การควบคุมมอเตอร์ 3 ทิศทางโดยใช้คอมพิวเตอร์บุคคล

3 DIMENSION PERSONAL COMPUTER CONTROLLER

โดย

ชาญชัย อนุกูลประเสริฐ

บัณฑิต วิเชียรเขต

ศักดา สุขใย

สุชาติ ภิญโญวรภาค

ปริญญานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีอิเลคทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

หัวข้อปริญญานิพนธ์ การควบคุมมอเตอร์สามทิศทางโดยใช้คอมพิวเตอร์บุคคล

โดย

นายชาญชัย อนุกูลประเสริฐ เลขประจำตัว 34131208

นายบัณฑิต วิเชียรเขต เลขประจำตัว 34131216

นายศักดา สุขใย เลขประจำตัว 34131229

นายสุชาติ ภิญโญวรภาค เลขประจำตัว 34131237

อาจารย์ที่ปรึกษา ดร.ไพศาล นาคพิพัฒน์

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2536

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังอนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

---------- ประธานกรรมการ
( )

---------- กรรมการ
( )

---------- กรรมการ
( )

---------- กรรมการ
( )

---------- กรรมการ
( )

การควบคุมมอเตอร์มอเตอร์สามทิศทางโดยใช้คอมพิวเตอร์บุคคล

3 DIMENSION PERSONAL COMPUTER CONTROLLER

โดย นายชาญชัย อนุกูลประเสริฐ
นายบัณฑิต วิเชียรเขต
นายศักดา สุขใย
นายสุชาติ ภิญโญวรภาค

อาจารย์ที่ปรึกษา ดร.ไพศาล นาคพิพัฒน์

บทคัดย่อ

3 Dimension Personal Computer Controller เป็นโครงงานที่จัดสร้างขึ้นมาเพื่อเป็นแขนกลแบบอัตโนมัติสามารถนำไปประยุกต์ใช้ในงานอุตสาหกรรมต่างๆได้เป็นอย่างดี แขนกลที่ใช้เป็นการเคลื่อนไหวในระบบพิกัด คาร์ทีเซียน ตามแนวแกน X แกน Y และ แกน Z ใช้ Personal Computer เป็นตัวสั่งงานผ่านอุปกรณ์ Interface RS 232C มาที่ Micro Controller เป็นตัวควบคุมการเคลื่อนที่ของแต่ละแกน และใช้ Stepping Motor ร่วมกับ Screw Shaft เป็นตัวขับด้าน Mechanic

ABSTRACT

3 Dimension personal computer controller is the robotic arms project which we can apply to be used in industrial. The robotic arms are move in cartesian coordinate x-axis , y-axis and z-axis.We used Personal Computer to send the movement command to micro controller which is a movement controller, used RS232C serial interface for transfer instruction command to micro controller and we used Stepping Moter which is working with the Screw Shaft are Mechanical driver.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้ ได้รับความกรุณาเอื้อเฟื้อจากบุคคลต่าง ๆ ในการให้คำแนะนำ ให้ข้อมูลและอำนวยความสะดวก ดังมีรายนามดังต่อไปนี้

1. ดร.ไพศาล นาคพิพัฒน์	อาจารย์ประจำคณะวิศวกรรมศาสตร์ และอาจารย์ที่ปรึกษาในการจัดทำปริญญานิพนธ์
2. บริษัท NS Electronice Bangkok	ให้การสนับสนุน อุปกรณ์บางส่วน ในการทำโครงงานนี้

สารบัญ

วัตถุประสงค์ และขอบเขตของโครงงาน

1. เครื่อง 3 Dimension Personal Computer Controller จะมีลักษณะเป็น แขนกล แบบแกน X,Y,Z ที่สั่งงานโดย Personal Computer ใช้ Micro Controller เป็นตัวควบคุมการเคลื่อนไหวและใช้ RS232C เป็นอุปกรณ์ Interfacc

2. สามารถสั่งให้แขนกลเคลื่อนไหว ตามแนวแกน X , แกน Y และ แกน Z โดยใช้ภาษา C ในการแปลคำสั่ง ผ่านทาง Personal Computer ส่งมาให้ Micro Controller

3. สามารถแสดงตำแหน่ง ในการเคลื่อนไหวบนจอภาพ ของ Personal Computer ได้

ลำดับขั้นตอน และวิธีดำเนินงาน

1. ศึกษา การทำงานของ Stepping Motor
2. ศึกษาการทำงานของ Z-80 Micro Controller และ RS232C Serial Interface และอุปกรณ์แต่ละตัวที่จะนำมาออกแบบวงจรควบคุม
3. ศึกษาการติดต่อระหว่าง Personal Computer กับ Micro Controller โดยผ่านทาง RS232 Serial Interface
4. ออกแบบ และสร้างอุปกรณ์ด้าน Machanic
5. ศึกษาการเขียน Program ภาษา C และ ภาษา Assembly ในการควบคุม Stepping Motor
6. ทดสอบการทำงานของเครื่องโดยสื่อทาง Personal Computer และใช้ Program ใน Micro-Controller

ประโยชน์และผลที่คาดว่าจะได้รับจากโครงงาน

1. เป็นพื้นฐานในการสร้างแขนกล ที่ใช้ในงานอุตสาหกรรม
2. เป็นประโยชน์ต่องานอุตสาหกรรม เพราะ 3 Dimension Personal Computer Controller สามารถนำไปประยุกต์ใช้งานได้หลายอย่าง

บทที่ 1

บทนำ

อุตสาหกรรมการผลิตในยุคปัจจุบัน ต้องมีการแข่งขันกันอย่างสูงในเรื่อง การเพิ่มผลผลิต การลดต้นทุน การผลิตและการควบคุมคุณภาพ ของผลิตภัณฑ์ การใช้เครื่องจักรแบบอัตโนมัติ เป็นหนทางหนึ่งที่สามารถช่วยใน บรรลุวัตถุประสงค์เหล่านี้ได้ ในปัจจุบันหุ่นยนต์ได้ถูกพัฒนาขึ้นมาเพื่อใช้ในโรงงานผลิตรถยนต์ เครื่องจักรจนแม้ แต่การผลิต อุปกรณ์ทางอิเลคทรอนิคส์ เข้ามาช่วยในขบวนการผลิต

กิจกรรม การผลิต ส่วนใหญ่ จะเป็นการ หยิบ,จับ การเคลื่อนที่ในบริเวณ ที่จำกัดซึ่งเป็นงานที่ซ้ำซาก น่าเบื่อ ถ้าเรายังคงใช้แรงงานคนจะทำให้ประสิทธิภาพในการทำงานลดลง โอกาศผิดพลาดจะเกิดมากขึ้น แต่หุ่นยนต์ หรือ เครื่องจักรอัตโนมัติจะทำให้ ผลงานออกมาดีกว่า เพราะเครื่องจักรเป็นสิ่งไม่มีชีวิต สามารถทำงานได้โดยไม่มีความเบื่อหน่าย ไม่ต้องการเวลาพักผ่อน ยกเว้น การซ่อมบำรุง ตามปกติ และเครื่องจักรสามารถทำงานบางอย่าง ในสิ่งที่คนเราทำไม่ได้ เช่น

- งานที่อันตราย หรืองานที่ มนุษย์ทำไม่สะดวก เช่น บริเวณที่มีรังสี สารพิษ ฯลฯ
- งานที่ต้องทำซ้ำบ่อย ๆ ซึ่งหากมนุษย์ทำอาจผิดพลาดได้ง่าย เพราะเป็นงานน่าเบื่อ
- ใช้ในงานที่ต้องการความสามารถเฉพาะ เช่น งานยกของหนัก งานต้องการความละเอียด
- ใช้ในงานที่ต้องทำต่อเนื่อง

ข้อดีของการใช้หุ่นยนต์ก็มีมากมายอาทิเช่น

- ความปลอดภัย งานที่อันตรายสามารถใช้หุ่นยนต์ทำได้
- ความสม่ำเสมอของงานดี เพราะเมื่อ Program แล้วก็จะทำตามขั้นตอนด้วยความเร็วที่กำหนด
- ใช้พลังงานน้อย เพราะการใช้หุ่นยนต์อาจไม่ต้องสนใจระบบถ่ายเทอากาศ

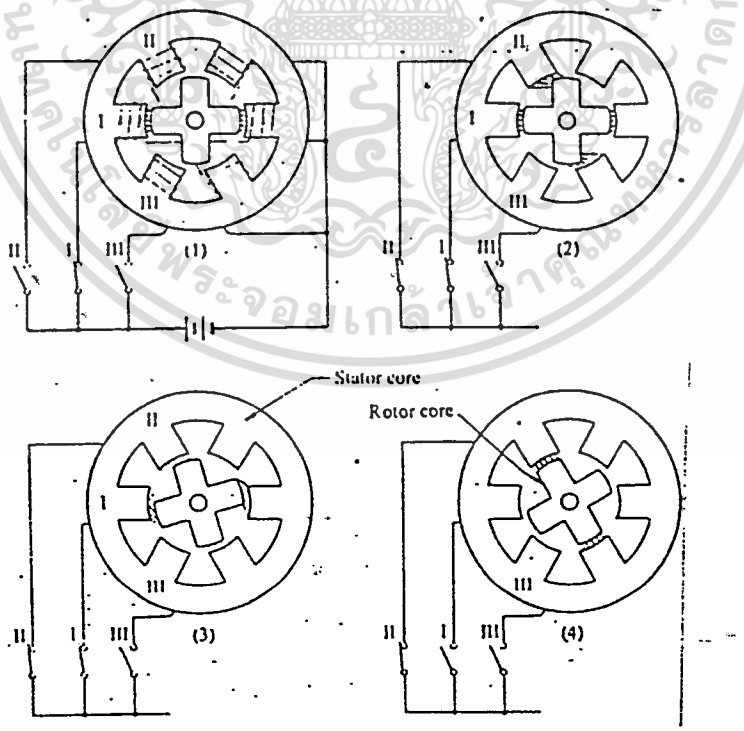
โครงงานนี้เป็นส่วนหนึ่งของการสร้างหุ่นยนต์หรือเครื่องจักรอัตโนมัติที่สามารถนำไปประยุกต์ใช้ในงานอุตสาหกรรม ดังที่กล่าวมาในข้างต้นได้

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีสเตปปิ้งมอเตอร์

สเตปปิ้งมอเตอร์และลักษณะโดยทั่วไป

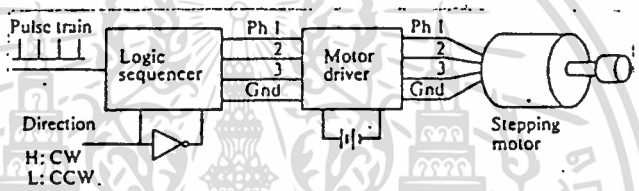
การทำงานของสเตปปิ้งมอเตอร์ดูจากรูป 2.1 เป็นสเตปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์ (Variable Reluctance) แกนเหล็กสเตเตอร์(Stator core) มีซี่ฟัน (Teeth) 6 ซี่ขณะที่โรเตอร์(Rotor) มีฟัน 4 ซี่ทั้งโรเตอร์และสเตเตอร์เป็นเหล็กอ่อนขดลวด (Coil) 3ชุดถูกต่ออยู่ดังรูปแต่ละชุดขดลวดมี 2 ขดลวดต่ออนุกรมกัน เรียกแต่ละชุดว่าเฟส (Phase) และผลจากการต่อแบบนี้เรียกว่า มอเตอร์ 3เฟส(3 PhaseMotor) กระแสถูกจ่ายไปยังขดลวดแต่ละขดโดยผ่านสวิทช์ I,II และ III ในสภาวะ(1) ขดลวดของเฟส 1 ได้รับกระแสโดยผ่านสวิทช์ I หรือเฟส 1 ถูกกระตุ้นเส้นแรงแม่เหล็กที่เกิดขึ้นในช่องอากาศ (Air Gap) เกิดขึ้นเนื่องจากการกระตุ้นซึ่งแสดงด้วยลูกศรในสภาวะนี้สเตเตอร์ 2ขั้ว ของเฟส 1 จะอยู่ในแนวเดียวกับ 2 ซี่ที่อยู่ตรงข้ามกันของโรเตอร์นี่เป็นสภาวะสมดุลย์ซึ่งอยู่ในเทอมของไดนามิค (Dynamics) เมื่อสวิทช์ II ปิดเพื่อกระตุ้นเฟส 2 กับเฟส1 เส้นแรงแม่เหล็กจะถูกสร้างขึ้นที่ขั้วของสเตเตอร์ของ



รูปที่ 2.1 วงจรกระตุ้นสเตปปิ้งมอเตอร์ 3 Phase

เฟส2ในลักษณะซึ่งแสดงในสภาวะ(2)ทอร์ค(Torgue)ทิศทางทวนเข็มนาฬิกาจะถูกสร้างขึ้นจากความเครียด (Tension) ในฟลักซ์แม่เหล็กเอียงไปยังแกนมอเตอร์ที่อยู่ใกล้หลังจากนั้นโรเตอร์จะมาอยู่ในสภาวะ (3)

ดังนั้นโรเตอร์จะหมุนไปด้วยมุมการเปลี่ยนแปลงที่คงที่ซึ่งเรียกว่ามุมสเตป(StepAngle)ในที่นี้คือ15 องศาขณะที่สวิทช์จะมีการเปลี่ยนแปลงอีกครั้งหนึ่งคือสวิทช์Iจะถูกเปิดเพื่อลดพลังงานในเฟส I โดยโรเตอร์จะหมุนไป 15องศา มาอยู่ในสภาวะ (4) ตำแหน่งมุมของโรเตอร์จะถูกควบคุมโดยการเปิดปิดสวิทช์ถ้าสวิทช์ถูกปิดเปิดเป็นลำดับโรเตอร์จะหมุนไปในลักษณะที่เป็นสเตปความเร็วเฉลี่ยจะสามารถควบคุมได้ด้วยการเปิด-ปิดสวิทช์ดังแผนภาพรูปที่ 2.1



รูปที่ 2.2 ไดอะแกรมของระบบขับสเตปปิ้งมอเตอร์

จากที่กล่าวพอที่จะกล่าวถึงคุณสมบัติของสเตปปิ้งมอเตอร์ได้ว่า

1. การหมุนของมอเตอร์จะเป็นสเตป (เป็นขั้น ๆ) สเตปละกี่องศาขึ้นอยู่กับชนิดของมอเตอร์
2. ความเร็วในการหมุนขึ้นอยู่กับสัญญาณพลัซที่ให้เข้ามาทางอินพุทของมอเตอร์ (ความถี่)
3. ความผิดพลาดในการหมุน 1 สเตปมีค่าน้อยมาก แต่ต้องจำกัดอยู่ในความเร็วที่พอดีด้วย
4. คุณสมบัติของการตอบสนองสัญญาณในขณะที่มอเตอร์เริ่มทำงานและหยุดทำงานดีมาก
5. เนื่องจากไม่มีแปลงถ่าน(Commutator)เหมือนมอเตอร์ในระบบดีซีดังนั้นจึงมีความแน่นอนในการทำงานสูง
6. แหล่งจ่ายแรงดันไฟที่ใช้ในการขับมอเตอร์มีค่าไม่มาก
7. ไม่ทำให้เกิดเสียงรบกวนและการสั่นได้ง่าย
8. ทำงานแบบ โอเพนลูป (Open loop)

สเตปปิ้งมอเตอร์ที่ใช้กันส่วนมากมี 3 ชนิด

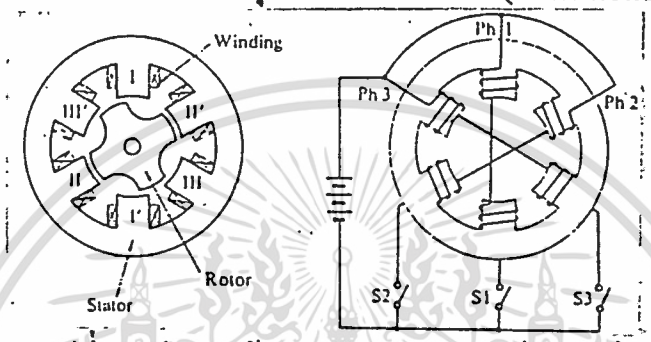
1. **วาริเอเบิลริลักแทนซ์สเตปปิ้งมอเตอร์** (Variable Stepping Motor)

2. สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร (Permanent Magnetic Stepping Motor)

3. สเตปปิ้งมอเตอร์แบบไฮบริด (Hybrid Stepping Motor

2.1.1 วาริเอเบิลรีลัคแทนซ์สเตปปิ้งมอเตอร์

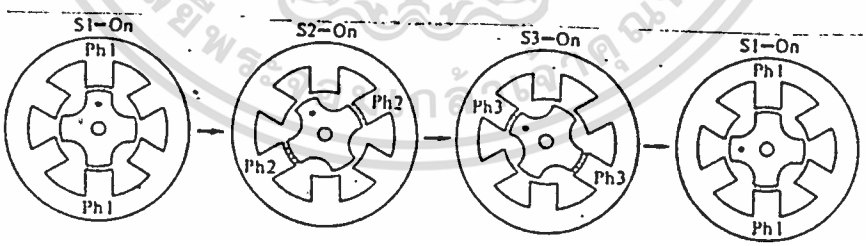
มอเตอร์แบบนี้โรเตอร์ทำด้วยเหล็กอ่อนซึ่งค่าซึมซาบแม่เหล็ก(Permeability)สูงและสามารถให้ฟลักซ์แม่เหล็กผ่านได้มากโดยโรเตอร์จะติดอยู่กับแกนมอเตอร์และสเตเตอร์ติดอยู่กับโครงของตัวมอเตอร์จากรูป 2.3 เป็นภาพตัดขวางของสเตปปิ้งมอเตอร์แบบนี้ซึ่งเป็นมอเตอร์ 3 เฟสมีซี่ฟัน6ซี่ฟันของสเตเตอร์จะอยู่ตรงข้ามจะต่อกันเป็นอนุกรมหรือขนานก็ได้ (ในที่นี้ต่อแบบอนุกรม)

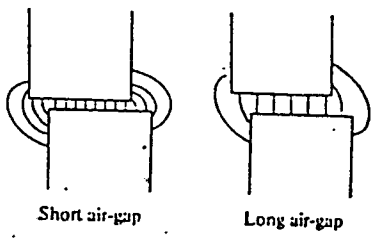


รูปที่ 2.3 โครงสร้างวาริเอเบิลรีลัคแทนซ์มอเตอร์

เราจะเห็นได้ว่าฟันของสเตเตอร์ 2 ซี่ที่มีเฟสเดียวกันจะมีขั้วแม่เหล็กตรงข้ามกันและกันดังแสดงดังรูป สมมุติว่าฟัน I,II และ III มีขั้วเป็นขั้วเหนือ ฟัน I,II และ III จะเป็น ขั้วใต้เมื่อถูกกระตุ้น

กระแสแต่ละเฟสจะถูกกระตุ้น ฟลักซ์แม่เหล็กก็จะเกิดดังรูป 2.4 ฟันของโรเตอร์ก็จะมีตำแหน่งในแนวเดียวกันกับฟันของสเตเตอร์ ซึ่งจะมีผลให้แมกเนติกรีลัคแทนซ์ (Megnetic Reluctance) น้อยที่สุด สภาวะนี้คือตำแหน่งสมดุลย์





รูปที่ 2.4 ฟลักซ์แม่เหล็กที่เกิดขึ้น

โครงสร้างเบื้องต้น ของมอเตอร์แบบนี้ จะมีลักษณะดังนี้

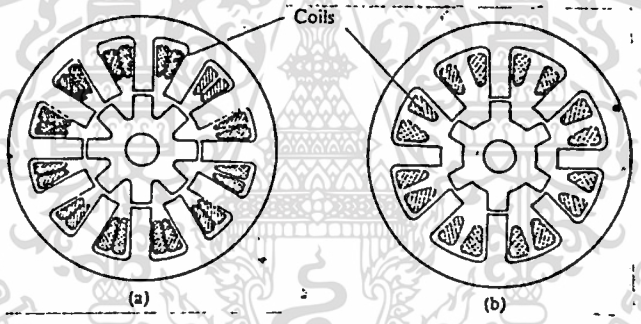
1.ช่องว่างอากาศควรจะเล็กที่สุดเท่าที่จะเป็นไปได้ช่องว่างอากาศระหว่างฟันของโรเตอร์กับฟันของสเตเตอร์ควรมีค่าความห่างกันน้อยมาก เพื่อที่จะได้ทอร์คสูง และตำแหน่งที่แน่นอนขึ้น

2. สำหรับมุมสเตปเล็ก ๆจากรูป2.5a แสดง 3เฟสมอเตอร์ที่สเตเตอร์มีฟัน12ซี่ และฟันมีโรเตอร์ 8ซี่รูป2.5b เป็นรูปที่ 4เฟส มอเตอร์ที่สเตเตอร์มีฟัน8ซี่และโรเตอร์มีฟัน6ซี่ซึ่งทั้งสองรูปนี้มีมุมสเตปเท่ากับ 15 องศา

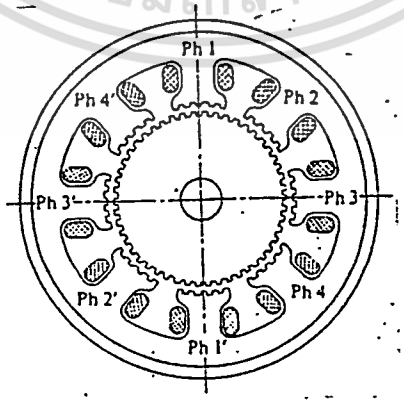
ความสัมพันธ์ระหว่างมุมสเตป θS,จำนวนเฟส m โดยจำนวนฟันของโรเตอร์Nrและจำนวนสเตปใน 1 รอบ S จะหาได้โดย

$$S = 360/\theta\ S = m\ (Nr)$$

นั่นคือ เราสามารถลดมุมสเตปลงได้โดยเพิ่มจำนวนฟันบนโรเตอร์ ดังแสดงในรูป 2.6



รูปที่ 2.5 โรเตอร์และสเตเตอร์

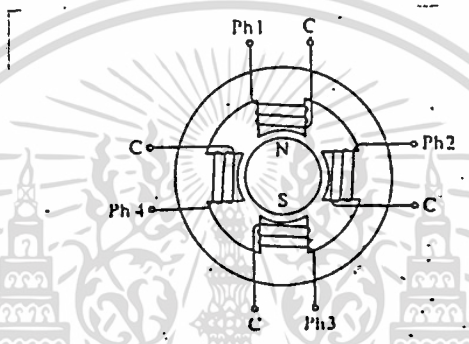


รูป 2.6 หน้าตัดของ 4 เฟส มอเตอร์มีฟันโรเตอร์ 50 ซี่ มุมสเตป 1.8 องศา

2.1.2 สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร

สเตปปิ้งมอเตอร์แบบนี้ใช้แม่เหล็กแบบถาวรรูปที่ 2.7 เป็นตัวอย่างของสเตปปิ้งมอเตอร์แบบถาวรแบบ4เฟส โรเตอร์เป็นทรงกระบอกสเตเตอร์มีฟัน4ซี่โดยที่แต่ละซี่มีขดลวดพันรอบถ้าจำนวนซี่บนสเตเตอร์และขั้วแม่เหล็กบนโรเตอร์เพิ่มขึ้นเป็น 2 เท่ามุมแต่ละสเตปลดลงจากเดิมครึ่งหนึ่งดังนั้นเพื่อที่จะลดมุมสเตปลงไปอีกในสเตปปิ้งมอเตอร์แบบแม่เหล็กถาวรจะต้องเพิ่มจำนวนแม่เหล็กและซี่ฟัน อย่างไรก็ตาม ขั้วแม่เหล็กที่จะเพิ่มขึ้นได้นั้นมีจำนวนจำกัด

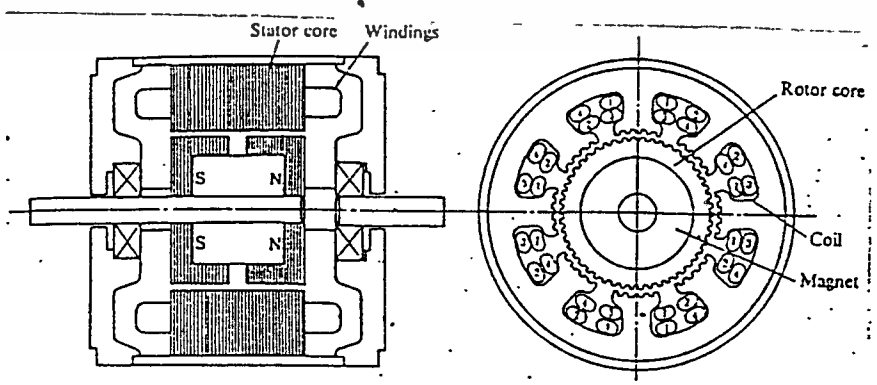
ลักษณะทั่วไปของมอเตอร์แบบนี้ก็คือโรเตอร์จะถูกยึดอยู่กับที่แม้ว่าไม่มีการกระตุ้นเฟสลักษณะเช่นนี้เรียกว่า ดีเทนท์ แมคคานิคซึ่ม (DETENT MECANISM)



รูปที่ 2.7 โครงสร้างสเตปปิ้งมอเตอร์ 4 เฟส

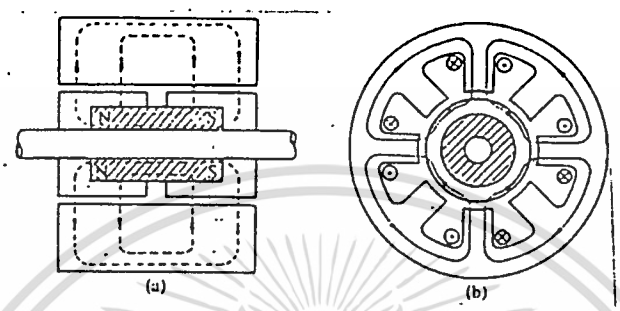
2.1.3 สเตปปิ้งมอเตอร์แบบไฮบริด

เป็นสเตปปิ้งมอเตอร์แบบหนึ่งที่มีโรเตอร์ เป็นแบบแม่เหล็กถาวรการใช้ชื่อไฮบริดได้มาจากการรวมหลักสำคัญของมอเตอร์แบบแม่เหล็กถาวรและแบบวาริเอเบิลรีลัคแทนซ์โครงสร้างแกนของสเตเตอร์จะคล้ายกับแบบวาริเอเบิลรีลัคแทนซ์แต่การพันและการต่อขดลวดจะต่างจากแบบวาริเอเบิลรีลักแทนซ์มอเตอร์ชุดขดลวด 4 เฟสที่แตกต่างกันจะถูกพันบนขั้วเดียวกัน ดังรูป 2.8 เพราะฉะนั้นขั้วหนึ่งจะไม่มีเพียงเฟสเดียว ขดลวด 2 ขดจะถูกพันเป็นขั้วเดียวกันแบบไปพิลา (Bifilar winding) ซึ่งจะทำให้ขั้วแม่เหล็กต่างกันขณะมีการกระตุ้น



รูปที่ 2.8 โครงสร้างของสเตปปิ้งมอเตอร์แบบไฮบริด

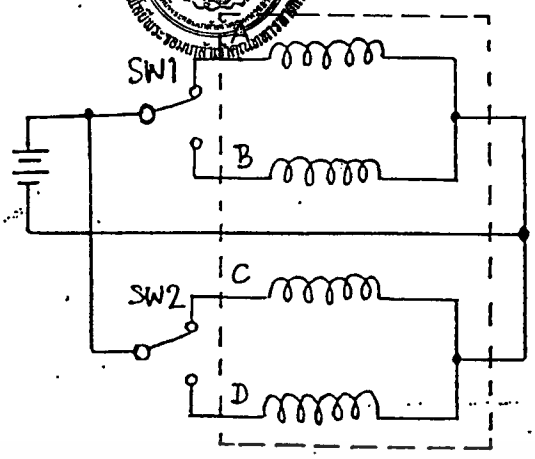
ลักษณะที่สำคัญอีกอย่างหนึ่งของไฮบริดมอเตอร์คือ โรเตอร์นั้นจะเป็นแม่เหล็ก รูปร่างทรงกระบอกอยู่ในแกนเหล็กของโรเตอร์ มันถูกทำให้เป็นแม่เหล็กตามขาวเพื่อสร้างสนามขั้วเดียวดังรูปที่ 2.9 แต่ละขั้วของแม่เหล็กจะถูกล้อมรอบด้วยฟันเหล็กอ่อนซึ่งฟันของโรเตอร์ กับสเตเตอร์อยู่เหลื่อมกันอยู่ครึ่งช่วงฟัน



รูปที่ 2.9 การวางแม่เหล็กตามขาวเพื่อสร้างสนามขั้วเดียวกัน

การทำงานพื้นฐาน

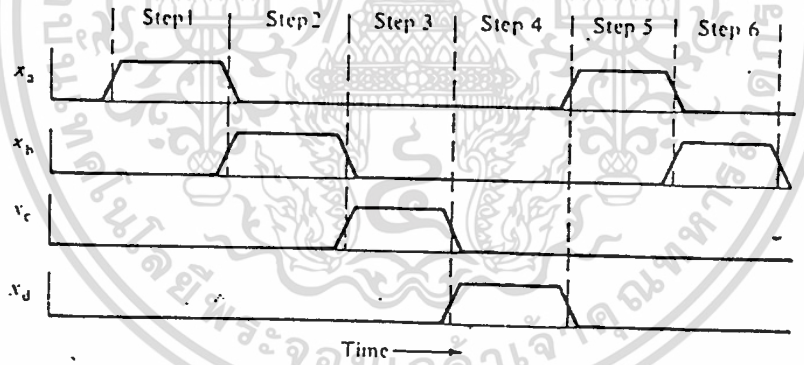
สเตปปิ้งมอเตอร์เป็นมอเตอร์ที่ถูกใช้ทำงานโดยสัญญาณอินพุทที่เป็นพัลซ์ทุกๆสัญญาณพัลซ์ที่ให้เข้ามาจะให้การเปลี่ยนแปลงสถานะสนามแม่เหล็กไฟฟ้าที่เกิดขึ้น และให้การหมุนของมอเตอร์เป็นมุมที่คงที่ซึ่งจะแตกต่างกับการหมุนของมอเตอร์แบบธรรมดาในระบบควบคุมที่ใช้สัญญาณดิจิตอลนี้เช่นในการส่งข้อมูลการควบคุมข้อมูลโดยที่เป็นปริมาณค่าของสัญญาณดิจิตอลทั้งหมดจึงทำให้การควบคุมการทำงานของสเตปปิ้งมอเตอร์โดยตรงได้เป็นอย่างดี การทำงานพื้นฐานของสเตปปิ้งมอเตอร์แสดงในรูปที่ 2.10 ซึ่งแสดงบล็อกของวงจรขับสเตปปิ้งมอเตอร์ ทำให้กระแสไฟดีซีเรียงลำดับเข้าไปตามเฟสต่าง ๆ ของมอเตอร์หมุนไปตามองศาที่กำหนดไว้การที่จะทำให้กระแสไฟเรียงลำดับเข้าไปที่มอเตอร์ได้ทำได้โดยการใช้งานของสวิทช์และเพื่อให้มอเตอร์หมุนจำเป็นต้องมีวงจรขับสเตปปิ้งมอเตอร์ที่จะควบคุมการไหลของกระแสไฟคือสวิทช์1และสวิทช์2ถ้าให้การทำงานของสวิทช์ตามตารางการทำงานที่แสดงไว้ในรูปที่ 2.11 ถ้าให้การทำงานโดยใช้รีเลย์(Relay) หรือมือโยกสเตปมอเตอร์ก็จะหมุนเหมือนกันแต่หมุนช้าๆ เนื่องจากรีเรย์หรือมือโยกให้การเปลี่ยนแปลงของสวิทช์ช้าซึ่งจะทำได้เร็วเท่ากับความถี่สูงสุดของสัญญาณพัลซ์ที่มอเตอร์ทำงานสามารถทำได้(เป็นจำนวนพัลซ์ต่อวินาที)



