
    EXIT;
    END;
  END;
  UNTIL CHOICE = 10;
END;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEYBOARD (HEAD OF ROBOT);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(25,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(15,7);
     TEXTBACKGROUND(5);
     WRITELN('USE LEFT_ARROW KEY TO CONTROL HEAD ROBOT MOVE LEFT');
     GOTOXY(15,9);
     TEXTBACKGROUND(7);
     WRITELN('USE RIGHT_ARROW KEY TO CONTROL HEAD ROBOT MOVE RIGHT')
     GOTOXY(35,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
         #72 : U := U+1;
         #75 : L := L+1;
```

```
      #77 : R := R+1;
      #80 : D := D+1;
      #27 : DONE := TRUE;
     END;

  BEGIN
    PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
    PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
    PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
    BEGIN
        FOR X :=1 TO 2 DO
          BEGIN
            FOR I:= 1 TO 2 DO
         PORT[PA1]:=CLR_DATA[I];
         PORT[PB1]:=CLR_DATA[I];
         PORT[PC1]:=CLR_DATA[I];
         PORT[PA2]:=CLR_DATA[I];
         PORT[PB2]:=CLR_DATA[I];
         PORT[PC2]:=CLR_DATA[I];
         PORT[PA3]:=CLR_DATA[I];
         PORT[PB3]:=CLR_DATA[I];
         PORT[PC3]:=CLR_DATA[I];
            END;
   END;

BEGIN
     CLRSCR;
     IF CH = #72 THEN
     GOTO LOOP;
     IF CH = #80 THEN
     GOTO LOOP;
     GOTOXY(20,12);
     TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
     WRITELN('NOW ! THE HEAD OF ROBOT IS OPERATING');
   BEGIN
     IF CH = #75  THEN
      BEGIN
      FOR X := 1 TO L DO
          BEGIN
           FOR I:= 1 TO 4 DO
             BEGIN
             PORT[PB1]:=DATA_OUT2[I];
             DELAY(6);
             END;
             IF X = L  THEN
             PORT[PB1]:=CLR_DATA[I];
           END;
      END;
      IF CH = #77   THEN
      BEGIN
      FOR X := 1 TO R DO
        BEGIN
          FOR I:= 1 TO 4 DO
            BEGIN
            PORT[PB1]:=DATA_OUT1[I];
            DELAY(6);
            END;
```

```
                IF X = R THEN
                PORT[PB1]:=CLR_DATA[I];
            END;
          END;
        END;
      END;
    END;


    UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEY BOARD (SHOULDER OF ROBOT);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(25,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(18,7);
     TEXTBACKGROUND(5);
     WRITELN('USE UP_ARROW KEY TO CONTROL SHOULDER  MOVE UP');
     GOTOXY(17,9);
     TEXTBACKGROUND(7);
     WRITELN('USE DOWN_ARROW KEY TO CONTROL SHOULDER MOVE DOWN');
     GOTOXY(35,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
          #72 : U := U+1;
          #75 : L := L+1;
```

```
    #77 : R := R+1;
    #80 : D := D+1;
    #27 : DONE := TRUE;
   END;

BEGIN
  PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
  PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
  PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
  BEGIN
      FOR X :=1 TO 2 DO
        BEGIN
          FOR I:= 1 TO 2 DO
       PORT[PA1]:=CLR_DATA[I];
       PORT[PB1]:=CLR_DATA[I];
       PORT[PC1]:=CLR_DATA[I];
       PORT[PA2]:=CLR_DATA[I];
       PORT[PB2]:=CLR_DATA[I];
       PORT[PC2]:=CLR_DATA[I];
       PORT[PA3]:=CLR_DATA[I];
       PORT[PB3]:=CLR_DATA[I];
       PORT[PC3]:=CLR_DATA[I];
          END;
  END;

  BEGIN
   CLRSCR;
   IF CH = #75 THEN
   GOTO LOOP;
   IF CH = #77 THEN
   GOTO LOOP;
   GOTOXY(20,12);
   TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
   WRITELN('NOW ! THE SHOULDER OF ROBOT IS OPERATING');
 BEGIN
   IF CH = #72  THEN
    BEGIN
    FOR X := 1 TO U DO
        BEGIN
         FOR I:= 1 TO 4 DO
           BEGIN
           PORT[PA2]:=DATA_OUT1[I];
           DELAY(9);
           END;
           IF X = U  THEN
           PORT[PA2]:=CLR_DATA[I];
         END;
     END;
     IF CH = #80   THEN
     BEGIN
     FOR X := 1 TO D DO
       BEGIN
         FOR I:= 1 TO 4 DO
           BEGIN
           PORT[PA2]:=DATA_OUT2[I];
           DELAY(9);
           END;
```

```
                IF X = D THEN
                PORT[PA2]:=CLR_DATA[I];
            END;
        END;
      END;
    END;
  END;


  UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEYBOARD (ARM OF ROBOT);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP,L1;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(25,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(20,7);
     TEXTBACKGROUND(5);
     WRITELN('USE UP_ARROW KEY TO CONTROL ARM  MOVE PULL');
     GOTOXY(19,9);
     TEXTBACKGROUND(7);
     WRITELN('USE DOWN_ARROW KEY TO CONTROL ARM MOVE EXTEND');
     GOTOXY(34,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
         #72 : U := U+1;
         #75 : L := L+1;
```

```
      #77 : R := R+1;
      #80 : D := D+1;
      #27 : DONE := TRUE;
     END;
BEGIN
   PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
   PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
   PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
   BEGIN
       FOR X :=1 TO 2 DO
         BEGIN
           FOR I:= 1 TO 2 DO
        PORT[PA1]:=CLR_DATA[I];
        PORT[PB1]:=CLR_DATA[I];
        PORT[PC1]:=CLR_DATA[I];
        PORT[PA2]:=CLR_DATA[I];
        PORT[PB2]:=CLR_DATA[I];
        PORT[PC2]:=CLR_DATA[I];
        PORT[PA3]:=CLR_DATA[I];
        PORT[PB3]:=CLR_DATA[I];
        PORT[PC3]:=CLR_DATA[I];
           END;
  END;
BEGIN
    CLRSCR;
    IF CH = #75 THEN
    GOTO LOOP;
    IF CH = #77 THEN
    GOTO LOOP;
    GOTOXY(20,12);
    TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
    WRITELN('NOW ! THE ARM OF ROBOT IS OPERATING');
 BEGIN
    IF CH = #72  THEN
     BEGIN
     FOR X := 1 TO U DO
         BEGIN
          FOR I:= 1 TO 4 DO
            BEGIN
            PORT[PB2]:=DATA_OUT1[I];
            DELAY(5);
            END;
            IF X = U  THEN
            PORT[PB2]:=CLR_DATA[I];
         END;
      END;
      IF CH = #80   THEN
      BEGIN
      FOR X := 1 TO D DO
        BEGIN
          FOR I:= 1 TO 4 DO
            BEGIN
            PORT[PB2]:=DATA_OUT2[I];
            DELAY(5);
            END;
            IF X = D THEN
            PORT[PB2]:=CLR_DATA[I];
```

```
            END;
          END;
        END;
      END;
    END;
    UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEYBOARD (WRIST PIVOT OF ROBOT);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP,L1;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(24,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(16,7);
     TEXTBACKGROUND(5);
     WRITELN('USE UP_ARROW KEY TO CONTROL WRIST PIVOT MOVE CLOCKWISE
     GOTOXY(14,9);
     TEXTBACKGROUND(7);
     WRITELN('USE DOWN_ARROW KEY TO CONTROL WRIST PIVOT MOVE
     ANTICLOCKWISE');
     GOTOXY(34,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
          #72 : U := U+1;
```

```
    #75 : L := L+1;
    #77 : R := R+1;
    #80 : D := D+1;
    #27 : DONE := TRUE;
   END;

BEGIN
  PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
  PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
  PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
  BEGIN
      FOR X :=1 TO 2 DO
        BEGIN
          FOR I:= 1 TO 2 DO
       PORT[PA1]:=CLR_DATA[I];
       PORT[PB1]:=CLR_DATA[I];
       PORT[PC1]:=CLR_DATA[I];
       PORT[PA2]:=CLR_DATA[I];
       PORT[PB2]:=CLR_DATA[I];
       PORT[PC2]:=CLR_DATA[I];
       PORT[PA3]:=CLR_DATA[I];
       PORT[PB3]:=CLR_DATA[I];
       PORT[PC3]:=CLR_DATA[I];
          END;
 END;

BEGIN
  CLRSCR;
  IF CH = #75 THEN
  GOTO LOOP;
  IF CH = #77 THEN
  GOTO LOOP;
  GOTOXY(20,12);
  TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
  WRITELN('NOW ! THE WRIST PIVOT OF ROBOT IS OPERATING');
BEGIN
  IF CH = #72  THEN
   BEGIN
   FOR X := 1 TO U DO
       BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PC3]:=DATA_OUT2[I];
          DELAY(7);
          END;
          IF X = U  THEN
          PORT[PC3]:=CLR_DATA[I];
        END;
    END;
    IF CH = #80   THEN
    BEGIN
    FOR X := 1 TO D DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PC3]:=DATA_OUT1[I];
          DELAY(7);
```

```
                END;
                IF X = D THEN
                PORT[PC3]:=CLR_DATA[I];
            END;
          END;
        END;
      END;
    END;


    UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEYBOARD (WRIST ROTATE OF ROBOT);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP,L1;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(24,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(15,7);
     TEXTBACKGROUND(5);
     WRITELN('USE RIGHT_ARROW KEY TO CONTROL WRIST ROTATE MOVE RIGHT
     GOTOXY(16,9);
     TEXTBACKGROUND(7);
     WRITELN('USE LEFT_ARROW KEY TO CONTROL WRIST ROTATE MOVE LEFT')
     GOTOXY(34,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
         #72 : U := U+1;
         #75 : L := L+1;
```

```
      #77 : R := R+1;
      #80 : D := D+1;
      #27 : DONE := TRUE;
     END;

  BEGIN
    PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
    PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
    PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
    BEGIN
        FOR X :=1 TO 2 DO
          BEGIN
             FOR I:= 1 TO 2 DO
         PORT[PA1]:=CLR_DATA[I];
         PORT[PB1]:=CLR_DATA[I];
         PORT[PC1]:=CLR_DATA[I];
         PORT[PA2]:=CLR_DATA[I];
         PORT[PB2]:=CLR_DATA[I];
         PORT[PC2]:=CLR_DATA[I];
         PORT[PA3]:=CLR_DATA[I];
         PORT[PB3]:=CLR_DATA[I];
         PORT[PC3]:=CLR_DATA[I];
            END;
   END;

BEGIN
    CLRSCR;
    IF CH = #72 THEN
    GOTO LOOP;
    IF CH = #80 THEN
    GOTO LOOP;
    GOTOXY(20,12);
    TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
    WRITELN('NOW ! THE WRIST ROTATE OF ROBOT IS OPERATING');
 BEGIN
    IF CH = #75  THEN
     BEGIN
     FOR X := 1 TO L DO
         BEGIN
          FOR I:= 1 TO 4 DO
            BEGIN
            PORT[PA3]:=DATA_OUT1[I];
            DELAY(6);
            END;
            IF X = L  THEN
            PORT[PA3]:=CLR_DATA[I];
          END;
     END;
     IF CH = #77   THEN
     BEGIN
     FOR X := 1 TO R DO
       BEGIN
         FOR I:= 1 TO 4 DO
           BEGIN
           PORT[PA3]:=DATA_OUT2[I];
           DELAY(6);
           END;
```

```
                IF X = R THEN
                PORT[PA3]:=CLR_DATA[I];
            END;
          END;
        END;
      END;
    END;


    UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEYBOARD (GRIPPER OF ROBOT);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP,L1;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(24,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(17,7);
     TEXTBACKGROUND(5);
     WRITELN('USE RIGHT_ARROW KEY TO CONTROL GRIPPER MOVE GRIP');
     GOTOXY(17,9);
     TEXTBACKGROUND(7);
     WRITELN('USE LEFT_ARROW KEY TO CONTROL GRIPPER MOVE EXPAND');
     GOTOXY(34,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
         #72 : U := U+1;
         #75 : L := L+1;
```

```
    #77 : R := R+1;
    #80 : D := D+1;
    #27 : DONE := TRUE;
   END;

BEGIN
  PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
  PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
  PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
  BEGIN
      FOR X :=1 TO 2 DO
        BEGIN
          FOR I:= 1 TO 2 DO
       PORT[PA1]:=CLR_DATA[I];
       PORT[PB1]:=CLR_DATA[I];
       PORT[PC1]:=CLR_DATA[I];
       PORT[PA2]:=CLR_DATA[I];
       PORT[PB2]:=CLR_DATA[I];
       PORT[PC2]:=CLR_DATA[I];
       PORT[PA3]:=CLR_DATA[I];
       PORT[PB3]:=CLR_DATA[I];
       PORT[PC3]:=CLR_DATA[I];
          END;
 END;

BEGIN
  CLRSCR;
  IF CH = #72 THEN
  GOTO LOOP;
  IF CH = #80 THEN
  GOTO LOOP;
  GOTOXY(20,12);
  TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
  WRITELN('NOW ! THE GRIPPER OF ROBOT IS OPERATING');
BEGIN
  IF CH = #75  THEN
   BEGIN
   FOR X := 1 TO L DO
       BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PB3]:=DATA_OUT1[I];
          DELAY(6);
          END;
          IF X = L  THEN
          PORT[PB3]:=CLR_DATA[I];
        END;
    END;
    IF CH = #77   THEN
    BEGIN
    FOR X := 1 TO R DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PB3]:=DATA_OUT2[I];
          DELAY(6);
          END;
```

```
                IF X = R THEN
                PORT[PB3]:=CLR_DATA[I];
              END;
          END;
        END;
      END;
    END;


    UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEYBOARD (FRONT WHEEL TURN);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP,L1;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(26,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(17,7);
     TEXTBACKGROUND(5);
     WRITELN('USE RIGHT_ARROW KEY TO CONTROL FRONT WHEEL TURN RIGHT'
     GOTOXY(17,9);
     TEXTBACKGROUND(7);
     WRITELN('USE LEFT_ARROW KEY TO CONTROL FRONT WHEEEL TURN LEFT')
     GOTOXY(34,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
         #72 : U := U+1;
         #75 : L := L+1;
```

```
    #77 : R := R+1;
    #80 : D := D+1;
    #27 : DONE := TRUE;
   END;

BEGIN
  PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
  PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
  PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
  BEGIN
      FOR X :=1 TO 2 DO
        BEGIN
          FOR I:= 1 TO 2 DO
       PORT[PA1]:=CLR_DATA[I];
       PORT[PB1]:=CLR_DATA[I];
       PORT[PC1]:=CLR_DATA[I];
       PORT[PA2]:=CLR_DATA[I];
       PORT[PB2]:=CLR_DATA[I];
       PORT[PC2]:=CLR_DATA[I];
       PORT[PA3]:=CLR_DATA[I];
       PORT[PB3]:=CLR_DATA[I];
       PORT[PC3]:=CLR_DATA[I];
          END;
  END;

  BEGIN
   CLRSCR;
   IF CH = #72 THEN
   GOTO LOOP;
   IF CH = #80 THEN
   GOTO LOOP;
   GOTOXY(20,12);
   TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
   WRITELN('NOW ! THE FRONT WHEEL TURN OF ROBOT IS OPERATING');
BEGIN
  IF CH = #75  THEN
   BEGIN
   FOR X := 1 TO L DO
       BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PA1]:=DATA_OUT2[I];
          DELAY(6);
          END;
          IF X = L  THEN
          PORT[PA1]:=CLR_DATA[I];
        END;
    END;
    IF CH = #77   THEN
    BEGIN
    FOR X := 1 TO R DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PA1]:=DATA_OUT1[I];
          DELAY(6);
          END;
```

```
                IF X = R THEN
                PORT[PA1]:=CLR_DATA[I];
              END;
          END;
        END;
      END;
    END;


    UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL BY KEYBOARD (FRONT WHEEL MOVE);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 LABEL    LOOP,L1;
 VAR      DONE : BOOLEAN;
          CH : CHAR;
          I:BYTE;
          D,L,R,U,X,Y,Z : INTEGER;
 TYPE     FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
           C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);
 BEGIN
LOOP:     CLRSCR;
     GOTOXY(26,5);
     TEXTCOLOR(LIGHTGREEN); TEXTBACKGROUND(3);
     WRITELN('MENU CONTROL ROBOT WITH KEYBOARD');
     GOTOXY(16,7);
     TEXTBACKGROUND(5);
     WRITELN('USE UP_ARROW KEY TO CONTROL FRONT WHEEL MOVE FORWARD'
     GOTOXY(15,9);
     TEXTBACKGROUND(7);
   WRITELN('USE DOWN_ARROW KEY TO CONTROL FRONT WHEEL MOVE BACKWARD
     GOTOXY(34,11);
     TEXTBACKGROUND(1);
     WRITELN('ESC  TO  EXIT');
     TEXTBACKGROUND(3);
       DONE := FALSE;
 REPEAT
      D := 0 ;
      L := 0;
      R := 0;
      U := 0;
      CH := READKEY;
        CASE CH OF
         #72 : U := U+1;
         #75 : L := L+1;
```

```
      #77 : R := R+1;
      #80 : D := D+1;
      #27 : DONE := TRUE;
     END;

BEGIN
  PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
  PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
  PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
  BEGIN
      FOR X :=1 TO 2 DO
        BEGIN
          FOR I:= 1 TO 2 DO
       PORT[PA1]:=CLR_DATA[I];
       PORT[PB1]:=CLR_DATA[I];
       PORT[PC1]:=CLR_DATA[I];
       PORT[PA2]:=CLR_DATA[I];
       PORT[PB2]:=CLR_DATA[I];
       PORT[PC2]:=CLR_DATA[I];
       PORT[PA3]:=CLR_DATA[I];
       PORT[PB3]:=CLR_DATA[I];
       PORT[PC3]:=CLR_DATA[I];
          END;
 END;

L1:  BEGIN
   CLRSCR;
   IF CH = #75 THEN
   GOTO LOOP;
   IF CH = #77 THEN
   GOTO LOOP;
   GOTOXY(20,12);
   TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(3);
   WRITELN('NOW ! THE FRONT WHEEL MOVE OF ROBOT IS OPERATING');
BEGIN
   IF CH = #72  THEN
    BEGIN
    FOR X := 1 TO U DO
        BEGIN
         FOR I:= 1 TO 4 DO
           BEGIN
           PORT[PC1]:=DATA_OUT3[I];
           DELAY(20);
           END;
           IF X = U  THEN
           PORT[PC1]:=CLR_DATA[I];
         END;
     END;
     IF CH = #80   THEN
     BEGIN
     FOR X := 1 TO D DO
       BEGIN
         FOR I:= 1 TO 4 DO
           BEGIN
           PORT[PC1]:=DATA_OUT4[I];
           DELAY(20);
           END;
```

```
                IF X = D THEN
                PORT[PC1]:=CLR_DATA[I];
            END;
          END;
        END;
      END;
    END;


    UNTIL DONE = TRUE;
END.
```

```
PROGRAM ROBOT_CONTROL_BY_COMPUTER (AUTO TEST2);
 { ETPC-8255 TURBO PASCAL V6.0 }
 {SET DIP SW PORT 300H }
 USES     CRT;
 VAR      I:BYTE;
          M1,M2,M3,M4,M5,M6,M7,M8,SP,X : INTEGER;
 TYPE
          SP_DATA = ARRAY [1..4] OF BYTE;
          FR_DATA = ARRAY [1..4] OF BYTE;
          RV_DATA = ARRAY [1..4] OF BYTE;
          MOVFORW_DATA = ARRAY [1..4] OF BYTE;
          MOVREV_DATA = ARRAY [1..4] OF BYTE;
          C_DATA = ARRAY [1..2] OF BYTE;
 CONST    PA1 = $0300;
          PB1 = $0301;
          PC1 = $0302;
          PCONT_1 =$0303;
          PA2 = $0304;
          PB2 = $0305;
          PC2 = $0306;
          PCONT_2 = $0307;
          PA3 = $0308;
          PB3 = $0309;
          PC3 = $030A;
          PCONT_3 = $030B;
          DATA_OUT  :SP_DATA=($10,$10,$10,$10);
          DATA_OUT1 :FR_DATA=($01,$02,$04,$08);
          DATA_OUT2 :RV_DATA=($08,$04,$02,$01);
          DATA_OUT3 :MOVFORW_DATA=($03,$03,$03,$03);
          DATA_OUT4 :MOVREV_DATA=($0A,$0A,$0A,$0A);
          CLR_DATA  :C_DATA=($00,$00);

  BEGIN
       PORT[PCONT_1]:=$80;   {PA1,PB1,PC1=OUTPUT PORT}
       PORT[PCONT_2]:=$80;   {PA2,PB2,PC2=OUTPUT PORT}
       PORT[PCONT_3]:=$80;   {PA3,PB3,PC3=OUTPUT PORT}
       BEGIN
            FOR X := 1 TO 2 DO
              BEGIN
                 FOR I:= 1 TO 2 DO
             PORT[PA1]:=CLR_DATA[I];
             PORT[PB1]:=CLR_DATA[I];
             PORT[PC1]:=CLR_DATA[I];
             PORT[PA2]:=CLR_DATA[I];
             PORT[PB2]:=CLR_DATA[I];
             PORT[PC2]:=CLR_DATA[I];
             PORT[PA3]:=CLR_DATA[I];
             PORT[PB3]:=CLR_DATA[I];
             PORT[PC3]:=CLR_DATA[I];
                END;
       END;

       CLRSCR;
       TEXTCOLOR(LIGHTGREEN+128); TEXTBACKGROUND(5);
       GOTOXY(25,12);
        WRITELN('NOW ! IT IS THE SOUND OF ROBOT ');
```

```
    FOR SP := 1 TO 500 DO
      BEGIN
        FOR I := 1 TO 4 DO
          BEGIN
          PORT[PC1]:=DATA_OUT[I];
          DELAY(5);
          END;
          IF SP = 500 THEN
          PORT[PC1]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE SHOULDER OF ROBOT IS TURNING');
    FOR M2 := 1 TO 500 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PA2]:=DATA_OUT1[I];
          DELAY(8);
          END;
          IF M2 = 500 THEN
          PORT[PA2]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE  WRIST PIVOT OF ROBOT IS TURNING');
    FOR M4:=1 TO 100 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PC3]:=DATA_OUT2[I];
          DELAY(5);
          END;
          IF M4 = 100 THEN
          PORT[PC3]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE WRIST ROTATE  OF ROBOT IS TURNING');
    FOR M5:=1 TO 100 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PA3]:=DATA_OUT1[I];
          DELAY(5);
          END;
          IF M5 = 100 THEN
          PORT[PA3]:=CLR_DATA[I];
     END;


CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE GRIPPER OF ROBOT IS MOVE GRIPPING');
```

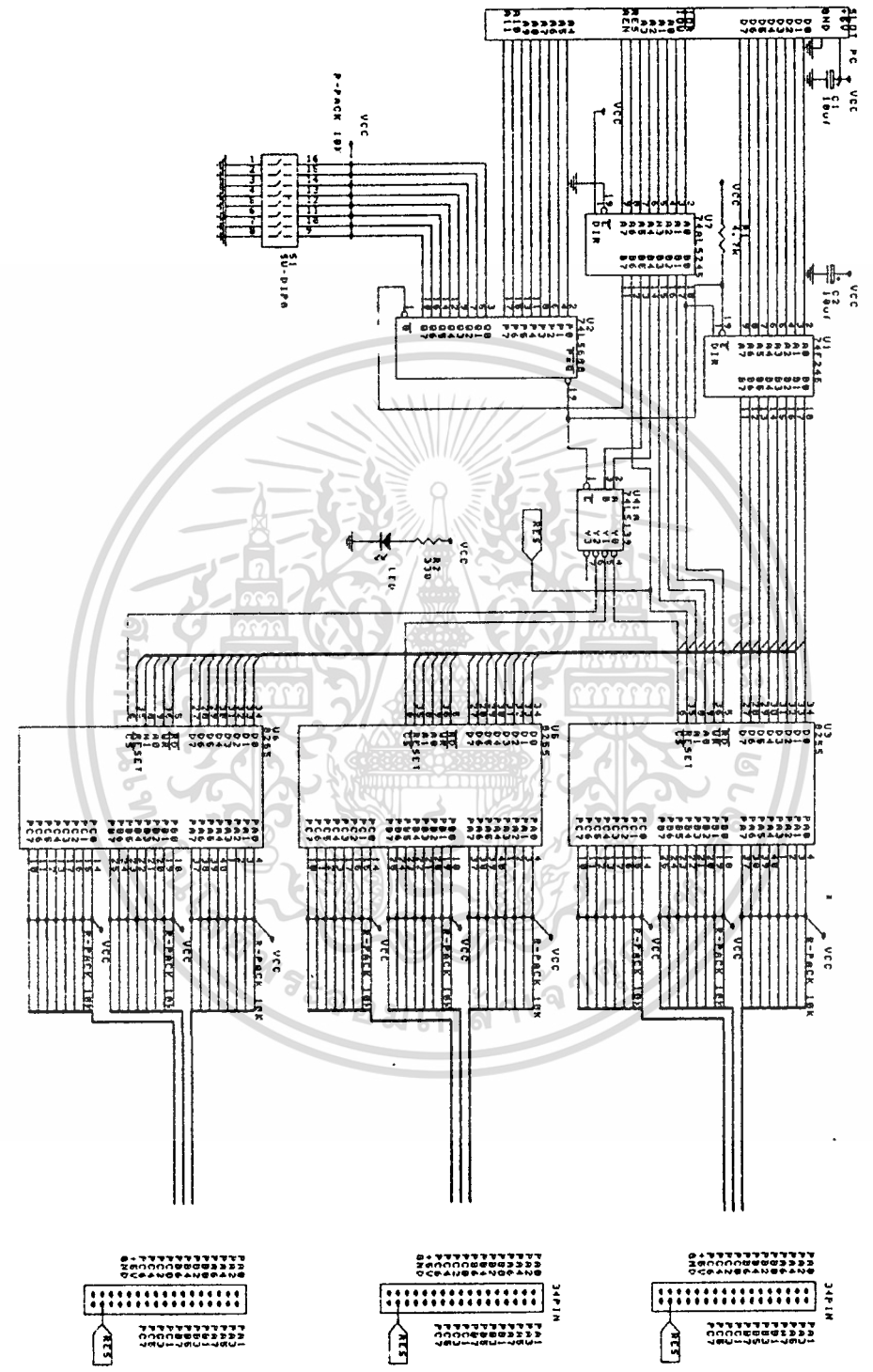
```
    FOR M6:=1 TO 100 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PB3]:=DATA_OUT2[I];
          DELAY(5);
          END;
          IF M6 = 100 THEN
          PORT[PB3]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
 WRITELN('NOW ! THE HEAD OF ROBOT IS TURNING ANTICLOCKWISE')
    FOR M1:=1 TO 400 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PB1]:=DATA_OUT2[I];
          DELAY(5);
          END;
          IF M1 = 400 THEN
          PORT[PB1]:=CLR_DATA[I];
     END;

   CLRSCR;
   GOTOXY(25,12);
   WRITELN('NOW ! THE SHOULDER OF ROBOT IS TURNING');
   FOR M2 := 1 TO 500 DO
       BEGIN
         FOR I:= 1 TO 4 DO
           BEGIN
           PORT[PA2]:= DATA_OUT2[I];
           DELAY(8);
           END;
           IF M2 = 500 THEN
           PORT[PA2]:=CLR_DATA[I];
       END;


  CLRSCR;
  GOTOXY(25,12);
  WRITE('NOW ! THE GRIPPER OF ROBOT IS EXPANDING');
  FOR M6 := 1 TO 100 DO
      BEGIN
        FOR I:= 1 TO 4 DO
          BEGIN
          PORT[PB3]:=DATA_OUT1[I];
          DELAY(5);
          END;
          IF M6 = 100 THEN
          PORT[PB3]:=CLR_DATA[I];
       END;


