# การประยุกต์ใช้โรบอทแขนกล(แขนกล)

# ROBOT APPLICATION


โดย

นาย อนุสรณ์ วิริยะพงศ์ไพบูลย์

นางสาว อลิสรา จรรยหาญ

นางสาว อุไรวรรณ บุญส่ง

อาจารย์ที่ปรึกษา

อาจารย์ มนัส สังวรศิลป์




ปริญญานิพนธ์สำหรับวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์การใช้งานหุ่นยนต์ (แขนกล)


ผู้จัดทำ

| | | |
|---|---|---|
| 1.นายอนุสรณ์ | วิริยะพงศ์ไพบูลย์ | 37014556 |
| 2.นางสาวอลิสรา | จรรยหาญ | 37014572 |
| 3.นางสาวอุไรวรรณ | บูญส่ง | 37014600 |


(อาจารย์มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

การประยุกต์การใช้งานหุ่นยนต์(แขนกล)

ROBOT APPLICATION


| | | |
|---|---|---|
| 1. นายอนุสรณ์ | วิริยะพงศ์ไพบูลย์ | 37014556 |
| 2. นางสาวอลิสรา | จรรยหาญ | 37014572 |
| 3. นางสาวอุไรวรรณ | บุญส่ง | 37014600 |


โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

(อาจารย์มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

# กิตติกรรมประกาศ

ในการทำโครงงานครั้งนี้ทางคณะผู้จัดทำใคร่ขอกราบขอบพระคุณ รศ.ดร. มนัส สังวรศิลป์ เป็นอย่างสูง ที่ท่านคอยดูแลเอาใจใส่และให้คำปรึกษาอย่างใกล้ชิด และขอขอบคุณ พี่ เป๋าพี่ภาควิชาวิศวกรรมเครื่องกลที่ให้คำแนะนำคำปรึกษาทางด้าน Mechanic ของตัวแขนกล , ขอขอบพระคุณ อาจารย์ จำลอง ปราบแก้ว อาจารย์ประจำภาควิชาวิศวกรรมเครื่องกลอำนวยความสะดวกทางด้านสถานที่ในการทำส่วน Mechanic และ คุณ มณฑา เจ้าหน้าที่ประจำ Shop Mechanic ที่คอยให้คำปรึกษาทางด้านเครื่องมือที่ใช้ในการทำ Mechanic และ เพื่อนๆ ( อุ๊ , ปิง ) ที่ให้ยืมพาหนะในการซื้ออุปกรณ์ตลอดช่วงที่ทำโครงงาน

.....

(นาย อนุสรณ์ วิริยะพงศ์ไพบูลย์)

.....

(นางสาว อลิสรา จรรยหาญ)

.....

(นางสาว อุไรวรรณ บุญส่ง)

ผู้จัดทำ

# การประยุกต์การใช้งานหุ่นยนต์(แขนกล)

นาย อนุสรณ์ วิริยะพงศ์ไพบูลย์
นางสาว อลิสรา จรรยหาญ
นางสาว อุไรวรรณ บุญส่ง
อ.มนัส สังวรศิลป์ ( อาจารย์ที่ปรึกษา )
ภาคการศึกษาที่ 2 ปีการศึกษา 2540

**บทคัดย่อ**

ปัจจุบันนี้ความก้าวหน้าทางด้านเทคโนโลยีได้มีการพัฒนาไปอย่างรวดเร็วซึ่งในด้านอุตสาหกรรมในโรงงานส่วนใหญ่ได้มีการนำหุ่นยนต์(robot) มาใช้งานกันมากขึ้น เนื่องจากสามารถทำงานได้อย่างมีประสิทธิภาพ รวดเร็วและแม่นยำกว่ามนุษย์ ดังนั้นจึงได้การศึกษาจัดทำโครงงานชิ้นนี้ขึ้นมาเพื่อตอบสนองความต้องการในปัจจุบัน ซึ่งหุ่นยนต์(แขนกล)ที่จัดทำขึ้นนี้ได้มีการนำ DC มอเตอร์ และ สเตปปิ้งมอเตอร์ มาใช้งานในการ drive แขนกลในส่วนต่างๆ ให้สามารถทำงานได้ตามต้องการ ซึ่งจะทำการศึกษาการทำงานของ gripper ให้สามารถจับวัตถุได้ตามต้องการ

# ROBOT APPLICATION

Mr. Anusorn Viriyapongphaibul
Miss. Alisara Jarnyaharn
Miss. Uraiwan Boonsong
Mr. Manus Sungvorasilp (Advisor)
1st Semestor,Educational Year1997

**Abstract**

Today technology has developed rapidly. In industrial has brought robot to use in their job. Due to it can work with efficiency, rapidly and accurracy so our group has made this project to response demand in the present. This project, Robot A rm is made by using d.c. motor and stepping motor for rotate parts of Robot Arm for it can work with efficiency and learn about work of gripper for catdhing object with efficiency