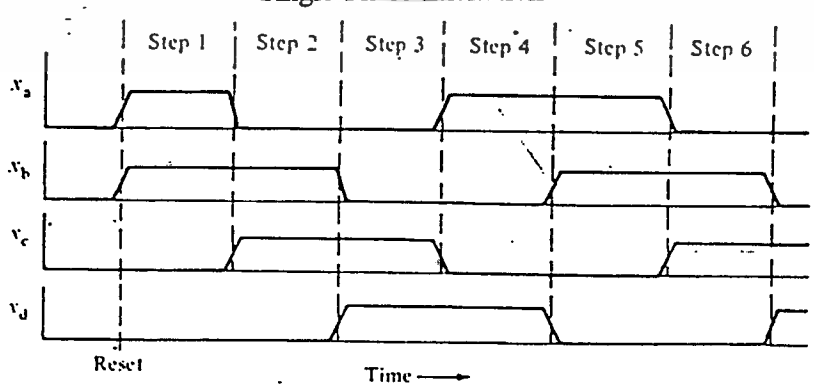
รูปที่ 2.10 การทำงานพื้นฐานของสเตปปิ้งมอเตอร์

วงจรขับมอเตอร์มีหลายชนิด ขึ้นอยู่กับชนิดสเตปปิ้งมอเตอร์ที่มีอยู่ 3 ระบบคือ

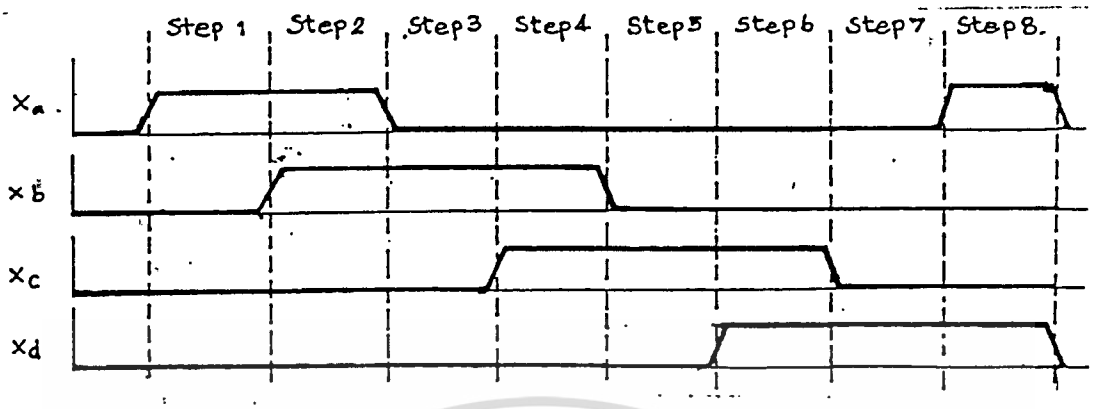
1. ระบบการกระตุ้นสนามแม่เหล็กเฟสเดียว (Single Phase Excitation)
2. ระบบการกระตุ้นสนามแม่เหล็ก 2 เฟส (2-Phase Excitation)
3. ระบบการกระตุ้นสนามแม่เหล็ก 1-2 เฟส (Half-Step Excitation)



Single Phase Excitation



2 - Phase Excitation

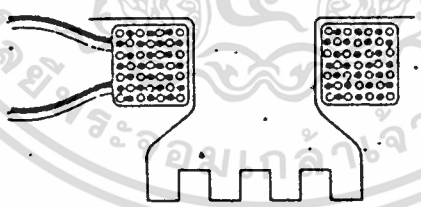


Haft - Step Excitation

รูปที่ 2.11 ชนิดของการขับสเตปปิ้งมอเตอร์

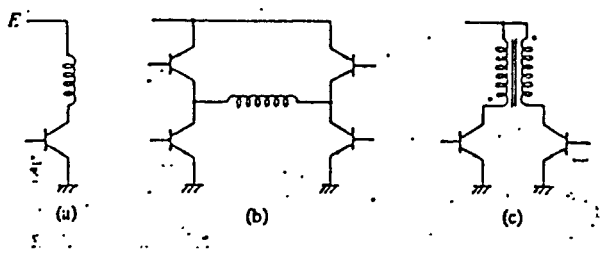
การพันลวดแบบโมโนฟิลาและไบฟิลา (Monofilar and Bifilar Winding)

แบบโมโนฟิลามีการพันลวดเส้นเดียวส่วนแบบไบฟิลาเส้นลวดทั้ง2เส้นนี้จะถูกพันเหมือนกับเส้นเดียวกันขั้วดังรูปที่ 2.12 และขดลวดทั้งสองเส้นนี้จะแยกกันที่ปลายเป็นลักษณะ 2 เส้นแยกกันถ้าขดลวดเป็นของเฟส 1 อีกขดหนึ่งเป็นของเฟส3และทำนองเดียวกันถ้าขดหนึ่งเป็นของเฟส2 หนึ่งของเฟส 4 (กรณีมอเตอร์ 4 เฟส)



รูปที่ 2.12 การพันแบบไบฟิลา

จุดประสงค์ของการพันแบบไบฟิลาก็คือเพื่อที่จะให้พลังงานกับขั้วแม่เหล็กสเตเตอร์โดยการสลับขั้วแม่เหล็ก การกระตุ้นแต่ละเฟสอาจเป็นแบบใดแบบหนึ่งใน3 แบบ ในรูปที่ 2.13



รูปที่ 2.13 การพันแบบโมโนฟิลา

ในวงจรรูปที่ 2.13a เป็นโมโนฟิลา ขั้วแม่เหล็กจะถูกกระตุ้นเป็นขั้วเหนือขั้วใต้เสมอ ซึ่งแสดงว่าไม่สามารถกลับขั้วแม่เหล็กได้การกระตุ้นแบบนี้เป็นการกระตุ้น แบบขั้วเดียว (Unipolar - Exited)

ในวงจรรูปที่ 2.13b ทิศทางของกระแสในขดสามารถสลับได้เนื่องจากเป็น วงจรบริดจ์ (Bridge Exited)

อย่างไรก็ตามจะต้องใช้ ทรานซิสเตอร์ถึง 4 ตัวต่อขดลวด1ขดแบบนี้จะทำให้ทอร์คที่ความเร็วต่ำมาก กว่าแบบไบฟิลา แต่ก็มีโอกาสที่ทรานซิสเตอร์จะพังได้เนื่องจากจัดเวลาผิดพลาด

ในวงจรรูปที่ 2.13c เกี่ยวกับคู่สายไบฟิลา และทรานซิสเตอร์ 2 ตัว โดยทำให้สเตเตอร์ถูกกระตุ้นเป็นขั้วแบบไบฟิลา ก็จะเกิดสนามแม่เหล็กคัปปลิ้ง (Coupling) เมื่อขั้วใดขั้วหนึ่งถูกกระตุ้น ถ้าแทนการพันแบบไบฟิลา ด้วยเส้นลวด 2 เส้น ที่แยกจากกันความแตกต่างของอินดัคแทนซ์ (Inductance) จะปรากฏระหว่าง ขด 2 ขด ทำให้ตำแหน่งผิดไป

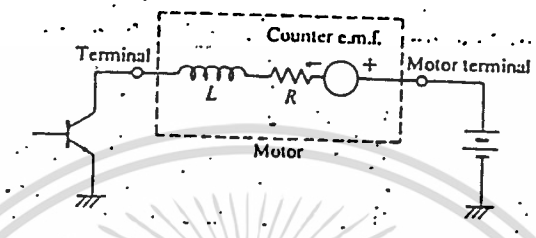
โดยทั่วไปประสิทธิภาพของมอเตอร์ แบบแม่เหล็กถาวรกับแบบไฮบริดที่ใช้แบบฟิลานี้จะได้ประสิทธิภาพดีกว่าแบบ โมโนฟิลา

2.1.4 วงจรขับมอเตอร์ (MOTOR DRIVE CIRCUIT)

จุดประสงค์ของวงจรขับ คือเพื่อที่จะให้ได้โวลท์เตจ และกระแสที่ถูกต้องไปยังมอเตอร์ในช่วงเวลาที่สั้นและลักษณะที่มีประสิทธิภาพ ทิศทางของกระแสในขดลวดของมอเตอร์แบบแม่เหล็กถาวรและแบบไฮบริดมีความสำคัญเพราะจะต้องใช้ในทิศทางที่เหมาะสมในเวลาต่างๆสำหรับการขับมอเตอร์ แบบวาริเอเบิลรีลัคแทนซ์จะไม่ขึ้นกับทิศทางของกระแสที่ไหลในขดลวดเนื่องจากไม่มีแม่เหล็กอยู่ในตัวมันเลย

วงจรขับมอเตอร์จะรับสัญญาณควบคุมมาจาก วงจรจัดลำดับลอจิกในการสร้างวงจรขับ

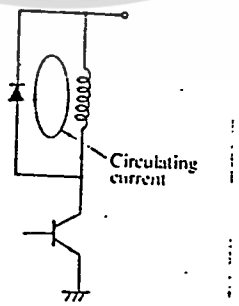
มอเตอร์จะมีปัญหาเกี่ยวกับ อินดัคแทนซ์และรีซิสแทนซ์ (Resistance) อนุกรมกันในวงจรสมมูลย์ของมอเตอร์ ดังรูปที่ 2.14 อีกทั้งในการหมุนยังมีปัญหาเพิ่มขึ้นเกี่ยวกับแรงดันไฟกลับ (BACK ELECTROMOTIVE FORCE : BACK EMF.) นอกจากนี้ยังต้องคำนึงถึงแหล่งจ่ายไฟ ดีซี และการใช้กับการป้องกัน Power ทรานซิสเตอร์ด้วย



รูปที่ 2.14 วงจรสมมูลย์ของสเตปปิ้งมอเตอร์

จากรูปที่ 2.14 เมื่อทรานซิสเตอร์หยุดทำงานจะเกิดสไปด์โวลท์เตจ (Spike Voltage) เนื่องจากขดลวด (ตัวเหนี่ยวนำ) ซึ่งอาจทำให้ทรานซิสเตอร์พังได้จึงต้องมีวงจรป้องกันและลดลักษณะแบบนี้อย่างรวดเร็ว โดยวิธีต่อไปนี้

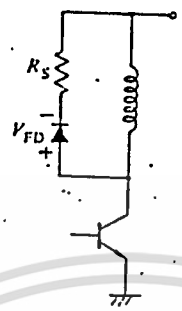
1. ลดโดยใช้ไดโอด (DIODE SUPPRESSOR) ใช้ไดโอดต่อคร่อมขนานกับขดลวดดังรูปที่ 2.15 กระแสจะไหลอยู่ในวงจรขดลวด กับไดโอดหลังจากทรานซิสเตอร์หยุดทำงาน ซึ่งกระแสนี้จะลดลงไปตามเวลาแต่ก็ใช้เวลานานมากจึงยังเกิดทอร์คยึดอยู่



รูปที่ 2.15 วงจร Diode Suppressor

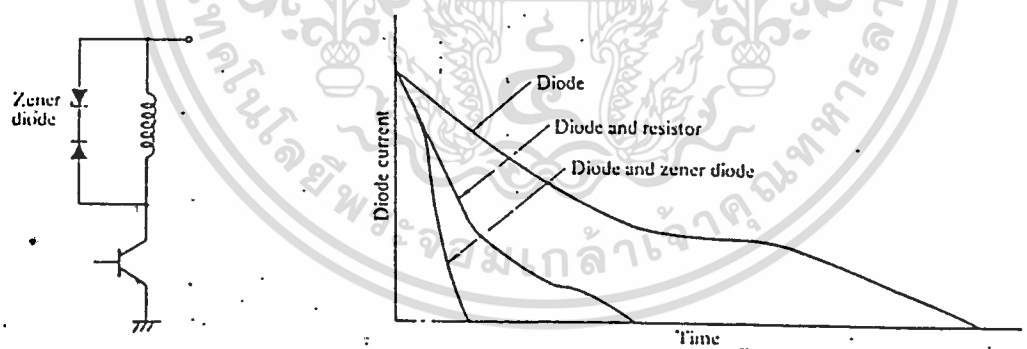
2.โดยใช้ไดโอดและตัวต้านทาน (DIODE AND RESISTOR SUPPRESSOR) โดยการต่อ

ความต้านทานอนุกรมกับไดโอด แล้วต่อขนานคร่อมขดลวดดังรูปที่2.16วิธีนี้จะช่วยลดเวลาในการที่จะทำให้กระแสหยุดไหลวนยิ่งใช้ตัวต้านทานค่ามากกระแสก็จะหยุดไหลเร็วขึ้นแต่ทรานซิสเตอร์ต้องทนโวลท์เตจคร่อม คอลเลคเตอร์ - อีมิเตอร์ เพิ่มขึ้นด้วย



รูปที่ 2.16 วงจร Diode/Resistor Suppressor

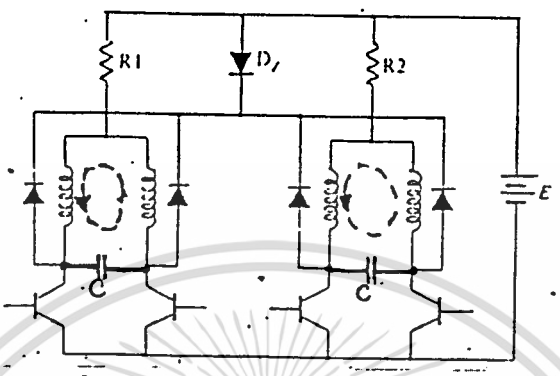
3. โดยใช้ซีเนอร์ไดโอด (ZENER DIODE SUPPRESSOR) โดยการต่อซีเนอร์ไดโอดดังรูปที่ 2.17 การต่อแบบนี้เมื่อเปรียบเทียบกับ 2 แบบแรกแล้ว จะมีประสิทธิภาพมากกว่า ซึ่งสามารถลดกระแสที่เกิดขึ้นได้เร็วกว่า



รูปที่ 2.17 วงจร Zener Diode Suppressor

4.โดยใช้คอนเดนเซอร์ (CONDENSER SUPPRESSOR) วิธีนี้มักจะใช้กับสเตปปิ้งมอเตอร์แบบไปฟีลา การต่อวงจรดังรูป 2.18 ของมอเตอร์ 4 เฟส ตัวคอนเดนเซอร์ที่ต่ออยู่ระหว่างเฟส 1 กับเฟส 3 และเฟส 2 กับเฟส 4 จะทำหน้าที่ดังนี้ในตอนที่ ทรานซิสเตอร์ T1 หยุดทำงานในการทำงานแบบกระตุ้นเฟสเดียว T2 หรือ T4 เริ่มทำงานแต่ T3 ยังไม่เริ่มทำงานเนื่องจาก เฟส1 เฟส3 ต่ออยู่ในลักษณะไบฟีลา กระแสที่เกิดขึ้นหลังจากที่ T1 หยุดทำงานจะไหลวนไปตามเส้นไข่ปลาดังแสดงในรูป ถ้า T3 หรือ T4 เริ่มทำงานแต่ T3 ยังไม่เริ่มทำงานประจุที่ชาร์จ (Charge) เก็บไว้ใน

คอนเดนเซอร์จะช่วยเพิ่มทอร์คให้โดยการ ปล่อยกระแสไหลผ่าน เฟส1 ทั้งนี้การต่อคอนเดนเซอร์สามารถใช้กับการกระตุ้นแบบ 2 เฟสได้ด้วย สำหรับค่า R1,R2 ใส่ไว้เพื่อปรับกระแสให้เหมาะสม นอกจากนั้น คอนเดนเซอร์ยังช่วยลดการสั่นของมอเตอร์ โดยเปลี่ยนพลังงานกลเป็นพลังงานความร้อนแทน



รูปที่ 2.18 วงจร (Condenser Suppreessor)

2.2 RS232C Serial Interface

ทำหน้าที่รับส่งข้อมูลในแบบอนุกรมเรียกว่า Universal Asynchronous Adapter เหตุที่มีชื่อเรียกว่า RS232C ก็เนื่องจากสมาคมผู้ผลิตอุปกรณ์อิเลคทรอนิคส์ของอเมริกาหรือ EIA ได้กำหนดมาตรฐานของอุปกรณ์การสื่อสารแบบอนุกรมเอาไว้ภายใต้ชื่อว่า RS232C ความจริงมาตรฐานของการส่งข้อมูลแบบอนุกรมมีหลายมาตรฐาน แต่ที่นิยมกันมากที่สุดสำหรับไมโครคอมพิวเตอร์ ก็คือ RS232C

หน้าที่สำคัญของการสื่อสารแบบอะซิงโครนัสก็คือ

รับสัญญาณ

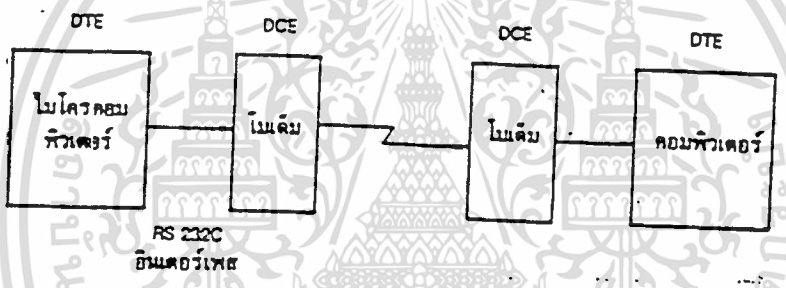
1. เปลี่ยนสัญญาณที่เข้ามาแบบอนุกรมให้เป็นแบบขนาน
2. ตรวจสอบความผิดพลาดของสัญญาณที่ได้รับ
3. ตัดสต๊อปบิต และ พาริตี้บิตออก
4. ส่งสัญญาณให้ซีพียูรู้ว่ารับสัญญาณไว้แล้ว

ส่งสัญญาณ

1. เปลี่ยนสัญญาณแบบขนานจากซีพียูค่อยทยอยส่งออกเป็นแบบอนุกรม
2. เพิ่มสต๊อปบิต และ พาริตี้
3. เพิ่มสัญญาณควบคุมโมเดมที่ต่อเชื่อม (ถ้ามี)

มาตรฐาน RS232C

มาตรฐาน RS232C ได้จัดพิมพ์ขึ้นเมื่อปี ค.ศ. 1969 โดยสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิคส์แห่งสหรัฐอเมริกา RS ย่อมาจาก Recommened Standard ส่วน 232 เป็นหมายเลขบ่งบอกของมาตรฐานตัวนี้ C เป็นหมายเลขของฉบับท้ายสุดของมาตรฐานตัวนี้ จุดประสงค์ของมาตรฐานตัวนี้ก็เพื่อบรรยายคุณลักษณะของการเชื่อมต่ออุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment DTE) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE ก็หมายถึงตัวไมโครคอมพิวเตอร์และ DCE ก็หมายถึงโมเดม อุปกรณ์อื่น ๆ เช่น เครื่องพิมพ์ที่รับสัญญาณแบบอนุกรมอาจจะเป็นได้ทั้ง DTE และ DCE จะเห็นได้จาก



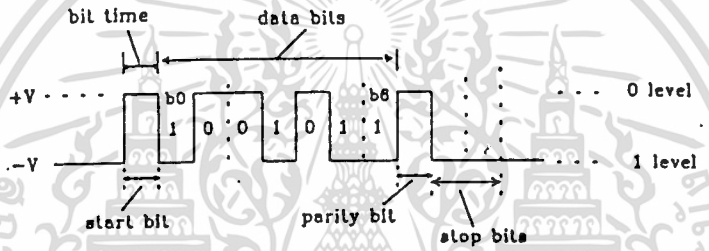
รูปที่ 2.19 การใช้ RS232C เชื่อมต่ออุปกรณ์

รูปที่ 2.19 จากรูปนี้เราจะเห็นได้ว่า RS232C มีส่วนสำคัญอย่างใหญ่หลวงสำหรับการสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์

ความจริงอีกประการหนึ่งของ RS232C ก็คือ ความเร็วและระยะทางของการเชื่อมต่อ RS232C สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้จาก 0 - 20,000 บิตต่อวินาทีซึ่งเพียงพอสำหรับไมโครคอมพิวเตอร์ที่มีขนาดอัตราบอด 110 ถึง 9600 บอด ความยาวของสายเชื่อมต่อโดยสัญญาณของมาตรฐานของ RS232 จำกัดอยู่แค่ 50 ฟุต ซึ่งเพียงพอสำหรับการสื่อสารไมโครคอมพิวเตอร์กับอุปกรณ์รอบนอก

ลักษณะที่สำคัญของระบบ RS232C นี้คือ ข้อมูลจะส่งในลักษณะอนุกรม กล่าวคือ แทนที่จะต้องมีสายสัญญาณ 8 เส้น สำหรับการส่ง 1 ไบท์ ระบบนี้อาจใช้เพียง 2 เส้น หรือมากกว่า (คืออย่างน้อยต้องมีสายสัญญาณ 1 เส้น และ Ground 1 เส้น) โดยจะให้สัญญาณผ่านทีละบิตเรียงต่อ ๆ กัน

ไปจนครบไบท์ ซึ่งลักษณะเช่นนี้มีข้อดีคือ จำนวนสายสัญญาณจะลดลงไปมากแม้ความเร็วในการส่งจะน้อยลงก็ตาม โดยสัญญาณที่ส่งไปนี้จะส่งไปในลักษณะเป็น Asynchronous Serial Data Format (ดังแสดงในรูป 2.20) กล่าวคือ ข้อมูลที่ส่ง จะประกอบด้วยสัญญาณ 2 ระดับ มาเรียงต่อกัน (เช่น + 5V สำหรับระดับ "0" และ - 5V สำหรับระดับ "1") โดยก่อนจะเริ่มส่งข้อมูล ระดับสัญญาณจะอยู่ที่ระดับ "1" จากนั้นก็จะมีสัญญาณเริ่มส่ง (Start Bit) ซึ่งเป็นสัญญาณระดับต่ำ "0" ส่งต่อเพื่อบอกว่ากำลังส่งข้อมูลตามมา จาก Start Bit จะตามด้วยสัญญาณขนาด 8 บิต เรียงต่อไป (ดูรูป 2.20 ประกอบ) จากนั้นก็จะมีสัญญาณบอกว่าจบการส่งข้อมูลแล้ว (เรียก Stop Bit) ซึ่งเป็นสัญญาณระดับสูง ("1") เมื่อหมดชุดข้อมูลก็ปรับให้สัญญาณมาอยู่ที่ระดับ "1" เพื่อคอยรับ Start Bit ต่อไป



รูปที่ 2.20 ลักษณะสัญญาณขณะส่งผ่าน RS232

การส่งสัญญาณลักษณะนี้ค่อยข้างช้า เพราะจำนวนบิตเรียงตัวต่อกันอย่างอนุกรม โดยอัตราเร็วของการส่ง (เรียก Baud Rate : จำนวนบิต / วินาที) จะมีค่าอยู่ ในช่วง 1200, 2400, 9600 ซึ่งจะเห็นได้ว่า การส่งแบบนี้ค่อนข้างช้ากว่าแบบขนานมาก เพราะการส่งลักษณะนี้ 1 ไบท์ จะต้องใช้จำนวนบิต ถึง 10 บิต (Start Bit + Data + Stop Bit)

ในการส่งข้อมูลผ่าน RS232 นี้สิ่งที่ต้องคำนึงถึงก็มีอาทิเช่น

- ต้องมีระดับสัญญาณตั้งแต่ +3 โวลต์ถึง +25 โวลต์ (สำหรับการส่งโดยคอมพิวเตอร์ไม่มีปัญหาแต่กรณีรับสัญญาณเข้า อาจต้องมีระบบป้องกัน เพราะหากสัญญาณเกิน +25 โวลต์อาจก่อให้เกิดความเสียหายได้)

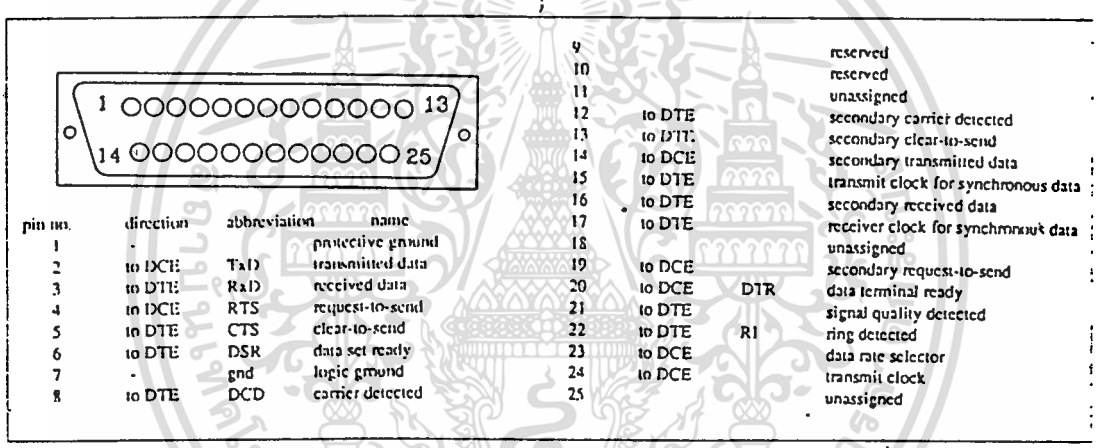
- ความเร็วในการส่ง (Baud Rate) ถ้าเป็นสายสัญญาณธรรมดา อาจส่งได้เร็วถึง 9600 แต่กรณีใช้ผ่านระบบโทรศัพท์ ความเร็วจะลดลง

- จำนวนบิตสำหรับข้อมูลว่าใช้ระบบใด เช่น 5,6,7 หรือ 8 บิต / ตัวอักษร

- จำนวน Stop Bit มีขนาดเท่าใด

- จะต้องมี Parity Check หรือไม่ ฯลฯ

ข้อดีของระบบ RS232 นี้ก็คือ สามารถติดต่ออุปกรณ์ที่วางอยู่ไกล ๆ ได้ (ความยาว < 15 เมตร) โดยอาจใช้สายเพียง 2 เส้น ในกรณีที่ต้องการส่งข้อมูลอย่างเดียว หรือ 8 เส้น ในกรณีต้องการตอบรับด้วย โดยข้อจำกัดก็มีบ้างเช่น สามารถใช้ได้กับอุปกรณ์ 1 ชิ้นต่อ 1 Port (คอมพิวเตอร์รุ่นใหม่ ๆ อาจมี Port ชนิดนี้ติดมาให้ 2 Port) ชนิดนี้ติดมาให้ 2 Port แต่อาจขยายจำนวนได้ นอกจากนี้การส่งแบบอนุกรม ทำให้ส่งข้อมูลได้ช้ากว่าแบบขนาน สำหรับ RS232 Port นี้จะใช้ตัวต่อ (Connector) แบบ 25 Pin (ดังรูป) ซึ่ง Pin ตำแหน่งต่าง ๆ อาจจะใช้งานไม่ทั้งหมด



รูปที่ 2.21 ลักษณะของ RS232C Port

Transmit Data (TD ขาที่ 2)

เป็นสัญญาณที่ส่งออกจาก DTE (หรือตัวไมโครคอมพิวเตอร์) ไปยังโมเด็มหรือต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น หรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออกสถานภาพของลอจิกที่ขานี้จะมีค่าเท่ากับ "1" หรือเทียบเท่ากับสต๊อปบิต

Receive Data (RD ขาที่ 3)

เป็นทางของสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์เมื่อไม่มีสัญญาณรับเข้ามา ขานี้จะมีสถานภาพทางลอจิก เป็น "1"

Request To Send (RTS ขาที่ 4)

ใช้สำหรับส่งสัญญาณไปยังโมเด็มหรือเครื่องพิมพ์เป็นการเรียกร้องที่จะส่งสัญญาณมาทางขา 2 สัญญาณนี้ใช้คู่กับ CTS หรือ Clear to send อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับสัญญาณก็จะส่งสัญญาณออกไปที่สาย CTS

Clear To Send (CTS ขาที่ 5)

ดังอธิบายไว้ใน RTS เมื่อสัญญาณนี้อยู่ในสถานะออฟ (Negative voltage หรือ ลอจิก "1") หมายความว่า อุปกรณ์รับกำลังบอกว่าพร้อมที่จะรับข้อมูลแล้ว

Data Set Ready (DSR ขาที่ 6)

เมื่อสัญญาณสายนี้อยู่ในสถานะออน (หรือลอจิก 0) เป็นการบอกไมโครคอมพิวเตอร์หรือฝ่ายส่งว่า โมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้วและพร้อมที่จะส่งได้แล้ว โมเด็มที่มีการหมุนหมายเลขอัตโนมัติจะส่งสัญญาณสายนี้ไปบอกให้คอมพิวเตอร์รู้ว่าต่อโทรศัพท์ได้สำเร็จแล้ว

Signal Ground (SG ขาที่ 7)

SG ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุก ๆ สายของสัญญาณ จะมีแรงต้นเป็น "0" เมื่อเทียบกับสัญญาณตัวอื่น

Carrier Detect (CD ขาที่ 8)

โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออน (ลอจิก "0") ไปบอกไมโครคอมพิวเตอร์ เมื่อได้รับสัญญาณจากโมเด็มของอีกฝ่ายหนึ่งสัญญาณนี้จะนำไปจุด LED บอกว่าได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่งแล้ว ไฟ LED จะอยู่บนหน้าปัดของโมเด็มเอง

Data Terminal Ready (DTR ขาที่ 20)

คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน (ลอจิก "0") เมื่อพร้อมที่จะติดต่อกับโมเด็ม โมเด็มส่วนมากจะไม่รายงานสถานภาพของตัวเอง (CD,USR และ CTS) ให้คอมพิวเตอร์รู้ หากคอมพิวเตอร์ไม่เปิดสัญญาณ DTR

Ring Indicator (RI ขาที่ 22)

สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบได้อัตโนมัติ (Auto - answer) สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งมา และ ออฟระหว่างเสียงดังของกระดิ่ง

2.3 8255 พอร์ทแบบขนานที่โปรแกรมได้

ในการนำเอาไมโครโปรเซสเซอร์ไปใช้งานนั้น จำเป็นต้องให้ไมโครโปรเซสเซอร์สามารถติดต่อกับโลกภายนอกได้ที่รู้จักกันดีคือ ให้มันสามารถส่งสัญญาณมาควบคุมอุปกรณ์ต่าง ๆ ได้เช่น สเตปปิ้งมอเตอร์, ควบคุมอุปกรณ์ไฟฟ้า ต่าง ๆ ส่วนที่ทำให้ไมโครโปรเซสเซอร์สามารถติดต่อกับโลกภายนอกได้ที่รู้จักกันก็คือ พอร์ท (PORT) ซึ่งมีอยู่หลายลักษณะด้วยกันเช่น เป็นแบบไอซีไตรสเตท (Tri - state) เบอร์ 74LS244 หรือพวกแลทช์ (Latch) เช่น เบอร์ 74LS374 เหล่านี้สามารถนำไปต่อใช้งานได้กับ CPU ได้โดยง่ายที่สุด โดยตัว CPU จะเป็นตัวควบคุมการอ่าน การเขียน พอร์ท หากเป็นการอ่านข้อมูลจากพอร์ทก็มักจะใช้ ไอซี แบบไตร-สเตท เป็นพอร์ทอินพุทโดยตัว CPU จะส่งสัญญาณไปเปิดเกทของไตรสเตทนี้ ให้ข้อมูลเข้ามาสู่สายข้อมูล (DATA BUS) และเข้าสู่ไมโครโปรเซสเซอร์หรือ CPU ต่อไป แต่สำหรับมาแลทช์ไว้ที่ตัวมัน (CPU) รับสัญญาณมาทริก เพื่อให้อุปกรณ์ภายนอกนั้นรับสัญญาณจากตัวแลทช์ตัวนี้ไปอีกทีหนึ่ง ที่ทำเช่นนี้ ก็เพราะตัวไมโครโปรเซสเซอร์หรือ CPU นี้ทำงานเร็วมาก ซึ่งในช่วงของการส่งข้อมูลออกจากพอร์ทจะใช้เวลาไม่กี่ไมโครเซค (μS)ซึ่งอาจทำให้อุปกรณ์ภายนอกรับไม่ทัน

มีบริษัทต่าง ๆ เล็งเห็นความสำคัญของการติดต่อระหว่าง CPU กับอุปกรณ์ภายนอกนี้จึงได้ทำไอซีสำเร็จรูปขึ้นหลายเบอร์และที่ขอกล่าวในที่นี้ก็คือ ไอซีเบอร์ 8255 ซึ่งเป็นของบริษัท อินเทล ซึ่งได้ออกแบบมาใช้กับ CPU เบอร์ 8080 แต่เราสามารถนำมาประยุกต์ใช้กับ เบอร์อื่น ๆ ได้โดยไม่ยาก

สาเหตุที่ 8255 เป็นที่นิยมมากก็เพราะว่า มันสามารถถูกโปรแกรมให้ทำงานในลักษณะต่าง ๆ ไม่ว่าจะเป็น อินพุท เอ้าท์พุท หรือแม้แต่แบบ แฮนด์เช็กกิ้ง (Handshaking) ได้ ทั้งยังราคาถูก