  CLRSCR;
  GOTOXY(25,12);
```

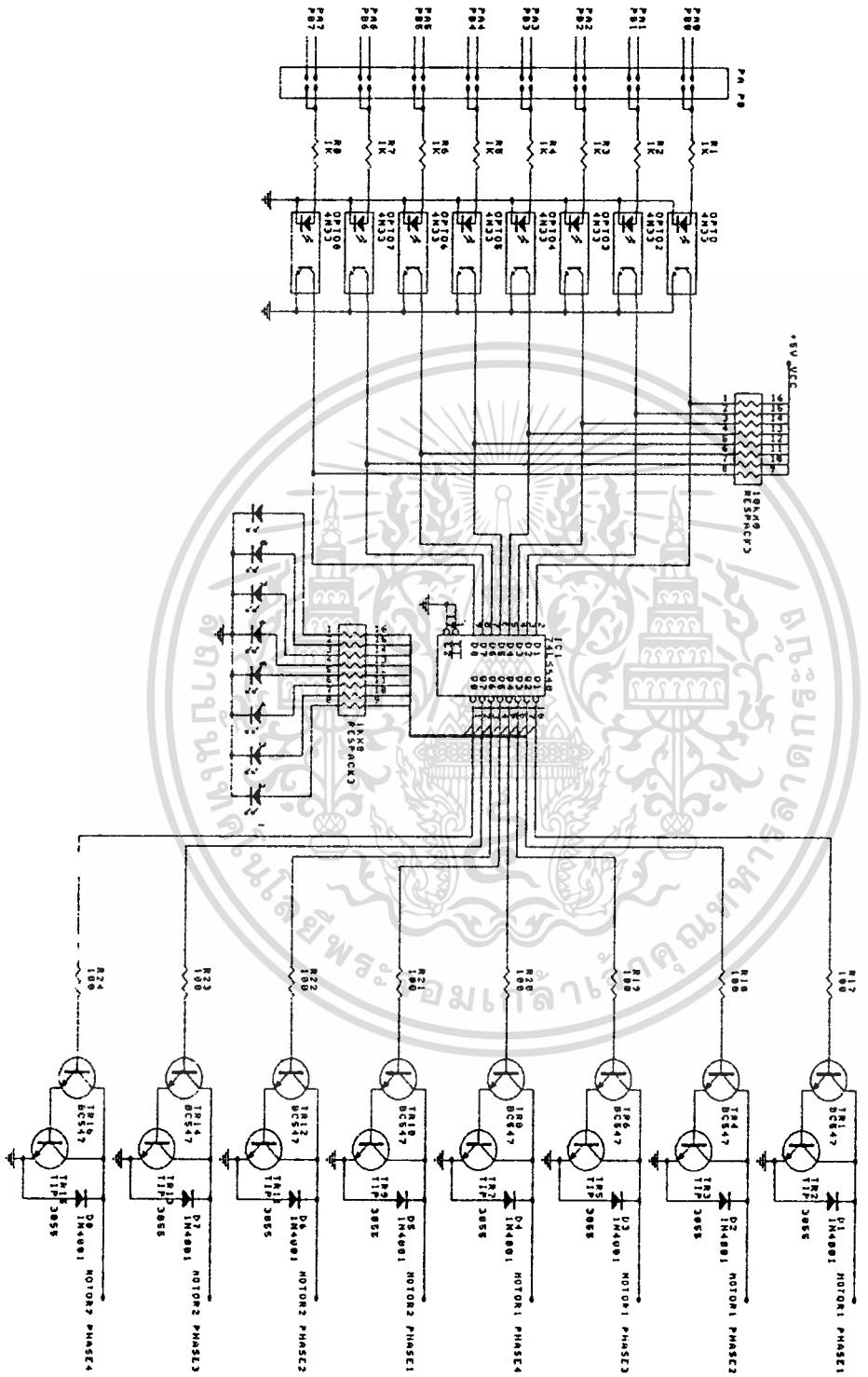
```
WRITE('NOW ! THE ARM OF ROBOT IS EXTENDING');
FOR M3 := 1 TO 600 DO
     BEGIN
       FOR I:=1 TO 4 DO
         BEGIN
         PORT[PB2]:=DATA_OUT2[I];
         DELAY(4);
         END;
         IF M3 = 600 THEN
         PORT[PB2]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE GRIPPER OF ROBOT IS GRIPPING');
FOR M6 := 1 TO 100 DO
     BEGIN
       FOR I:= 1 TO 4 DO
         BEGIN
         PORT[PB3]:=DATA_OUT2[I];
         DELAY(5);
         END;
         IF M6 = 100 THEN
         PORT[PB3]:=CLR_DATA[I];
      END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE ARM OF ROBOT IS PULLING ');
FOR M3:= 1 TO 600 DO
     BEGIN
       FOR I:=1 TO 4  DO
         BEGIN
         PORT[PB2]:=DATA_OUT1[I];
         DELAY(4);
         END;
         IF M3 = 600  THEN
         PORT[PB2]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE HEAD OF ROBOT IS TURNING CLOCKWISE');
FOR M1:= 1 TO 400 DO
     BEGIN
       FOR I:=1 TO 4  DO
         BEGIN
         PORT[PB1]:=DATA_OUT1[I];
         DELAY(5);
         END;
         IF M1 = 400  THEN
         PORT[PB1]:=CLR_DATA[I];
     END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE FRONT WHEEL OF ROBOT IS TURNING ');
```

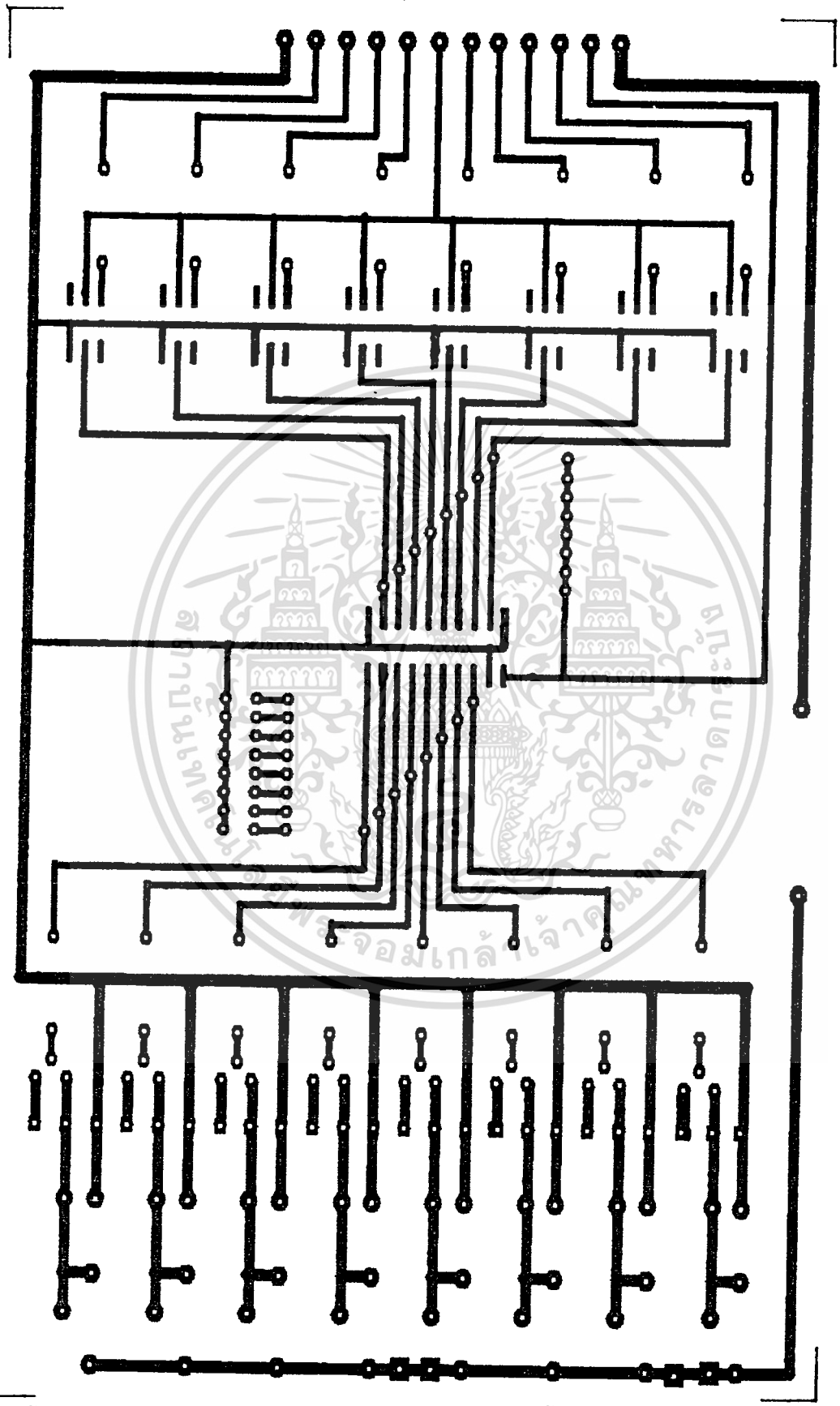
```
FOR M7:= 1 TO  50 DO
    BEGIN
      FOR I:=1 TO 4 DO
        BEGIN
        PORT[PA1]:=DATA_OUT2[I];
        DELAY(5);
        END;
        IF M7  = 50 THEN
        PORT[PA1]:=CLR_DATA[I];
    END;
FOR M7:= 1 TO  50 DO
    BEGIN
      FOR I:=1 TO 4 DO
        BEGIN
        PORT[PA1]:=DATA_OUT1[I];
        DELAY(5);
        END;
        IF M7  = 50 THEN
        PORT[PA1]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE  FRONT WHEEL OF ROBOT IS  MOVE FORWARD');
FOR M8:= 1 TO 100 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PC1]:=DATA_OUT3[I];
        DELAY(10);
        END;
        IF M8= 100 THEN
        PORT[PC1]:=CLR_DATA[I];
    END;

 CLRSCR;
 GOTOXY(25,12);
 WRITE('NOW ! THE ARM OF ROBOT  IS EXTENDING');
 FOR M3:= 1 TO 600 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PB2]:=DATA_OUT2[I];
        DELAY(4);
        END;
        IF M3 = 600 THEN
        PORT[PB2]:=CLR_DATA[I];
    END;


 CLRSCR;
 GOTOXY(25,12);
 WRITE('NOW ! THE GRIPPER OF ROBOT  IS EXPANDING');
 FOR M6:= 1 TO 100 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
```

```
        PORT[PB3]:=DATA_OUT1[I];
        DELAY(5);
        END;
        IF M6 = 100 THEN
        PORT[PB3]:=CLR_DATA[I];
    END;


CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE ARM OF ROBOT  IS PULLING');
FOR M3:= 1 TO 600 DO
    BEGIN
      FOR I:= 1 TO 4 DO
        BEGIN
        PORT[PB2]:=DATA_OUT1[I];
        DELAY(4);
        END;
        IF M3 = 600 THEN
        PORT[PB2]:=CLR_DATA[I];
    END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE FRONT WHEEL OF ROBOT  IS MOVING BACKWARD'
FOR M8:= 1 TO 100 DO
    BEGIN
     FOR I:= 1 TO 4 DO
       BEGIN
       PORT[PC1]:=DATA_OUT4[I];
       DELAY(5);
       END;
       IF M8 = 100 THEN
       PORT[PC1]:=CLR_DATA[I];
     END;


CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE WRIST PIVOT OF ROBOT  IS TURNING');
FOR M4:= 1 TO 100 DO
   BEGIN
     FOR I:= 1 TO 4 DO
       BEGIN
       PORT[PC3]:=DATA_OUT1[I];
       DELAY(5);
       END;
       IF M4 = 100 THEN
       PORT[PC3]:=CLR_DATA[I];
   END;

CLRSCR;
GOTOXY(25,12);
WRITE('NOW ! THE WRIST ROTATE OF ROBOT  IS TURNING');
FOR M5:= 1 TO 100 DO
    BEGIN
      FOR I:= 1 TO 4 DO
```

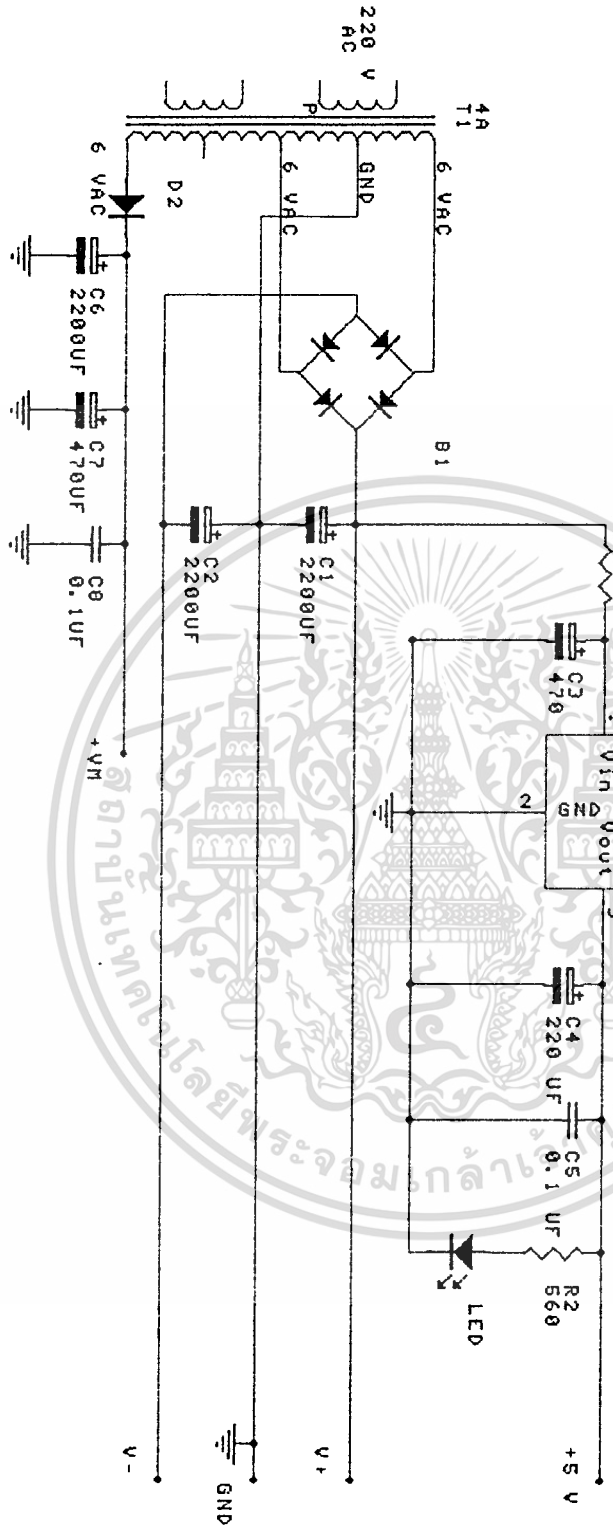
```
                BEGIN
                PORT[PA3]:=DATA_OUT2[I];
                DELAY(5);
                END;
                IF M5 = 100 THEN
                PORT[PA3]:=CLR_DATA[I];
            END;