# สารบัญ


| เรื่อง | หน้า |
|---|---|
| กิตติกรรมประกาศ | I |
| บทคัดย่อ | II |
| Abstract | III |
| สารบัญ | IV |
| สารบัญรูป | |
| สารบัญตาราง | |
| บทที่ 1 บทนำ | 1 |
| บทที่ 2 แขนกล | 2 |
| 2.1 ชนิดของหุ่นยนต์ | 2 |
| 2.2 มือของหุ่นยนต์ ( Robot end effector ) | 3 |
| 2.2.1 เมคคานิคกริปเปอร์ ( Mechanical gripper ) | 4 |
| 2.3 ความแม่นยำของการเคลื่อนที่ | 6 |
| 2.3.1 สปาเชียนรีโซลูชั่น | 6 |
| 2.3.1.1 ระบบควบคุม ( Control system ) | 6 |
| 2.3.1.2 ความคลาดเคลื่อนเชิงกล ( Mechanical inaccuracy ) | 6 |
| 2.3.2 แอคคูเรซี | 6 |
| 2.3.3 รีพีททีบิลิตี้ | 7 |
| 2.4 การเคลื่อนที่ของหุ่นยนต์ | 7 |
| 2.4.1 การเคลื่อนที่ของหุ่นยนต์สามารถแบ่งการเคลื่อนที่ออกได้เป็น | 7 |
| 2.4.1.1การเคลื่อนที่แบบเส้นตรง ( Linear ) | 7 |
| 2.4.1.2การเคลื่อนที่แบบหมุนรอบจุดหมุน ( Rotational ) | 7 |
| 2.4.1.3การเคลื่อนที่แบบบิดรอบจุดหมุน ( Twisting ) | 7 |
| 2.4.1.1การเคลื่อนที่แบบหมุนตั้งฉาก ( Revolving ) | 7 |
| 2.4.2 การเคลื่อนที่ของมือ | 8 |
| 2.4.2.1 หมุน ( Rool ) | 8 |

| เรื่อง | หน้า |
|---|---|
| 2.4.2.2บิด ( Pitch ) | 8 |
| 2.4.2.3ส่าย ( Yaw ) | 8 |
| บทที่ 3 การทำงานของสเตปปิ้งมอเตอร์และการทำงานของ DC มอเตอร์ | 9 |
| 3.1 การทำงานของสเตปปิ้งมอเตอร์ | 9 |
| 3.1.1 การกระตุ้นและควบคุมการหมุนของสเตปปิ้งมอเตอร์ | 11 |
| 3.2 การทำงานของมอเตอร์กระแสตรง ( DC MOTOR ) | 14 |
| 3.1.2 PULSE WIDTH MODULATION | 14 |
| บทที่ 4 ส่วนประกอบของแขนกล | 17 |
| 4.1 ส่วนฐาน ( base part ) | 19 |
| 4.2 ส่วนเฟืองที่หัวไหล่ ( shoulder part ) | 20 |
| 4.3 ส่วนข้อศอก ( elbow part ) | 21 |
| 4.4 ส่วนข้อมือ ( hand connecter part ) | 22 |
| 4.5 ส่วนมือจับ ( gripper part ) | 23 |
| บทที่ 5 โครงสร้างของโครงงาน | 25 |
| 5.1 อัลกอริทึมในการควบคุมแขนกล | 25 |
| 5.2 สมการ Direct kinematic | 26 |
| 5.2.1 การอ้างอิงมุมของแขนกล | 27 |
| 5.3 สมการ Inverse kinematic | 30 |
| 5.4 การประยุกต์ใช้งาน | 34 |
| บทที่ 6 การควบคุมและโปรแกรมการควบคุม | 36 |
| 6.1 ส่วน serial port | 36 |
| 6.1.1 การสื่อสารข้อมูล | 36 |
| 6.1.2 การส่งข้อมูลแบบอนุกรม | 37 |
| 6.1.3 อสมวารกับสมวาร | 38 |
| 6.1.3.1 การส่งข้อมูลแบบอสมวาร | 38 |
| 6.1.3.2 การส่งาข้อมูลแบบสมวาร | 38 |
| 6.1.4 รูปแบบการสื่อสาร | 39 |
| 6.1.5 มาตรฐาน RS-232 | 40 |
| 6.1.6 การใช้งาน | 41 |