2.3.1 ลักษณะทั่วไปของ 8255

เป็นไอซีขนาด 40 ขา ตัวแบนโดยแยกเป็นลักษณะของบล็อกง่าย ๆ ดังรูปที่ 2.22 คือมีพอร์ทใช้งานได้ถึง 3 พอร์ท (เป็นขนาด 8 บิต) พอร์ท A, พอร์ท B, และพอร์ท C โดยพอร์ท C นี้สามารถแยกได้เป็น 2 ส่วนคือ พอร์ท C บน ตั้งแต่ PC4 - PC7 จำนวน 4 บิตและพอร์ท C ล่างตั้งแต่ PC0 - PC3 โดยพอร์ททุกพอร์ท (A,B,C) สามารถโปรมแกรมได้ให้เป็น อินพุท เอ้าท์พุทซึ่งจะได้กล่าวถึงรายละเอียดต่อไป

การทำงานของ 8255 จะใช้สัญญาณควบคุมจากตัวไมโครโปรเซสเซอร์มาควบคุมโดยมีคำสั่ง (Control Word) มาที่กลุ่มควบคุมชุด A,B แล้ว กลุ่มควบคุมชุดนี้ก็จะสั่งต่อไปที่พอร์ทเพื่อให้เป็นไปตามข้อกำหนดของคำสั่งนั้น ๆ เช่นให้ พอร์ท A เป็นอินพุท พอร์ท B เป็นเอ้าท์พุท เหล่านี้เป็นต้น ส่วนกรณีเมื่อมีการอ่านเขียนพอร์ทจาก CPU นั้น กลุ่มควบคุมลอจิกการเขียนอ่าน จะเป็นตัวส่งสัญญาณไปบอกแก่ กลุ่มควบคุมชุด ในแต่ละชุดอีกทีทั้งนี้แล้วแต่ว่า CPU จะมีการอ่านเขียนพอร์ทของ กลุ่มควบคุมชุดใด

ความหมายของขาต่าง ๆ ของ ไอซี 8255 มีดังนี้

D0 - D7 เป็นขาข้อมูลของ 8255 ที่ใช้ติดต่อกับตัวไมโครโปรเซสเซอร์ซึ่งข้อมูลที่จะเข้าออกสู่พอร์ทต่าง ๆ ของ 8255 จะต้องผ่านขาข้อมูลนี้

CS เป็นขาอินพุทที่รับสัญญาณลอจิก "0" จากภายนอกเพื่อแสดงว่าต้องการเลือกใช้ไอซีเบอร์นี้ หากได้รับลอจิก "1" ก็จะทำให้ไอซีตัวนี้ไม่ทำงานคือไม่รับรู้สัญญาณใด ๆ ทั้งสิ้น

RD เป็นขาอินพุทที่รับสัญญาณจากตัวไมโครโปรเซสเซอร์โดยหากมีลอจิกเป็น "0" จะแสดงว่า CPU ต้องการที่จะอ่านข้อมูลจากตัว 8255

WR เป็นอินพุทที่รับสัญญาณจากตัวไมโครโปรเซสเซอร์โดยหากมีลอจิกเป็น "0" ก็จะแสดงว่า CPU ต้องการที่จะเขียนข้อมูลจากตัว 8255

A0-A1 เป็นอินพุทที่รับแอดเดรสจากตัวไมโครโปรเซสเซอร์ ที่ถอดรหัสตำแหน่งของ 8255 เรียบร้อยแล้วโดยจะมีตำแหน่งใช้งาน 4 ตำแหน่ง เพื่ออ่าน เขียนรีจีสเตอร์ (พอร์ท) ของ 8255 ที่มีอยู่ด้วยกัน 4 ตัว

RESET เป็นอินพุท ที่รับสัญญาณจากภายนอกเข้ามาทำการรีเซตตัว 8255 โดยหากได้รับลอจิก "1" จะทำให้พอร์ทเป็นอินพุทพอร์ทหมดทั้งนี้ เพื่อไม่ต้องการให้มีสัญญาณออกไปกวนต่อระบบภายนอกเพื่อ 8255 ได้รับสัญญาณรีเซต

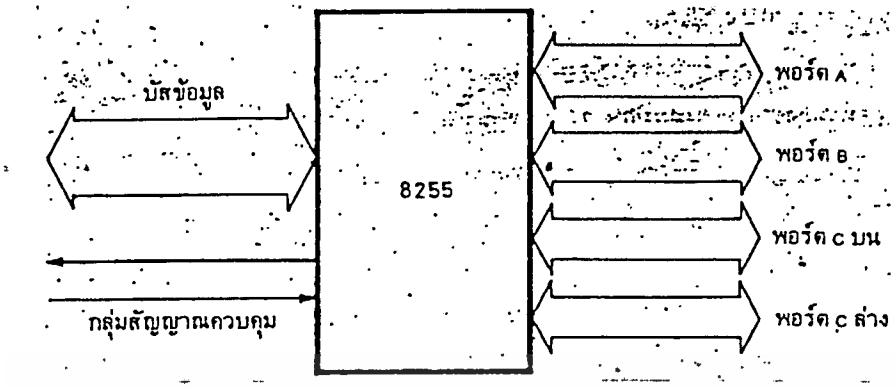
PA0-PA7 เป็นขาสัญญาณพอร์ท A ที่ใช้ติดต่อกับโลกภายนอก

PB0-PB7 เป็นขาสัญญาณพอร์ท B ที่ใช้ติดต่อกับโลกภายนอก

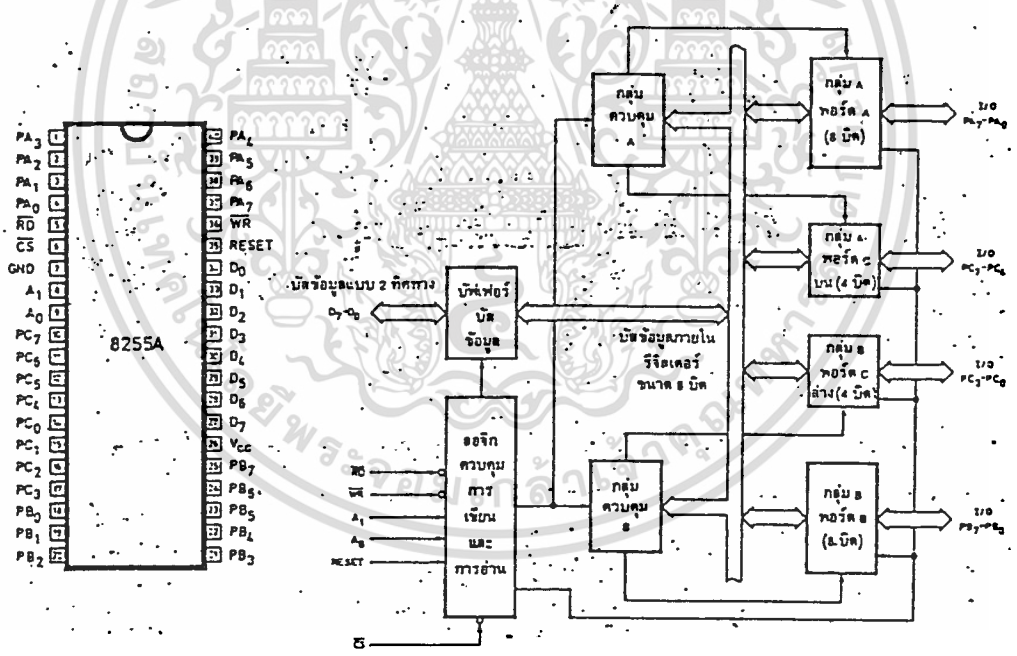
PC0-PC7 เป็นขาสัญญาณพอร์ท C ที่ใช้ติดต่อกับโลกภายนอก ซึ่งพอร์ทนี้จะแบ่งเป็น 2 กลุ่ม คือ PC0-PC7 ซึ่งสามารถโปรแกรมแยกกันได้

2.3.2 การต่อใช้งาน 8255

หากดูที่ขาของ 8255 ที่รูปที่ 2.23 แล้วจะสังเกตุเห็นได้ว่าเราสามารถต่อขา 8255 บางส่วนได้โดยตรงกับขา ไมโครเซสเซอร์เลย (Z-80) เช่นขา D0-D7 , A0-A7 เป็นต้น หากแต่บางขาเราจำเป็นต้องมีการดัดแปลงสัญญาณที่ได้จาก CPU (ซึ่งกรณีนี้ เราใช้เบอร์ Z-80) เสียก่อน โดยหากเราถอด



รูปที่ 2.22 แสดงบล๊อกเส้นทางของพอร์ท

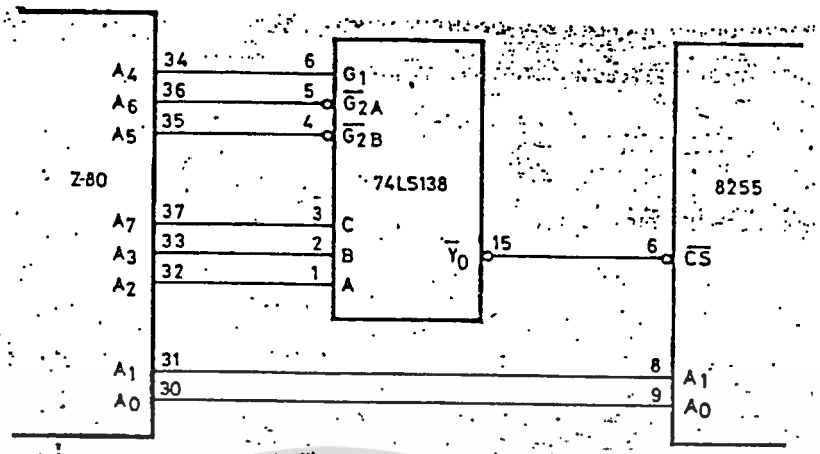


รูปที่ 2.23 แผนผังวงจรภายในและการจัดขา ไอซี

ในรูปที่ 2.23 จะเห็นโครงสร้างภายในที่แสดงถึงกลุ่มควบคุมที่มีอยู่ 3 กลุ่ม คือ

- กลุ่มควบคุม A จะควบคุม พอร์ท A และพอร์ท C บน
- กลุ่มควบคุมชุด B จะควบคุม พอร์ท B และพอร์ท C ล่าง
- กลุ่มควบคุมลอจิกการเขียนและการอ่าน

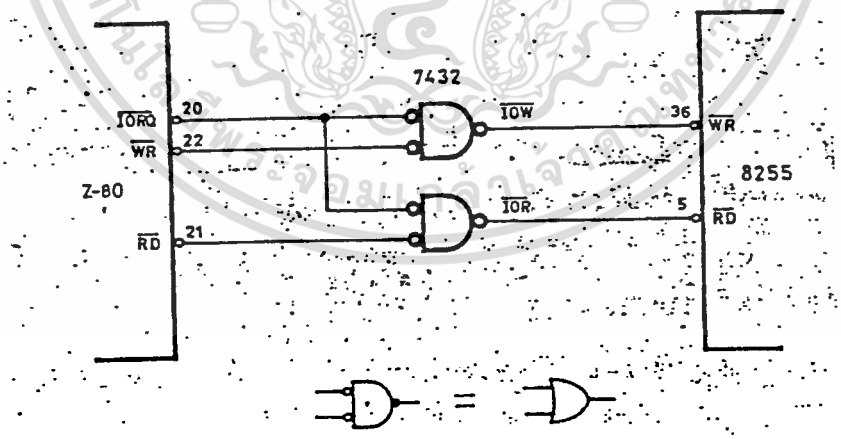
รหัสแอดเดรสของ 8255 ให้เป็นพอร์ทที่แอดเดรส 10H,11H,12H,13H เราจะสามารถทำ ดีโค๊ดเดอร์โดยง่ายดังรูปที่ 2.24



รูปที่ 2.24 แสดงการดีโค๊ดแอดเดรสพอร์ทให้ 8255

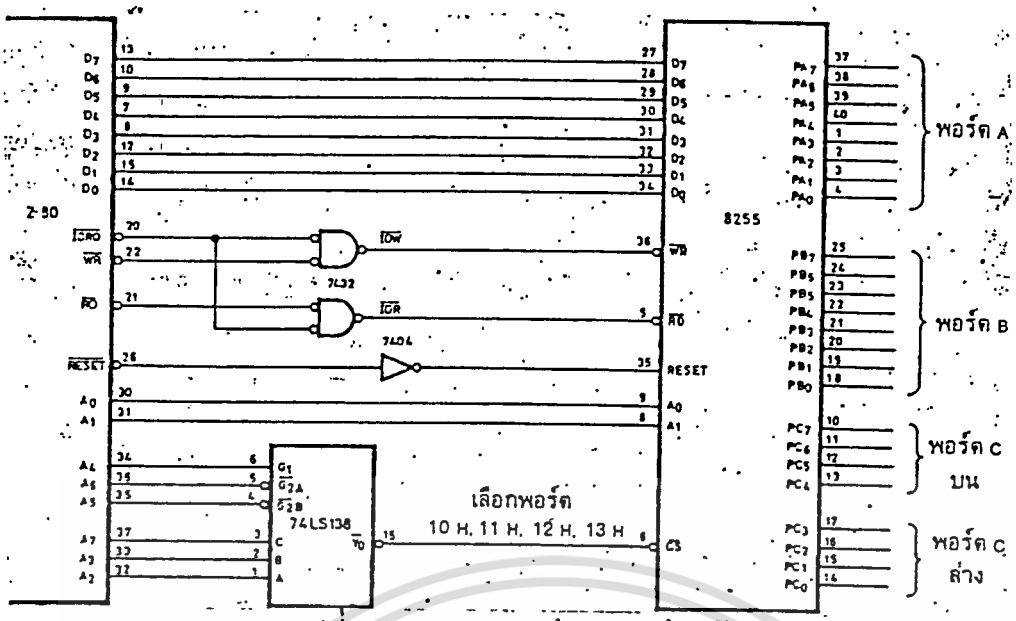
สังเกตุได้ว่า CS จะแอคทีฟทุกครั้งหาก A-80 อ้างพอร์ทที่แอดเดรส 0001000XX โดยค่าของ XX คือ A0 ,A1 ที่เราจะต่อตรงเข้ากับ 8255 เพื่อทำการเลือกรีจีสเตอร์ควบคุมและพอร์ททั้งสามของ 8255

สัญญาณการควบคุมอีกส่วนที่สำคัญก็คือ สัญญาณการอ่านเขียนพอร์ทของ 8255 โดยหาก WR ได้รับแอคทีฟ "0" ก็จะเป็นการเขียนข้อมูลจาก CPU เข้าสู่ตัว 8255 หรือออกสู่พอร์ทที่ต้องการ และเช่นเดียวหาก RD ได้รับลอจิก "0" ก็จะเป็นการอ่านข้อมูลพอร์ทจากตัว 8255 เข้าสู่ CPU เข้ากับตัว Z - 80 แล้ว เราจะต้องทำเป็นสัญญาณการอ่านเขียนพอร์ทของ Z - 80 ให้ถูกต้องเสียก่อนดังรูป 2.25



รูปที่ 2.25 การเชื่อมต่อสัญญาณ อ่านเขียนพอร์ทระหว่าง Z - 80 , 8255

และสุดท้ายคือสัญญาณ RESET แอคทีฟ "0" ฉะนั้นเราจะต้องมาผ่านอินเวอร์เตอร์ก่อนก็จะได้ลักษณะการต่อร่วมกับ CPU ดังนี้



รูปที่ 2.26 แสดงการต่อ 8255 ร่วมกับ Z - 80

2.3.3 การโปรแกรม 8255

เราได้ทราบมาแล้วในรูปที่ 2.23 ว่าโครงสร้างภายในของ 8255 มีการควบคุมชุดอยู่ 3 กลุ่ม ซึ่งทั้งสามกลุ่มนี้จะทำงานร่วมกันดังกล่าวมา และเราสามารถจะควบคุมการทำงานของพอร์ทจาก CPU ได้โดยสั่งงานมาที่ กลุ่มควบคุมดังกล่าว แต่ตัว CPU จะมองเห็น 8255 เป็น 4 พอร์ทด้วยกัน โดยแต่ละพอร์ทเสมือนเป็นรีจิสเตอร์ที่ CPU สามารถจะทำการอ่าน / การเขียนได้ แต่ละพอร์ทจะอยู่กันคนละแอดเดรส ดังที่ทำการดีโค๊ดให้ 8255 ที่แอดเดรส 10H,11H,12H,13H (ตามสัญญาณ A0 - A1) และเราจะสามารถได้ตำแหน่งของพอร์ท 8255 แต่ละตัวดังรูป

10H ----------> พอร์ท A

11H ----------> พอร์ท B

12H ----------> พอร์ท C

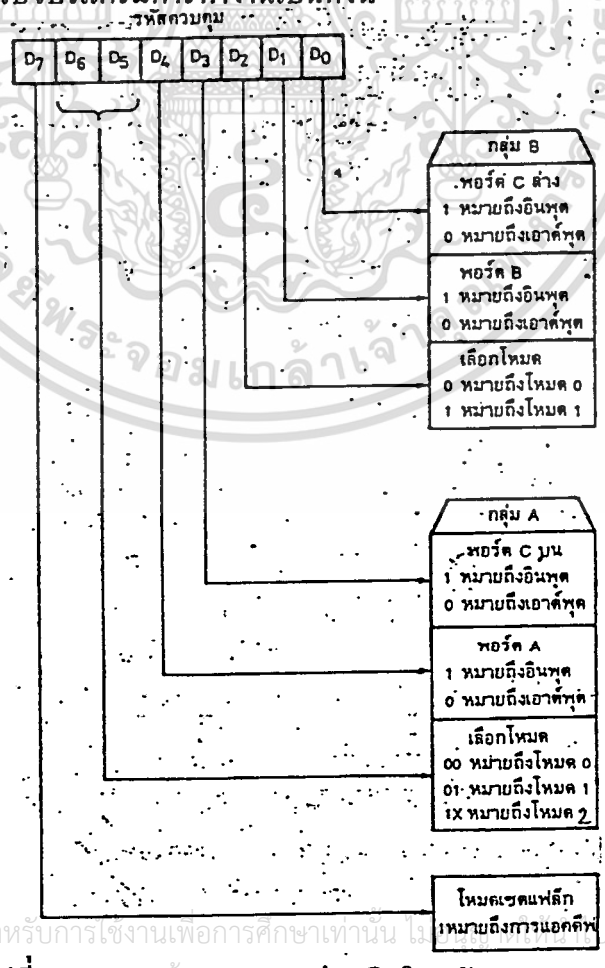
13H ----------> พอร์ท Control

ซึ่งหากมีการอ่านเขียนไปยังพอร์ทดังกล่าว ก็จะใช้ร่วมกับสัญญาณ RD, WR โดย WR หมายถึงเอ้าท์พุทข้อมูลและ RD แอกทีฟหมายถึง อินพุทข้อมูล ดังนั้นเราจะได้ลอจิกที่ขาของ 8255 ในลักษณะต่าง ๆ ดังนี้

RD	WR	A1	A0	ความหมาย
1	0	0	0	เขียนพอร์ท A ซึ่งเป็นข้อมูล
0	1	0	0	อ่านพอร์ท A ซึ่งเป็นข้อมูล
1	0	0	1	เขียนพอร์ท B ซึ่งเป็นข้อมูล
0	1	0	1	อ่านพอร์ท B ซึ่งเป็นข้อมูล
1	0	1	0	เขียนพอร์ท C ซึ่งเป็นข้อมูล
0	1	1	0	อ่านพอร์ท C ซึ่งเป็นข้อมูล
1	0	1	1	เขียนข้อมูล ซึ่งเป็นระหัสควบคุม
0	0	1	1	อ่านเข้ามา ซึ่งไม่มีความหมายใด

ตารางที่ 2.1 แสดงลอจิกเมื่อทำการติดต่อกับ 8255

การใช้งาน เราจะต้องส่งรหัสควบคุม (Control code) หรือเรียกอีกอย่างหนึ่งว่า รีจีสเตอร์ควบคุม ซึ่งจะเป็นข้อมูลขนาด 1 ไบท์ ส่งไปที่แอดเดรส 13H (กรณีนี้เราถอดรหัสไว้ที่ 13H) โดยหมายความของแต่ละบิตที่เราส่งไปโปรแกรมการทำงานเป็นดังนี้



รูปที่ 2.27 ความหมายของแต่ละบิตในรหัสควบคุม

บิต D7 เป็นบิตที่แสดงว่าในไบท์นี้เป็นรหัสควบคุม ถ้าเป็น "1" โดยแต่ละบิตจะมีผลการเปลี่ยนแปลงโหมดต่าง ๆ ของ 8255 หากเป็น "0" การเซทพอร์ทของ C

บิต D6,D5 เป็นการเลือกโหมดของพอร์ท A ซึ่งจะมีอยู่ ด้วยกัน 3 โหมดคือ 0,1,2

บิต D4 เป็นการกำหนดให้พอร์ท A เป็นอินพุทหรือเอ้าท์พุท โดยหากเป็น "1" ก็จะเป็นอินพุท หากเป็น "0" ก็จะเป็นเอ้าท์พุท

บิต D3 เป็นการกำหนดให้พอร์ท C บน ให้เป็นอินพุทหรือ เอ้าท์พุท โดยหากเป็น "1" ก็เป็นอินพุท หากเป็น "0" ก็จะเป็นเอ้าท์พุท

บิต D2 เป็นการกำหนดโหมดการทำงานของพอร์ท B โดยหากเป็น "0" หมายถึงเลือกให้พอร์ท B ทำงานในโหมด 0 หากเป็น "1" เป็นการเลือกให้พอร์ท "1" เป็นการเลือกให้พอร์ท B ทำงานในโหมด 1

บิต D1 เป็นการกำหนดให้พอร์ท B ให้เป็นอินพุท หรือ เอ้าท์พุท โดยหากเป็น "1" ก็จะเป็นอินพุท หากเป็น "0" ก็แสดงว่าให้เป็นเอ้าท์พุท

บิต D0 เป็นการกำหนดให้พอร์ท C ล่าง ให้เป็นอินพุท โดยหากเป็น "1" ก็จะเป็นอินพุท หากเป็น "0" ก็จะแสดงว่าให้เป็นเอ้าท์พุท

การโปรแกรมก็จะเริ่มจากการส่งค่าระหัสควบคุม 1ไบท์ดังกล่าวนี้ไปสู่พอร์ทควบคุมหลังจากนั้นหากต้องการเรียกไปถึงพอร์ทใดก็พร้อมสามารถอ้างได้ตามแอดเดรสทันที เช่นต้องการโปรแกรมให้พอร์ท A,BC ทั้งหมดให้เป็นเอ้าท์พุทพอร์ท เราก็จะได้รหัสควบคุมเป็น 10000000 หรือ 80H เราก็จะสั่งได้เป็น

```
LD    A,80H      ; กำหนดรหัสควบคุม
OUT   (13H),A    ; เป็นการส่งรหัสควบคุมสู่รีจีสเตอร์ควบคุม 8255
```

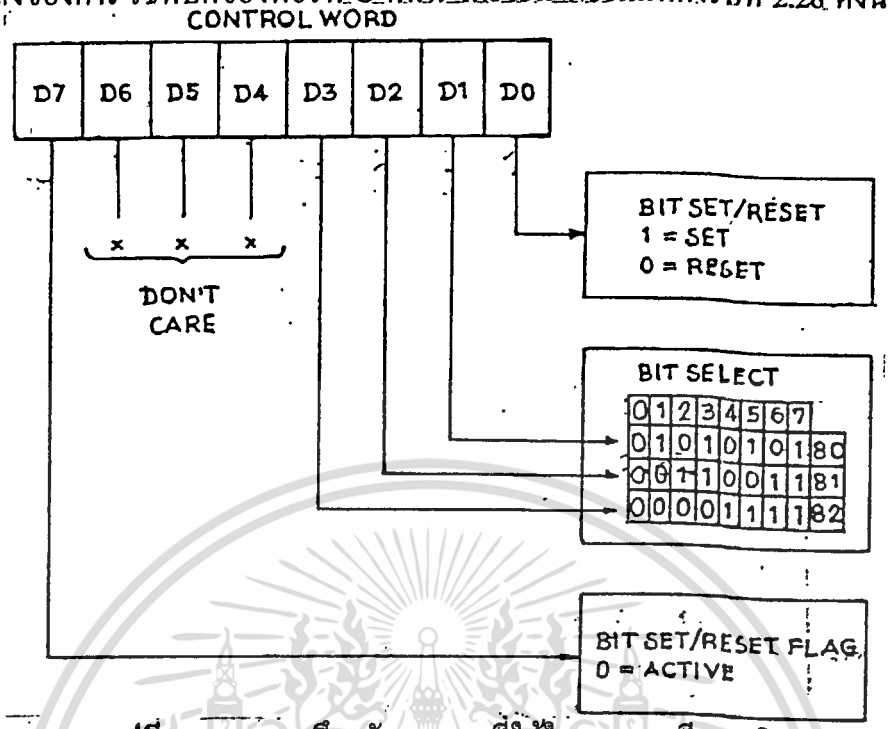
จากนี้ทั้งสามพอร์ท ก็จะเป็นเอ้าท์พุทพอร์ท ให้เราตามต้องการเมื่อเราจะส่งค่าออกไปก็สามารถกระทำได้โดยง่าย เช่น ต้องการส่งค่า 88H ออกไปที่พอร์ท B,C เราจะทำดังนี้

```
LD    A,88H      ; ค่าของข้อมูลที่ต้องการส่ง
OUT   (11H),A    ; ส่งออกไปพอร์ท B
OUT   (12H),A    ; ส่งออกไปพอร์ท C
```

<u>กรณีพิเศษของรหัสควบคุม</u>

ปกติรหัสที่เราต้องส่งออกไปถึงพอร์ทควบคุมหรือรีจีสเตอร์ควบคุมนี้ จะต้องเป็นการเซทโหมด,พอร์ทอินพุท,เอ้าท์พุทและรหัสนั้น บิต 7 จะต้องเป็น "1" เสมอที่นี้หากบิต 7 นี้เป็น "0" บ้างจะเกิดอะไรขึ้นถ้าหากบิต 7 เป็น "0" และถูกส่งไปที่ แอดเดรสของพอร์ทควบคุมแล้ว 8255 จะ

ถือว่าเป็นคำสั่งของการ เซทบิตของพอร์ท C ทันที โดยจะมี ฟอร์แมทดังรูปที่ 2.28 ดังนี้



รูปที่ 2.28 แสดงถึงรหัสควบคุมที่ใช้ในการเซท/รีเซท บิต

ตัวอย่างเช่น

```
LD    A,80H          ; รหัสควบคุมที่ใช้ทั้งสามพอร์ทเป็น OUT PUT
OUT   (13H),A        ; ส่งสู่รีจีสเตอร์ควบคุม
LD    A,00010001B    ; รหัสควบคุมใช้เซทบิตที่ 4 ของพอร์ท C "1"
OUT   (13H),A        ; บิต 4 พอร์ท C เป็น "1"
DEC   A              ; เปลี่ยนเป็นรีเซท
OUT   (13H),A        ; กำหนดให้บิต 4 พอร์ท C เป็น "0"
```

จะเห็นว่าสามารถนำไปประยุกต์ใช้งานได้เช่น เป็นตัวสร้าง พัลซ์ และกำหนดใช้งาน เปิด-ปิด อุปกรณ์ด้วย พอร์ท C ที่มีคำสั่งไม่ยุ่งยากและเป็นอิสระเป็นต้น

2.3.4 การทำงานในโหมด 0

จัดว่าเป็นโหมดพื้นฐานที่ นิยมใช้กันมากที่สุด เนื่องด้วยความง่ายตรงไปตรงมา คือทั้งสามพอร์ทเราสามารถจะโปรแกรมได้ชุดใดชุดหนึ่ง เป็นอินพุทหรือเป็นเอ้าท์พุทได้อีก ฉะนั้นโดยสรุปแล้วก็จะมีพอร์ทที่จะโปรแกรมให้เป็นอินพุทหรือเอ้าท์พุทได้เสมือน 4 พอร์ท คือ พอร์ท A,พอร์ท B, พอร์ท C บนและพอร์ท C ล่าง (แต่โปรแกรมแยกเฉพาะบิตในแต่ละพอร์ทไม่ได้)

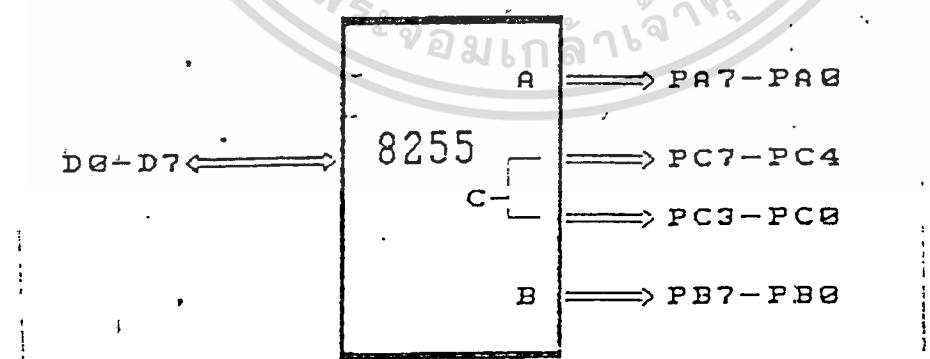
1	0	0	X	X	0	X	X
BIT	7	6	5	4	3	2	1	0

ตารางที่ 2.2 แสดงรหัสคำสั่งของโหมด 0

ซึ่งหากเราดูที่รหัสคำสั่งแล้วจะเห็นว่ามีอยู่ 4 บิต ที่ถูกกำหนดตายตัวคือ บิต 7,บิต 5,บิต 6, และบิต 2 ส่วนที่เหลืออีก 4 บิต ก็คือกำหนดว่าให้พอร์ทใดเป็นพอร์ทอินพุท/เอ้าท์พุท นั่นเองซึ่งหากเราให้พอร์ทใดเป็นอินพุท เราก็ใส่ลอจิก "1" ที่บิตนั้น หรือหากเราต้องการให้พอร์ทใดเป็นเอ้าท์พุท ก็ให้ใส่ "0" ที่บิตนั้น จะเห็นว่ามีความเป็นไปได้ในการกำหนดลักษณะของพอร์ทใน โหมดนี้อยู่ 16 อย่างด้วยกัน

<u>ตัวอย่างเช่น</u> 1. ต้องการให้ทุกพอร์ทเป็นเอ้าท์พุทหมด เราจะได้รหัสคำสั่งเป็น

1	0	0	0	0	0	0	0



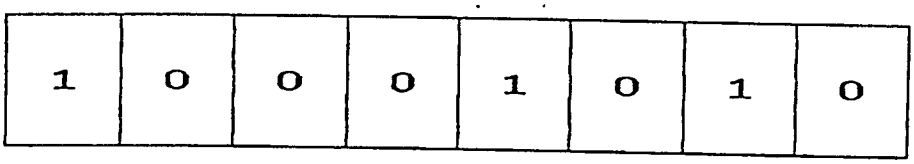
2. ต้องการให้ พอร์ท A ----------> เอ้าท์พุท

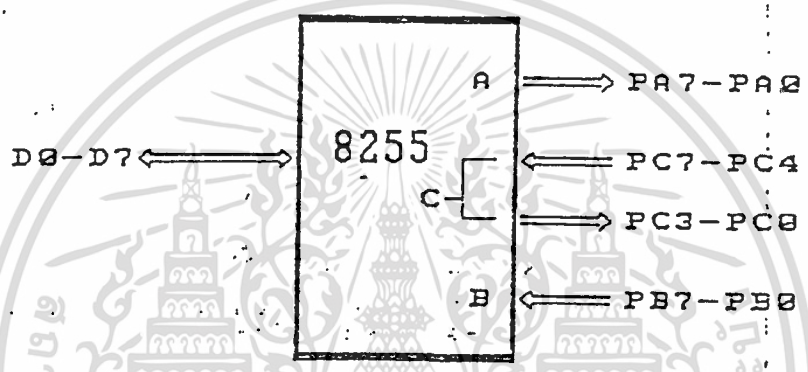
พอร์ท B ----------> อินพุท

พอร์ท C บน ----------> อินพุท

พอร์ท C ล่าง ----------> เอ้าท์พุท

เราจะได้รหัสคำสั่งเป็น





2.3 ชนิดของแขนกล (Type of robotic arm)

แขนกลมีรูปร่างลักษณะแตกต่างกันไปแล้วแต่ลักษณะการใช้งานโดยขึดความสามารถในการเคลื่อนที่ จะแตกต่างกันไปแต่โดยทั่วไปหากจะจัดแบ่งออกเป็นกลุ่มก็คงจะแบ่งได้เป็น 4 กลุ่มคือ