        CLRSCR;
END.
```

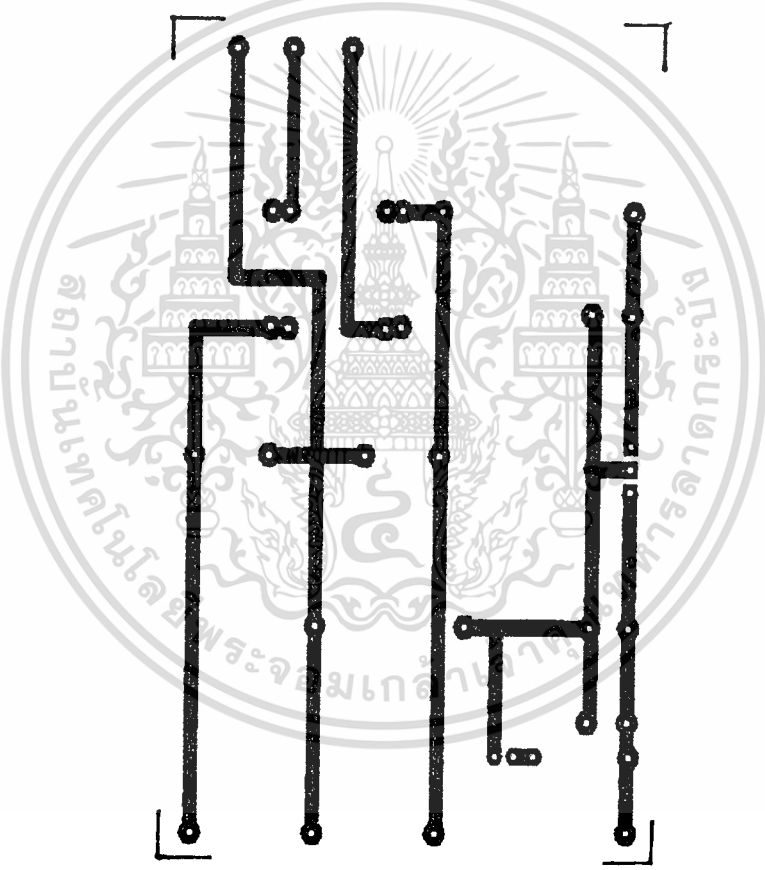




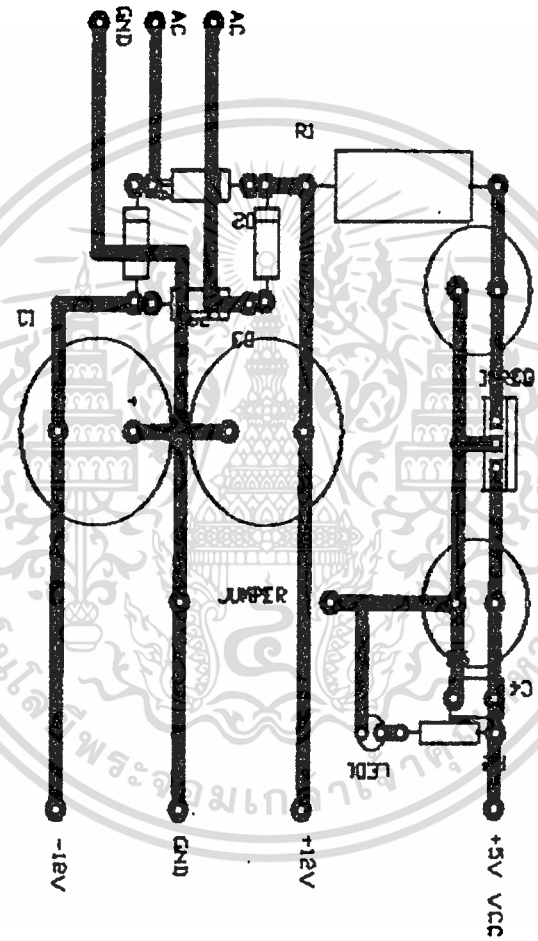


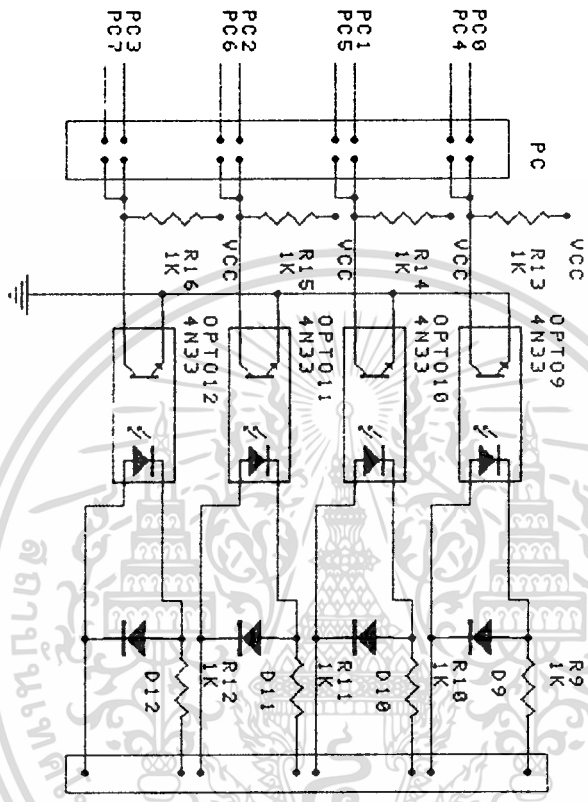
DRIVER4 Check Plot

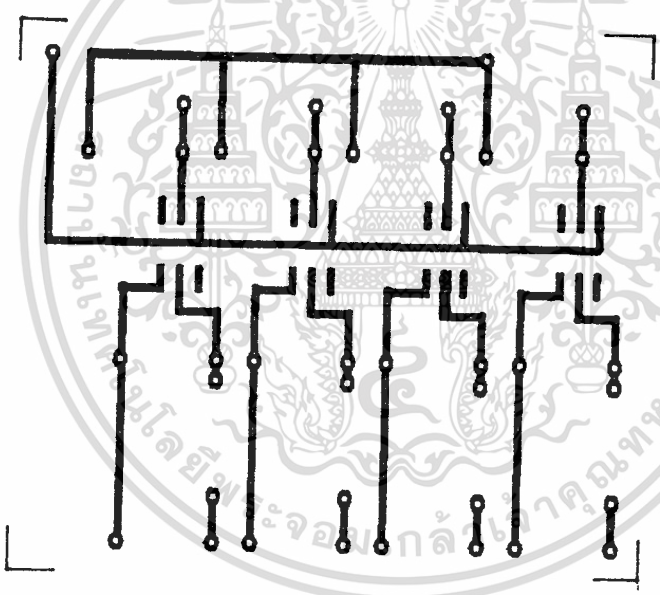




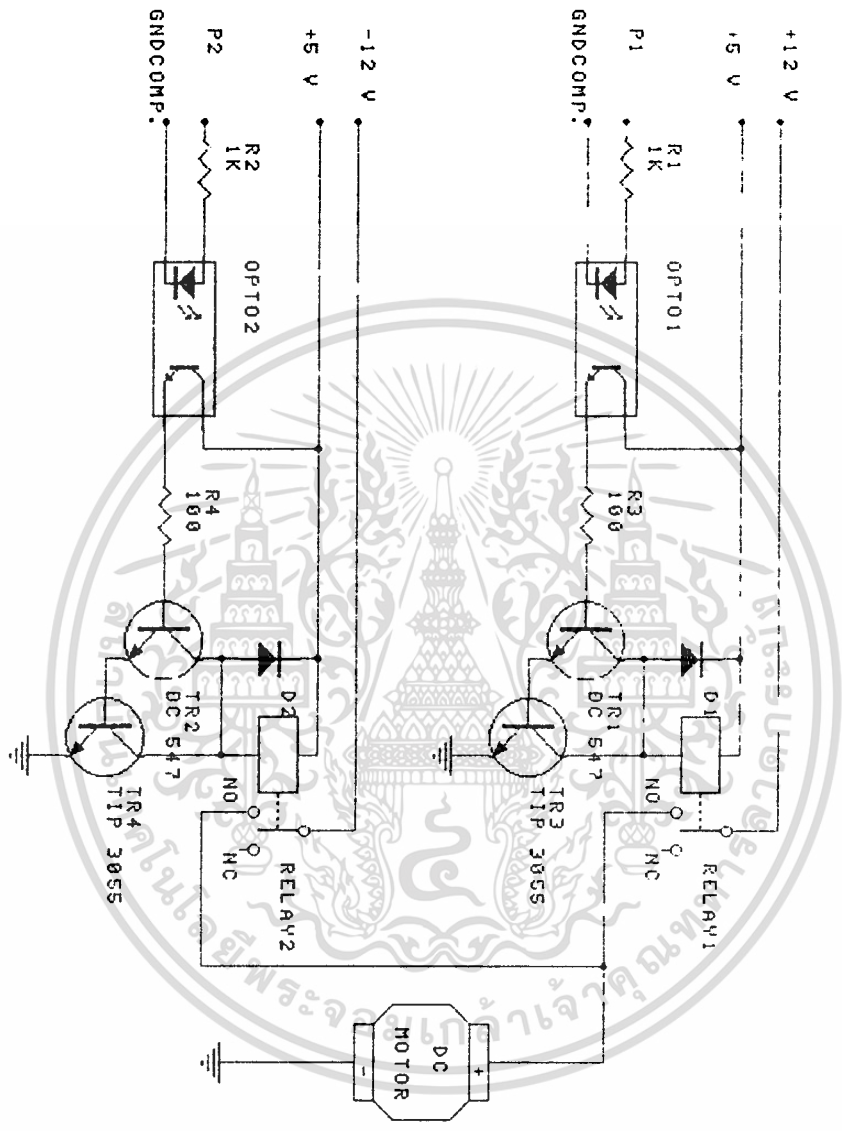
SUPPLY Check Plot

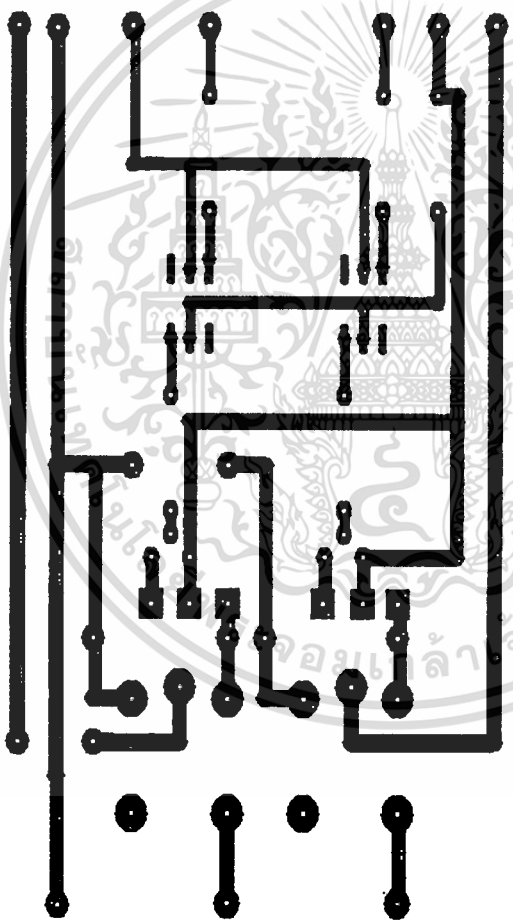


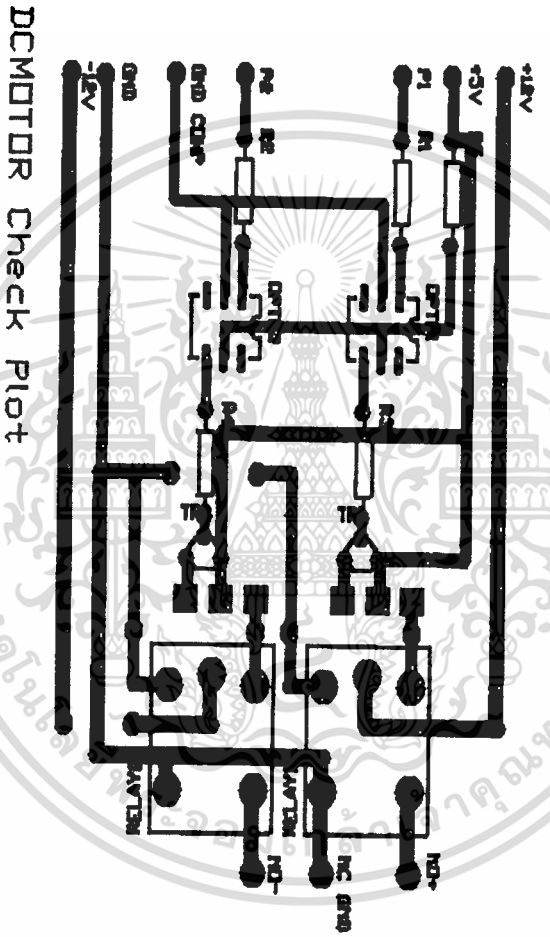


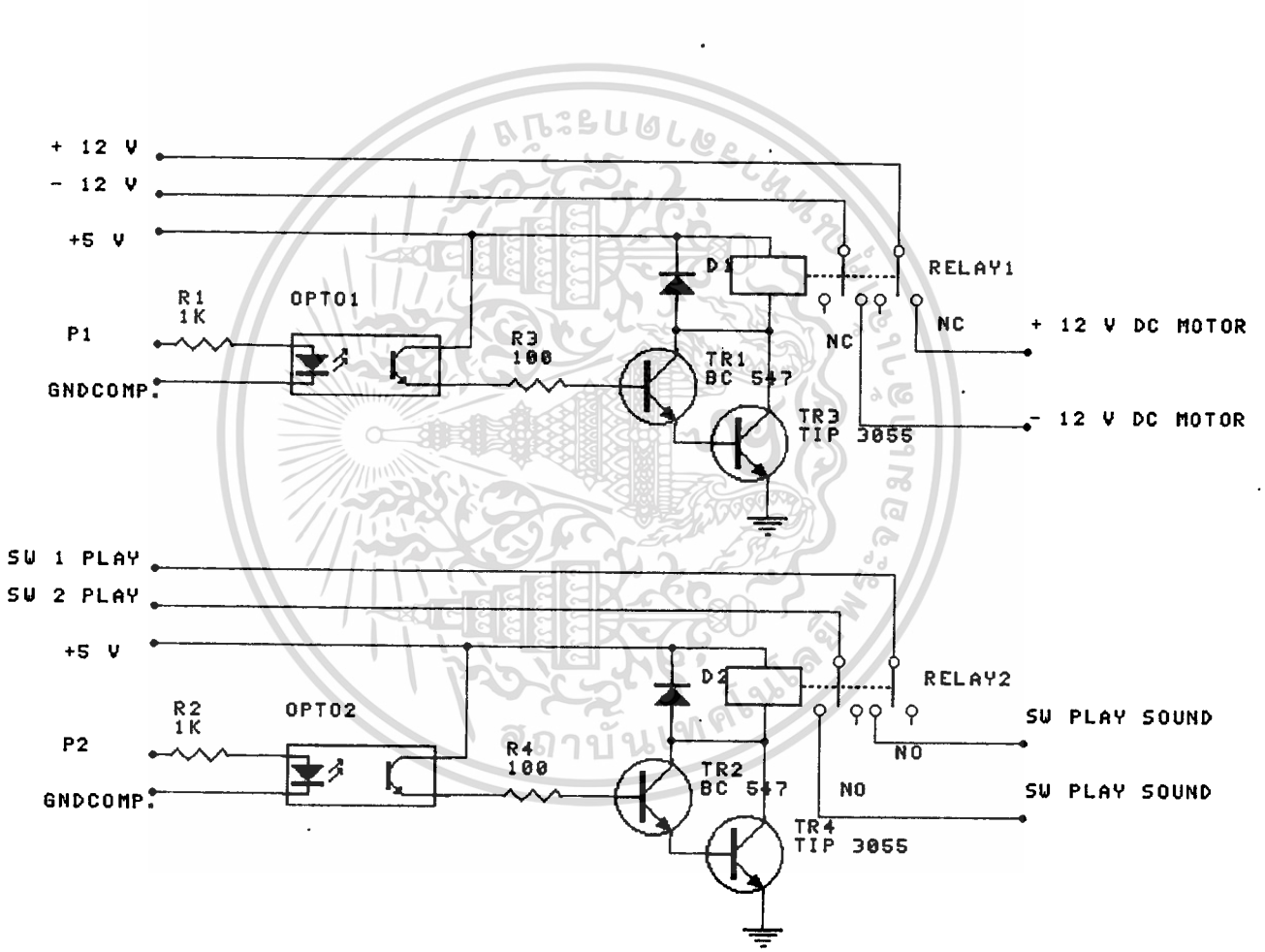


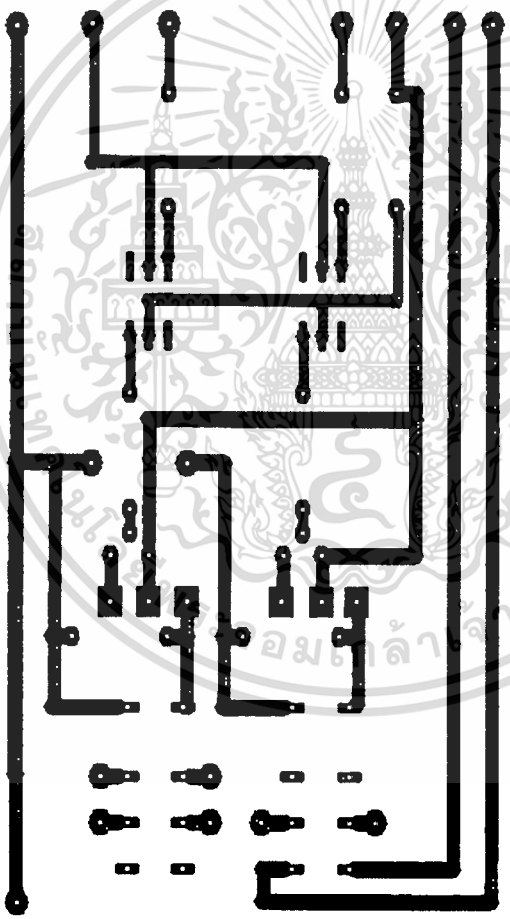
INPUT Check Plot



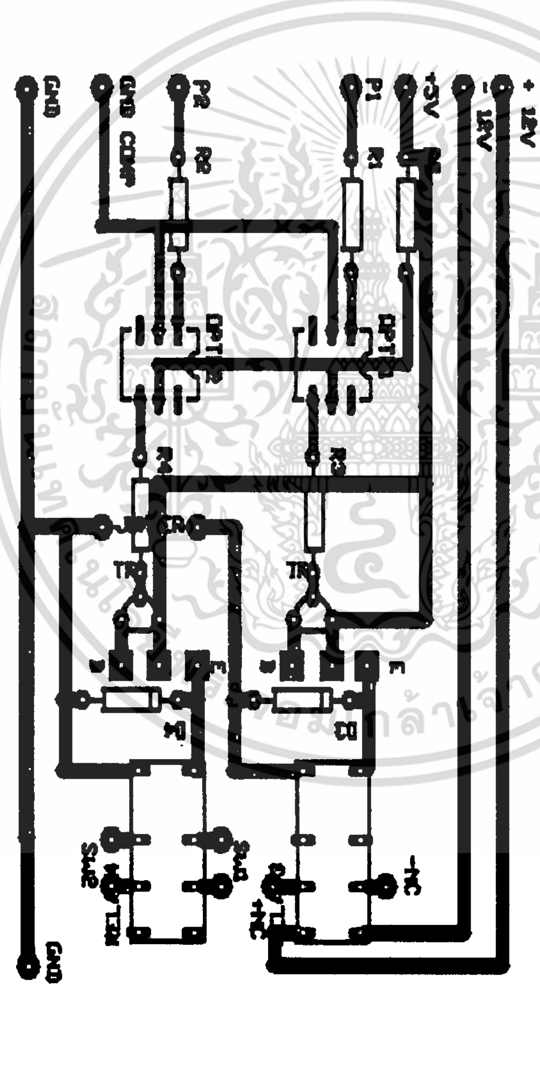








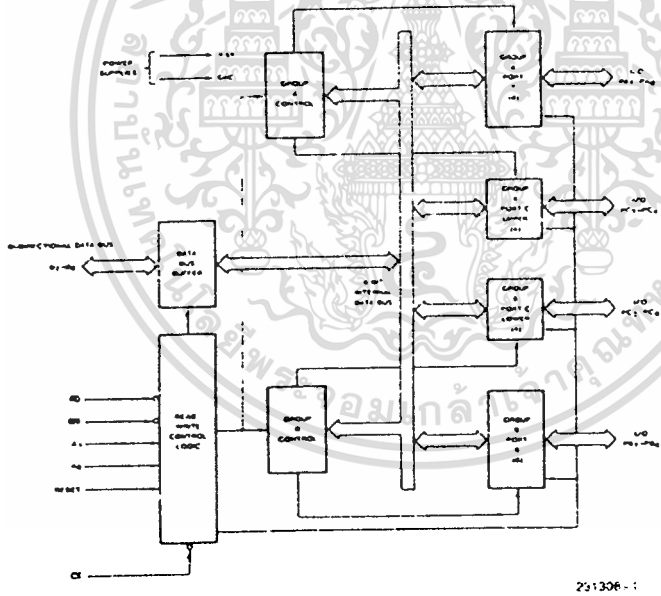
SOUND Check Plot



8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

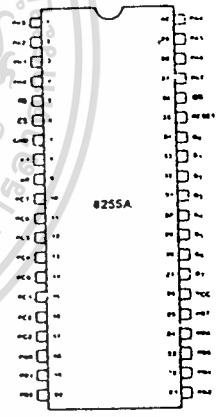
- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- Reduces System Package Count
- Improved DC Driving Capability
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range
- 40 Pin DIP Package or 44 Lead PLCC
  (See Intel Packaging Order Number, 231369)

The Intel 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.



231306-1

Figure 1. 8255A Block Diagram



231306-2

Figure 2. Pin Configuration

September 1987
Order Number: 231306-002

8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select. A "low" on this input pin enables the communication between the 8255A and the CPU.

($\overline{RD}$)

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

($\overline{WR}$)

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

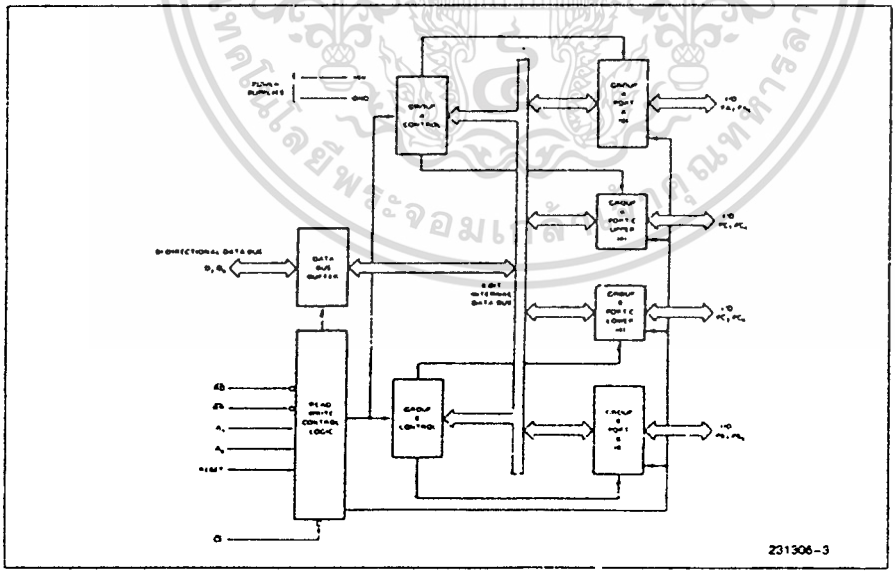


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input Operation (READ)
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
					Output Operation (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					Disable Function
X	X	X	X	1	Data Bus → 3-State
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus → 3-State

(RESET)

Reset. A "high" on this input clears the control register and all ports (A, B, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A—Port A and Port C upper (C7–C4)

Control Group B—Port B and Port C lower (C3–C0)

The Control Word Register can Only be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

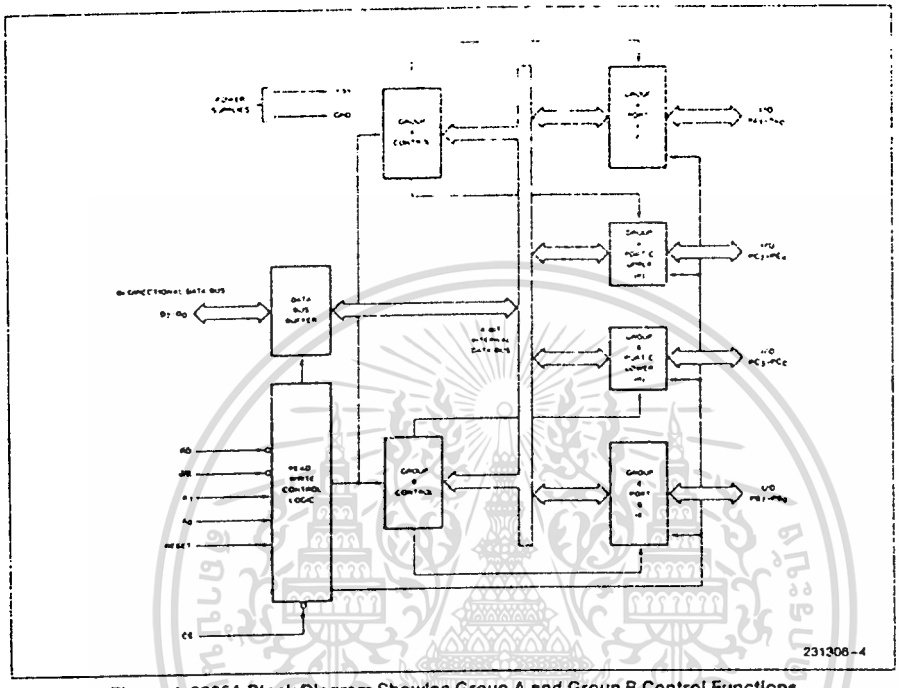
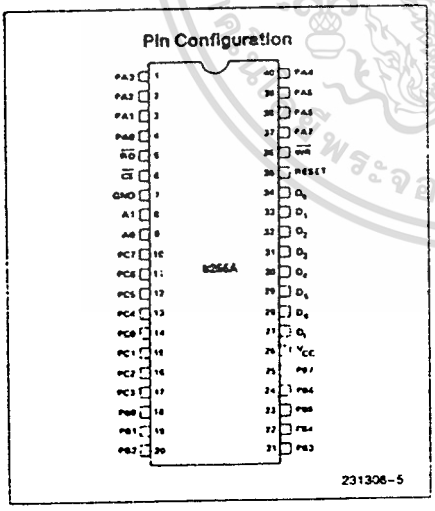


Figure 4. 8225A Block Diagram Showing Group A and Group B Control Functions



Pin Names

D_7-D_0	Data Bus (Bi-Directional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A0, A1	Port Address
PA7–PA0	Port A (BIT)
PB7–PB0	Port B (BIT)
PC7–PC0	Port C (BIT)
V_{CC}	+5 Volts
GND	0 Volts

8255A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0—Basic Input/Output

Mode 1—Strobed Input/Output

Mode 2—Bi-Directional Bus

When the reset input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

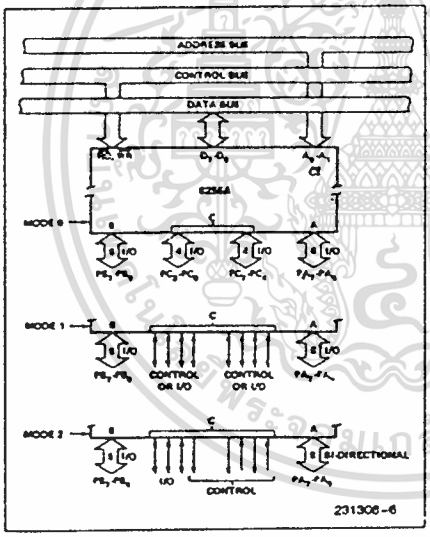


Figure 5. Basic Mode Definitions and Bus Interface

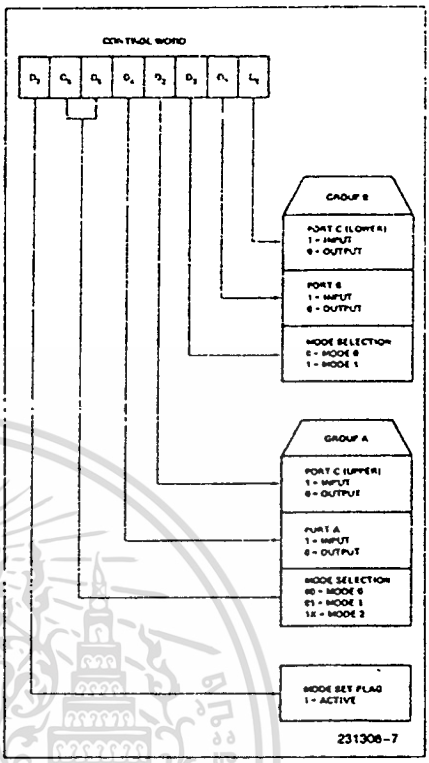


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

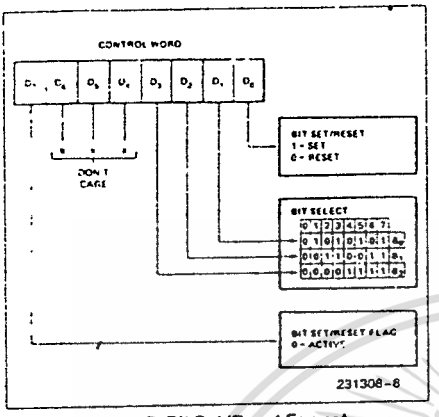


Figure 7. Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure

INTE flip-flop definition:

(BIT-SET)—INTE is set—interrupt enable

(BIT-RESET)—INTE is RESET—interrupt disable

NOTE:
All Mask flip-flops are automatically reset during mode selection and device Reset.

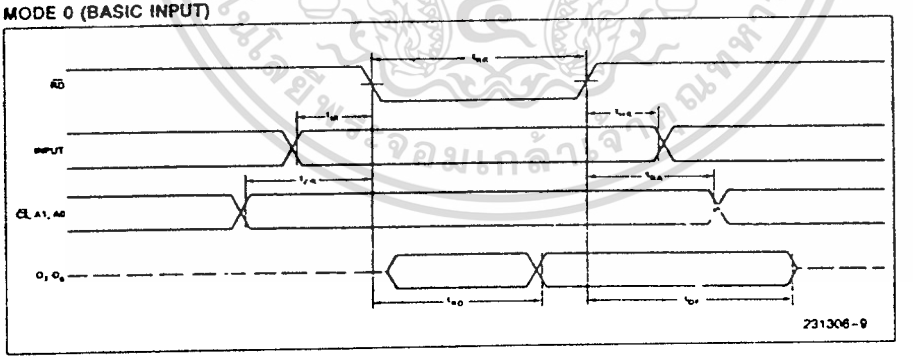
Operating Modes

MODE 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

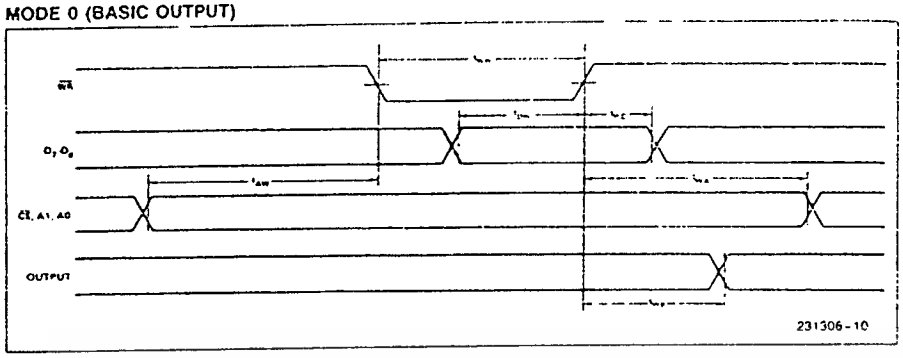
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



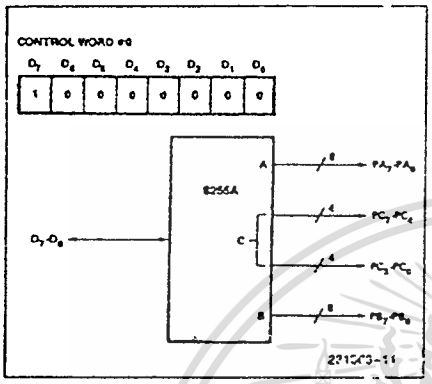
MODE 0 (BASIC OUTPUT)

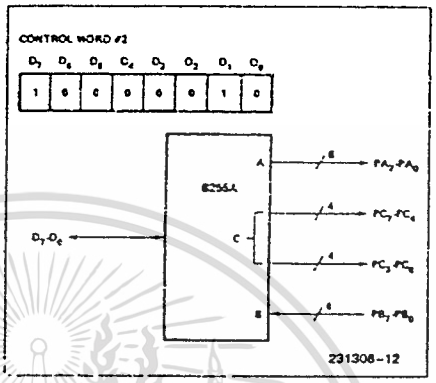


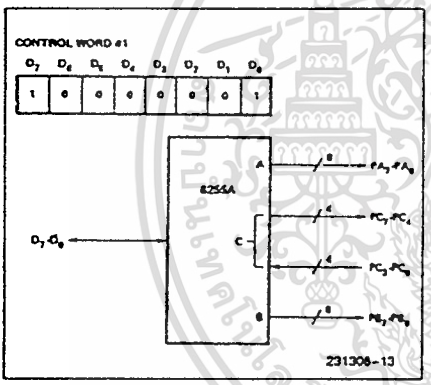
MODE 0 PORT DEFINITION

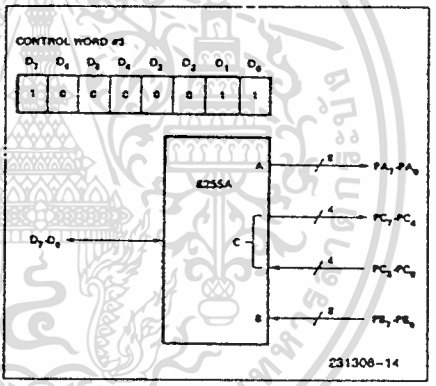
A		B		Group A			Group B	
D_4	D_3	D_1	D_0	Port A	Port C (Upper)	#	Port B	Port C (Lower)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

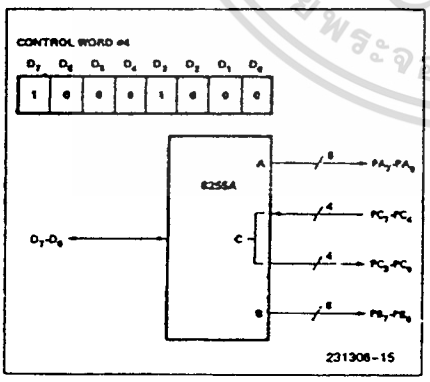
MODE CONFIGURATIONS

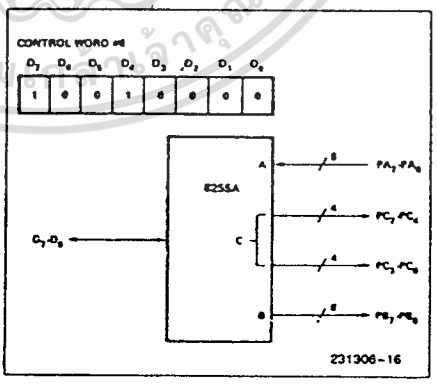


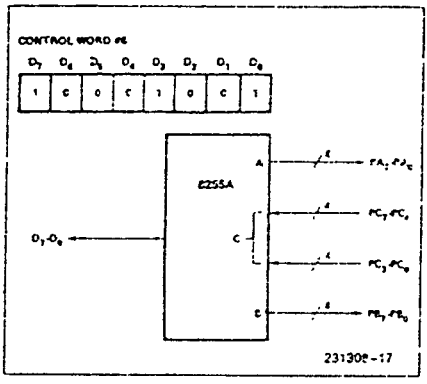


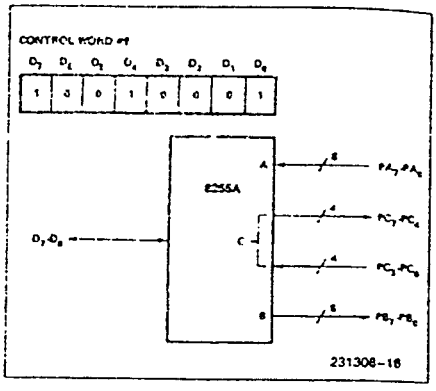


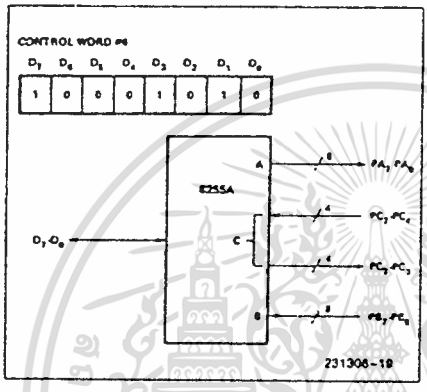


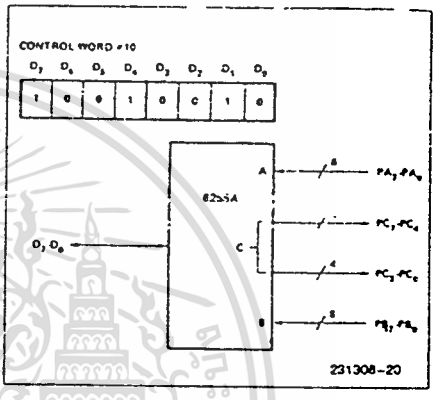


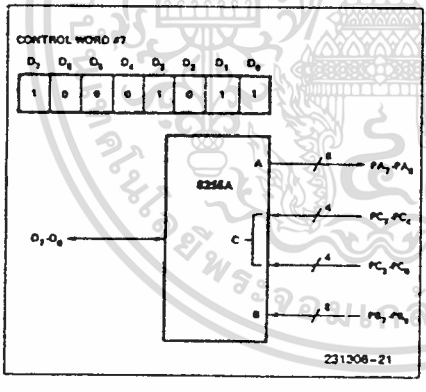


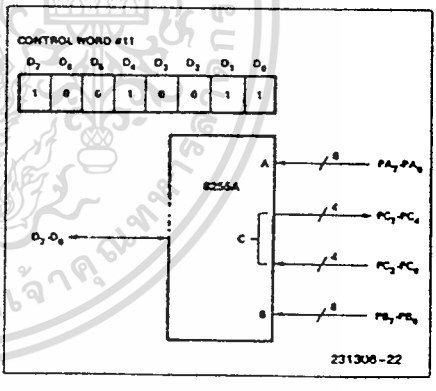


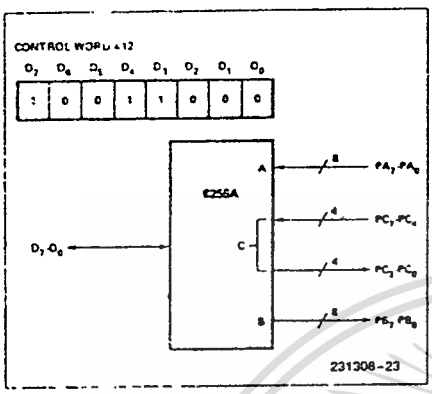


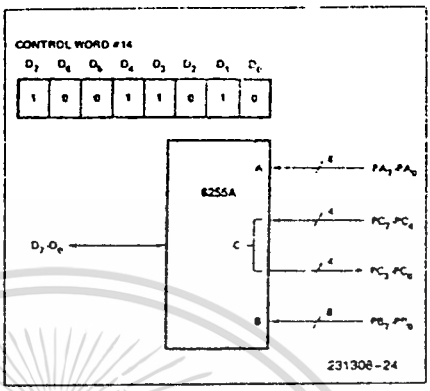


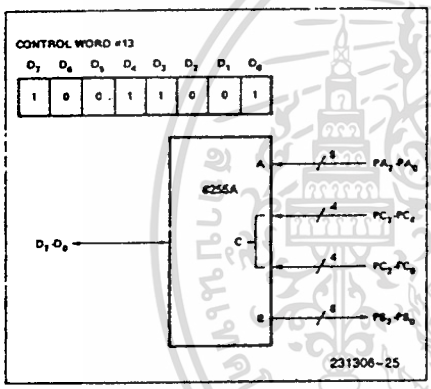


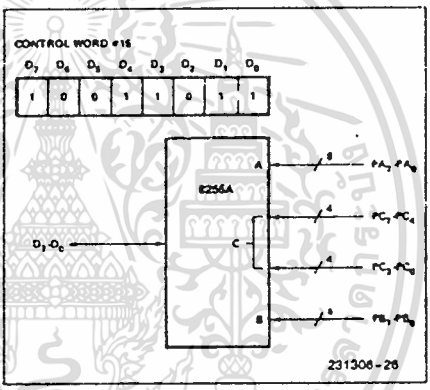












Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and port B use the lines on port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

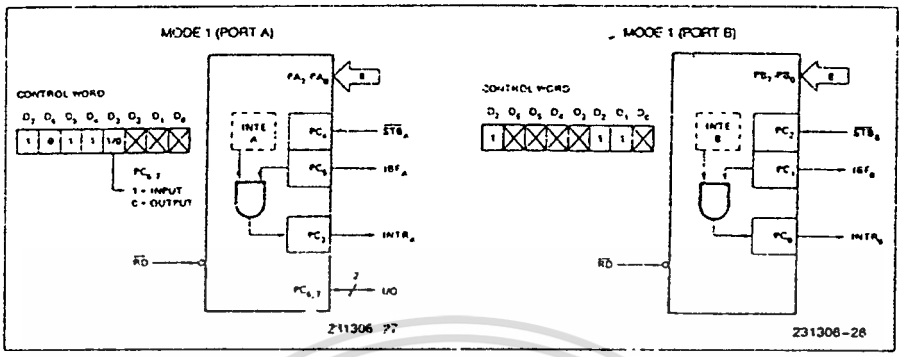


Figure 8. MODE 1 Input

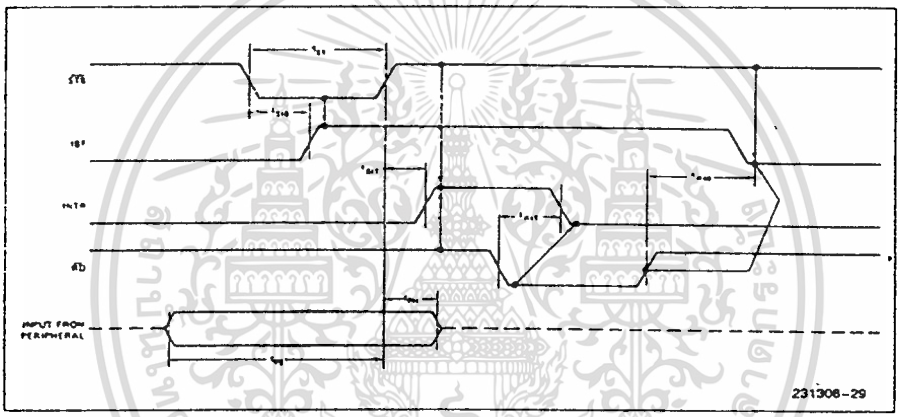


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{OBF}$ F/F will be set by the rising edge of the $\overline{WR}$ input and reset by $\overline{ACK}$ input being low.

$\overline{ACK}$ (Acknowledge Input). A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{ACK}$ is a "one", $\overline{OBF}$ is a "one", and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

INTE A

Controlled by bit set/reset of PC_6

INTE B

Controlled by bit set/reset of PC_2

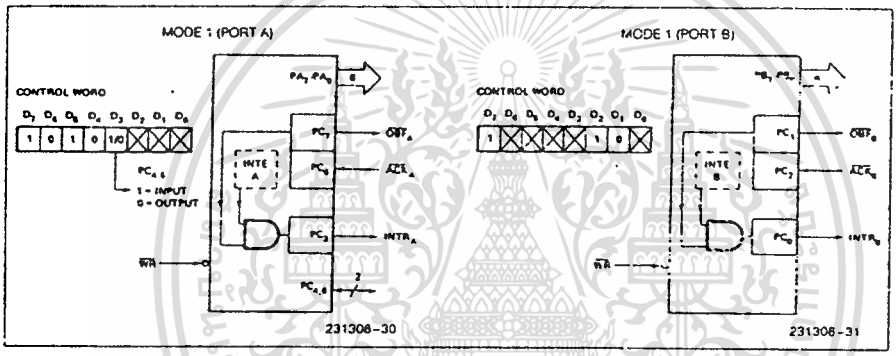


Figure 10. MODE 1 Output

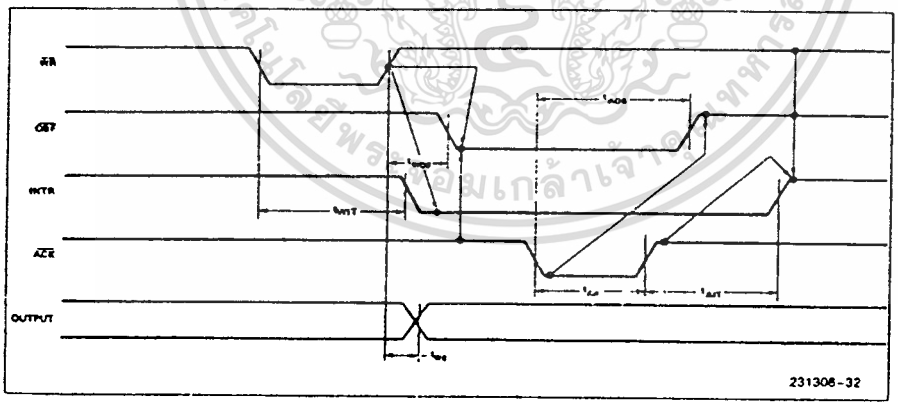


Figure 11. MODE 1 (Strobed Output)

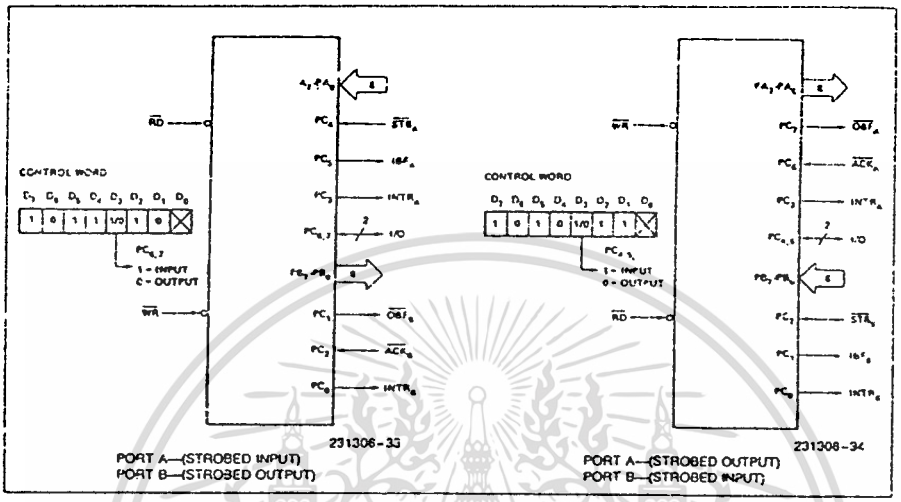


Figure 12. Combinations of MODE 1

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in MODE 1 to support a wide variety of strobed I/O applications.

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Full). The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with $\overline{OBF}$). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