1- แขนกลที่มีการเคลื่อนไหวได้ใน ระบบพิกัดเชิงขั้ว (Polar Coordinate Configuration)

2- แขนกลที่มีการเคลื่อนไหวได้ใน ระบบพิกัดทรงกระบอก (Cylindrical Coordinate Configuration)

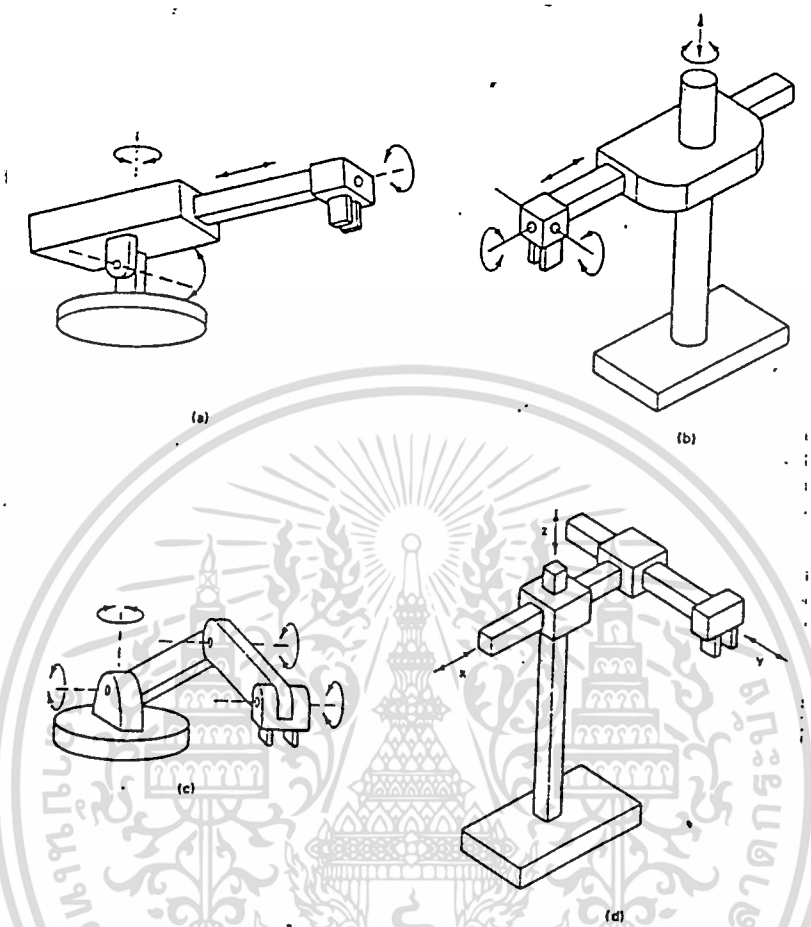
3- แขนกลที่มีแขนในลักษณะเป็นข้อต่อ (Joint Arm Configuration)

4- แขนกลที่มีการเคลื่อนไหวได้ใน ระบบพิกัดคาร์ทิเซียน (Cartesian Coordinate Configuration)

ลักษณะของแขนกลทั้ง 4 ชนิด แสดงไว้ในรูป 2.29 โดยรายละเอียดจะเป็นดังนี้

ก) แขนกลที่มีการเคลื่อนไหวได้ในระบบพิกัดเชิงขั้ว :- ขอบเขตการเคลื่อนที่ของแขนกลที่มีลักษณะเช่นนี้ อาจเรียกอีกอย่างหนึ่งว่าอยู่ใน ระบบพิกัดทรงกลม (Spherical Coordinate Configuration) ทั้งนี้เพราะลักษณะบริเวณที่แขนกลจะกวาดไปได้จะมีลักษณะเป็นรูปทรงกลม (ดังรูป 2.29ก)

ข) แขนกลที่มีการเคลื่อนไหวได้ในระบบพิกัดทรงกระบอก :- หุ่นยนต์ลักษณะนี้มีลักษณะการกวาดของแขนเป็นรูปทรงกระบอก (ดังรูป 2.29ข)



รูปที่ 2.29 ลักษณะของแขนกล

ก) แบบที่มีการเคลื่อนไหวได้ในระบบพิกัดเชิงขั้ว

ข) แบบที่มีการเคลื่อนไหวได้ในระบบพิกัดทรงกระบอก

ค) แบบที่มีแขนในลักษณะเป็นข้อต่อ

ง) แบบที่มีการเคลื่อนที่ได้ในระบบพิกัดคาร์ทีเซียน

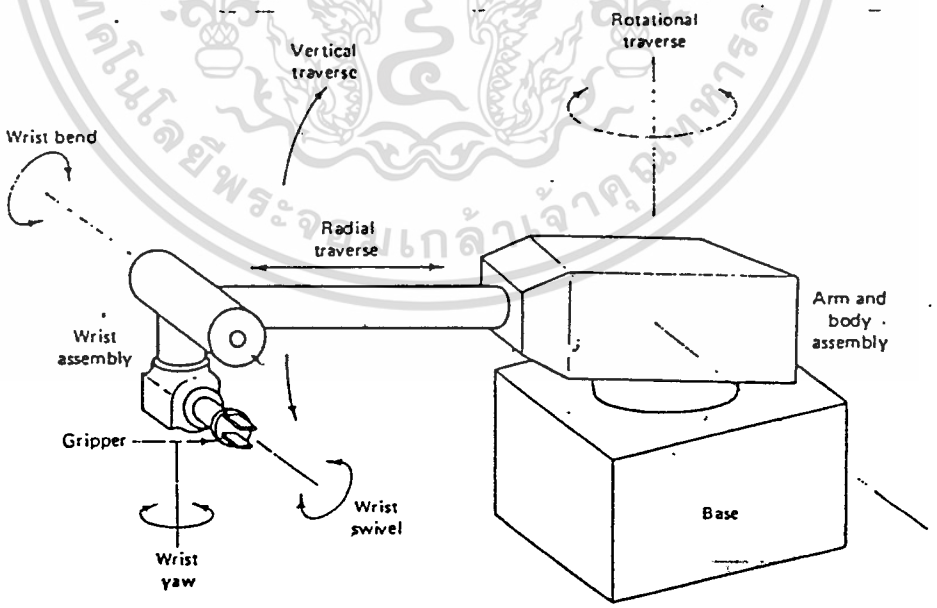
ค) แขนกลที่มีแขนในลักษณะเป็นข้อต่อ :- แขนกลประเภทนี้จะเลียนแบบลักษณะของแขนมนุษย์โดยส่วนต่าง ๆ สามารถยืดหดได้เช่นเดียวกับส่วนไหล่ ตัวแขน ข้อศอก และข้อมือ ทำให้การเคลื่อนที่คล่องตัว โดยจะมีรัศมีที่กวาดได้ เป็นลักษณะรูปครึ่งทรงกลม (ดังรูป 2.29ค)

ง) หุ่นยนต์นี้มีการเคลื่อนไหวได้ในระบบพิกัดคาร์ทีเซียน :- หุ่นยนต์ประเภทนี้จะเคลื่อนที่ได้ใน 3 แกนหลักคือ แกน X แกน Y และแกน Z ของระบบพิกัดคาร์ทีเซียน ดังนั้นขอบเขตรัศมีการกวาดของมือจะเป็นรูปกล่องสี่เหลี่ยม (ดังรูป 2.29 ง)

ทั้งนี้ไม่ว่าจะเป็นแขนกลในลักษณะใด งานหลักของแขนกลก็คือการจัดการกับเป้าหมายโดยจะใช้ส่วนที่อยู่ที่ปลายสุดของแขนเป็นตัวการ ส่วนนี้จะทำหน้าที่คล้ายมือของมนุษย์ มีชื่อเรียกเฉพาะว่า " End Effector" ดังนั้นในการนำไปใช้งานจะต้องคำนึงถึงความสามารถของส่วนนี้เป็นหลัก

โดยปกติแล้วการเคลื่อนที่ของส่วน "มือ" หรือ End Effector นี้จะทำได้ใน 6 ระดับขั้นความเสรี (Degree of Freedom) ซึ่งจะทำให้คล้ายกับการทำงานของมือมนุษย์ แต่ก็มิใช่ว่าแขนกลทุกตัวจะมีมือที่หมุนได้ใน 6 ระดับขั้นความเสรีโดย 6 ระดับขั้นความเสรีนี้ จะเป็นการหมุนของแขนและตัวเสีย 3 และเป็นการหมุนของมืออีก 3 ดังแสดงในรูป 2.30 โดยรายละเอียดจะเป็นดังนี้

1. การเคลื่อนที่ในแนวดิ่ง (Vertical Transverse) โดยการยกทั้งแขนขึ้น หรือยกลงตามแนวดิ่ง
2. การเคลื่อนที่ทางแนวรัศมี (Radial Transverse) เป็นการยืดหดมือออกจากแกน
3. การหมุน (Rotation Transverse) เป็นการหมุนรอบแกน ส่วนของมือจะมีอีก 3 อย่างคือ
4. การหมุนของข้อมือ (Wrist Swivel) (ดูรูป 2.30 ประกอบ)
5. การบิดของข้อมือ (Wrist Bend) เช่นยกขึ้น / ยกลง (ดูรูป 2.30 ประกอบ)
6. การหันเหข้อมือ (Wrist Yew)



รูป 2.30 ลักษณะการเคลื่อนที่ทั้ง 6 ระดับขั้นความเสรีของแขนกล

การเคลื่อนที่ของแขนกล

ลักษณะการเคลื่อนที่ของแขนกลอุตสาหกรรมนี้ จะคล้ายกับการเคลื่อนที่ของอุปกรณ์ใน NC Machine กล่าวคือ มีได้ 2 ลักษณะคือ เป็นการเคลื่อนที่จากจุดไปจุด (Point-To-Point : PTP) และแบบต่อเนื่อง (Continuous Path) โดยแบบแรกจะเป็นการโปรแกรมตำแหน่ง (เริ่มต้น/สิ้นสุด) ให้กับแขนกลเพื่อให้มันเคลื่อนที่จากจุดจุดหนึ่งไปยังมีอีกจุดหนึ่งโดยไม่สนใจว่าจะใช้เส้นทางใดซึ่งแขนกลที่มีการเคลื่อนที่ลักษณะนี้ สามารถนำไปใช้ในงานประกอบ หยิบ จับ ยกของ หรืองานเชื่อมตามจุด (Spot Welding) ฯลฯ ส่วนแบบต่อเนื่องหรือแบบ Contour นี้แนวทางการเคลื่อนที่จะกำหนดแน่นอน ดังนั้นจึงใช้ได้กับงานที่อาศัยแนวทางการเคลื่อนที่ด้วย เช่น การพ่นสีชิ้นส่วน การเชื่อมแบบต่อเนื่อง (Arc Welding) ฯลฯ โดยแบบหลังนี้จำเป็นต้องใช้หน่วยความจำในการสั่งงานมากกว่าแบบแรก

คุณสมบัติอื่น ๆ ของแขนกลอุตสาหกรรมทั่วไป

นอกเหนือจากลักษณะของแขนกล และการเคลื่อนที่ที่กล่าวไปแล้วนั้น คุณสมบัติทางเทคนิคอื่น ๆ ของแขนกลอุตสาหกรรมจะแตกต่างกันไปแล้วแต่ลักษณะของงาน ซึ่งคุณสมบัติเหล่านี้จะเป็นตัวบอกถึงประสิทธิภาพของแขนกลแต่ละประเภท โดยลักษณะเหล่านี้ก็มีอาทิเช่น

- บริเวณการทำงาน (Work Volume) ซึ่งเป็นบริเวณหรือขอบเขตที่แขนกลต้องให้ในการทำงาน แต่ทั้งนี้ก็ขึ้นกับขนาดและลักษณะของแขนกล ด้วยโดย Work Volume ของแขนกลแบบต่าง ๆ จะเป็นดังนี้:

แบบระบบพิกัดคาร์ทีเซียน :- Work Volume จะเป็นรูปทรงกล่องสี่เหลี่ยม

แบบระบบพิกัดทรงกระบอก :- Work Volume จะเป็นรูปทรงกล่องสี่เหลี่ยม

แบบระบบแขนลักษณะข้อต่อ :- Work Volume จะมีลักษณะคล้ายทรงกลม

แบบระบบพิกัดเชิงขั้ว :- Work Volume จะมีลักษณะเป็นรูปกึ่งทรงกลม

- ความละเอียดในการเคลื่อนที่ (Precision of Movement) จะหมายถึงการที่แขนกลสามารถจัดการกับงานที่มีความละเอียดขนาดไหน การทำงานเที่ยงตรงแค่ไหน และหากทำซ้ำจะกลับมายังตำแหน่งเดิมได้ใกล้เคียงขนาดไหน ฯลฯ

- ความเร็วในการเคลื่อนที่ หมายถึง ความเร็วที่หุ่นยนต์สามารถบังคับ "มือ" ให้จัดการกับงานโดยอาจ ทำได้ถึง 1.5 เมตร / วินาที

- ความสามารถในการรับน้ำหนัก ส่วนนี้จะขึ้นอยู่กับการออกแบบให้เหมาะกับงาน โดยสูงสุดอาจรับได้ถึง 1000 ปอนด์

- ลักษณะของระบบขับ (Drive System) ซึ่งเป็นระบบที่ขับให้ส่วนต่าง ๆ เคลื่อนที่โดยในปัจจุบันที่มีใช้กันมากก็มี 3 แบบด้วยกันคือ

1) ระบบไฮดรอลิก (Hydraulic) ซึ่งเป็นระบบที่ง่ายต่อการดูแลรักษา ทนทาน และยังทำให้การเคลื่อนที่ทำได้เร็ว

2) ระบบมอเตอร์ไฟฟ้า (Electric Motor) เป็นระบบที่ใช้ Stepping Motor หรือ Servo Motor เป็นตัวไปควบคุมการเคลื่อนที่ของส่วนต่าง ๆ แขนกลที่ใช้ระบบนี้จะไม่มีกำลังเหมือนกับแบบแรก แต่จะมีความ ละเอียดไม่ว่าจะในแง่ของตำแหน่งที่เคลื่อนไปหรือการทำซ้ำจะดีกว่า

3) ระบบลมอัด (Pneumatic) ระบบนี้จะใช้กับแขนกลขนาดเล็ก อีกทั้งระบบไม่ยุ่งยาก

บทที่ 3

3.1 โครงสร้างของงาน

เครื่อง 3 Dimension Personal Computer Controller จะมีโครงสร้างของงานแบ่งเป็นส่วนใหญ่ ๆ ได้ดังนี้

1. Personal Computer : จะเป็นเครื่อง Personal Computer ที่ใช้ CPU เบอร์ 286, 386 หรือ 486 ก็ได้ แต่ต้องมี Serial Port

2. Control Module : จะประกอบอยู่ในกล่องขนาดใหญ่ ซึ่งภายใน Module นี้จะประกอบด้วย RS232 Serial Interface, Micro Controller Z-80 2 ตัวที่เป็น Master และ Slave, 8255 Programmable Port และ Optical Sensor Board

3. Power Module : ประกอบไปด้วย Power Supply ขนาด +24V และ -10V, Driver Board ของ Stepping Motor ทั้ง 3 ตัว

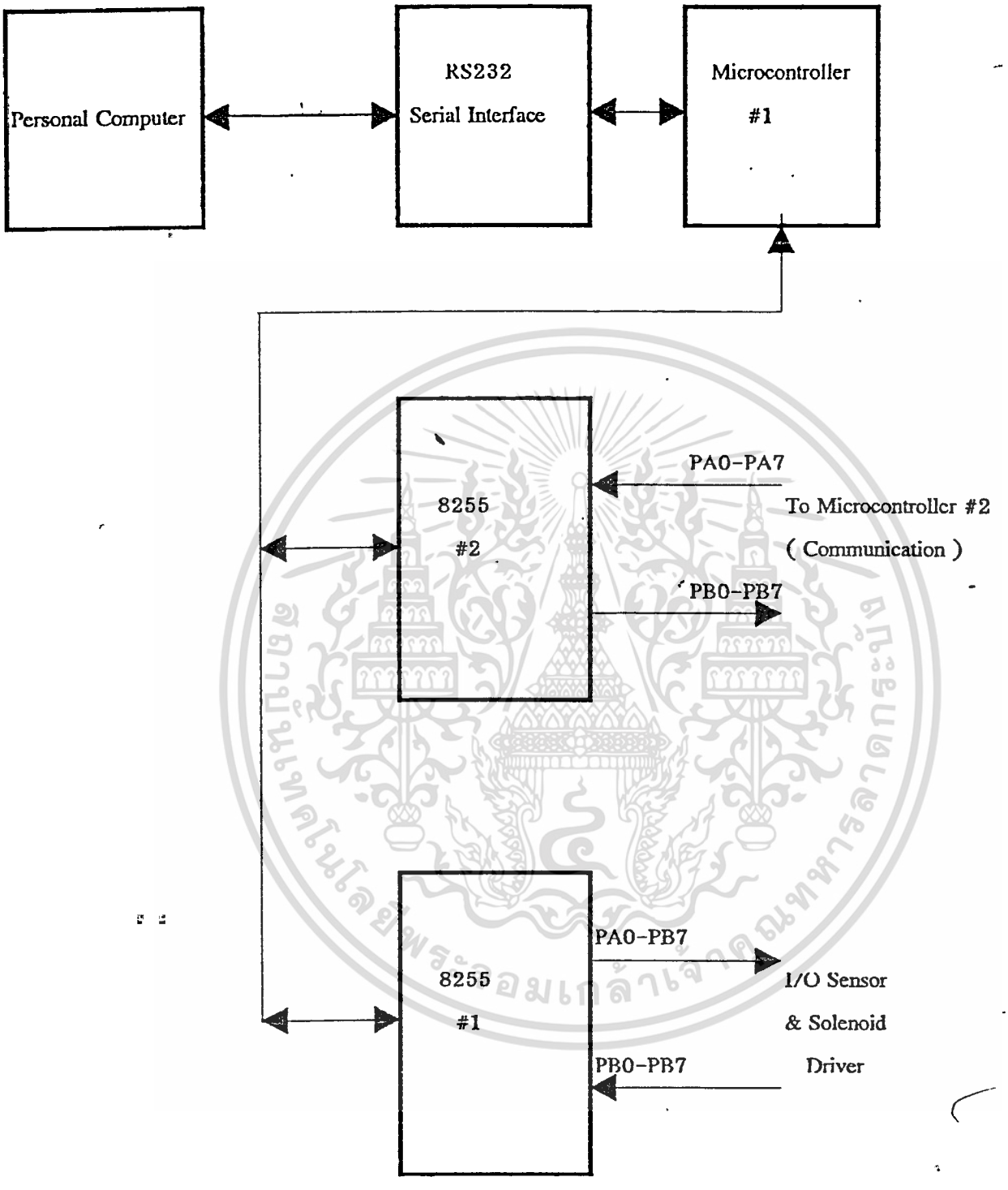
4. ส่วนแขนกล จะเป็นแขนกลที่เคลื่อนไหวแบบพิกัด คาร์ทีเซียน ตามแกน X, แกน Y และแกน Z

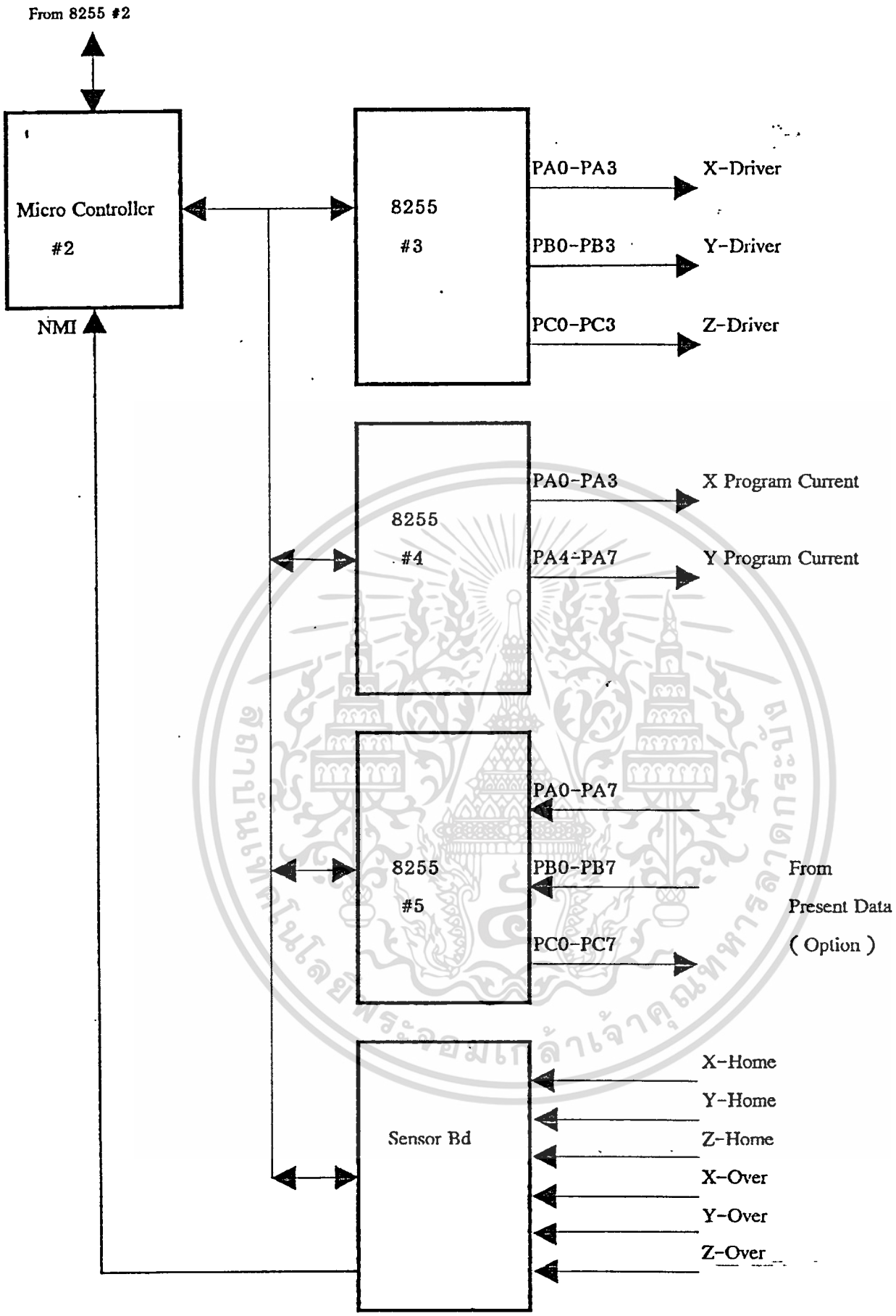
แกน X และ แกน Y ใช้ Stepping Motor ขนาด 2A. 200 Step ต่อรอบ ใช้ Machanical Drive แบบ Screw Shaft ขนาดเส้นผ่าศูนย์กลาง 15.8 mm. ยาว 50 Cm. เกลียวเอียง 45 องศา ส่วนแกน Z ใช้ Motor ขนาด 1A., 200 Step ต่อรอบ ปลายทั้ง 2 ของ แต่ละแกนจะมี Optical Sensor ติดอยู่เพื่อเป็นตัวบอกให้ Controller รู้ว่าตำแหน่งเริ่มต้น (Home) และสิ้นสุด (Over)ของแต่ละแกน ส่วนด้านล่างของแขนกลจะมี Solenoid Driver Board ขนาด 8 Channel ติดตั้งอยู่ในกล่องขนาดเล็ก

3.2 การทำงานและการควบคุม

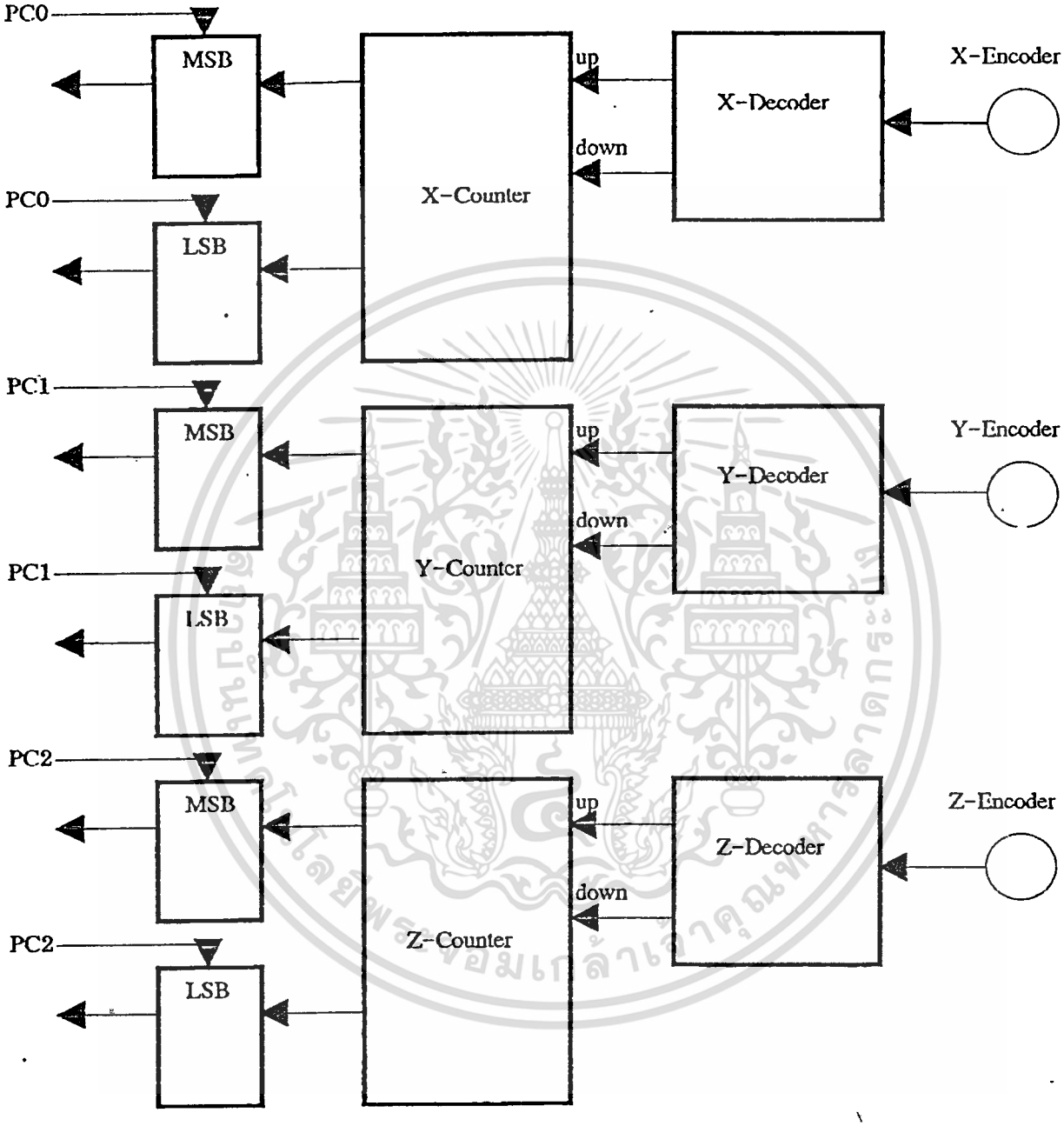
ในบทนี้จะกล่าวถึง การทำงานของเครื่อง 3 Dimension Personal Computer Controller ในแต่ละส่วน รวมถึง Software และการใช้งาน

Block Diagram





Present Data (Option)



จาก Block Diagram

Personal Computer จะส่ง Command ผ่านทาง Serial port ซึ่งเลือกที่ใช้การ Interface แบบนี้ เพราะว่าเราสามารถจะใช้ Personal computer รุ่นใหม่ ๆ ก็ได้ที่มี Serial port ไม่ต้องเปิดเครื่องมาเสียบ Card หลังจากที่ Personal computer ส่ง Command ไปยังชุด Micro controller แล้ว (Micro Controller นี้จะมี CPU อยู่ 2 ตัว ตัวแรกเป็น Master ตัวที่ 2 เป็น Slave) ตัว Master จะมีหน้าที่รับคำสั่งจาก Personal Computer ทาง Serial Port และทำการ Process คำสั่งว่า คำสั่งที่สั่งมาจาก Personal computer นั้นเป็นคำสั่งให้ทำอะไร คำสั่งที่ได้รับจาก Personal computer นั้นจะถูกเปรียบเทียบกับ Command Bank ที่อยู่ใน Rom (Monitor Program) ถ้า Command ที่ใช้ถูกต้องก็จะ Process คำสั่งต่อไปจนหมด Command line ถ้าเกิดมีการเขียนผิด Master CPU จะสั่ง Load Error Code ไปยัง Personal Computer เพื่อ display เป็น Syntax Error หรือถ้ากรณีที่ค่า X,Y,Z ค่าเกิน ค่าที่กำหนดไว้ จะสั่ง Error code เป็น X Value Error , Y Value Error , Z Value Error

ถ้าเป็นคำสั่งที่เป็นการควบคุมการทำงานของ Motor จะถูกส่งไปให้ CPU ตัวที่เป็น Slave ซึ่งเป็นตัวควบคุมการหมุนของ Motor ทั้ง 3 แกน ทั้งนี้รวมกับ Limit Sensor ด้วย หลังจาก CPU ตัวที่เป็น Slave ได้รับค่า X,Y,Z Co-ordinate ที่ Master CPU ส่งมาแล้วก็จะทำการหมุน Motor เคลื่อนชุด Mechanic แขนกลไปยังตำแหน่งที่ต้องการตามคำสั่ง Master CPU จะมีหน้าที่ Update ค่า X,Y,Z Co-ordinate ไป Display ที่อยู่ข้างกล่องด้วยและส่งค่า X, Y ,Z Co-ordinate ไปให้ Personal Computer เพื่อทำการ Display ที่ Personal Computer อีกทีหนึ่ง ถ้ามีการ Error ที่ได้ทีหลังของตัวควบคุม ไม่ว่าจะเป็นคำสั่งที่ผิด Key ผิด CPU ที่เป็น Master จะสั่งการ Error นี้ไป Display ที่ Personal Computer ด้วย

การทำงานของ Controller ตัวที่ 2 (Slave)

- จะเป็นตัวควบคุมเกี่ยวกับการหมุนของ Motor แกน X แกน Y และแกน Z
- ส่งผ่านข้อมูลไปยัง Master Controller เมื่อ Motor หมุนไปยังตำแหน่งที่ถูกกำหนดโดย Master Controller
- กำเนิดสัญญาณเพื่อส่งให้ Motor Driver
- Program Current ของ Stepping Motor แกน X และแกน Y ว่าในขณะ Motor หมุนต้องการกระแสเท่าใด และขณะที่ Hold ต้องการเท่าใด
- (Option) ตรวจสอบ ตำแหน่งปัจจุบัน (Present Distance Data) โดยใช้ Encoder จะทำให้ Controller รับรู้ถึงความผิดพลาดทางด้าน Mechanic หรือ Step ของ Motor ทำงานผิดพลาด

การทำงานของ Driver

Driver ที่เลือกใช้จะเป็นแบบที่สามารถกำหนด Current ได้โดยการ Program ควบคุม ในขณะที่มอเตอร์ เริ่มต้นการหมุน จะต้องใช้แรงมาก วงจรนี้จะทำหน้าที่ Boost Current ที่จ่ายให้กับ มอเตอร์และหลังจากที่มอเตอร์จะสิ้นสุดการหมุนแล้ว Program Current ก็จะสั่งให้ลดกระแสลง เมื่อ มอเตอร์หยุดนิ่ง (Hold) เพื่อลด Power ที่สูญเสียในมอเตอร์

สำหรับการใช้งาน Stepping Motor จะมีโอกาสสูญเสียจำนวน Step ในสภาวะมี Load มากนั้นได้เลือกวิธีแก้ไขโดยการค่อย ๆ เพิ่มจำนวนรอบความเร็วของมอเตอร์จากต่ำ ไปหาสูง ที่ละน้อยและจะค่อย ๆ ลดจำนวน รอบของมอเตอร์ลงเมื่อเคลื่อนที่ใกล้จะถึงตำแหน่งที่ต้องการ