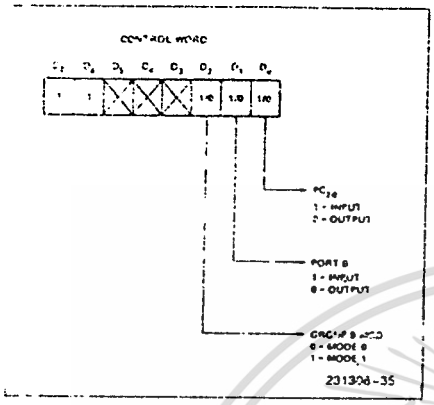


Figure 13. MODE Control Word

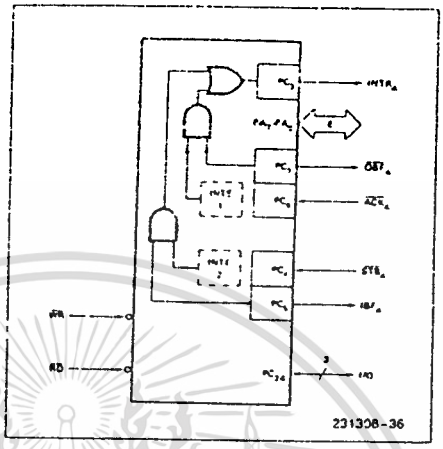
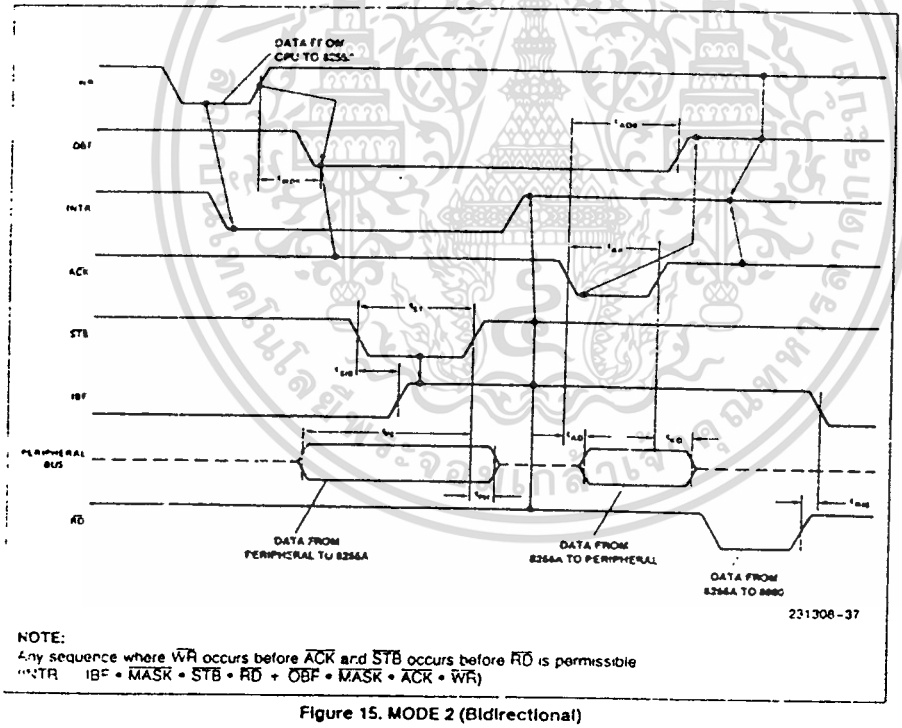


Figure 14. MODE 2



NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible
(INTR = IBF • MASK • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • MASK • $\overline{ACK}$ • $\overline{WR}$)

Figure 15. MODE 2 (Bidirectional)

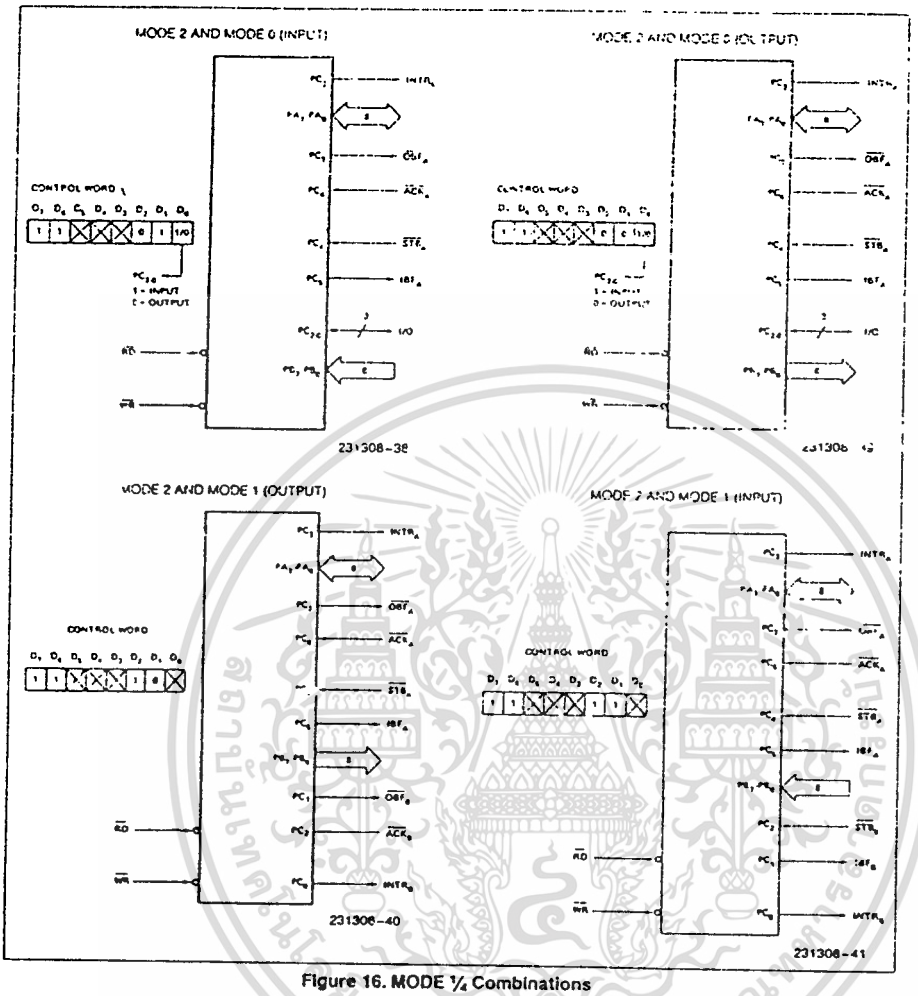


Figure 16. MODE 1/4 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	⟷	
PA_1	IN	OUT	IN	OUT	⟷	
PA_2	IN	OUT	IN	OUT	⟷	
PA_3	IN	OUT	IN	OUT	⟷	
PA_4	IN	OUT	IN	OUT	⟷	
PA_5	IN	OUT	IN	OUT	⟷	
PA_6	IN	OUT	IN	OUT	⟷	
PA_7	IN	OUT	IN	OUT	⟷	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	$\overline{IBF_B}$	$\overline{OBF_B}$	I/O	
PC_2	IN	OUT	$\overline{STB_B}$	$\overline{ACK_B}$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB_A}$	I/O	$\overline{STB_A}$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK_A}$	$\overline{ACK_A}$	
PC_7	IN	OUT	I/O	$\overline{OBF_A}$	$\overline{OBF_A}$	

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs—

All input lines can be accessed during a normal Port C read.

If Programmed as Outputs—

Bits in C upper (PC_7–PC_4) must be individually accessed using the bit set/reset function.

Bits in C lower (PC_3–PC_0) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

Source Current Capability on Port B and Port C

Any set of **eight** output buffers, selected randomly from Ports B and C can source 1 mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

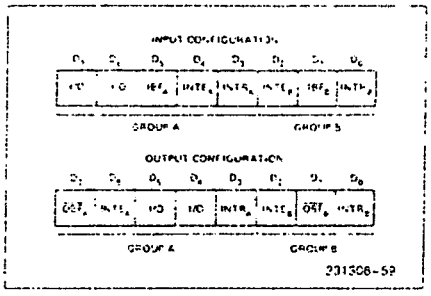


Figure 17. MODE 1 Status Word Format

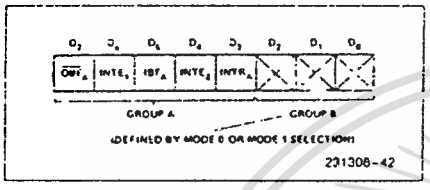


Figure 18. MODE 2 Status Word Format

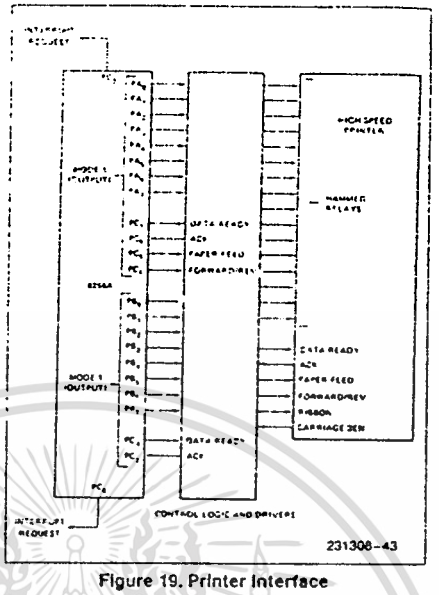


Figure 19. Printer Interface

APPLICATIONS OF THE 8255A

The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 19 through 25 represent a few examples of typical applications of the 8255A.

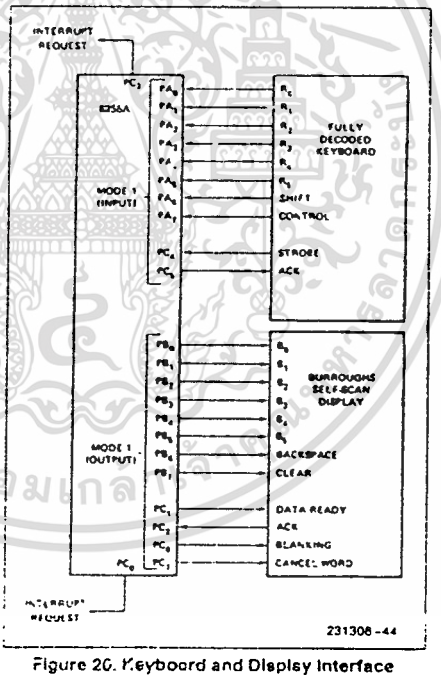


Figure 20. Keyboard and Display Interface

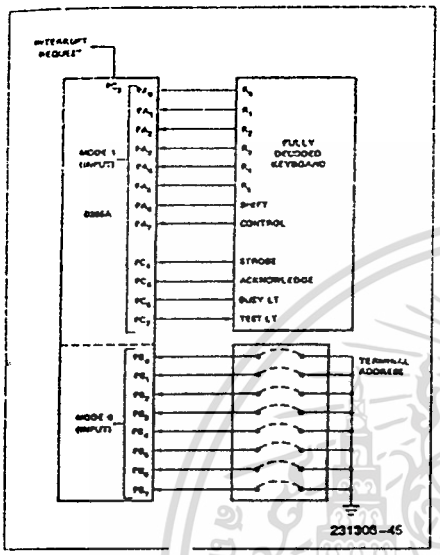


Figure 21. Keyboard and Terminal Address Interface

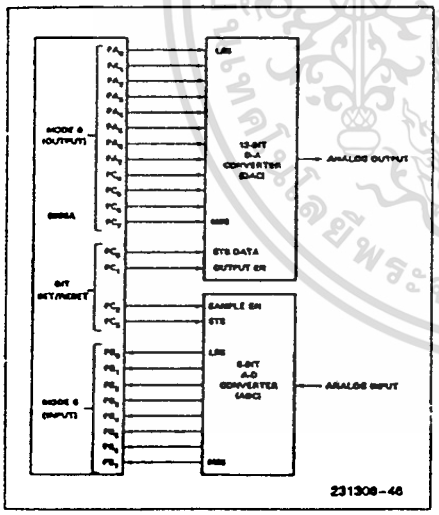


Figure 22. Digital to Analog, Analog to Digital

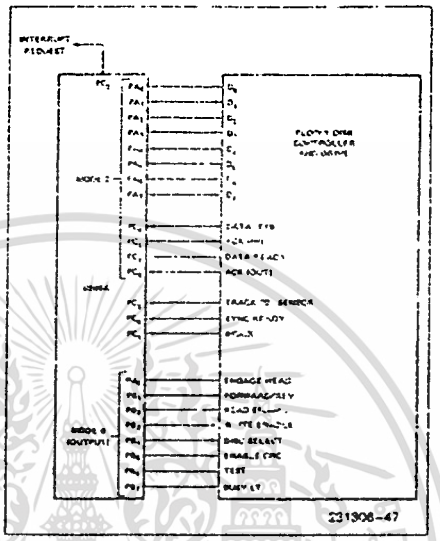


Figure 23. Basic Floppy Disk Interface

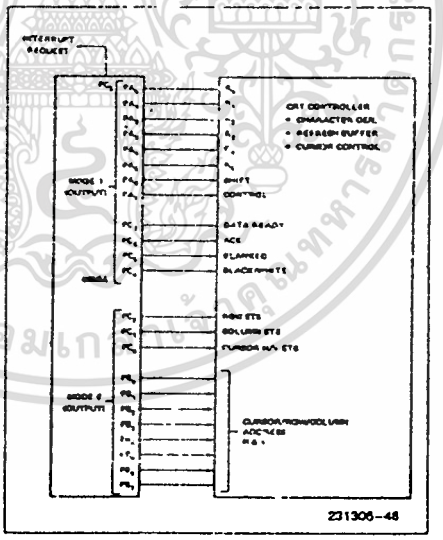


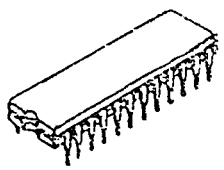
Figure 24. Basic CRT Controller Interface

2048-word×8-bit High Speed Static CMOS RAM

FEATURES

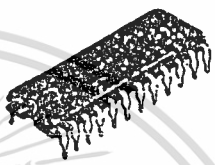
- Single 5V Supply and High Density 24 Pin Package
- High speed: Fast Access Time 120ns/150ns/200ns (max.)
- Low Power Standby and Low Power Operation Standby: 100μW (typ.) Operation: 180mW (typ.)
- Completely Static RAM: No clock or Timing Strobe Required
- Directly TTL Compatible: All Input and Output
- Pin Out Compatible with Standard 16K EPROM/MASK ROM
- Equal Access and Cycle Time

HM6116-2, HM6116-3, HM6116-4

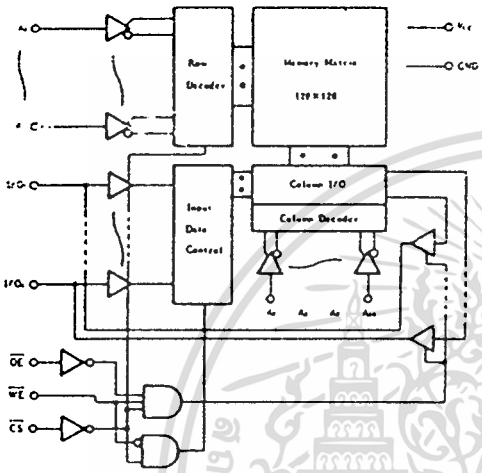


(DG-24)

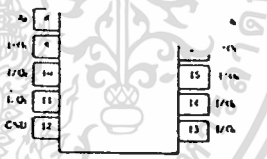
HM6116P-2, HM6116P-3, HM6116P-4



FUNCTIONAL BLOCK DIAGRAM



PIN A



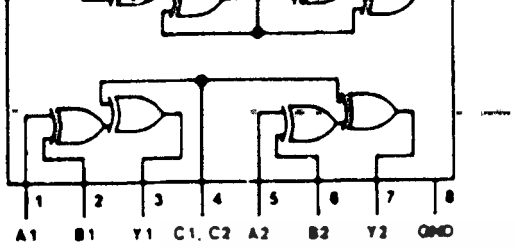
ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Rating	Unit
Voltage on Any Pin Relative to GND	V_T	−0.5* to +7.0	V
Operating Temperature	T_{opr}	0 to +70	°C
Storage Temperature (Plastic)	T_{stg}	−55 to +125	°C
Storage Temperature (Ceramic)	T_{stg}	−65 to +150	°C
Temperature Under Bias	T_{bias}	−10 to +85	°C
Power Dissipation	P_T	1.0	W

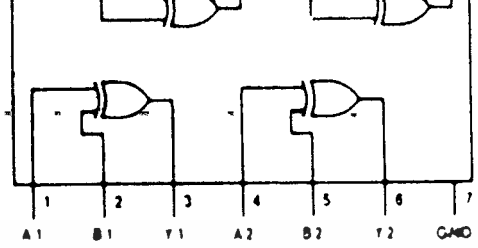
* Pulse Width 50ns : −1.5 V

TRUTH TABLE

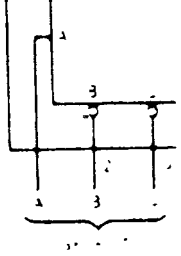
$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	Mode	V_{CC} Current	I/O Pin	Ref. Cycle
H	×	×	Not Selected	I_{SB}, I_{SB1}	High Z	
L	L	H	Read	I_{CC}	Dout	Read Cycle (1)~(3)
L	H	L	Write	I_{CC}	Din	Write Cycle (1)
L	L	L	Write	I_{CC}	Din	Write Cycle (2)



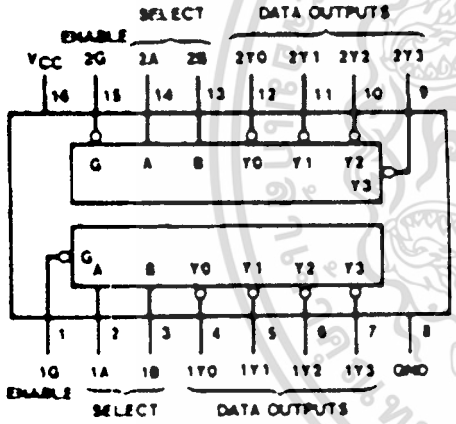
74S135



74130
74LS130
74ALS136
74S136

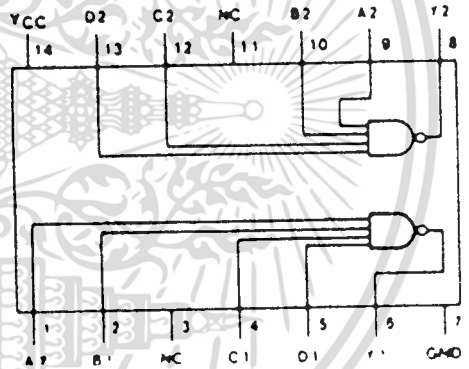


139
Dual 2 to 4 Line Decoder



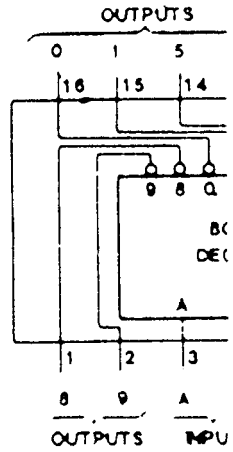
74LS139 74HC139
74S139 74HCT139
74ALS139 40H139
74AS139
74F139

140
Dual 4 Input NAND Line Driver

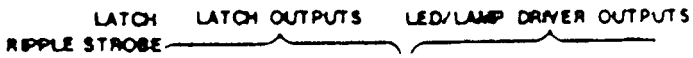


74S140

141
BCD to Decimal

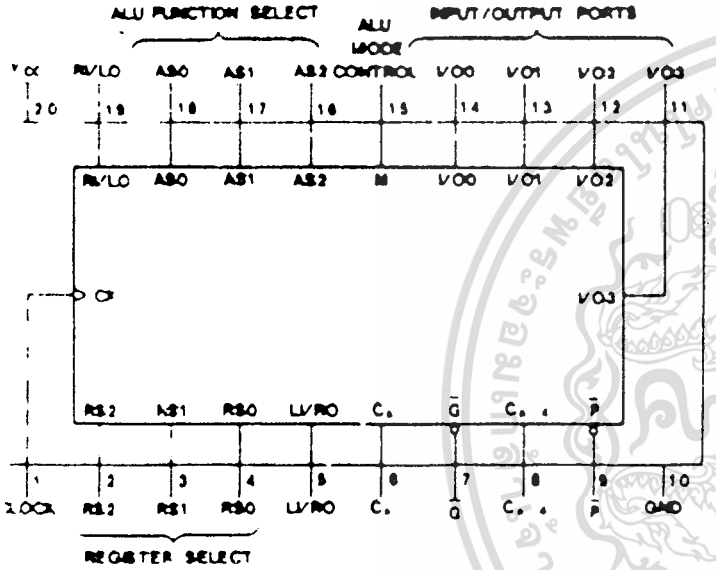


143
BCD Counter Latch, 7 Segment Decoder / Driver



144
BCD Counter L

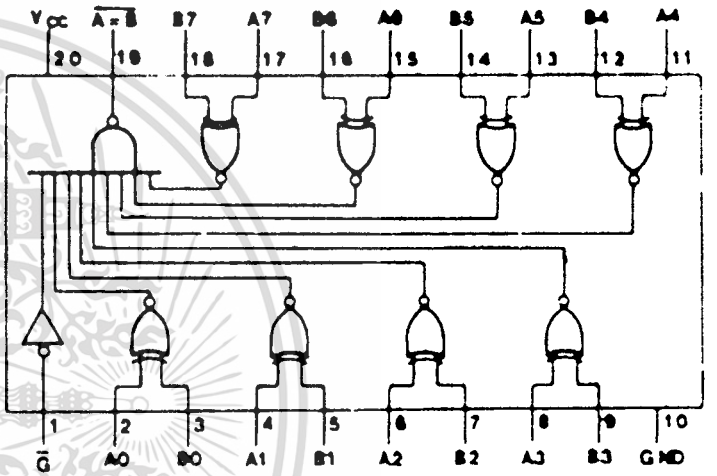
: Parallel Binary Accumulator



74LS681

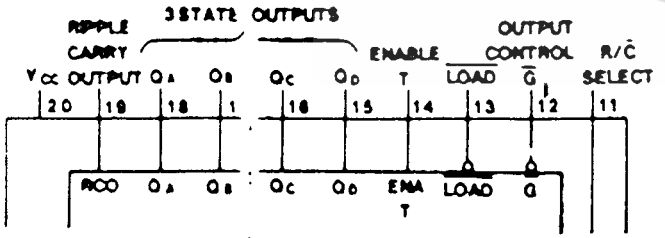
688

8 Bit Equal to Comparator with OC



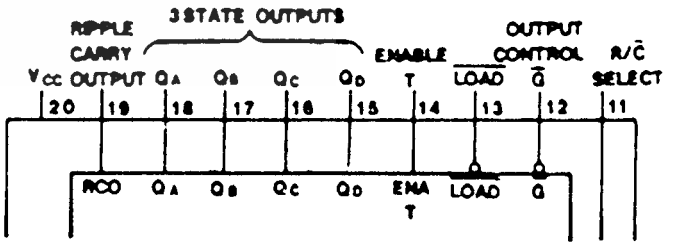
74LS688 74HC688
74ALS688 74HCT688

Synchronous Counter



691

Binary Synchronous Counter

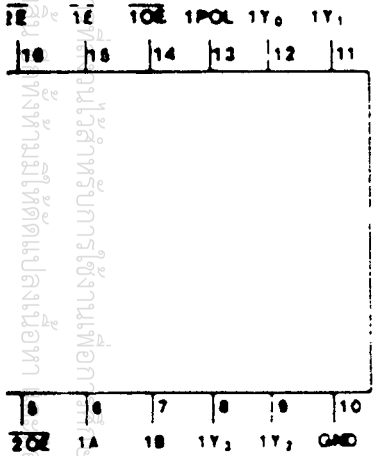


74HC534
74HCT534

74LS537 25LS2537
74ALS537
74F537

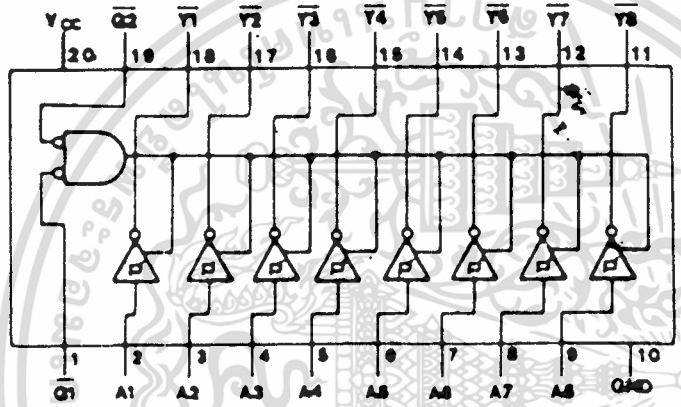
74LS538 25LS2538
74ALS538
74F538

ler (3 state)



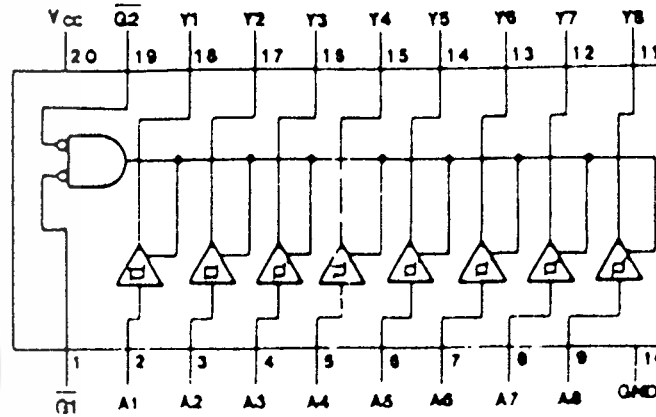
25LS2539

540
Octal 3 state Bus Buffer (Inverted)



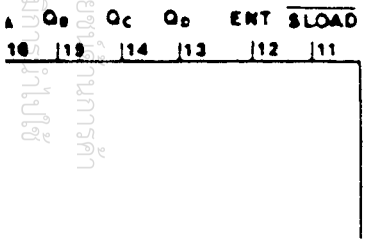
74LS540 74HC540
74ALS540 74HCT540

541
Octal 3 state Bus Buffer

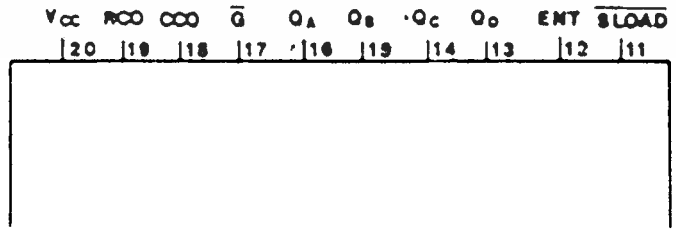


74LS541 74HC541
74ALS541 74HCT541

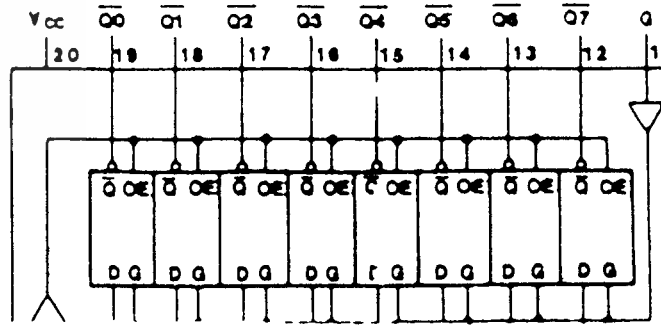
Decade Counter (3 state)



561
Synchronous Presettable Decade Counter (3 state)



563
Octal 3 state Transparent Latch (Inverted)

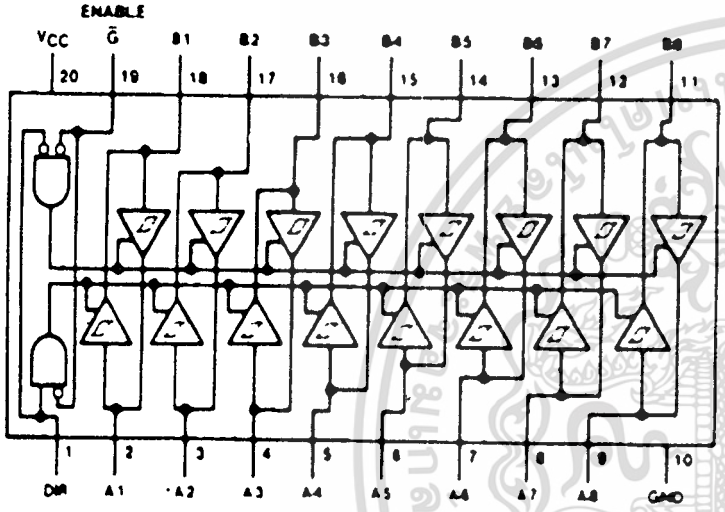


74ALS241 74HC241
74S241 74HCT241
74F241 40H241
74ALS1241