การตรวจสอบสัญญาณ

X - Home, Y- Home, Z - Home

X - Over, Y - Over, Z - Over

การตรวจสอบสัญญาณในตำแหน่ง

X = 0 , Y = O , Z = 0

Controller จะสั่งการในมอเตอร์แกน X,Y,Z หมุนทวนเข็มนาฬิกาและเมื่อ Sensor " Home" ทำงานจะมีสัญญาณ Interupt แบบ NMI ส่งให้ CPU แล้ว CPU ก็จะตอบสนองโดยการหยุดการหมุน ของมอเตอร์และ ส่งให้มอเตอร์หมุนตามเข็มนาฬิกาอีก 5 Step

3.3 ซอฟท์แวร์และการใช้งาน

เครื่อง 3 Dimension Personal Computer Controller นี้เป็นเครื่องที่ใช้ Personal Computer ควบคุมการทำงานของ Stepping Motor ของแกน X ,Y และ Z การทำงานจะแบ่งเป็น 2 ส่วน คือส่วนของ Personal Computer และส่วนของ Micro Controller ซึ่งส่วนของ Micro Controller จะติดต่อกับ Personal Computer ทาง Serial Port และส่วนของ Motor X,Y และ Z จะ Interface กับส่วนของ Micro Controller ซอฟท์แวร์ ที่ใช้ก็ถูกแบ่งออกเป็น 2 ส่วนเช่นกันในส่วนของSoftware ที่ใช้บน Personal Computer จะใช้ภาษา C และ Software ที่ใช้กับ Micro Controller ใช้ภาษา Assembly ของ CPU Z-80

Command ที่ใช้ใน 3 Dimension Personal Computer Controller

1. MA-X0000,Y0000 <CR>
2. MR-X0000,Y0000 <CR>
3. MA-Z0000 <CR>
4. MR-Z0000 <CR>
5. IN-0 <CR>
6. OUT-0 <CR>
7. OUTO-0 <CR>
8. OUTI-0 <CR>
9. AND-0,0 <CR> ---------- Option
10. OR-0,0 <CR> ---------- Option
11. DL7-0000 <CR>
12. LOAD-FILENAME . FILETYPE <CR>
13. RUN <CR>

แต่ละคำสั่งมีความหมายดังนี้

1. MA-X0000,Y0000 <CR> (MOVE ABSOLUTE XY)

เป็นคำสั่งบอกให้ X และ Y Motor หมุนและทำการเคลื่อนที่แขนกลไปยังจุดที่มีค่า Co-Ordinate ที่ตามหลังค่า X และ Y ดังตัวอย่าง

MA-X123,Y5678 <CR> เป็นการให้เคลื่อนที่ชุดแขนกลไปยัง Co-ordinate X=1234 และ Y=5678 ซึ่งค่าของ X และ Y จะมีค่าไม่เกิน 9999 ซึ่งถ้าค่าเกิน 9999 หลังจากที่เครื่อง Micro Controller รับคำสั่งจาก Personal Computer ไปแล้วจะส่งค่า Error

กลับมาที่ Personal Computer เพื่อ Display เป็น X Value Error หรือ Y Value Error

2. MR-X0000,Y0000 <CR> (Move Relative XY)

เป็นคำสั่งที่บอกให้ X และ Y Motor หมุนและทำการเคลื่อนที่ชุดแขนกลไปเป็นแบบค่าต่อเนื่อง ต่อจากค่า Co-ordinate เดิมไปเท่ากับค่า X และ Y ที่กำหนดในคำสั่งตัวอย่างเช่น

MR-X1234 , Y5678 <CR>

สมมติว่าเดิม X และ Y มีค่า Co-ordinate X=1000, Y=1000 หลังจากได้รับคำสั่งข้างบนค่าใหม่ของ X จะเป็น 1000+1234 = 2234 และค่าใหม่ของ Y จะเป็น 1000+5678 = 6678 และทั้งนี้ทั้งนั้นค่าของ X และ Y จะต้องไม่เกิน 9999 ซึ่งถ้าเกินค่านี้ หลังจากเครื่อง Micro Computer รับคำสั่งไปแล้ว ก็จะส่งค่า Error กลับมาที่ Personal Computer เพื่อ Display เป็น X Value Error หรือ 7 - Value Error

3. MA-Z0000 <CR> (Move Absulute Z)

เป็นคำสั่งที่บอกให้ Motor แกน Z (แนวดิ่ง) หมุนและทำการเคลื่อนชุด Machanic ไปยังจุดที่มีค่า Z ตามคำสั่ง ตัวอย่างเช่น

MA-Z1234 <CR> เป็นการให้ชุด Mechanic ของแกน Z (แนวดิ่ง) ทำการเคลื่อนที่ไปยังจุดที่มีค่า Z = 1234 ซึ่งค่าสูงสุดในแกน Z จะมีค่า = 3000 ถ้าค่าเกิน 3000 หลังจากที่คำสั่งส่งไปยัง Micro Controller แล้ว Micro Controller จะสั่ง Error กลับมาเป็น Z Value Error แสดงที่จอภาพของ Personal Computer

4. MR-Z0000 <CR> (Move Relative Z)

เป็นคำสั่งที่บอกให้ Motor แกน Z (แนวดิ่ง) หมุนและทำการเคลื่อนชุด Machanic ไปเป็นแบบต่อเนื่อง ต่อจากค่าเดิม ตัวอย่างเช่น

MR-Z1234 <CR>

สมมติเดิมค่าในแกน Z มีค่า = 1000 หลังจากทำคำสั่งข้างบนแล้ว แกน Z จะมีค่าเท่ากับ 1000 + 1234 = 2234 ทั้งนี้ทั้งนั้นแกน Z จะมีค่าได้ไม่เกิน 3000 ถ้ามีค่าเกินค่านี้แล้ว หลังจากที่ Micro Controller รับคำสั่งจาก Personal Computer จะสั่ง Error กลับมาที่ Personal Computer เป็น Z Value Error

5. IN-0 <CR>

เป็นคำสั่งที่บอกให้ชุด Micro computer รับข้อมูลที่ Input port ซึ่งจะมีค่า 0-9 มาเก็บไว้ที่ Input port Buffer เพื่อรอการ Process ต่อไป ตัวอย่างเช่น

IN-1 <CR> หมายถึงรับข้อมูลเข้าทาง Input 1

6. OUT-0 <CR>

เป็นคำสั่งที่บอกให้ชุด Micro computer ส่งข้อมูลของ Input port Buffer ออกไปยัง Port 6-9 เพื่อไปควบคุมการทำงานของเอนกประสงค์ ตัวอย่างเช่น

OUT-1 <CR> หมายถึงส่งข้อมูลจาก Input port Buffer ออกไปทาง Output 1

7. OUTO-0 <CR>

เป็นคำสั่งที่บอกให้ชุด Micro Controller ส่ง Signal Logic 0 ออกทาง Output port 0-9 ดังตัวอย่าง

OUTO-1 <CR> หมายถึง ส่ง Signal Logic 0 ออกทาง Output 1

8. OUTI-0 <CR>

เป็นคำสั่งที่บอกให้ชุด Micro computer ส่ง Signal Logic 1 ออกทาง Output port 0-9 ดังตัวอย่าง

OUTI-1 <CR> หมายถึง ส่ง Signal Logic 1 ออกทาง Output 1

9. DL7-0000 <CR>

เป็นคำสั่งที่บอกให้ชุด Micro computer ทำ Loop delay time ค่า Delay time จะมีค่าอยู่ระหว่าง 0-9999 msec ดังตัวอย่างเช่น

DL7-1234 <CR> หมายถึงให้ Micro computer ทำ Loop delay time นานเท่ากับ 1234 msec

10. LOAD-FILENAME . FILETYPE <CR>

เป็นคำสั่งที่ให้ Personal Computer ทำการ Load file ที่ระบุ Filename Filetype ตามคำสั่งลงใน Memory

ของ Personal Computer ที่จองไว้ เพื่อที่จะรอส่งไปยัง Micro Controller ที่ละคำสั่ง ตัวอย่างเช่น

LOAD-FRILL . TXT <CR>

11. RUN <CR>

เป็นคำสั่งที่ให้ PC ทำการ Run file ที่ถูก Load ด้วยคำสั่ง Load โดยวิธีการ Run ก็คือนำคำสั่งที่อยู่ใน Memory ของ PC ที่ถูก Load จาก File ส่งไปยัง Micro Controller ทีละคำสั่งจนกว่าจะ End of file

บทที่ 4

สรุปผลและวิจารณ์โครงงาน

โครงงาน 3 Dimension Personal Computer Controller ที่ได้จัดทำขึ้นในครั้งนี้ เป็นเพียงพื้นฐานการควบคุมแขนกล ที่สามารถนำไป ประยุกต์และพัฒนาให้สามารถทำงานได้จริง โดยการต่อแกน Z ให้เป็นหัวจับชิ้นงาน, สว่านใช้เจาะ และ อื่น ๆ ตามความเหมาะสม เพราะทางผู้จัดสร้างโดยเตรียม Driver Board สำหรับ Solenoid หรือ Switch ต่าง ๆ ได้อีก 8 Channel

และการเคลื่อนที่ของ แต่ละแกนยังไม่มีการ Feed Back กลับไปว่า Machanic เคลื่อนที่ไปไปตามที่ Controller สั่งให้จริงหรือไม่ แต่หากผู้สร้างได้เล็งเห็นจุดบกพร่องข้อนี้แล้ว จึงได้ทำส่วนที่เป็น Decoder สำหรับใช้ติดตั้ง Encoder เพื่อให้มีการ Feed Back ตำแหน่งจริง ๆ กลับเข้าไปบอก Controller ได้

ภาคผนวก

```
#define PORT      1
#define ESC       27
#define CR        13
#define SPACE     32
#define BKSP      8

#define NO_ERR    0
#define ILL_CMD   1
#define SYN_ERR   2
#define XV_ERR    3
#define YV_ERR    4
#define ZV_ERR    5

#define HOME      71
#define M_LEFT    77
#define M_RIGHT   75
#define M_UP      72
#define M_DOWN    80
#define Z_UP      73
#define Z_DOWN    81



#include <dos.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>


void port_init(),sport();


main()
{
  char rbuff[80];
  char cmdbuf[80];
  char *cmdptr;
  char chr;
  char c[80];
  int i;

  port_init(PORT,227); /* baud rate 9600,no parity,1 stop bit, 8 data bit *
  clrscr();
  printf("    3 D   P E R S O N A L   C O M P U T E R    C O N T R O L L E
  printf("                                               X =           \n\n
  printf("                                               Y =           \n\n
  printf("                                               Z =


   do
    {
     i=0;
     for (i=0;i<=79;i++)
     cmdbuf[i] = 0;
     i=0;
     gotoxy(5,20);
     printf("COMMAND : ");
     do
      {

  cmdbuf[i] = getch();

      switch(cmdbuf[i])
       {
     case CR:
     case ESC:
      break;
```

```
      case BKSP:
       if(i == 0)
          break;
       i--;
       printf("\b");
       printf(" ");
       printf("\b");
       break;

      case 0 :
       sport(PORT,cmdbuf[i]);
       cmdbuf[i] = getch();
       if (cmdbuf[i] == M_LEFT || cmdbuf[i] == M_RIGHT ||
           cmdbuf[i] == M_UP   || cmdbuf[i] == M_DOWN  ||
           cmdbuf[i] == Z_UP   || cmdbuf[i] == Z_DOWN)
          {
           sport(PORT,cmdbuf[i]);
           xyzupd(PORT);
           };
       if (cmdbuf[i] == HOME)
          {
           sport(PORT,cmdbuf[i]);
           xyzerr(PORT);
           }
       break;
      default:
       printf("%c",cmdbuf[i]);
       i++;
       break;

      }



       } while (cmdbuf[i] != CR && cmdbuf[i] != ESC);

        i=0;
        do
     {
       sport(PORT,cmdbuf[i]); i++;
     } while (cmdbuf[i-1] != CR);

    xyzerr(PORT);
    gotoxy(14,20);
      printf("
     } while ( cmdbuf[0] != ESC);                    /* end doloop 1 */
}   /* main loop */

/* ======================== */

void port_init(port,code)
int port;
unsigned char code;
{
  union REGS r;
  r.x.dx = port;
  r.h.ah = 0;
  r.h.al = code;
  int86 (0x14,&r,&r);
}

/* ======================== */

void sport(port,c)
int port;
char c;
```

```
{
  union REGS r;
  r.x.dx = port;
  r.h.al = c;
  r.h.ah = 1;
  int86 (0x14,&r,&r);
  if (r.h.ah & 128) {
    printf ("send error detected in serial port");
    exit(1);
    }
}

/* ======================== */
rport(port)
int port;
{
  union REGS r;

  while(!(check_stat(PORT)&256))
   if(kbhit()){
      getch();
      exit(1);
      }

   r.x.dx = port;
   r.h.ah = 2;
   int86(0x14,&r,&r);
   if(r.h.ah & 128)
      printf("read error detected in serial port");
   return r.h.al;
}

/* ============================ */
check_stat(port)
int port;
{
  union REGS r;

  r.x.dx = port;
  r.h.ah = 3;
  int86(0x14,&r,&r);
  return r.x.ax;
}
/* ============================ */
xyzupd(port)
int port;
{
int i;
char rbuff[80];
     for (i=0;i<=79;i++) rbuff[i] = 0;
     i=0;
   do {
       rbuff[i] = rport(port);
   i++;
     } while (rbuff[i-1] != CR);
   if (rbuff[0] == 'X')
    {  gotoxy(54,4);
       printf("           ");
       gotoxy(54,4);
       i=1;
      while (rbuff[i] != CR)
       {
       printf("%c",rbuff[i]); i++;
       }
    }
```

```
   else if (rbuff[0] == 'Y')
    {  gotoxy(54,6);
       printf("           ");
       gotoxy(54,6);
       i=1;
      while (rbuff[i] != CR)
       {
       printf("%c",rbuff[i]); i++;
       }
    }
   else if (rbuff[0] == 'Z')
    {  gotoxy(54,8);
       printf("           ");
       gotoxy(54,8);
       i=1;
      while (rbuff[i] != CR)
       {
       printf("%c",rbuff[i]); i++;
       }
    }
}
/* ============================ */
xyzerr(port)
int port;
{
int i;
char rbuff[80];
     for (i=0;i<=79;i++) rbuff[i] = 0;
     i=0;
   do {
       rbuff[i] = rport(port);
       i++;
     } while (rbuff[i-1] != '.');
   if (rbuff[0] == 'X')
    {  gotoxy(54,4);
       printf("           ");
       gotoxy(54,4);
       i=1;
      while (rbuff[i] != CR)
       {
       printf("%c",rbuff[i]); i++;
       }
       i++;
       if (rbuff[i] == 'Y')
        {  gotoxy(54,6);
      printf("           ");
      gotoxy(54,6);
      i++;
      while (rbuff[i] != CR)
          {
           printf("%c",rbuff[i]); i++;
          }
        }
       i++;
       if (rbuff[i] == 'Z')
        {  gotoxy(54,8);
      printf("           ");
      gotoxy(54,8);
      i++;
      while (rbuff[i] != CR)
          {
           printf("%c",rbuff[i]); i++;
          }
      }
    }
   else if (rbuff[0] == 'Y')
```

```
    {  gotoxy(54,6);
       printf("           ");
       gotoxy(54,6);
       i=1;
      while (rbuff[i] != CR)
       {
       printf("%c",rbuff[i]); i++;
       }
       i++;
       if (rbuff[i] == 'X')
        {  gotoxy(54,4);
      printf("           ");
      gotoxy(54,4);
      i++;
      while (rbuff[i] != CR)
          {
           printf("%c",rbuff[i]); i++;
          }
        }

    }
  else if (rbuff[0] == 'Z')
    {  gotoxy(54,8);
       printf("           ");
       gotoxy(54,8);
       i=1;
      while (rbuff[i] != CR)
       {
       printf("%c",rbuff[i]); i++;
       }
    }
  else if (rbuff[0] == ILL_CMD)
    {  gotoxy(11,18);
       printf("                                                    ");
       gotoxy(11,18);
       printf("*** ILLEGAL COMMAND ***");
    }
  else if (rbuff[0] == SYN_ERR)
    {  gotoxy(11,18);
       printf("                                                    ");
       gotoxy(11,18);
       printf("*** SYNTAX ERROR ***");
    }
  else if (rbuff[0] == XV_ERR)
    {  gotoxy(11,18);
       printf("                                                    ");
       gotoxy(11,18);
       printf("*** X VALUE ERROR ***");
    }
  else if (rbuff[0] == YV_ERR)
    {  gotoxy(11,18);
       printf("                                                    ");
       gotoxy(11,18);
       printf("*** Y VALUE ERROR ***");
    }
  else if (rbuff[0] == ZV_ERR)
    {  gotoxy(11,18);
       printf("                                                    ");
       gotoxy(11,18);
       printf("*** Z VALUE ERROR ***");
    }
  else if (rbuff[0] == '.')
    {  gotoxy(11,18);
       printf("                                                    ");
    }
}
```

```
;       ******************************************
;       *    XYZ MOTOR CONTROLLER MONITOR PROGRAM   *
;       *      INTERFACE WITH PERSONAL COMPUTER     *
;       *         BY S.  SAKDA      AUG 27,93       *
;       ******************************************
;
        CPU   "Z80.TBL"
        HOF   "INT16"
;
        ORG   0000H
PDATA:    EQU   60H                ;PA- P8255 LCD
PSIGN:    EQU   61H                ;PB- P8255 LCD
KIN:      EQU   62H                ;PC- P8255 LCD
PCONT:    EQU   63H                ;Control port for P8255 LCD
BDATA:    EQU   40H                ;PA- P8255 interface to SLAVE CPU
PORTB:    EQU   41H                ;PB- P8255 interface to SLAVE CPU
HCHK:     EQU   42H                ;PC- P8255 interface to SLAVE CPU
PCONT1:   EQU   43H                ;Control port for P8255 interface
IPORT:    EQU   70H                ;ASSUME
OPORT:    EQU   71H                ;ASSUME
CTRSR:    EQU   0A1H               ;Control port for 8251
IOSERL:   EQU   0A0H               ;IN/OUT port for 8251
XMAX:     EQU   1500               ;X MAX-value
YMAX:     EQU   1180               ;Y MAX-value
ZMAX:     EQU   150                ;Z MAX-value
MOTOR_X:        EQU   40H          ; MOTOR X  USE PA0-PA3.
MOTOR_Y:        EQU   41H          ; MOTOR Y  USE PB0-PC3.
MOTOR_Z:        EQU   42H          ; MOTOR Z  USE PC0-PC3.
MOTOR_INIT:     EQU   43H          ; MOTOR INITIALIZE
PATTN:          EQU   33H          ; OUTPUT PATTN MX,MY,MZ.
DELAY_DATA:     EQU   01FFH        ; DELAY TIME MOTOR
INIT_PORT1:     EQU   80H
;
        LD    HL,0000H
LP:       DEC   HL
        LD    A,H
        OR    L
        JR    NZ,LP
;
        LD    A,PATTN
        LD    (XPATT),A
        LD    (YPATT),A
        LD    (ZPATT),A
;
        LD    A,0
        LD    (XCACT),A
        LD    (XCACT+1),A
        LD    (YCACT),A
        LD    (YCACT+1),A
        LD    (ZCACT),A
        LD    (ZCACT+1),A
;
;*** INITIAL 8255 ****
        LD    A,89H              ;PA,PB=O/P,PC=I/P
        OUT   (PCONT),A
;
;*** INITIAL 8255 ***
;
        LD    A,INIT_PORT1
        OUT   (MOTOR_INIT),A
;
;*** INITIAL 8251 ****
        LD    A,4EH              ;1 stop bit,no parity,8 bit data,baud rate
        OUT   (CTRSR),A          ;16x
        LD    A,27H              ;TXRDY,INT
```

```
         OUT   (CTRSR),A
        DI
        LD    SP,SYSSTK          ;Set stack
        CALL  INITLCD            ;Initialize LCD
        CALL  WHEAD
        CALL  CLRBUF
MAIN:    CALL  CLRCOM             ;clear buffer
        LD    HL,COMBUF          ;set COMMAND BUFFER pointer
        LD    (COMPTR),HL
;
LOOP:    CALL  RXBYTE
        LD    A,D
        CP    0
        JP    Z,MVS
        LD    HL,(COMPTR)
        LD    (HL),D
        INC   HL
        LD    (COMPTR),HL
        LD    A,D
        CP    0DH
        JP    Z,COMEX
        JR    LOOP
;
;**************************
;
MVS:     CALL  RXBYTE
        LD    A,D
        CP    4DH
        JP    Z,XPLUS
        CP    4BH
        JP    Z,XMINUS
        CP    48H
        JP    Z,YPLUS
        CP    50H
        JP    Z,YMINUS
        CP    49H
        JP    Z,ZPLUS
        CP    51H
        JP    Z,ZMINUS
        CP    47H
        JP    Z,HOME
        JP    LOOP
;
;=========================
;
XPLUS:   CALL  UPDATE
        LD    HL,(XCBUF)
        CALL  ADDX1
        LD    (XCBUF),HL
        LD    (DBUF),HL
        CALL  DECML4
        LD    HL,DBUF
        LD    A,0CH
        CALL  WRLINE
        LD    D,"X"
        CALL  TXBYTE
        LD    HL,DBUF
        CALL  TXMSG
        JP    LOOP
;
;=========================
;
XMINUS:  CALL  UPDATE
        LD    HL,(XCBUF)
        CALL  SUB1
```

```
          LD    (XCBUF),HL
          LD    (DBUF),HL
          CALL  DECML4
          LD    HL,DBUF
          LD    A,0CH
          CALL  WRLINE
          LD    D,"X"
          CALL  TXBYTE
          LD    HL,DBUF
          CALL  TXMSG
          JP    LOOP
;
;=========================
;
YPLUS:    CALL  UPDATE
          LD    HL,(YCBUF)
          CALL  ADDY1
          LD    (YCBUF),HL
          LD    (DBUF),HL
          CALL  DECML4
          LD    HL,DBUF
          LD    A,4CH
          CALL  WRLINE
          LD    D,"Y"
          CALL  TXBYTE
          LD    HL,DBUF
          CALL  TXMSG
          JP    LOOP
;
;=========================
;
YMINUS:   CALL  UPDATE
          LD    HL,(YCBUF)
          CALL  SUB1
          LD    (YCBUF),HL
          LD    (DBUF),HL
          CALL  DECML4
          LD    HL,DBUF
          LD    A,4CH
          CALL  WRLINE
          LD    D,"Y"
          CALL  TXBYTE
          LD    HL,DBUF
          CALL  TXMSG
          JP    LOOP
;
;=========================
;
ZPLUS:    CALL  UPDATE
          LD    HL,(ZCBUF)
          CALL  ADDZ1
          LD    (ZCBUF),HL
          LD    (DBUF),HL
          CALL  DECML4
          LD    HL,DBUF
          LD    A,1CH
          CALL  WRLINE
          LD    D,"Z"
          CALL  TXBYTE
          LD    HL,DBUF
          CALL  TXMSG
          JP    LOOP
;
;=========================
;
```

```
ZMINUS:  CALL UPDATE
         LD   HL,(ZCBUF)
         CALL SUB1
         LD   (ZCBUF),HL
         LD   (DBUF),HL
         CALL DECML4
         LD   HL,DBUF
         LD   A,1CH
         CALL WRLINE
         LD   D,"Z"
         CALL TXBYTE
         LD   HL,DBUF
         CALL TXMSG
         JP   LOOP
;
;=========================
;
HOME:     LD   A,0
         LD   (XCBUF),A
         LD   (XCBUF+1),A
         LD   (YCBUF),A
         LD   (YCBUF+1),A
         LD   (ZCBUF),A
         LD   (ZCBUF+1),A
         CALL UPDATX
         CALL UPDATY
         CALL UPDATZ
         LD   D,"."
         CALL TXBYTE
         JP   LOOP
;
;=== COMMAND EXECUTE SUB. ===
;
COMEX:    LD   HL,COMBUF
         LD   A,50H
         CALL WRLINE
         CALL COMCMP
         LD   A,C
         CP   0
         JP   Z,ILL
         CP   1
         JP   Z,MA
         CP   2
         JP   Z,MR
         CP   3
         JP   Z,IN
         CP   4
         JP   Z,OUT
         CP   5
         JP   Z,OUT0
         CP   6
         JP   Z,OUT1
         CP   7
         JP   Z,AND1
         CP   8
         JP   Z,OR1
         CP   9
         JP   Z,DLY
NOERR:    LD   D,"."
         CALL TXBYTE
         JP   MAIN
;
ILL:      LD   D,01H
         CALL TXBYTE
         LD   D,0DH
```

```
          CALL TXBYTE
          LD   D,"."
          CALL TXBYTE
          JP   MAIN
;
;=========================
;
MA:        INC  DE
          LD   A,(DE)
          INC  DE                ;Increment HL to next character
          CP   "X"
          JP   Z,GETAX
          CP   "Y"
          JP   Z,GETAY
          CP   "Z"
          JP   Z,GETAZ
          JP   ILL
;
MR:        INC  DE
          LD   A,(DE)
          INC  DE                ;Increment HL to next character
          CP   "X"
          JP   Z,GETRX
          CP   "Y"
          JP   Z,GETRY
          CP   "Z"
          JP   Z,GETRZ
          JP   ILL
;
;
GETAX:     CALL DECBIN
          LD   (XPBUF),HL
          PUSH HL
          LD   BC,XMAX
          SBC  HL,BC
          JR   NC,AXER
          POP  HL
          JP   AX1
AXER:      POP  BC
SXERR:     LD   D,3
          CALL TXBYTE
          LD   D,0DH
          CALL TXBYTE
          LD   D,"."
          CALL TXBYTE
          JP   MAIN
AX1:        LD   BC,(XCBUF)
          XOR  A
          SBC  HL,BC
          JR   C,AA
          JR   BB
AA:        LD   A,L
          CPL  A
          LD   L,A
          LD   A,H
          CPL  A
          LD   H,A
          LD   BC,1
          ADD  HL,BC
          SETT 7,H
BB:        LD   (XCACT),HL
          LD   HL,(XPBUF)
          LD   (XCBUF),HL        ;save X values into XCBUF
          LD   A,(DE)            ;use DE as pointer instead of HL
          CP   0DH               ;check [CR]
```

```
          JP    Z,UPDATA
          CP    ","
          JP    NZ,ILL
          INC   DE
          LD    A,(DE)
          CP    "Y"
          JP    NZ,ILL
          INC   DE
          CALL  DECBIN
          LD    (YPBUF),HL
          PUSH  HL
          LD    BC,YMAX
          SBC   HL,BC
          JR    NC,AYER
          POP   HL
          JP    AY1
AYER:     POP   BC
SYERR:    LD    D,4
          CALL  TXBYTE
          LD    D,0DH
          CALL  TXBYTE
          LD    D,"."
          ;Z MAX-valu
          JP    MAIN
AY1:        LD    BC,(YCBUF)
          XOR   A
          SBC   HL,BC
          JR    C,CC
          JR    DD
CC:       LD    A,L
          CPL   A
          LD    L,A
          LD    A,H
          CPL   A
          LD    H,A
          LD    BC,1
          ADD   HL,BC
          SETT  7,H
DD:       LD    (YCACT),HL
          LD    HL,(YPBUF)
          LD    (YCBUF),HL      ;save Y values into YCBUF
          JP    UPDATA
;
;
GETAY:    CALL  DECBIN
          LD    (YPBUF),HL
          PUSH  HL
          LD    BC,YMAX
          SBC   HL,BC
          JP    NC,AYER
          POP   HL
          LD    BC,(YCBUF)
          XOR   A
          SBC   HL,BC
          JR    C,EE
          JR    FF
EE:       LD    A,L
          CPL   A
          LD    L,A
          LD    A,H
          CPL   A
          LD    H,A
          LD    BC,1
          ADD   HL,BC
          SETT  7,H
```

```
FF:      LD   (YCACT),HL
         LD   HL,(YPBUF)
         LD   (YCBUF),HL      ;save Y values into YCBUF
         LD   A,(DE)          ;use DE as pointer instead of HL
         CP   0DH             ;check [CR]
         JP   Z,UPDATA
         CP   ","
         JP   NZ,ILL
         INC  DE
         LD   A,(DE)
         CP   "X"
         JP   NZ,ILL
         INC  DE
         CALL DECBIN
         LD   (XPBUF),HL
         PUSH HL
         LD   BC,XMAX
         SBC  HL,BC
         JP   NC,AXER
         POP  HL
         LD   BC,(YCBUF)
         XOR  A
         SBC  HL,BC
         JR   C,GG
         JR   HH
GG:      LD   A,L
         CPL  A
         LD   L,A
         LD   A,H
         CPL  A
         LD   H,A
         LD   BC,1
         ADD  HL,BC
         SETT 7,H
HH:      LD   (YCACT),HL
         LD   HL,(XPBUF)
         LD   (XCBUF),HL      ;save X values into XCBUF
         JP   UPDATA
;
;
GETAZ:   CALL DECBIN
         LD   (ZPBUF),HL
         PUSH HL
         LD   BC,ZMAX
         SBC  HL,BC
         JP   NC,UPDATA
         POP  HL
         JP   AZ1
AZER:    POP  BC
SZERR:   LD   D,5
         CALL TXBYTE
         LD   D,0DH
         CALL TXBYTE
         LD   D,"."
         CALL TXBYTE
         JP   MAIN
AZ1:     LD   BC,(ZCBUF)
         XOR  A
         SBC  HL,BC
         JR   C,II
         JR   JJ
II:      LD   A,L
         CPL  A
         LD   L,A
         LD   A,H
```

```
          CPL  A
          LD   H,A
          LD   BC,1
          ADD  HL,BC
          SETT 7,H
JJ:        LD   (YCACT),HL
          LD   HL,(XPBUF)
          LD   (ZCBUF),HL      ;save Z values into ZCBUF
          JP   UPDATA
;
;
GETRX:    CALL DECBIN
          LD   (XCACT),HL
          LD   BC,(XCBUF)      ;get XCBUF
          PUSH BC              ;save old values
          ADD  HL,BC
          PUSH HL              ;save new values
          LD   BC,XMAX         ;check X-max limit
          SBC  HL,BC
          JR   C,NXERR         ;if the value less than the max limit,no error
          POP  HL              ;new value
          POP  HL              ;get old value
          LD   HL,XERRM        ;display error massage
          LD   A,50H
          CALL WRLINE
          JP   SXERR
NXERR:    POP  HL              ;get new value
          POP  BC              ;old value
;
          LD   (XCBUF),HL      ;save X values into XCBUF
          LD   A,(DE)          ;use DE as pointer instead of HL
          CP   0DH             ;check [CR]
          JP   Z,UPDATA
          CP   ","
          JP   NZ,ILL
          INC  DE
          LD   A,(DE)
          CP   "Y"
          JP   NZ,ILL
          INC  DE
          CALL DECBIN
          LD   (YCACT),HL
          LD   BC,(YCBUF)      ;get YCBUF
          PUSH BC              ;save old values
          ADD  HL,BC
          PUSH HL              ;save new values
          LD   BC,YMAX         ;check Y-max limit
          SBC  HL,BC
          JR   C,NYERR         ;if the value less than the max limit,no error
          POP  HL              ;new value
          POP  HL              ;get old value
          LD   HL,YERRM        ;display error massage
          LD   A,50H
          CALL WRLINE
          JP   SYERR
NYERR:    POP  HL              ;get new value
          POP  BC              ;old value
          LD   (YCBUF),HL      ;save Y values into YCBUF
          JP   UPDATA
;
;
GETRY:    CALL DECBIN
          LD   (YCACT),HL
          LD   BC,(YCBUF)      ;get YCBUF
          PUSH BC              ;save old values
```

```
          ADD  HL,BC
          PUSH HL.              ;save new values
          LD   BC,YMAX          ;check Y-max limit
          SBC  HL,BC
          JR   C,NYERR1         ;if the value less than the max limit,no error
          POP  HL               ;new value
          POP  HL               ;get old value
          LD   HL,YERRM         ;display error massage
          LD   A,50H
          CALL WRLINE
          JP   SYERR
NYERR1:    POP  HL               ;get new value
          POP  BC               ;old value
;
          LD   (YCBUF),HL       ;save Y values into XCBUF
          LD   A,(DE)           ;use DE as pointer instead of HL
          CP   0DH              ;check [CR]
          JP   Z,UPDATA
          CP   ","
          JP   NZ,ILL
          INC  DE
          LD   A,(DE)
          CP   "X"
          JP   NZ,ILL
          INC  DE
          CALL DECBIN
          LD   (XCACT),HL
          LD   BC,(XCBUF)       ;get XCBUF
          PUSH BC               ;save old values
          ADD  HL,BC
          PUSH HL               ;save new values
          LD   BC,XMAX          ;check X-max limit
          SBC  HL,BC
          JR   C,NXERR1         ;if the value less than the max limit,no error
          POP  HL               ;new value
          POP  HL               ;get old value
          LD   HL,XERRM         ;display error massage
          LD   A,50H
          CALL WRLINE
          JP   SXERR
NXERR1:    POP  HL               ;get new value
          POP  BC               ;old value
          LD   (XCBUF),HL       ;save X values into XCBUF
          JP   UPDATA
;
;
GETRZ:     CALL DECBIN
          LD   (ZCACT),HL
          LD   BC,(ZCBUF)       ;get ZCBUF
          PUSH BC               ;save old values
          ADD  HL,BC
          PUSH HL               ;save new values
          LD   BC,ZMAX          ;check Z-max limit
          SBC  HL,BC
          JR   C,NZERR          ;if the value less than the max limit,no error
          POP  HL               ;new value
          POP  HL               ;get old value
          LD   HL,ZERRM         ;display error massage
          LD   A,50H
          CALL WRLINE
          JP   SZERR
NZERR:     POP  HL               ;get new value
          POP  BC               ;old value
;
          LD   (ZCBUF),HL       ;save Z values into ZCBUF
```

```
UPDATA:  CALL MOTORXY
         CALL MOTORZ
         CALL UPDATX
         CALL UPDATY
         CALL UPDATZ
         LD   D,"."
         CALL TXBYTE
         JP   MAIN
;
;
;**************************
;
IN:       INC  DE
         CALL DECBIN          ;max = 8  input port
         LD   B,L
         LD   C,10000000B
IN1:      RLC  C
         DJNZ IN1
         IN   A,(IPORT)
         AND  C               ;mask bit by following instruction bit
         LD   C,A
         LD   A,(CTRSTT)      ;get status value from memory
         OR   C
         LD   (CTRSTT),A      ;keep to buffer
         JP   MAIN
;
;
OUT:      INC  DE
         CALL DECBIN          ;max = 8  output port
         LD   B,L
         LD   C,10000000B
OUT2:     RLC  C
         DJNZ OUT2
         LD   A,(CTROUT)
         AND  C               ;mask bit by following instruction bit
         OUT  (OPORT),A
         JP   MAIN
;
;
OUT0:     INC  DE
         CALL DECBIN          ;max = 8  output port
         LD   B,L
         LD   A,01111111B
OUT01:    RLC  A
         DJNZ OUT01
         OUT  (OPORT),A
         JP   MAIN
;
;
OUT1:     INC  DE
         CALL DECBIN          ;max = 8  output port
         LD   B,L
         LD   A,10000000B
OUT11:    RLC  A
         DJNZ OUT11
         OUT  (OPORT),A
         JP   MAIN
;
;
AND1:     INC  DE
AND2:     CALL DECBIN           ;max = 8  output port
         LD   B,L
         LD   A,(CTRSTT)
         LD   C,A
AND3:     RRC  C
```

```
          DJNZ AND3
          JR   NC,ANDX
          LD   A,(DE)
          CP   ","
          JP   NZ,LAST
          INC  DE
          JR   AND2
ANDX:      LD   A,0
          LD   (CTROUT),A
          JP   MAIN
LAST:      LD   A,1
          LD   (CTROUT),A
          JP   MAIN
;
;
OR1:       INC  DE
OR2:       CALL DECBIN          ;max = 8  output port
          LD   B,L
          LD   A,(CTRSTT)
          LD   C,A
OR3:       RRC  C
          DJNZ OR3
          JR   C,ORX
          LD   A,(DE)
          CP   ","
          JP   NZ,ORLAST
          INC  DE
          JR   AND2
ORX:       LD   A,1
          LD   (CTROUT),A
          JP   MAIN
ORLAST:    LD   A,0
          LD   (CTROUT),A
          JP   MAIN
;
;
DLY:       CALL DECBIN
          DEC  L
LPP:       LD   B,0FFH
LPP1:      DJNZ LPP1
          DEC  L
          JR   NZ,LPP
          DEC  H
          JR   NZ,LPP
          JP   MAIN
;
;
;=========================
;
CLRBUF:    PUSH BC
          PUSH DE
          PUSH HL
          LD   A,0
          LD   HL,XCBUF              ;Clear all buffer
          LD   DE,XCBUF+1
          LD   BC,120
          LD   (HL),0
          LDIR
          POP  HL
          POP  DE
          POP  BC
          RET
;
;=========================
;
```

```
;*** ADDX1 SUB. ***
;
ADDX1:    PUSH BC
          PUSH HL                    ;Save old value
          LD   BC,1
          ADC  HL,BC
          PUSH HL                    ;Save new value
          LD   BC,XMAX               ;Max X-value
          SBC  HL,BC
          JR   C,EX
          POP  BC
          POP  HL
          POP  BC
          RET
EX:        POP  HL                    ;Get new value
          POP  BC
          POP  BC
          RET
;
;==========================
;
;*** ADDY1 SUB. ***
;
ADDY1:    PUSH BC
          PUSH HL                    ;Save old value
          LD   BC,1
          ADC  HL,BC
          PUSH HL                    ;Save new value
          LD   BC,YMAX               ;Max Y-value
          SBC  HL,BC
          JR   C,EY
          POP  BC
          POP  HL
          POP  BC
          RET
EY:        POP  HL                    ;Get new value
          POP  BC
          POP  BC
          RET
;
;==========================
;
;*** ADDZ1 SUB. ***
;
ADDZ1:    PUSH BC
          PUSH HL                    ;Save old value
          LD   BC,1
          ADC  HL,BC
          PUSH HL                    ;Save new value
          LD   BC,ZMAX               ;Max Z-value
          SBC  HL,BC
          JR   C,EZ
          POP  BC
          POP  HL
          POP  BC
          RET
EZ:        POP  HL                    ;Get new value
          POP  BC
          POP  BC
          RET
;
;*** SUB1 SUB. ***
;
SUB1:     PUSH BC
          PUSH HL                   ;Save old value
```

```
         LD    BC,1                ;Decrement by one
        SBC   HL,BC
        JR    C,SXX
        POP   BC
        POP   BC
        RET
SXX:     POP   HL
        POP   BC
        RET
;
;*** CLEAR SUB.****
;
CLEAR:   LD    A,00H
        CALL GOTO
        RET
;
;*** CLEAR BUFFER SUB. ***
;
CLRBF:    PUSH HL
        PUSH BC
        LD    B,16
        LD    HL,DBUF
CLRBF1:  LD    A,20H
        LD    (HL),A
        INC   HL
        DJNZ CLRBF1
        POP   BC
        POP   HL
        RET
;
;*** CLEAR COMMAND BUFFER SUB. ***
;
CLRCOM:  PUSH HL
        PUSH BC
        LD    B,80
        LD    HL,COMBUF
CLRCM1:  LD    A,20H
        LD    (HL),A
        INC   HL
        DJNZ CLRCM1
        POP   BC
        POP   HL
        RET
;
;*** UPDATE SUB.***
;
UPDATE:  PUSH HL
        LD    HL,DBUF
        LD    (OUTPTR),HL
        POP   HL
        RET
;
;*** UPDATX SUB. ***
;
UPDATX:  PUSH HL
        PUSH DE
        CALL UPDATE
        LD    HL,(XCBUF)
        CALL DECML4
        LD    HL,DBUF
        LD    A,0CH
        CALL WRLINE
        LD    D,"X"
        CALL TXBYTE
        LD    HL,DBUF
```

```
         CALL TXMSG
        POP  DE
        POP  HL
        RET
;
;*** UPDATY SUB. ***
;
UPDATY:  PUSH HL
        PUSH DE
        CALL UPDATE
        LD   HL,(YCBUF)
        CALL DECML4
        LD   HL,DBUF
        LD   A,4CH
        CALL WRLINE
        LD   D,"Y"
        CALL TXBYTE
        LD   HL,DBUF
        CALL TXMSG
        POP  DE
        POP  HL
        RET
;
;*** UPDATZ SUB. ***
;
UPDATZ:  PUSH HL
        PUSH DE
        CALL UPDATE
        LD   HL,(ZCBUF)
        CALL DECML4
        LD   HL,DBUF
        LD   A,1CH
        CALL WRLINE
        LD   D,"Z"
        CALL TXBYTE
        LD   HL,DBUF
        CALL TXMSG
        POP  DE
        POP  HL
        RET
;
;**** INITIAL LCD DISPLAY ****
;PA0-PA7  :PIN D0-D7 (DATA READ/WRITE LCD)
;PB2      :PIN E (ENABLE SIGNAL PULSE)
;PB1      :PIN R/W (READ/WRITE)
;PB0      :PIN RS (REGISTER SELECTION)
;
INITLCD: LD   A,0
        OUT  (PSIGN),A
        LD   A,00111000B        ;function set  38H
        OUT  (PDATA),A          ;DL=1 8 bit,N=1 1/16 duty,F=0 5*7
        CALL EPULSE
        CALL DELAYX             ;DELAY > 4.1 ms
        LD   A,00001111B        ;display on/off control
        OUT  (PDATA),A          ;D=1 off,C=1 cursor on,B=1 blink
        CALL EPULSE
        LD   A,00000110B        ;entry mode set
        OUT  (PDATA),A          ;I/D=1 increment,S=0 right
        CALL EPULSE
        LD   A,00000001B        ;clear all display
        OUT  (PDATA),A
        CALL EPULSE
        CALL DELAYX
        RET
;
```

```
;**** WRITE HEAD ****
;
WHEAD:    LD    HL,HMSG
          LD    A,00H
          CALL  WRLINE
          LD    A,40H
          CALL  WRLINE
          LD    A,10H
          CALL  WRLINE
          LD    A,50H
          CALL  WRLINE
          RET
;
;============================
;
COMCMP:   LD    C,1
          LD    HL,COMTAB       ;set HL point to COMTAB
COMCP1:   LD    DE,COMBUF       ;set DE point to COMBUF
COMCP2:   LD    A,(DE)
          CP    20H             ;compare with space
          RET   Z
          CP    (HL)
          INC   DE
          INC   HL
          JR    NZ,COMCP3
          JR    COMCP2
COMCP3:   LD    A,(HL)
          CP    "."
          JR    Z,COMCP4
          CP    0DH
          INC   HL
          JR    NZ,COMCP3
          INC   C
          JR    COMCP1
COMCP4:   LD    C,0
          RET
;
;**** WRITE LINE 16 CHAR ****
;  INPUT (HL) = DATA
;  INPUT A = DISPLAY ADDRESS
WRLINE:   CALL  GOTO
          AND   0FH
          LD    D,A
          LD    A,16
          SUB   D
          LD    B,A             ;16 CHAR
WRL:      LD    D,(HL)
          PUSH  BC
          CALL  WRBYTE
          POP   BC
          INC   HL
          DJNZ  WRL
          RET
;
;**** ENABLE PULSE SUB. ****
;
EPULSE:   IN    A,(PSIGN)
          SETT  2,A
          OUT   (PSIGN),A       ;enable bit 2=1
          LD    B,00H
EP1:      DJNZ  EP1
          RES   2,A
          OUT   (PSIGN),A       ;enable bit 2=0
          RET
;
```

```
;**** GOTO POSITION ****
;    INPUT REG A=DATA
;
GOTO:     PUSH BC
          PUSH AF
          SETT 7,A                ;set DD RAM
          OUT  (PDATA),A
          XOR  A
          OUT  (PSIGN),A          ;set RS=0,R/W=0
          CALL EPULSE
          POP  AF
          POP  BC
          RET
;
;**** WRITE DATA SUB. ****
;   INPUT REG D=DATA
;
WRBYTE:   LD   A,00000001B        ;data write
          OUT  (PSIGN),A
          LD   A,D                ;data byte
          OUT  (PDATA),A
          CALL EPULSE
          RET
;
;**** DELAYX  SUB. ****
;
DELAYX:    LD   B,0
DE1:      NOP
          NOP
          DJNZ DE1
          RET
;
;*****************************************************************
;Function: Convert ASC II codes to corresponding decimal values in binary
;          until met none decimal digit.
;
DECBIN:   LD   HL,0
          LD   A,(DE)
NDIGIT:   SUB  "0"
          RET  C
          CP   10
          RET  NC
          ADD  HL,HL              ;[HL] = 10*[HL]
          LD   B,H
          LD   C,L
          ADD  HL,HL
          ADD  HL,HL
          ADD  HL,BC
          LD   B,0
          LD   C,A
          ADD  HL,BC
          INC  DE
          LD   A,(DE)
          JR   NDIGIT

;
;*** DECML SUB. ***
;
DECML:    LD   C,0                ;Zero supress flag
CLOOP:    LD   E,(IY+0)
          INC  IY
          LD   D,(IY+0)
          INC  IY
          XOR  A
DECLP:    SBC  HL,DE
```

```
         JR   C,ADDBK
        INC  A
        JR   DECLP
ADDBK:   ADD  HL,DE
        CALL SUPRES
        DJNZ CLOOP
        RET
SUPRES:  AND  A
        JR   Z,YES0          ;If zero then check zero supress flag
        LD   C,A             ;Else
        ADD  A,30H           ;Convert to ASCII code format and output
        JP   CHRWR
YES0:    LD   A,C
        AND  A
        JR   Z,BLANK0        ;Supress leading zero
PRINT0:  LD   A,30H
        JP   CHRWR
BLANK0:  LD   A,B             ;Still check for last digit
        DEC  A
        JR   Z,PRINT0        ;If last digit then print '0'
        LD   A,20H
        JP   CHRWR
;
;*** CHRWR SUB. ****
;
CHRWR:   PUSH HL
        LD   HL,(OUTPTR)
        LD   (HL),A
        INC  HL
        LD   (OUTPTR),HL
        POP  HL
        RET
;
;*** DECML4 SUB. ***
;
DECML4:  LD   IY,TENS4
        LD   B,4
        CALL DECML
        RET
;
;*** DECML3 SUB. ***
;
DECML3:  LD   IY,TENS3
        LD   B,3
        CALL DECML
        RET
;
;*** MPY8 SUB. ***
;Input E,A   Result in HL
;
MPY8:    LD   B,8
        LD   D,0
        LD   H,D
        LD   L,D
LMPY:    ADD  HL,HL
        RLC  A
        JR   NC,NADD
        ADD  HL,DE
NADD:    DJNZ LMPY
        RET
;
;*** RXBYTE SUB ***
;
RXBYTE:  IN   A,(CTRSR)
        BIT  1,A
```

```
        JR    Z,RXBYTE
        IN    A,(IOSERL)
        LD    D,A
        RET
;
;*** TXBYTE SUB ***
;
TXBYTE: IN    A,(CTRSR)
        BIT   0,A
        JR    Z,TXBYTE
        LD    A,D
        OUT   (IOSERL),A
        RET
;
;*** TXMSG SUB ***
;Function : send massage which is pointed by reg HL until met space bar(20H)
;
TXMSG:  LD    B,4
TXM1:   LD    D,(HL)
        CALL  TXBYTE
        INC   HL
        DJNZ  TXM1
        LD    D,0DH
        CALL  TXBYTE
        RET
;
;*******************************
;    TEST DIRECTION  X-AXIS SUB
;*******************************
;
MOTORXY: LD   HL,(XCACT)
       LD    (XDIR),HL
       BIT   7,H
       JP    Z,XX1
       RES   7,H
       LD    (XCACT),HL
       JP    XCCW
XX1:   RES   7,H
       LD    (XCACT),HL
       JP    XCW
       LD    HL,(YCACT)
       LD    (YDIR),HL
       BIT   7,H
       JP    Z,YY1
       RES   7,H
       LD    (YCACT),HL
       JP    YCCW
YY1:   RES   7,H
       LD    (YCACT),HL
       JP    YCW
;
;
XCCW:  LD    HL,(XCACT)
       LD    A,H
       OR    L
       JP    Z,YSTOP
       DEC   HL
       LD    (XCACT),HL
       LD    A,(XPATT)
       OUT   (MOTOR_X),A
       RLCA
       LD    (XPATT),A
       CALL  DELAY
       LD    HL,(YDIR)
       BIT   7,H
       JP    NZ,YCCW
```