74F242 40H242
74S242
74AS242
74ALS1242

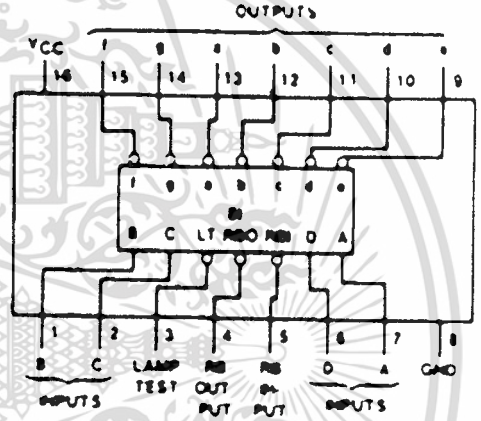
74F243
74S243
74AS243

245
Octal 3 state Bus Transceiver



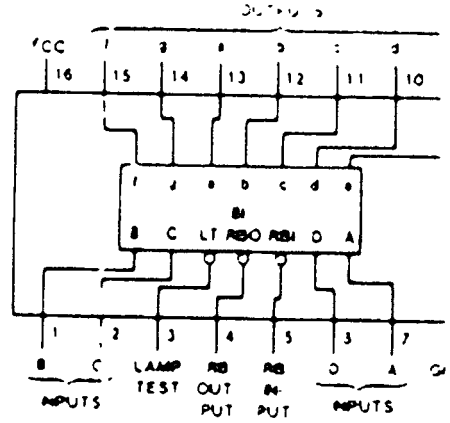
74LS245 74AS245 74HC245
74ALS245 74ALS1245 74HCT245
74F245 40H245

246: BCD to Seven Segment Decoder 30V with OC
247: BCD to Seven Segment Decoder 15V with OC



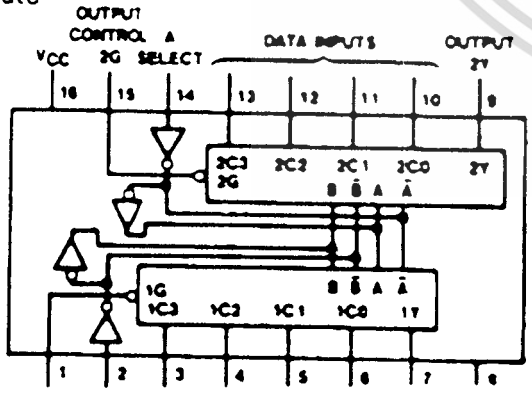
74246~247
74LS247

248: BCD to Seven Segment 24 V pullup with OC
249: BCD to Seven Segment 15V with OC

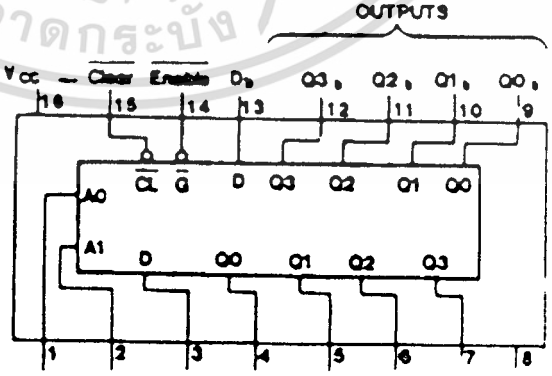


74248~249
74LS248 249

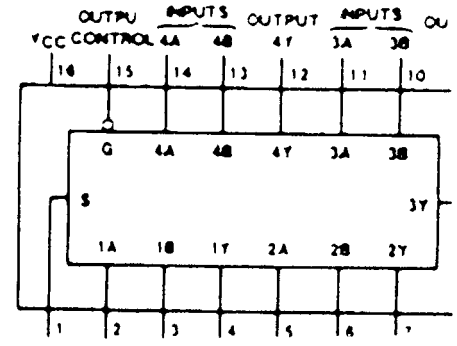
253
Dual 4 to 1 Line Data Selector 3 State

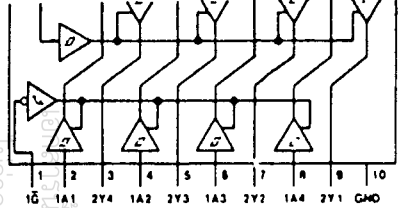


256
Dual 4 Bit Addressable Latch

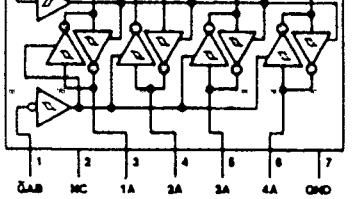


257
Quad 2 to 1 Data Selector

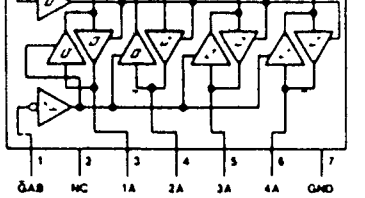




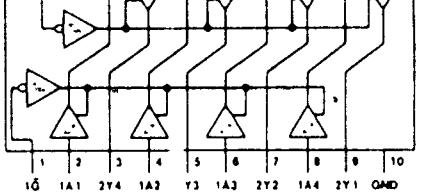
74LS241 74AS241
74ALS241 74HC241
74S241 74HCT241
74F241 40H241
74ALS1241



74LS242 74HC242
74ALS242 74HCT242
74F242 40H242
74S242
74AS242
74ALS1242

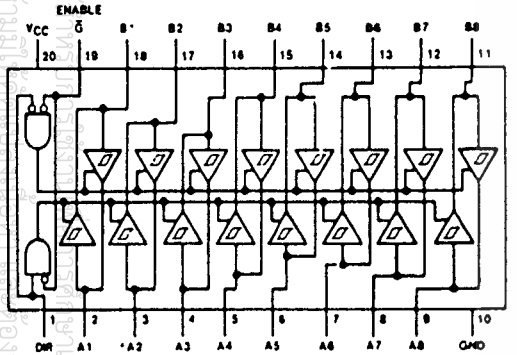


74LS243 74HC243
74ALS243 74HCT243
74F243 40H243
74S243
74AS243
74ALS1243



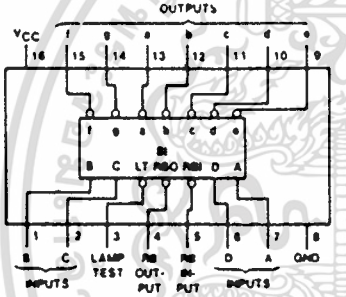
74LS244 74HC244
74ALS244 74HCT244
74S244 40H244
74F244
74AS244
74ALS1244

245
Octal 3 state Bus Transceiver



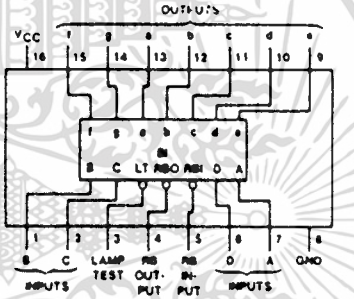
74LS245 74AS245 74HC245
74ALS245 74ALS1245 74HCT245
74F245 40H245

246: BCD to Seven Segment Decoder 30V with OC
247: BCD to Seven Segment Decoder 15V with OC



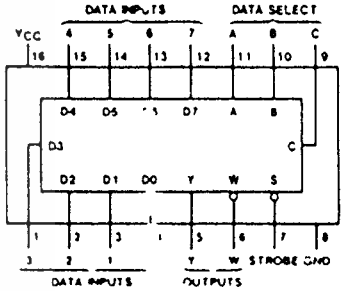
74246~247
74LS247

248: BCD to Seven Segment Decoder 2kΩ pullup with OC
249: BCD to Seven Segment Decoder 5.5V with OC



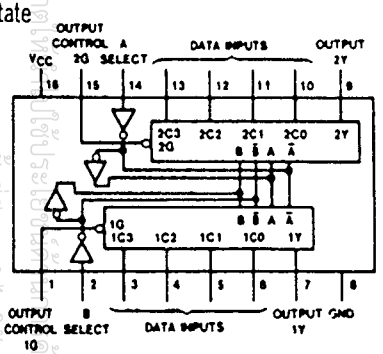
74248~249
74LS248~249

251
8 to 1 Line Data Selector 3 State



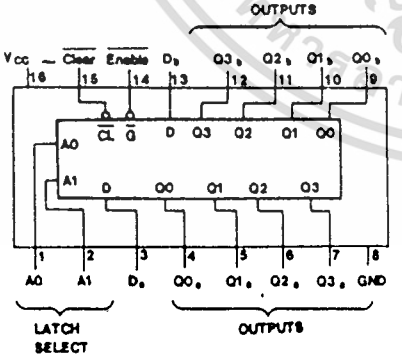
74241 74ALS251
74LS25 74AS251
74S251 74HC251
74F251 74HCT251

253
Dual 4 to 1 Line Data Selector 3 State



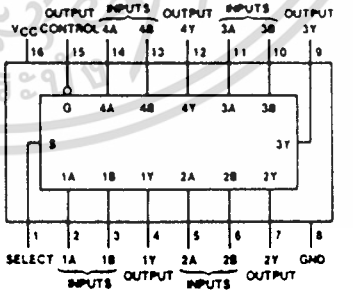
74253 74ALS253 74HC253
74LS253 74AS253 74HCT253
74S253
74F253

256
Dual 4 Bit Addressable Latch



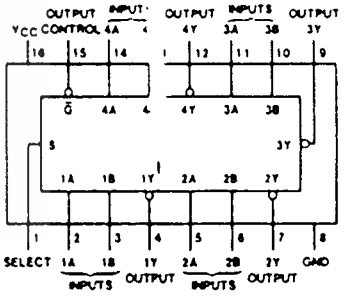
74LS256
74F256

257
Quad 2 to 1 Data Selector 3 State



74257 74ALS257 74HC257
74LS257 74AS257 74HCT257
74S257
74F257

258
Quad 2 to 1 Data Selector 3 State



74LS258 74HC258
74S258
74F258
74ALS258
74AS258

บรรณานุกรม

1. ทฤษฎีและการประยุกต์ใช้ไมโครโปรเซสเซอร์ Z-80
   ยืน ภู่วรรณ

2. เซมิคอนดัคเตอร์อิเล็กทรอนิกส์ ฉบับที่ 146
   โครงงาน เรื่อง บอร์ดควบคุมสเต็ปปิ้งมอเตอร์ด้วยเครื่องคอมพิวเตอร์
   บ. ซีเอ็ดยูเคชั่น

3. คู่มือการทดลอง ET - HARDWARE LAB
   บ. อีทีที จำกัด

4. การอินเตอร์เฟส IBM / PC
   ธานินทร์ ถาวรศาสนวงศ์
   ทินกร ดุ๊ก
   B.Eng., (KMIT - LADKRABANG)

5. การเขียนโปรแกรมและประมวลผลด้วยเทอร์โบปาสคาล
   นุกูล กระจาย