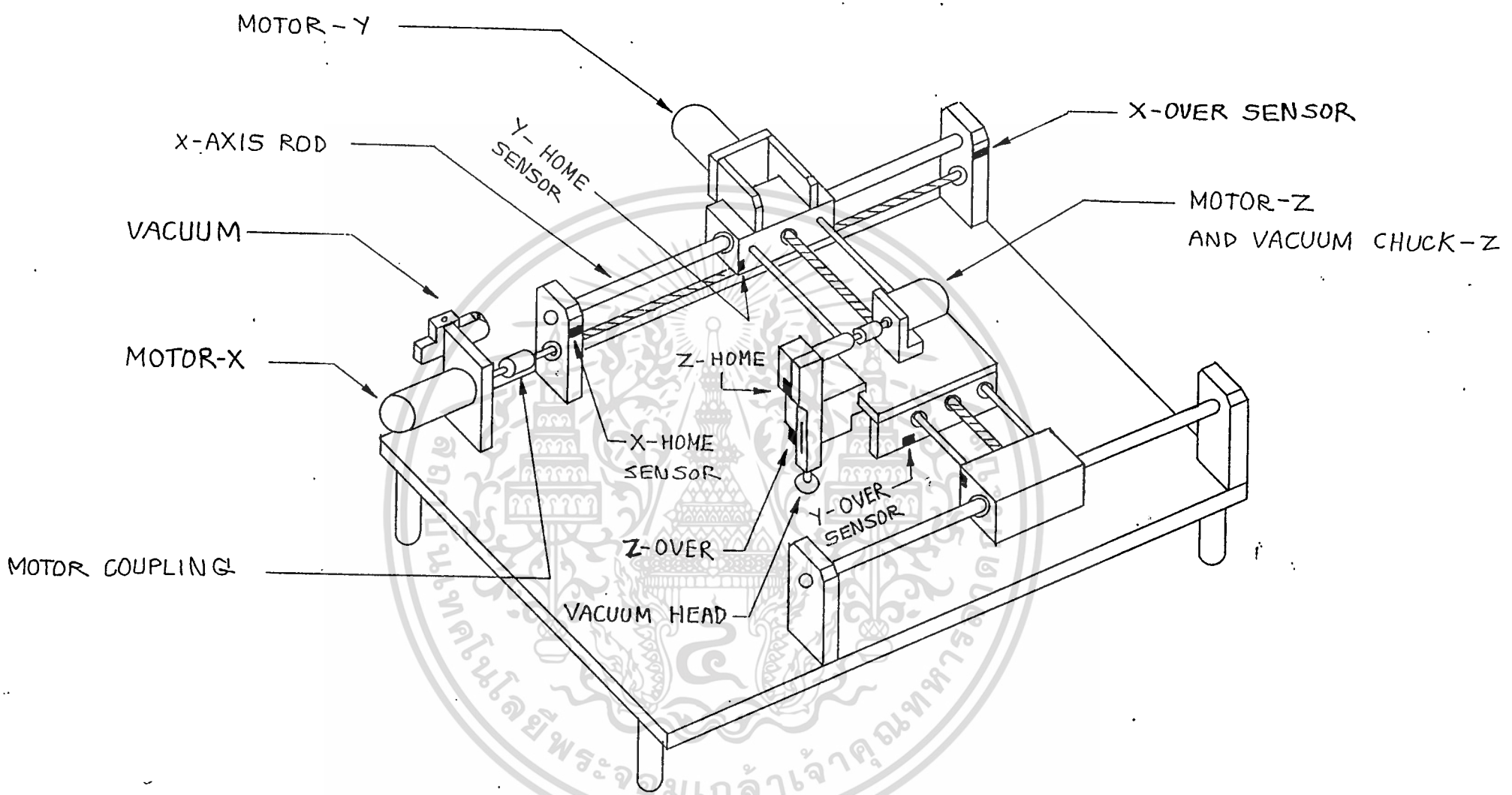
```
        JP    YCW
XCW:    LD    HL,(XCACT)
        LD    A,H
        OR    L
        JP    Z,YSTOP
        DEC   HL
        LD    (XCACT),HL
        LD    A,(XPATT)
        OUT   (MOTOR_X),A
        RRCA
        LD    (XPATT),A
        CALL  DELAY
        LD    HL,(YDIR)
        BIT   7,H
        JP    NZ,YCCW
        JP    YCW
;
YCCW:   LD    HL,(YCACT)
        LD    A,H
        OR    L
        JP    Z,XSTOP
        DEC   HL
        LD    (YCACT),HL
        LD    A,(YPATT)
        OUT   (MOTOR_Y),A
        RLCA
        LD    (YPATT),A
        CALL  DELAY
        LD    HL,(XDIR)
        BIT   7,H
        JP    NZ,XCCW
        JP    XCW
YCW:    LD    HL,(YCACT)
        LD    A,H
        OR    L
        JP    Z,XSTOP
        DEC   HL
        LD    (YCACT),HL
        LD    A,(YPATT)
        OUT   (MOTOR_Y),A
        RRCA
        LD    (YPATT),A
        CALL  DELAY
        LD    HL,(XDIR)
        BIT   7,H
        JP    NZ,XCCW
        JP    XCW
;
YSTOP:  LD    HL,(XCACT)
        LD    A,H
        OR    L
        RET   Z
        LD    HL,(YDIR)
        BIT   7,H
        JP    NZ,YCCW
        JP    YCW
;
XSTOP:  LD    HL,(YCACT)
        LD    A,H
        OR    L
        RET   Z
        LD    HL,(XDIR)
        BIT   7,H
        JP    NZ,XCCW
        JP    XCW
;
```

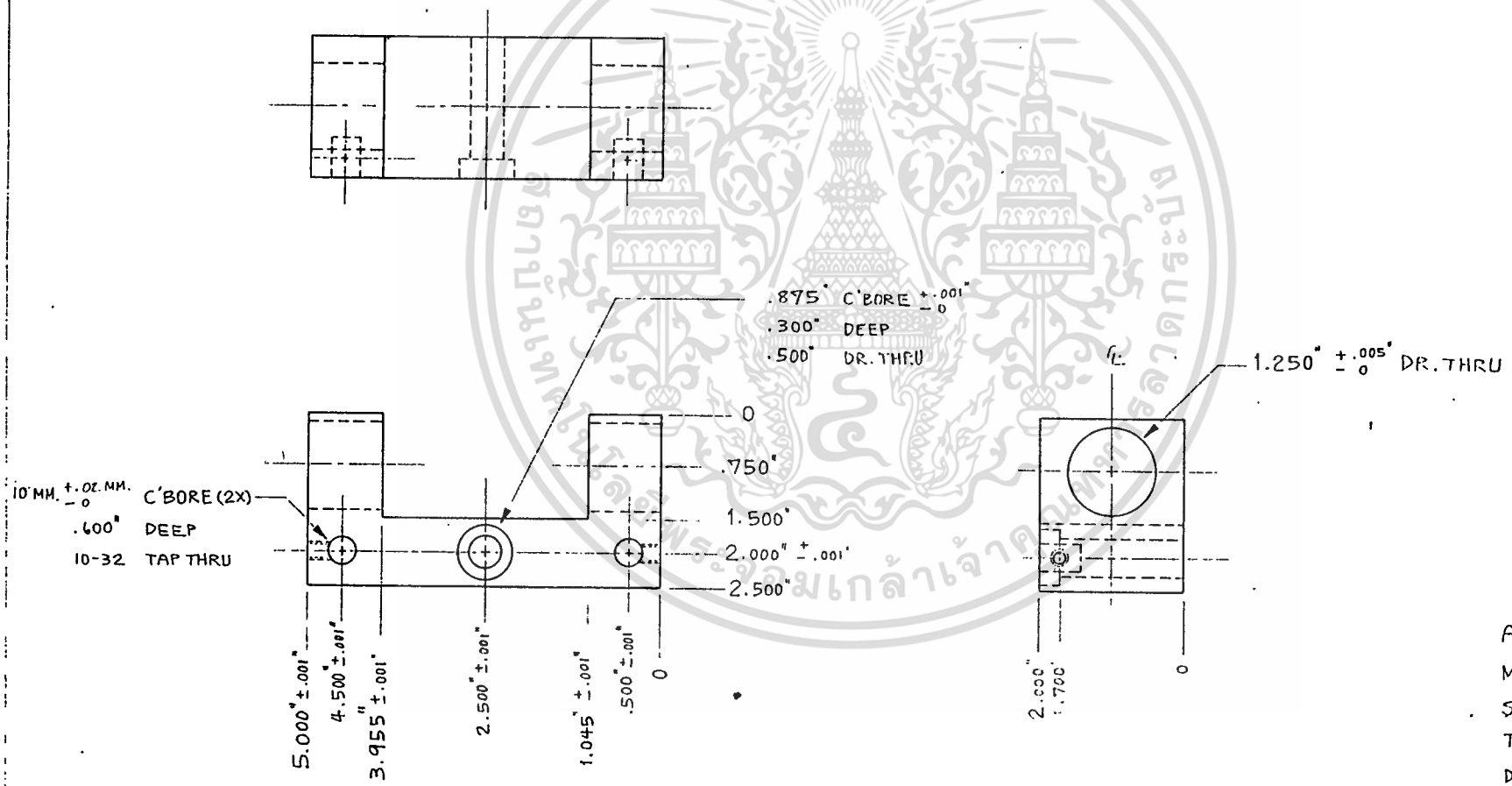
```
;
MOTORZ: LD    HL,(ZCACT)
        LD    (ZDIR),HL
        BIT   7,H
        JP    Z,ZZ1
        RES   7,H
        LD    (ZCACT),HL
        JP    ZCCW
ZZ1:    RES   7,H
        LD    (ZCACT),HL
        JP    ZCW
;
ZCCW:   LD    HL,(ZCACT)
        LD    A;H
        OR    L
        RET   Z
        DEC   HL
        LD    (ZCACT),HL
        LD    A,(ZPATT)
        OUT   (MOTOR_Z),A
        RLCA
        LD    (ZPATT),A
        CALL  DELAY
        JP    ZCCW
ZCW:    LD    HL,(ZCACT)
        LD    A,H
        OR    L
        RET   Z
        DEC   HL
        LD    (ZCACT),HL
        LD    A,(ZPATT)
        OUT   (MOTOR_Z),A
        RRCA
        LD    (ZPATT),A
        CALL  DELAY
        JP    ZCW
;
;
;**************************
;   DELAY SUB
;**************************
;
DELAY:  EX    AF,AF'
      EXX
      LD    HL,DELAY_DATA
DELAY1:  DEC  HL
      LD    A,H
      OR    L
      JR    NZ,DELAY1
      EXX
      EX    AF,AF'
      RET
;
;
;*******************************
;
TENS4:    DWL  1000
TENS3:    DWL  100
          DWL  10
          DWL  1
;
HMSG:     DFB  "3D-CTR          X=      "
LN2:      DFB  "                Y=      "
LN3:      DFB  "                Z=      "
LN4:      DFB  "                        "
XERRM:    DFB  "  X VALUE ERROR "
```

```
YERRM:    DFB   "   Y VALUE ERROR "
ZERRM:    DFB   "   Z VALUE ERROR "
;
COMTAB:   DFB   "MA"
          DFB   0DH
          DFB   "MR"
          DFB   0DH
          DFB   "IN".
          DFB   0DH
          DFB   "OUT"
          DFB   0DH
          DFB   "OUTI"
          DFB   0DH
          DFB   "OUTO"
          DFB   0DH
          DFB   "AND"
          DFB   0DH
          DFB   "OR"
          DFB   0DH
          DFB   "DLY"
          DFB   0DH
          DFB   "."              ;End of command table
;
;**** STORAGE MEMORY ****
          ORG   8000H
;
OUTPTR:   DFS   2                ;Output Buffer pointer
DBUF:     DFS   16               ;Display buffer
POSPTR:   DFS   2                ;Coordinate position pointer
POSCNT:   DFS   2                ;For keep position counter
XCBUF:    DFS   2                ;X axis current buffer
YCBUF:    DFS   2                ;Y axis current buffer
ZCBUF:    DFS   2                ;Z axis current buffer
XCACT:    DFS   2                ;X axis activity
YCACT:    DFS   2                ;Y axis activity
ZCACT:    DFS   2                ;Z axis activity
XPBUF:    DFS   2                ;X axis prior value
YPBUF:    DFS   2                ;Y axis prior value
ZPBUF:    DFS   2                ;Z axis prior value
CTRSTT:   DFS   1                ;Control status buffer
CTROUT:   DFS   1                ;Control output buffer
CTRPT:    DFS   2                ;For keep pointer of CTRBF
COMPTR:   DFS   2                ;For keep pointer of CTRBF
;
XPATT:    DFS   1
YPATT:    DFS   1
ZPATT:    DFS   1
XDIR:     DFS   1
YDIR:     DFS   1
ZDIR:     DFS   1

;
;
COMBUF:   DFS   80               ;For keeping command
BLANK:    DFS   200              ;Area for system stack
SYSSTK:   DFS   20


END
```


MOTOR-Y
X-AXIS ROD
Y-HOME SENSOR
X-OVER SENSOR
VACUUM
MOTOR-Z AND VACUUM CHUCK-Z
MOTOR-X
Z-HOME
X-HOME SENSOR
Y-OVER SENSOR
Z-OVER
MOTOR COUPLING
VACUUM HEAD



AXIS-Y BRACKET 1

MAT : A_L

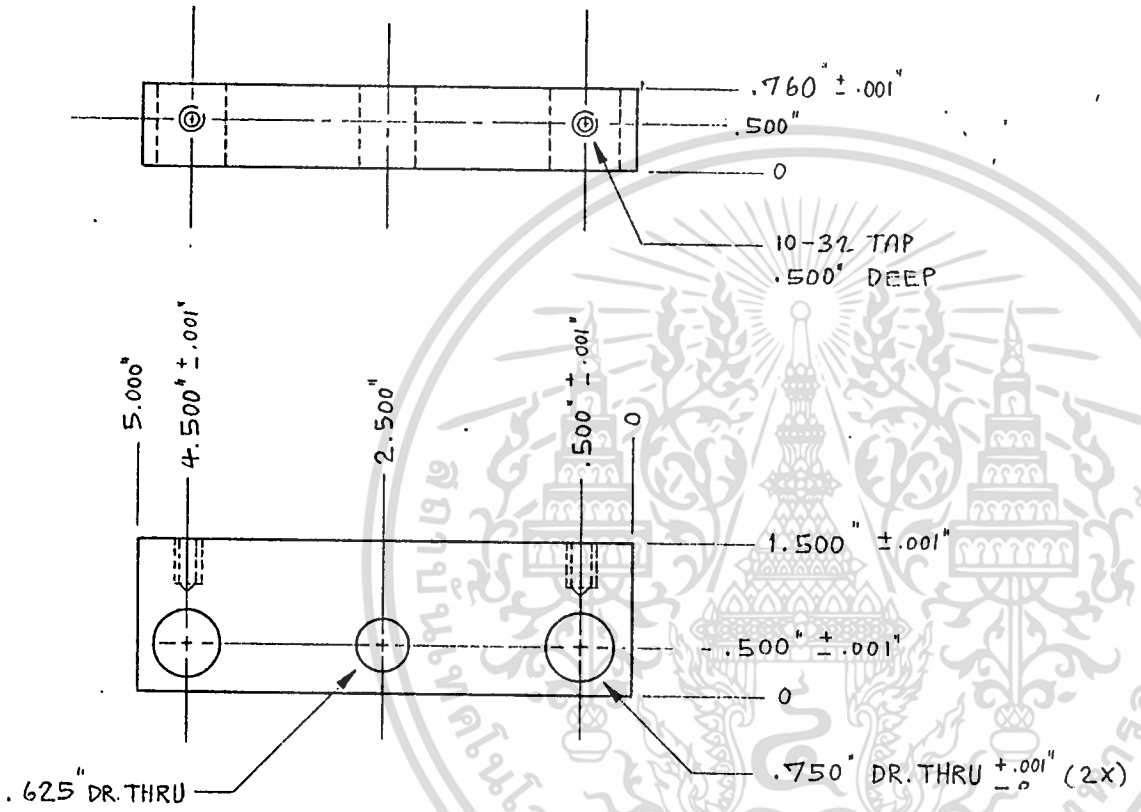
SUR : REMOVE ALL BURR

TOL : ±.005" BLACK ANODIZ

DWG : SUCHART P.

U/P :

QTY : 2



AXIS-Y BRACKET 2

MAT: A_L

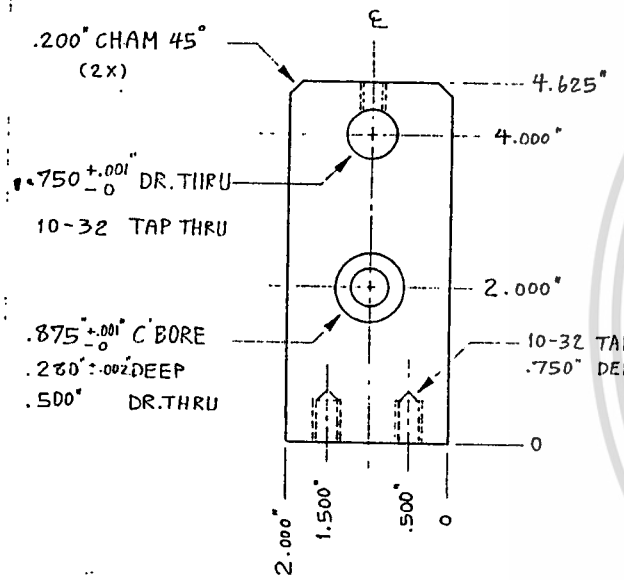
SUR: REMOVE ALL BURR

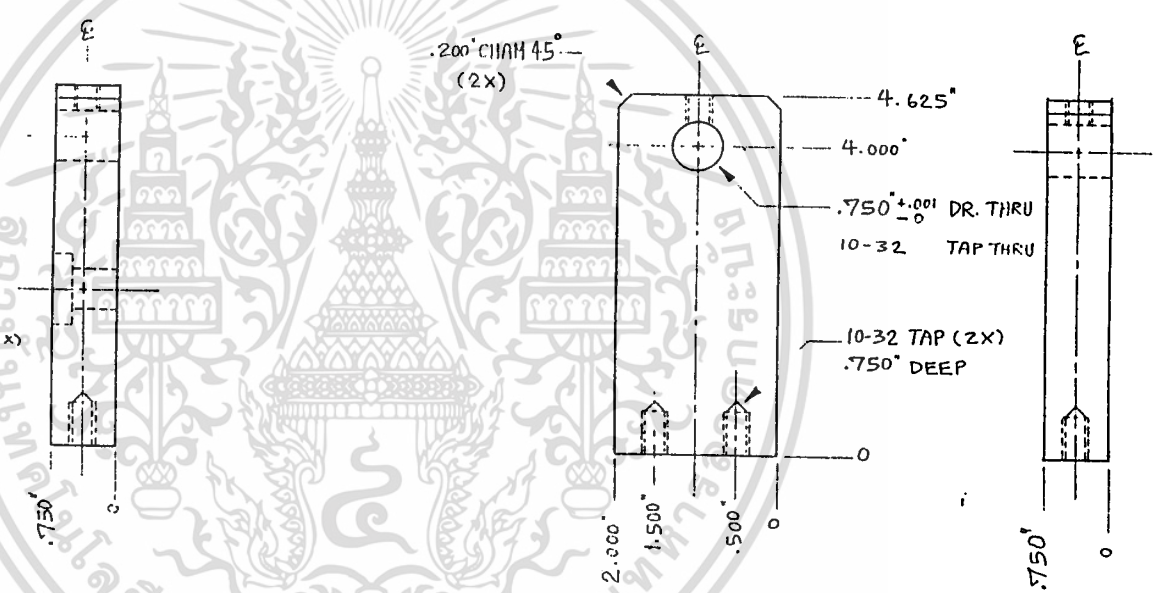
TOL: ± .005" BLACK ANODIZE

DWG: SUCHART P

U/P:

QTY: 2





AXIS-X BRACKET 1

MAT : A_L

SUR : ▽ REMOVE ALL BURR

TOL : ±.005" BLACK ANODIZE

DWG : SUCHART P.

U/P :

QTY : 2

AXIS-X BRACKET 2

MAT : A_L

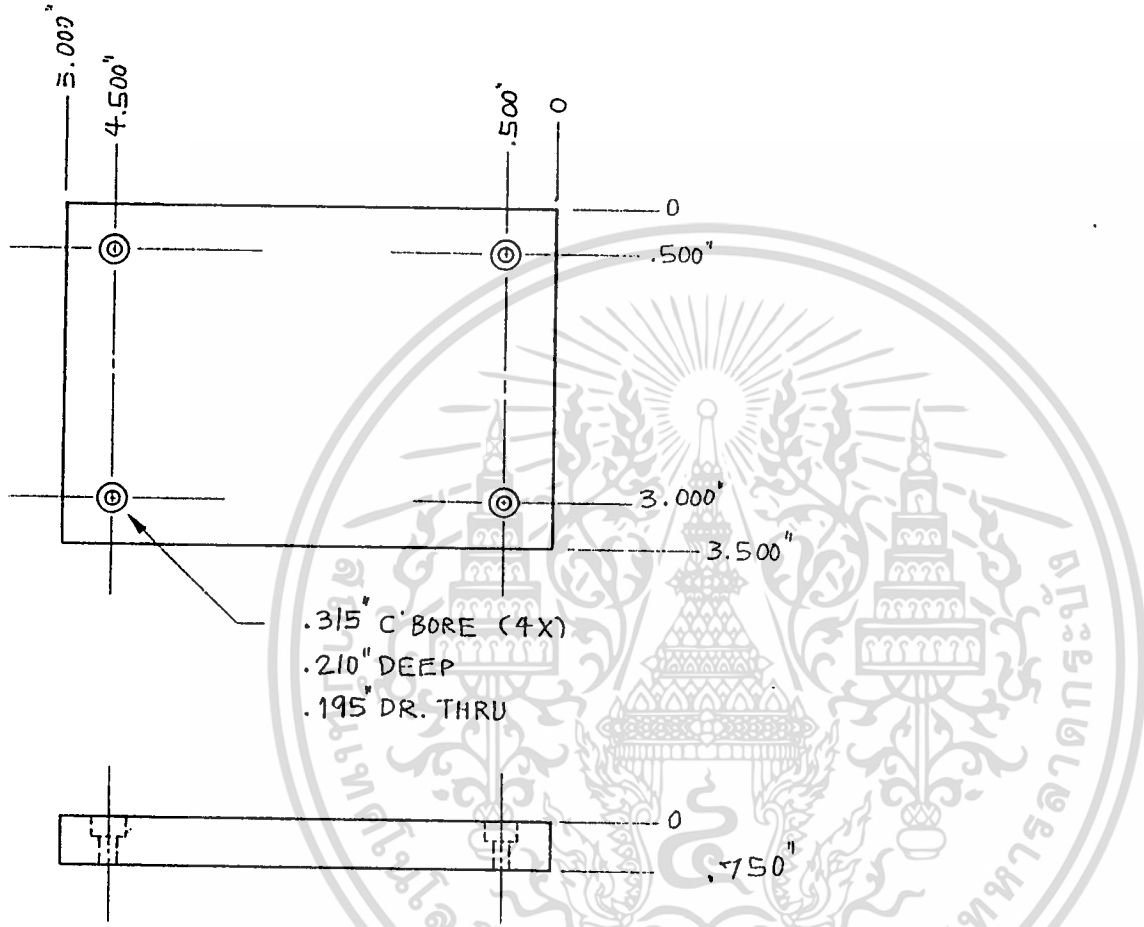
SUR : ▽ REMOVE ALL BURR

TOL : ±.005" BLACK ANODIZE

DWG : SUCHART P.

U/P :

QTY : 2

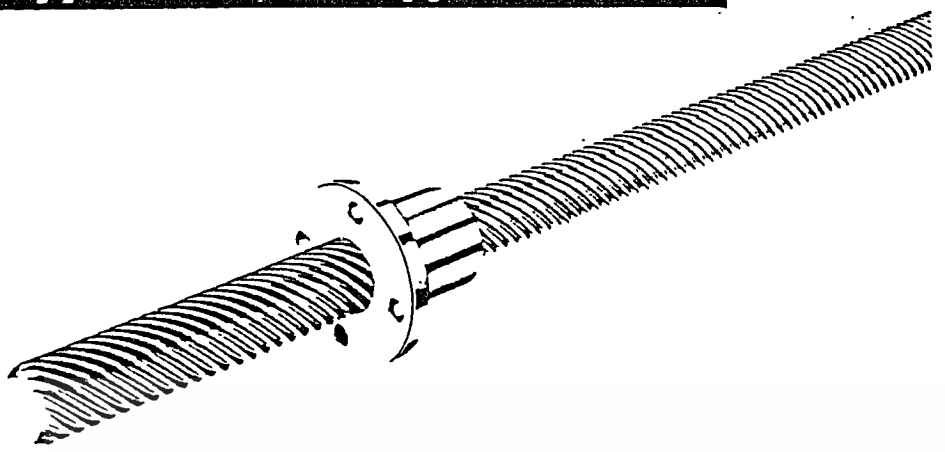


REMOVE ALL BURR

AXIS-Y BASE

MAT : A_L
SUR : ▽▽ BLACK ANODIZE
TOL : ±.005"
DWG : SUCHART P
U/P :
QTY : 1

THK type DCMA/type DCMB



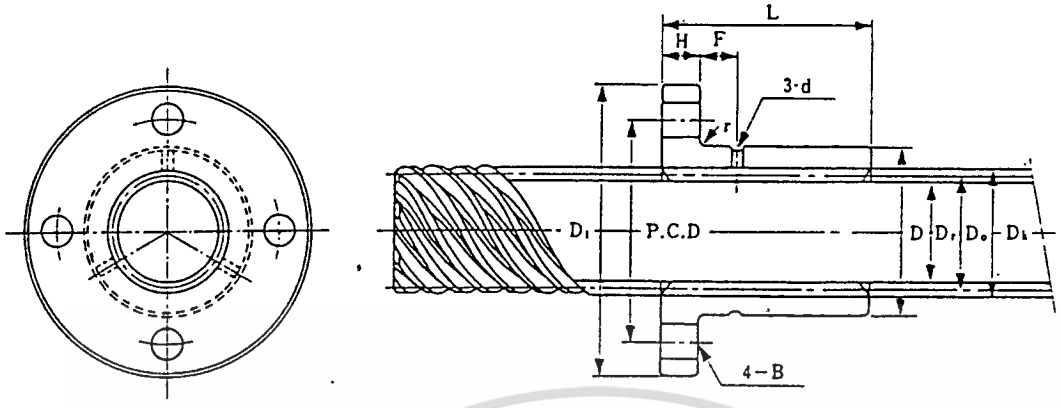
Type DCMB

Note 1) Change nut model No.	Main dimensions mm: Outer diameter D	Tolerance	Length L	Flange diameter D_1	H	B	P.C.D	r	F	d	Note 2) Dynamic permissible torque T kgf·m	Note 2) Dynamic permissible thrust F kgf
DCM B 8T	15	0 −0.1	16	28	4	3.4	21	0.8	—	—	0.33	88
DCM B 12T	20		25	36	5	4.5	27	1.0	—	—	1.3	140
DCM A 15T	22	0 −0.052	15	44	6	5.4	31	1.5	4.5	1.5	1.7	235
DCM B 15T			30								3.3	470
DCM A 17T	28		15	51	7	6.6	38	1.5	4.5	1.5	2.1	265
DCM B 17T			35								4.9	620
DCM A 20T	32		20	56	7	6.6	42	1.5	6.5	2	4.1	425
DCM B 20T			40								8.1	850
DCM A 25T	36	0 −0.062	25	61	8	6.6	47	2	8.5	2	7.6	650
DCM B 25T			50								15.1	1300
DCM A 30T	44		28	76	10	9	58	2	9	2	13.3	825
DCM B 30T			56								27.4	1650
DCM A 35T	52		30	84	10	9	66	2.5	10	3	14.7	945
DCM B 35T			60								29.3	1890
DCM A 40	58		35	98	12	11	76	2.5	11.5	3	38.9	2040
DCM B 40		0 −0.074	70								77.8	4080
DCM A 45	64		37	104	12	11	80	2.5	12.5	3	48.3	2340
DCM B 45			75								97.9	4750
DCM A 50	68		40	109	12	11	85	2.5	14	3	69.5	2910
DCM B 50			80								139	5820

1: The symbol T indicates combination with a rolled screw shaft

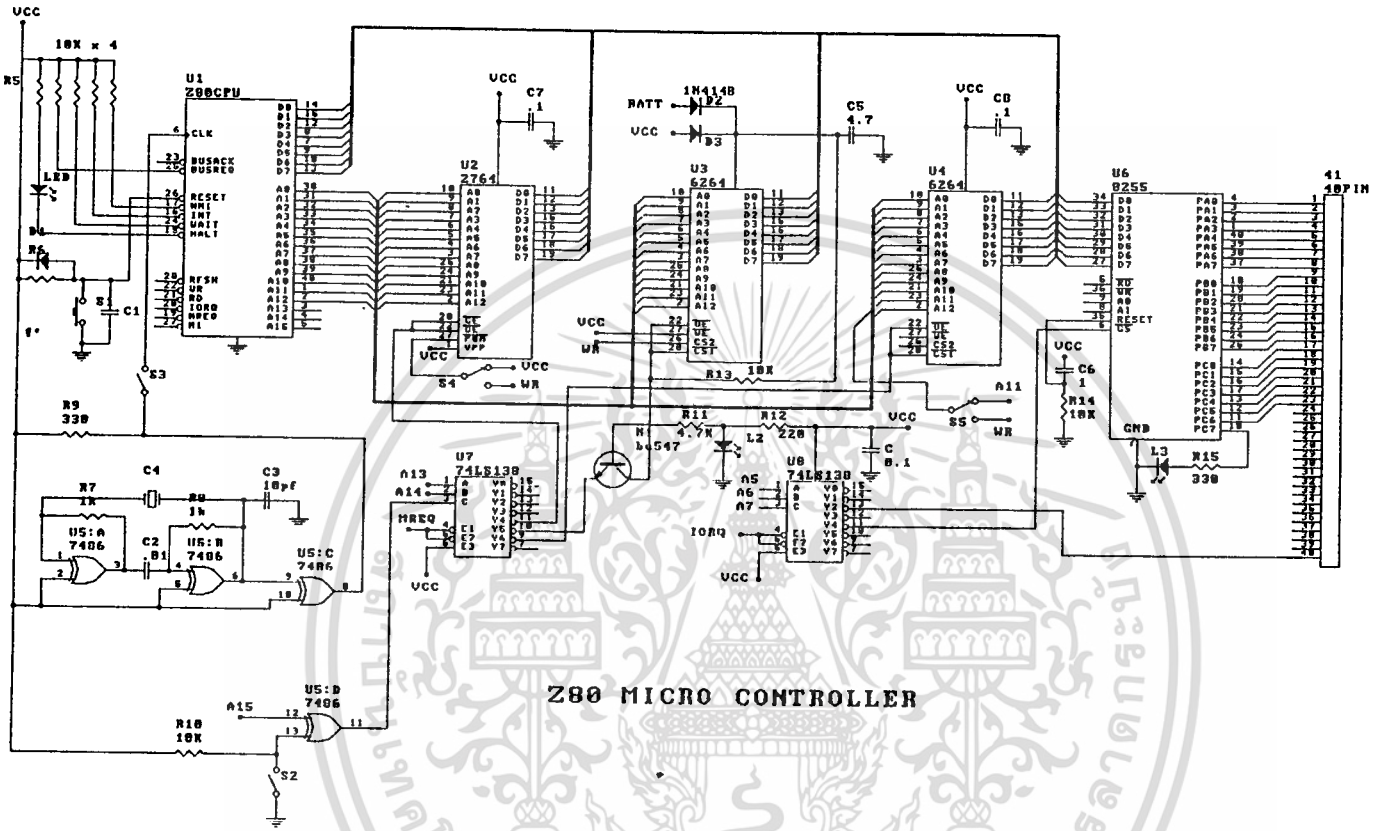
2: The dynamic permissible thrust F is the thrust with a tooth contact surface pressure of 1 kgf/mm².

3: The material for the miniature change nuts DCMB8T and DCMB12T is oil-impregnated plastic.



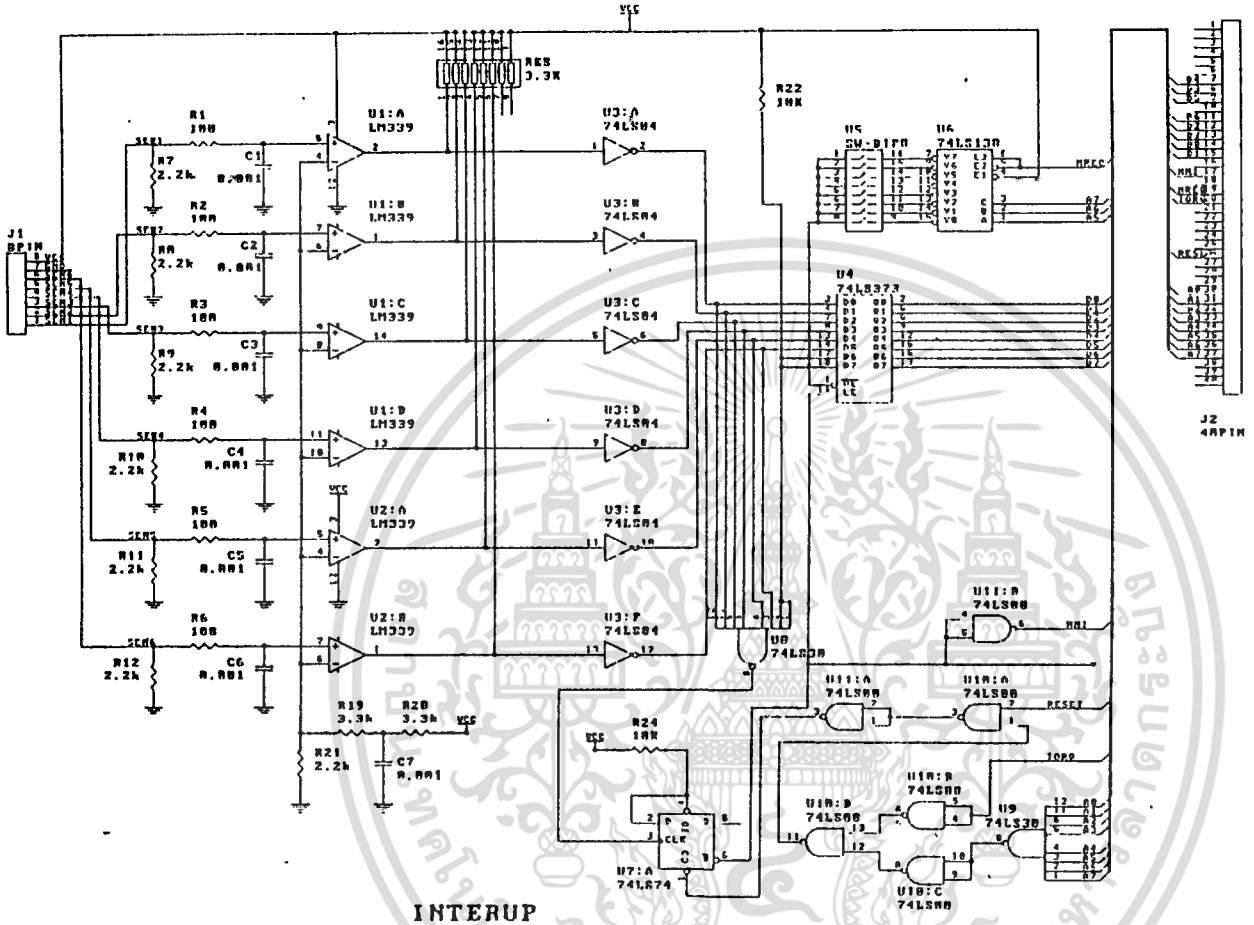
Change nut weight grf	Screw shaft model No	Multiple-start screw part details mm: Lead angle α	Outer diameter Dk	Effective diameter Do	Root diameter Dr	Lead ℓ	Number of shafts Z	Standard screw shaft length mm	Screw shaft weight kgf/m
5	CT 8T	(45°)	9.0	7.6	6.2	24	6	500	0.35
10	CT 12T	(45°)	13.3	11.5	9.7	36	7	500 1000	0.82
60	CT 15T	(45°)	15.8	13.7	11.6	44.4	8	500 1000	1.2
85									
95	CT 17T	(45°)	17.8	15.7	13.6	50	9		1.5
140									
135	CT 20T	(45°)	21.2	18.7	16.2	60	9	500 1000 1500	2.6
210									
175	CT 25T	(45°)	25.6	23.1	20.6	73.3	11		3.3
280									
290	CT 30T	(45°)	31.9	29.4	26.9	93.3	14		5.3
465									
425	CT 35T	(45°)	34.1	31.1	28.1	97.7	11	500 1000 2000	5.8
670									
715	☆CT 40	(45°)	44	38.18	33.3	119.9	12	-	9.0
1065									
820	☆CT 45	(45°)	47	41.37	36.4	129.9	13	1000 2000 3000	10.6
1270									
925	☆CT 50	(45°)	52	47.73	42.9	149.9	15		14.0
1375									

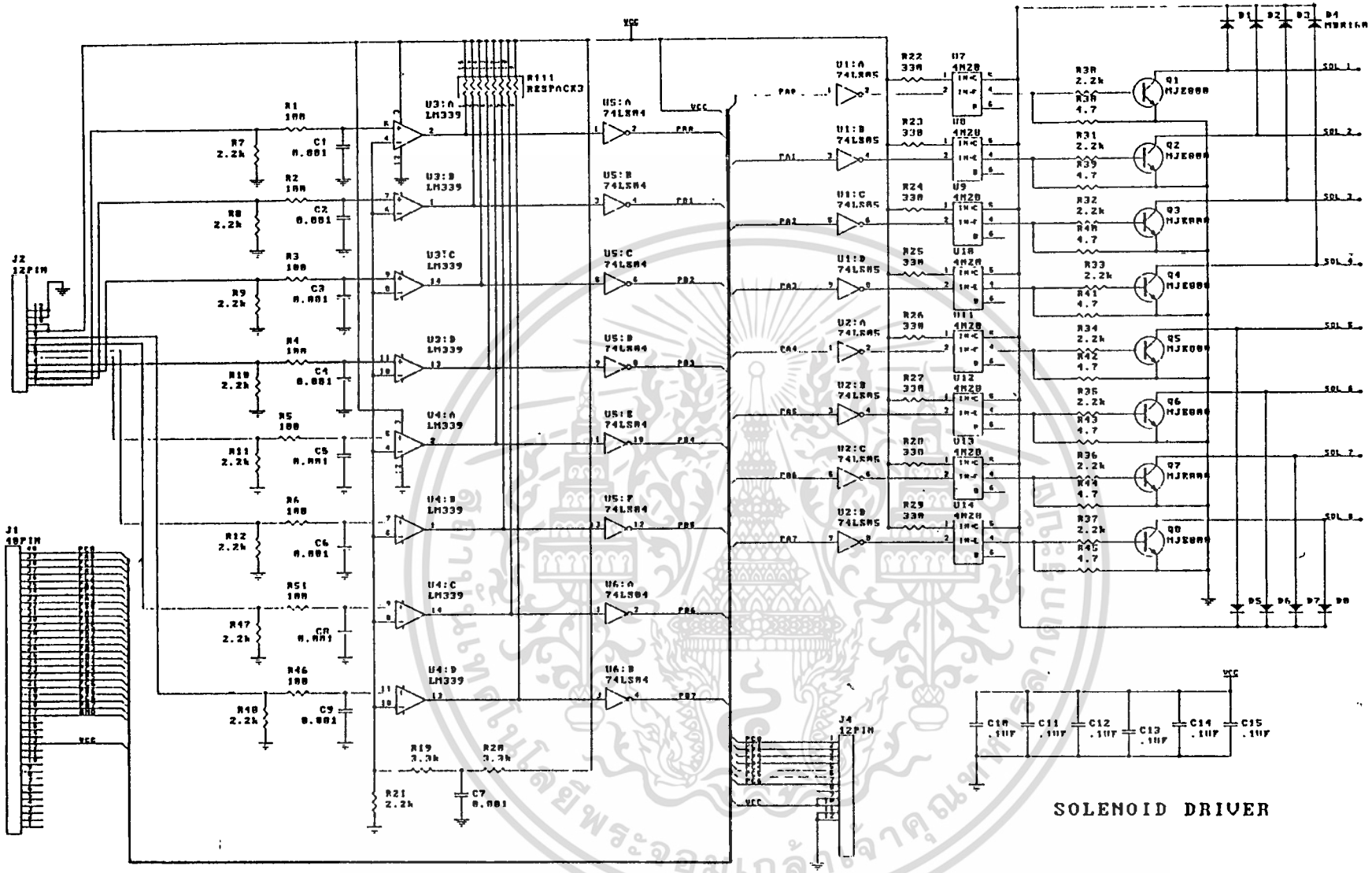
● The screw shafts marked by ☆ are manufactured on order.

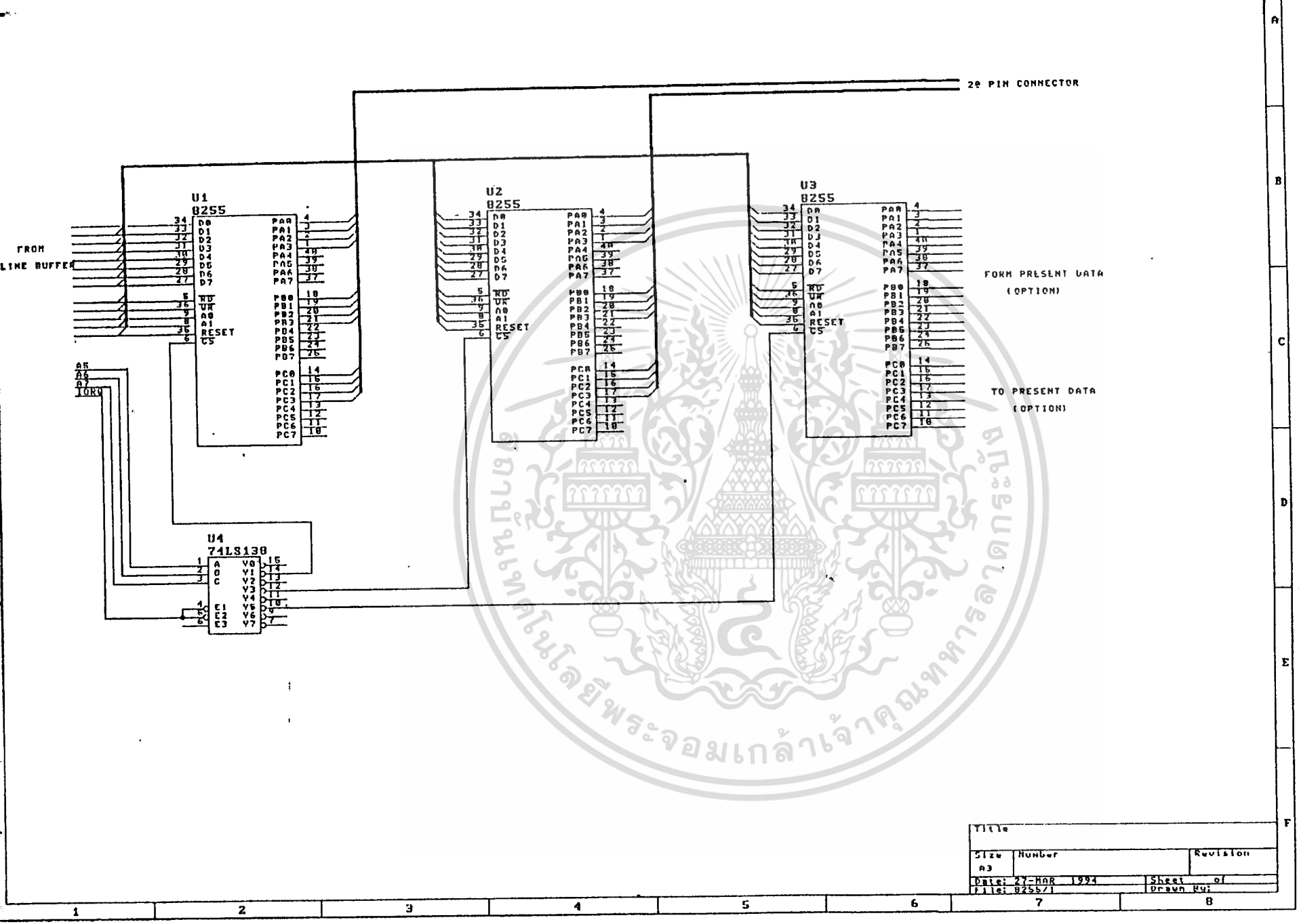


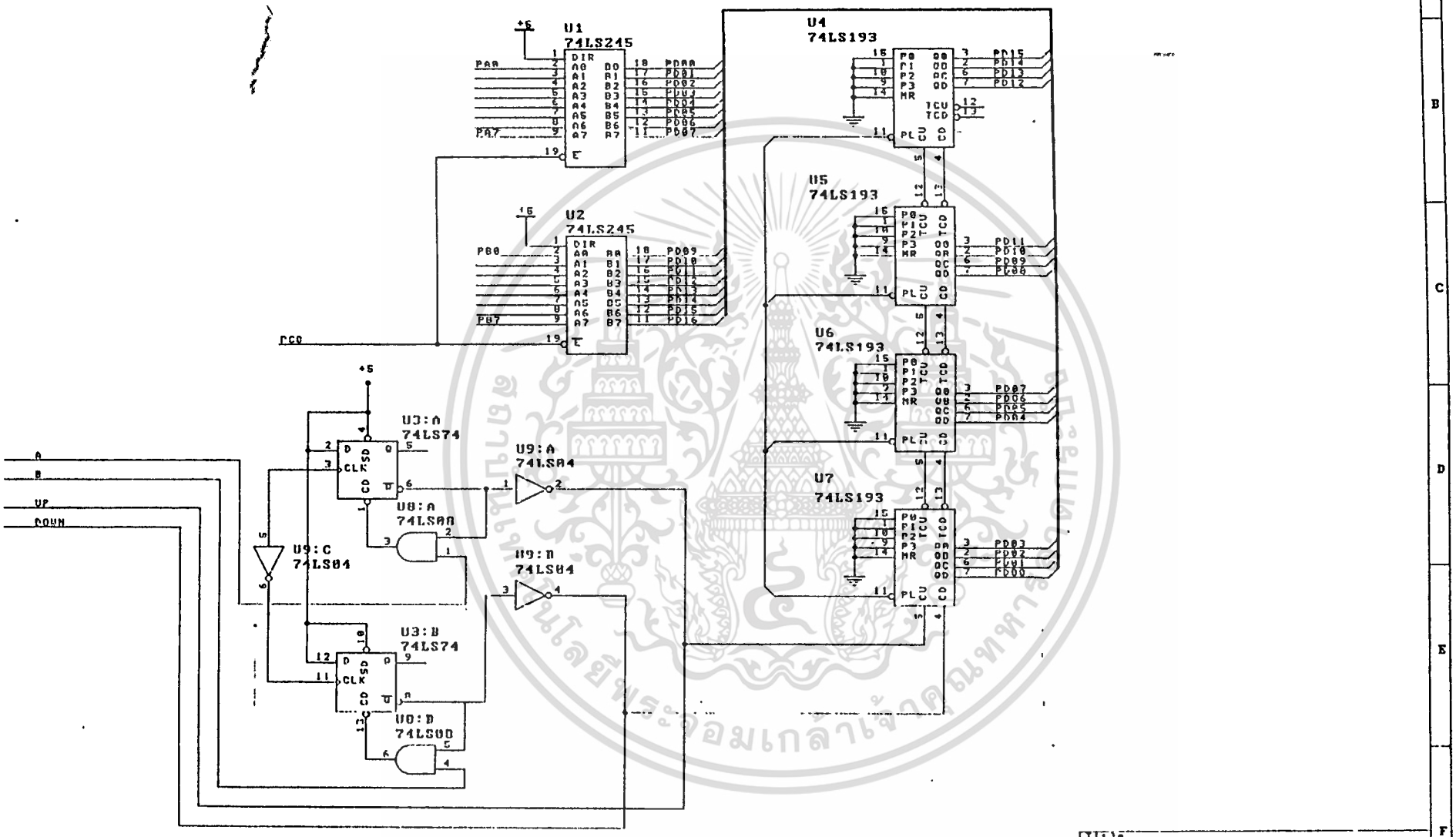
Z80 MICRO CONTROLLER

Title		
Size A2	Number	Revision
Date: 26-JUN-1994		Sheet of
File: 280A7		Drawn By:

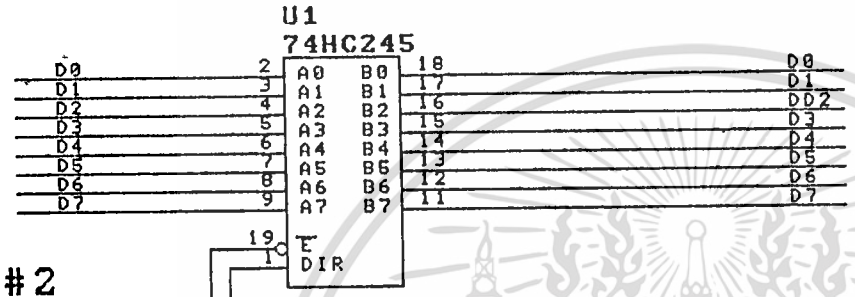



SOLENOID DRIVER
J1 40PIN
J2 12PIN
J4 12PIN
RES1 RESPACK8
R1 100
R2 100
R3 100
R4 100
R5 100
R6 100
R7 100
R46 100
R7 2.2k
R8 2.2k
R9 2.2k
R10 2.2k
R11 2.2k
R12 2.2k
R47 2.2k
R48 2.2k
R19 3.3k
R20 3.3k
R21 2.2k
C1 0.001
C2 0.001
C3 0.001
C4 0.001
C5 0.001
C6 0.001
C8 0.001
C9 0.001
C7 0.001
U3:A LM339
U3:B LM339
U3:C LM339
U4:A LM339
U4:B LM339
U4:C LM339
U4:D LM339
U5:A 74LS04
U5:B 74LS04
U5:C 74LS04
U5:D 74LS04
U5:E 74LS04
U5:F 74LS04
U6:A 74LS04
U6:B 74LS04
U1:A 74LS05
U1:B 74LS05
U1:C 74LS05
U1:D 74LS05
U2:A 74LS05
U2:B 74LS05
U2:C 74LS05
U2:D 74LS05
R22 330
R23 330
R24 330
R25 330
R26 330
R27 330
R28 330
R29 330
U7 4N28
U8 4N28
U9 4N28
U10 4N28
U11 4N28
U12 4N28
U13 4N28
U14 4N28
R30 2.2k
R38 4.7
R31 2.2k
R39 4.7
R32 2.2k
R40 4.7
R33 2.2k
R41 4.7
R34 2.2k
R42 4.7
R35 2.2k
R43 4.7
R36 2.2k
R44 4.7
R37 2.2k
R45 4.7
Q1 MJE800
Q2 MJE800
Q3 MJE800
Q4 MJE800
Q5 MJE800
Q6 MJE800
Q7 MJE800
Q8 MJE800
D1 D2 D3 D4 MBR16A
D5 D6 D7 D8
C10 .1UF
C11 .1UF
C12 .1UF
C13 .1UF
C14 .1UF
C15 .1UF
VCC
SOL 1
SOL 2
SOL 3
SOL 5
SOL 6
SOL 7
SOL 8

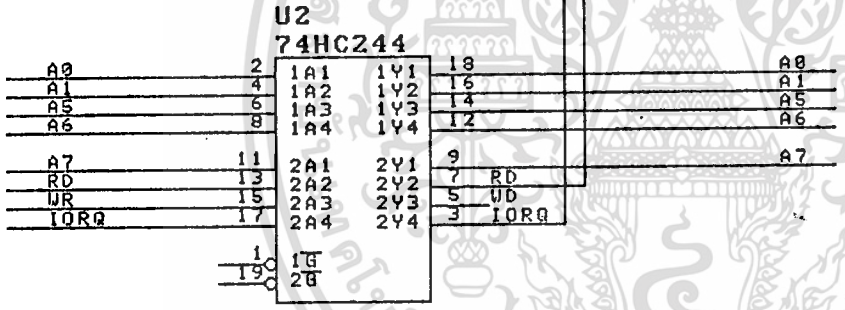

20 PIN CONNECTOR
FROM
LINE BUFFER
U1
8255
U2
8255
U3
8255
U4
74LS138
FORM PRESENT DATA
(OPTION)
TO PRESENT DATA
(OPTION)
RESET
Title
Size
A3
Number
Revision
Date: 27-MAR 1994
File: 8255/1
Sheet of
Drawn By:



Title	PRESENT COUNTER
Size	A3
Number	123003
Revision	A
Date:	27-MAR-1994
Sheet	1 of 5
File:	PRESENT/1
Drawn By:	CHANCHAI



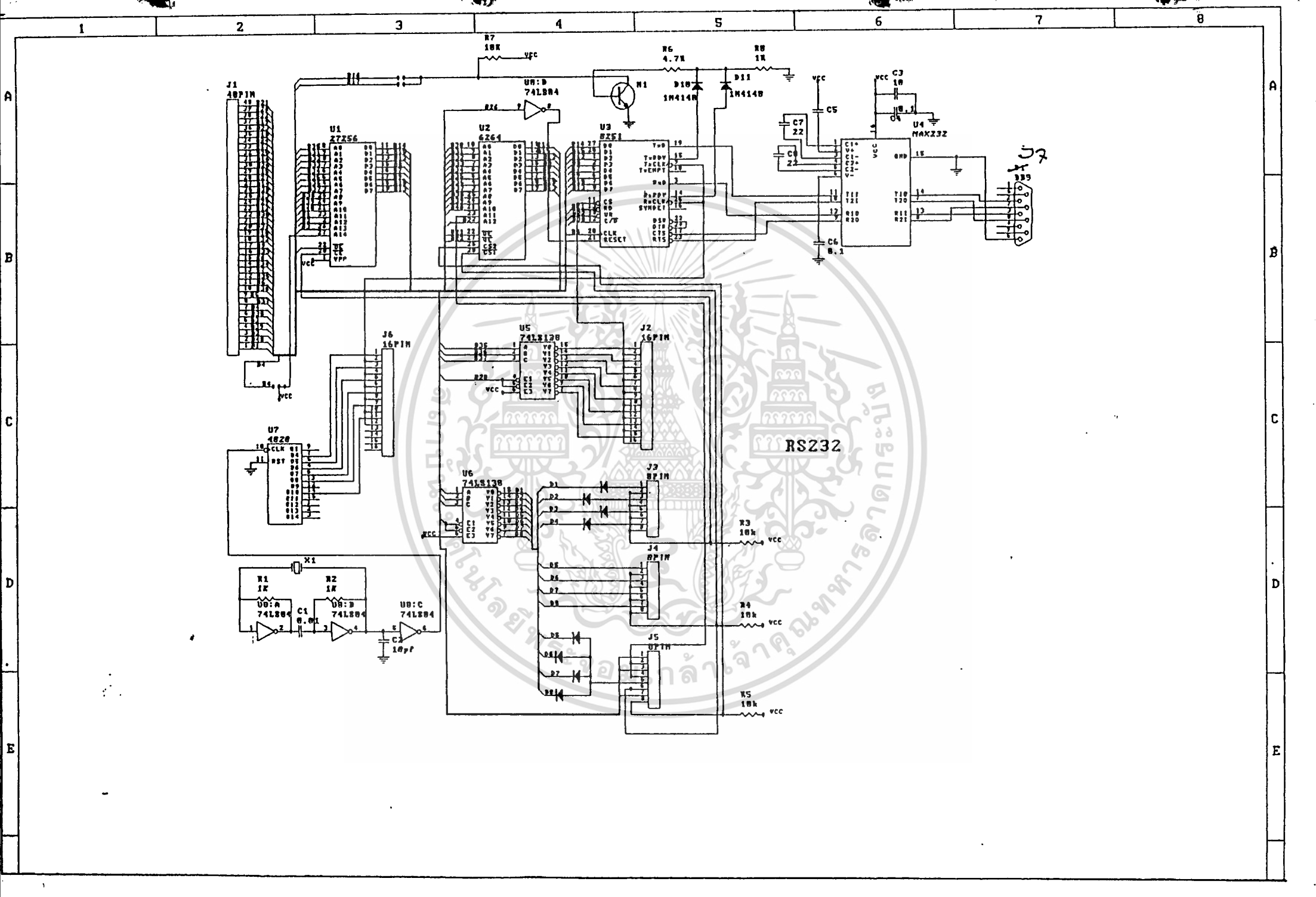
To Z-80 #2
BUS

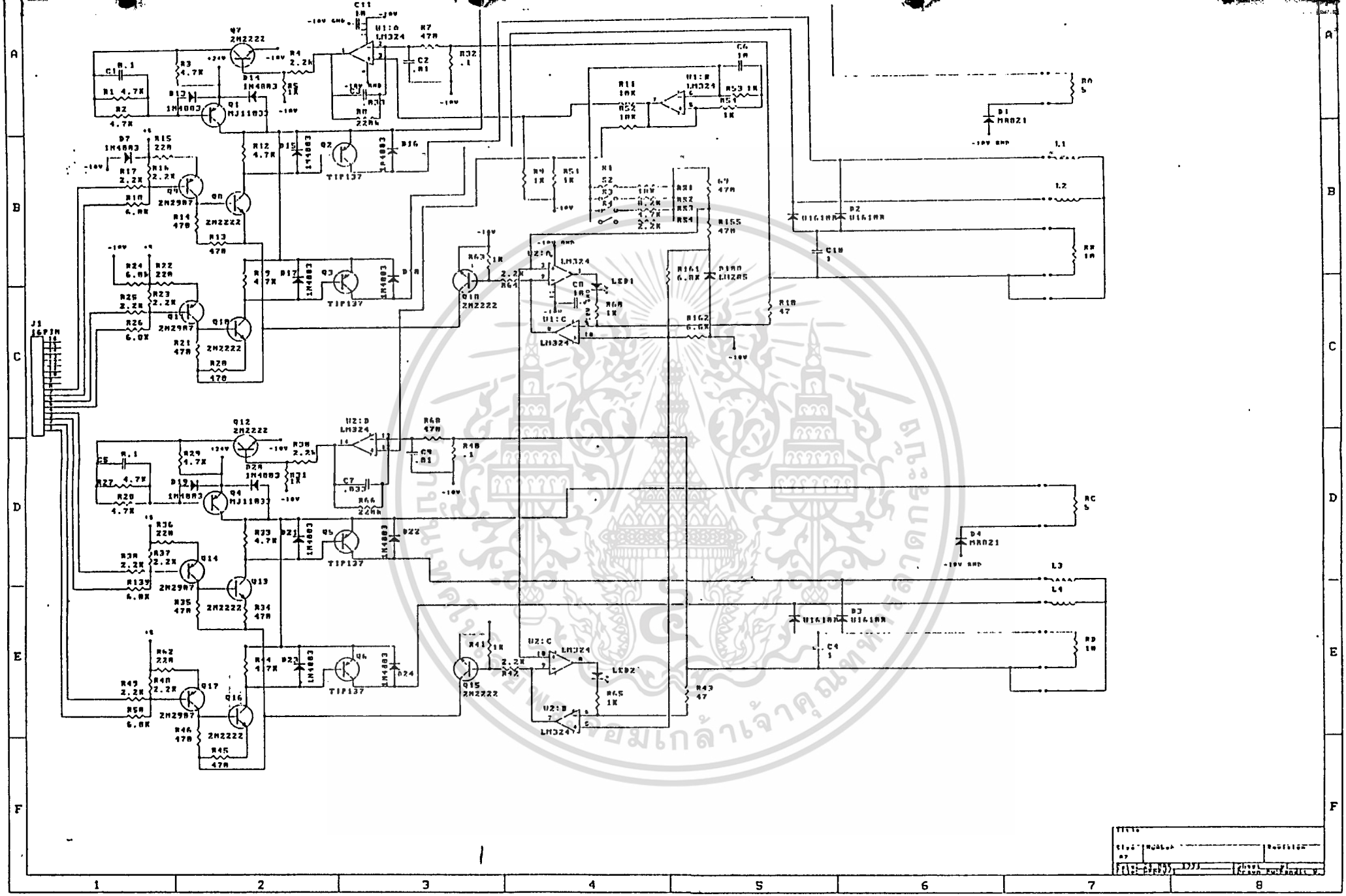


To 8255 PORT

LINE BUFFER

Title		
Size A4	Number	Revision
Date: 27-MAR 1994		Sheet of
File: BUFF/1		Drawn By:


RS232
J1
40PIN
U1
27256
U2
6264
U3
8251
U4
MAX232
C3
10
C5
C7
22
C8
22
C6
0.1
R7
10K
R6
4.7K
R8
1K
D10
1N4148
D11
1N4148
Q1
U8:B
74LS04
U8:A
74LS04
U8:C
74LS04
U8:D
74LS04
X1
R1
1K
R2
1K
C1
0.001
C2
10pf
J6
16PIN
U5
74LS138
J2
16PIN
U7
4020
U6
74LS138
J3
8PIN
J4
8PIN
J5
8PIN
R3
10k
R4
10k
R5
10k
vcc


J1
16PIN
Q7
2N2222
Q1
MJ11033
Q8
2N2222
Q2
TIP137
Q9
2N2907
Q10
2N2222
Q3
TIP137
Q11
2N2907
Q12
2N2222
Q4
MJ11033
Q13
2N2222
Q5
TIP137
Q14
2N2907
Q15
2N2222
Q6
TIP137
Q16
2N2222
Q17
2N2907
U1:A
LM324
U1:B
LM324
U1:C
LM324
U1:D
LM324
U2:A
LM324
U2:B
LM324
U2:C
LM324
U2:D
LM324
D1
MR821
D2
U1618R
D3
U1618R
D4
MR821
D5
U1618R
D6
U1618R
D7
1N4883
D8
1N4883
D9
1N4883
D10
1N4883
D11
1N4883
D12
1N4883
D13
1N4883
D14
1N4883
D15
1N4883
D16
1N4883
D17
1N4883
D18
1N4883
D19
1N4883
D20
1N4883
D21
1N4883
D22
1N4883
D23
1N4883
D24
1N4883
D25
1N4883
D26
1N4883
D27
1N4883
D28
1N4883
D29
1N4883
D30
1N4883
D31
1N4883
D32
1N4883
D33
1N4883
D34
1N4883
D35
1N4883
D36
1N4883
D37
1N4883
D38
1N4883
D39
1N4883
D40
1N4883
D41
1N4883
D42
1N4883
D43
1N4883
D44
1N4883
D45
1N4883
D46
1N4883
D47
1N4883
D48
1N4883
D49
1N4883
D50
1N4883
D51
1N4883
D52
1N4883
D53
1N4883
D54
1N4883
D55
1N4883
D56
1N4883
D57
1N4883
D58
1N4883
D59
1N4883
D60
1N4883
D61
1N4883
D62
1N4883
D63
1N4883
D64
1N4883
D65
1N4883
D66
1N4883
D67
1N4883
D68
1N4883
D69
1N4883
D70
1N4883
D71
1N4883
D72
1N4883
D73
1N4883
D74
1N4883
D75
1N4883
D76
1N4883
D77
1N4883
D78
1N4883
D79
1N4883
D80
1N4883
D81
1N4883
D82
1N4883
D83
1N4883
D84
1N4883
D85
1N4883
D86
1N4883
D87
1N4883
D88
1N4883
D89
1N4883
D90
1N4883
D91
1N4883
D92
1N4883
D93
1N4883
D94
1N4883
D95
1N4883
D96
1N4883
D97
1N4883
D98
1N4883
D99
1N4883
D100
1N4883
D101
1N4883
D102
1N4883
D103
1N4883
D104
1N4883
D105
1N4883
D106
1N4883
D107
1N4883
D108
1N4883
D109
1N4883
D110
1N4883
D111
1N4883
D112
1N4883
D113
1N4883
D114
1N4883
D115
1N4883
D116
1N4883
D117
1N4883
D118
1N4883
D119
1N4883
D120
1N4883
D121
1N4883
D122
1N4883
D123
1N4883
D124
1N4883
D125
1N4883
D126
1N4883
D127
1N4883
D128
1N4883
D129
1N4883
D130
1N4883
D131
1N4883
D132
1N4883
D133
1N4883
D134
1N4883
D135
1N4883
D136
1N4883
D137
1N4883
D138
1N4883
D139
1N4883
D140
1N4883
D141
1N4883
D142
1N4883
D143
1N4883
D144
1N4883
D145
1N4883
D146
1N4883
D147
1N4883
D148
1N4883
D149
1N4883
D150
1N4883
D151
1N4883
D152
1N4883
D153
1N4883
D154
1N4883
D155
1N4883
D156
1N4883
D157
1N4883
D158
1N4883
D159
1N4883
D160
1N4883
D161
1N4883
D162
1N4883
D163
1N4883
D164
1N4883
D165
1N4883
D166
1N4883
D167
1N4883
D168
1N4883
D169
1N4883
D170
1N4883
D171
1N4883
D172
1N4883
D173
1N4883
D174
1N4883
D175
1N4883
D176
1N4883
D177
1N4883
D178
1N4883
D179
1N4883
D180
1N4883
D181
1N4883
D182
1N4883
D183
1N4883
D184
1N4883
D185
1N4883
D186
1N4883
D187
1N4883
D188
1N4883
D189
1N4883
D190
1N4883
D191
1N4883
D192
1N4883
D193
1N4883
D194
1N4883
D195
1N4883
D196
1N4883
D197
1N4883
D198
1N4883
D199
1N4883
D200
1N4883
D201
1N4883
D202
1N4883
D203
1N4883
D204
1N4883
D205
1N4883
D206
1N4883
D207
1N4883
D208
1N4883
D209
1N4883
D210
1N4883
D211
1N4883
D212
1N4883
D213
1N4883
D214
1N4883
D215
1N4883
D216
1N4883
D217
1N4883
D218
1N4883
D219
1N4883
D220
1N4883
D221
1N4883
D222
1N4883
D223
1N4883
D224
1N4883
D225
1N4883
D226
1N4883
D227
1N4883
D228
1N4883
D229
1N4883
D230
1N4883
D231
1N4883
D232
1N4883
D233
1N4883
D234
1N4883
D235
1N4883
D236
1N4883
D237
1N4883
D238
1N4883
D239
1N4883
D240
1N4883
D241
1N4883
D242
1N4883
D243
1N4883
D244
1N4883
D245
1N4883
D246
1N4883
D247
1N4883
D248
1N4883
D249
1N4883
D250
1N4883
D251
1N4883
D252
1N4883
D253
1N4883
D254
1N4883
D255
1N4883
D256
1N4883
D257
1N4883
D258
1N4883
D259
1N4883
D260
1N4883
D261
1N4883
D262
1N4883
D263
1N4883
D264
1N4883
D265
1N4883
D266
1N4883
D267
1N4883
D268
1N4883
D269
1N4883
D270
1N4883
D271
1N4883
D272
1N4883
D273
1N4883
D274
1N4883
D275
1N4883
D276
1N4883
D277
1N4883
D278
1N4883
D279
1N4883
D280
1N4883
D281
1N4883
D282
1N4883
D283
1N4883
D284
1N4883
D285
1N4883
D286
1N4883
D287
1N4883
D288
1N4883
D289
1N4883
D290
1N4883
D291
1N4883
D292
1N4883
D293
1N4883
D294
1N4883
D295
1N4883
D296
1N4883
D297
1N4883
D298
1N4883
D299
1N4883
D300
1N4883
D301
1N4883
D302
1N4883
D303
1N4883
D304
1N4883
D305
1N4883
D306
1N4883
D307
1N4883
D308
1N4883
D309
1N4883
D310
1N4883
D311
1N4883
D312
1N4883
D313
1N4883
D314
1N4883
D315
1N4883
D316
1N4883
D317
1N4883
D318
1N4883
D319
1N4883
D320
1N4883
D321
1N4883
D322
1N4883
D323
1N4883
D324
1N4883
D325
1N4883
D326
1N4883
D327
1N4883
D328
1N4883
D329
1N4883
D330
1N4883
D331
1N4883
D332
1N4883
D333
1N4883
D334
1N4883
D335
1N4883
D336
1N4883
D337
1N4883
D338
1N4883
D339
1N4883
D340
1N4883
D341
1N4883
D342
1N4883
D343
1N4883
D344
1N4883
D345
1N4883
D346
1N4883
D347
1N4883
D348
1N4883
D349
1N4883
D350
1N4883
D351
1N4883
D352
1N4883
D353
1N4883
D354
1N4883
D355
1N4883
D356
1N4883
D357
1N4883
D358
1N4883
D359
1N4883
D360
1N4883
D361
1N4883
D362
1N4883
D363
1N4883
D364
1N4883
D365
1N4883
D366
1N4883
D367
1N4883
D368
1N4883
D369
1N4883
D370
1N4883
D371
1N4883
D372
1N4883
D373
1N4883
D374
1N4883
D375
1N4883
D376
1N4883
D377
1N4883
D378
1N4883
D379
1N4883
D380
1N4883
D381
1N4883
D382
1N4883
D383
1N4883
D384
1N4883
D385
1N4883
D386
1N4883
D387
1N4883
D388
1N4883
D389
1N4883
D390
1N4883
D391
1N4883
D392
1N4883
D393
1N4883
D394
1N4883
D395
1N4883
D396
1N4883
D397
1N4883
D398
1N4883
D399
1N4883
D400
1N4883
D401
1N4883
D402
1N4883
D403
1N4883
D404
1N4883
D405
1N4883
D406
1N4883
D407
1N4883
D408
1N4883
D409
1N4883
D410
1N4883
D411
1N4883
D412
1N4883
D413
1N4883
D414
1N4883
D415
1N4883
D416
1N4883
D417
1N4883
D418
1N4883
D419
1N4883
D420
1N4883
D421
1N4883
D422
1N4883
D423
1N4883
D424
1N4883
D425
1N4883
D426
1N4883
D427
1N4883
D428
1N4883
D429
1N4883
D430
1N4883
D431
1N4883
D432
1N4883
D433
1N4883
D434
1N4883
D435
1N4883
D436
1N4883
D437
1N4883
D438
1N4883
D439
1N4883
D440
1N4883
D441
1N4883
D442
1N4883
D443
1N4883
D444
1N4883
D445
1N4883
D446
1N4883
D447
1N4883
D448
1N4883
D449
1N4883
D450
1N4883
D451
1N4883
D452
1N4883
D453
1N4883
D454
1N4883
D455
1N4883
D456
1N4883
D457
1N4883
D458
1N4883
D459
1N4883
D460
1N4883
D461
1N4883
D462
1N4883
D463
1N4883
D464
1N4883
D465
1N4883
D466
1N4883
D467
1N4883
D468
1N4883
D469
1N4883
D470
1N4883
D471
1N4883
D472
1N4883
D473
1N4883
D474
1N4883
D475
1N4883
D476
1N4883
D477
1N4883
D478
1N4883
D479
1N4883
D480
1N4883
D481
1N4883
D482
1N4883
D483
1N4883
D484
1N4883
D485
1N4883
D486
1N4883
D487
1N4883
D488
1N4883
D489
1N4883
D490
1N4883
D491
1N4883
D492
1N4883
D493
1N4883
D494
1N4883
D495
1N4883
D496
1N4883
D497
1N4883
D498
1N4883
D499
1N4883
D500
1N4883
D501
1N4883
D502
1N4883
D503
1N4883
D504
1N4883
D505
1N4883
D506
1N4883
D507
1N4883
D508
1N4883
D509
1N4883
D510
1N4883
D511
1N4883
D512
1N4883
D513
1N4883
D514
1N4883
D515
1N4883
D516
1N4883
D517
1N4883
D518
1N4883
D519
1N4883
D520
1N4883
D521
1N4883
D522
1N4883
D523
1N4883
D524
1N4883
D525
1N4883
D526
1N4883
D527
1N4883
D528
1N4883
D529
1N4883
D530
1N4883
D531
1N4883
D532
1N4883
D533
1N4883
D534
1N4883
D535
1N4883
D536
1N4883
D537
1N4883
D538
1N4883
D539
1N4883
D540
1N4883
D541
1N4883
D542
1N4883
D543
1N4883
D544
1N4883
D545
1N4883
D546
1N4883
D547
1N4883
D548
1N4883
D549
1N4883
D550
1N4883
D551
1N4883
D552
1N4883
D553
1N4883
D554
1N4883
D555
1N4883
D556
1N4883
D557
1N4883
D558
1N4883
D559
1N4883
D560
1N4883
D561
1N4883
D562
1N4883
D563
1N4883
D564
1N4883
D565
1N4883
D566
1N4883
D567
1N4883
D568
1N4883
D569
1N4883
D570
1N4883
D571
1N4883
D572
1N4883
D573
1N4883
D574
1N4883
D575
1N4883
D576
1N4883
D577
1N4883
D578
1N4883
D579
1N4883
D580
1N4883
D581
1N4883
D582
1N4883
D583
1N4883
D584
1N4883
D585
1N4883
D586
1N4883
D587
1N4883
D588
1N4883
D589
1N4883
D590
1N4883
D591
1N4883
D592
1N4883
D593
1N4883
D594
1N4883
D595
1N4883
D596
1N4883
D597
1N4883
D598
1N4883
D599
1N4883
D600
1N4883
D601
1N4883
D602
1N4883
D603
1N4883
D604
1N4883
D605
1N4883
D606
1N4883
D607
1N4883
D608
1N4883
D609
1N4883
D610
1N4883
D611
1N4883
D612
1N4883
D613
1N4883
D614
1N4883
D615
1N4883
D616
1N4883
D617
1N4883
D618
1N4883
D619
1N4883
D620
1N4883
D621
1N4883
D622
1N4883
D623
1N4883
D624
1N4883
D625
1N4883
D626
1N4883
D627
1N4883
D628
1N4883
D629
1N4883
D630
1N4883
D631
1N4883
D632
1N4883
D633
1N4883
D634
1N4883
D635
1N4883
D636
1N4883
D637
1N4883
D638
1N4883
D639
1N4883
D640
1N4883
D641
1N4883
D642
1N4883
D643
1N4883
D644
1N4883
D645
1N4883
D646
1N4883
D647
1N4883
D648
1N4883
D649
1N4883
D650
1N4883
D651
1N4883
D652
1N4883
D653
1N4883
D654
1N4883
D655
1N4883
D656
1N4883
D657
1N4883
D658
1N4883
D659
1N4883
D660
1N4883
D661
1N4883
D662
1N4883
D663
1N4883
D664
1N4883
D665
1N4883
D666
1N4883
D667
1N4883
D668
1N4883
D669
1N4883
D670
1N4883
D671
1N4883
D672
1N4883
D673
1N4883
D674
1N4883
D675
1N4883
D676
1N4883
D677
1N4883
D678
1N4883
D679
1N4883
D680
1N4883
D681
1N4883
D682
1N4883
D683
1N4883
D684
1N4883
D685
1N4883
D686
1N4883
D687
1N4883
D688
1N4883
D689
1N4883
D690
1N4883
D691
1N4883
D692
1N4883
D693
1N4883
D694
1N4883
D695
1N4883
D696
1N4883
D697
1N4883
D698
1N4883
D699
1N4883
D700
1N4883
D701
1N4883
D702
1N4883
D703
1N4883
D704
1N4883
D705
1N4883
D706
1N4883
D707
1N4883
D708
1N4883
D709
1N4883
D710
1N4883
D711
1N4883
D712
1N4883
D713
1N4883
D714
1N4883
D715
1N4883
D716
1N4883
D717
1N4883
D718
1N4883
D719
1N4883
D720
1N4883
D721
1N4883
D722
1N4883
D723
1N4883
D724
1N4883
D725
1N4883
D726
1N4883
D727
1N4883
D728
1N4883
D729
1N4883
D730
1N4883
D731
1N4883
D732
1N4883
D733
1N4883
D734
1N4883
D735
1N4883
D736
1N4883
D737
1N4883
D738
1N4883
D739
1N4883
D740
1N4883
D741
1N4883
D742
1N4883
D743
1N4883
D744
1N4883
D745
1N4883
D746
1N4883
D747
1N4883
D748
1N4883
D749
1N4883
D750
1N4883
D751
1N4883
D752
1N4883
D753
1N4883
D754
1N4883
D755
1N4883
D756
1N4883
D757
1N4883
D758
1N4883
D759
1N4883
D760
1N4883
D761
1N4883
D762
1N4883
D763
1N4883
D764
1N4883
D765
1N4883
D766
1N4883
D767
1N4883
D768
1N4883
D769
1N4883
D770
1N4883
D771
1N4883
D772
1N4883
D773
1N4883
D774
1N4883
D775
1N4883
D776
1N4883
D777
1N4883
D778
1N4883
D779
1N4883
D780
1N4883
D781
1N4883
D782
1N4883
D783
1N4883
D784
1N4883
D785
1N4883
D786
1N4883
D787
1N4883
D788
1N4883
D789
1N4883
D790
1N4883
D791
1N4883
D792
1N4883
D793
1N4883
D794
1N4883
D795
1N4883
D796
1N4883
D797
1N4883
D798
1N4883
D799
1N4883
D800
1N4883
D801
1N4883
D802
1N4883
D803
1N4883
D804
1N4883
D805
1N4883
D806
1N4883
D807
1N4883
D808
1N4883
D809
1N4883
D810
1N4883
D811
1N4883
D812
1N4883
D813
1N4883
D814
1N4883
D815
1N4883
D816
1N4883
D817
1N4883
D818
1N4883
D819
1N4883
D820
1N4883
D821
1N4883
D822
1N4883
D823
1N4883
D824
1N4883
D825
1N4883
D826
1N4883
D827
1N4883
D828
1N4883
D829
1N4883
D830
1N4883
D831
1N4883
D832
1N4883
D833
1N4883
D834
1N4883
D835
1N4883
D836
1N4883
D837
1N4883
D838
1N4883
D839
1N4883
D840
1N4883
D841
1N4883
D842
1N4883
D843
1N4883
D844
1N4883
D845
1N4883
D846
1N4883
D847
1N4883
D848
1N4883
D849
1N4883
D850
1N4883
D851
1N4883
D852
1N4883
D853
1N4883
D854
1N4883
D855
1N4883
D856
1N4883
D857
1N4883
D858
1N4883
D859
1N4883
D860
1N4883
D861
1N4883
D862
1N4883
D863
1N4883
D864
1N4883
D865
1N4883
D866
1N4883
D867
1N4883
D868
1N4883
D869
1N4883
D870
1N4883
D871
1N4883
D872
1N4883
D873
1N4883
D874
1N4883
D875
1N4883
D876
1N4883
D877
1N4883
D878
1N4883
D879
1N4883
D880
1N4883
D881
1N4883
D882
1N4883
D883
1N4883
D884
1N4883
D885
1N4883
D886
1N4883
D887
1N4883
D888
1N4883
D889
1N4883
D890
1N4883
D891
1N4883
D892
1N4883
D893
1N4883
D894
1N4883
D895
1N4883
D896
1N4883
D897
1N4883
D898
1N4883
D899
1N4883
D900
1N4883
D901
1N4883
D902
1N4883
D903
1N4883
D904
1N4883
D905
1N4883
D906
1N4883
D907
1N4883
D908
1N4883
D909
1N4883
D910
1N4883
D911
1N4883
D912
1N4883
D913
1N4883
D914
1N4883
D915
1N4883
D916
1N4883
D917
1N4883
D918
1N4883
D919
1N4883
D920
1N4883
D921
1N4883
D922
1N4883
D923
1N4883
D924
1N4883
D925
1N4883
D926
1N4883
D927
1N4883
D928
1N4883
D929
1N4883
D930
1N4883
D931
1N4883
D932
1N4883
D933
1N4883
D934
1N4883
D935
1N4883
D936
1N4883
D937
1N4883
D938
1N4883
D939
1N4883
D940
1N4883
D941
1N4883
D942
1N4883
D943
1N4883
D944
1N4883
D945
1N4883
D946
1N4883
D947
1N4883
D948
1N4883
D949
1N4883
D950
1N4883
D951
1N4883
D952
1N4883
D953
1N4883
D954
1N4883
D955
1N4883
D956
1N4883
D957
1N4883
D958
1N4883
D959
1N4883
D960
1N4883
D961
1N4883
D962
1N4883
D963
1N4883
D964
1N4883
D965
1N4883
D966
1N4883
D967
1N4883
D968
1N4883
D969
1N4883
D970
1N4883
D971
1N4883
D972
1N4883
D973
1N4883
D974
1N4883
D975
1N4883
D976
1N4883
D977
1N4883
D978
1N4883
D979
1N4883
D980
1N4883
D981
1N4883
D982
1N4883
D983
1N4883
D984
1N4883
D985
1N4883
D986
1N4883
D987
1N4883
D988
1N4883
D989
1N4883
D990
1N4883
D991
1N4883
D992
1N4883
D993
1N4883
D994
1N4883
D995
1N4883
D996
1N4883
D997
1N4883
D998
1N4883
D999
1N4883
D1000
1N4883

หนังสืออ้างอิง

1. Takashi Kenjo , 1985 , Stepping Motor and their Microprocessor Controls, Clarendon press., Oxford

2. E.P.Popov and E.I. Yurevich, Robotics, Mir.Publeshers Moscow

3. ผศ.ดร.พูลพงษ์ บุญพราหมณ์, คอมพิวเตอร์ช่วยงานอุตสาหกรรม, สมาคมส่งเสริมเทคโนโลยี่ (ไทย-ญี่ปุ่น)