| เรื่อง | หน้า |
|---|---|
| 6.1.7 ขั้วต่อแบบย่อ | 41 |
| 6.2 ส่วนประมวลผล(8031) และ output ที่ใช้ในการควบคุม | 42 |
| 6.2.1 ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวเบอร์ 8031 | 42 |
| 6.2.2 IC ขยายพอร์ตเบอร์ 8255 | 50 |
| 6.3 ส่วนที่เป็นวงจร drive motor ใช้ IC เบอร์ 18245 และ 18200 | 52 |
| 6.3.1 วงจร drive motor LMD 18245 | 52 |
| 6.3.2 วงจร drive motor LMD 18200 | 53 |
| 6.4 ส่วนโปรแกรมการควบคุม | 55 |
| 6.4.1 ส่วนโปรแกรมแอสแซมบลี้(8051) | 55 |
| 6.4.2 ส่วนโปรแกรม delphi | 55 |
| 6.4.2.1 ส่วนที่ใช้ในการคอนโทรล dc motor | 55 |
| 6.4.2.2 ส่วนที่ใช้ในการคอนโทรล stepping motor | 56 |
| บทที่ 7 วงจรและส่วนที่ใช้ในการควบคุม | 60 |
| 7.1 วงสเตปปิ้งมอเตอร์ | 60 |
| 7.2 วงจรในส่วนมอเตอร์กระแสตรง | 65 |
| 7.3 ส่วนที่ใช้ในการเชื่อมต่อกับส่วนต่างๆ | 65 |
| 7.3.1 ส่วน connect led | 65 |
| 7.3.2 ส่วน connect 8031 | 66 |
| 7.3.3 ส่วน connect supply and dc motor | 67 |
| 7.3.4 ส่วน connect stepping motor | 68 |
| 7.3.5 ส่วน connect supply microcontroller | 68 |
| 7.3.6 ส่วน connect robot arm | 69 |
| 7.3.7 ส่วน connect sensor | 69 |
| บทที่ 8 ผลการทดลอง | 70 |
| 8.1 ด้าน mechanic | 70 |
| 8.2 ด้าน hardware | 70 |
| 8.3 ด้าน software | 71 |
| บทที่ 9 สรุปและวิจารณ์ | 72 |
| ภาคผนวก ก | |
| ภาคผนวก ข | |

# สารบัญรูป


| รูปที่ | หน้า |
|---|---|
| รูปที่ 2.1 ชนิดของหุ่นยนต์ (a) โพล่าร์ (b) ไซเลนดริคัลคาร์ทีเซียน(d) อาร์ติคูเลต | 3 |
| รูปที่ 2.2 มือจับที่ออกแบบนิ้วให้มีรูปทรงคล้ายกับวัตถุ | 5 |
| รูปที่ 2.3 มือจับที่อาศัยแรงเสียดทานของนิ้วกับวัตถุ | 5 |
| รูปที่ 2.4 ความแม่นยำและความสามารถในการทำซ้ำของหุ่นยนต์ | 7 |
| รูปที่ 2.5 ลักษณะการเคลื่อนที่ของแขนแบบต่างๆ | 8 |
| รูปที่ 2.6 การเคลื่อนที่ของมือ | 8 |
| รูปที่ 3.1(ก)แสดงสเตปปิ้งมอเตอร์ที่มีการต่อวงจรขดลวดภายในเพื่อกระตุ้นให้เกิดขั้วแม่เหล็กขึ้น 1 ขั้ว ในทิศทางตรงกันข้าม ส่วนขดลวดอื่นๆ จะไม่ถูกกระตุ้นเลย (ข) แสดงการต่อวงจรขดลวดแบบกระตุ้นให้เกิดขั้วแม่เหล็กพร้อมกัน 2 ขั้วที่อยู่ใกล้กันทำให้โรเตอร์เคลื่อนที่มาอยู่ระหว่างขั้วแม่เหล็กทั้งสอง | 11 |
| รูปที่ 3.2 แสดง PWM amplifier สำหรับมอเตอร์กระแสตรง | 14 |
| รูปที่ 4.1 แสดงโครงสร้างของแขนกล | 17 |
| รูปที่ 4.2 ภาพแสดงโครงสร้างส่วนฐานของแขนกล | 19 |
| รูปที่ 4.3 ภาพแสดงส่วนเฟืองที่หัวไหล่ | 20 |
| รูปที่ 4.4 ภาพแสดงส่วนข้อศอก ( elbow part ) | 21 |
| รูปที่ 4.5 ภาพแสดงส่วนข้อมือ | 22 |
| รูปที่ 4.6 ภาพแสดงส่วน gripper | 23 |
| รูปที่ 5.1 บล็อกไดอะแกรมของแขนกล | 24 |
| รูปที่ 5.2 ไดอะแกรมการทำงานของแขนกลในรูปแบบ 5 ดีกรีออฟฟรีดอม | 26 |
| รูปที่ 5.3 บล็อกไดอะแกรมของสมการ direct kinematic | 26 |
| รูปที่ 5.4 ตำแหน่งโฮมของแขนกล | 27 |
| รูปที่ 5.5 แสดงตัวอย่างในการอ้างอิงมุม | 27 |
| รูปที่ 5.6 ตำแหน่งด้านของแขนกลชนิดอาร์ติคูเลตแบบ 5 ดีกรีออฟฟรีดอม | 28 |
| รูปที่ 5.7 บล็อกไดอะแกรมของสมการ inverse kinematic | 30 |

รูปที่ หน้า

รูปที่ 5.8 แสดงอัลกอริทึมของสมการ inverse kinematic 31
รูปที่ 5.9 แสดงการอ้างอิง $q_{234}$ 31
รูปที่ 5.10 ตัวอย่างการอ้างอิง $q_{234}$ 33
รูปที่ 6.1 การส่งข้อมูลแบบอนุกรม 37
รูปที่ 6.2 แสดงรูปแบบการสื่อสาร 39
รูปที่ 6.3 การเชื่อมต่อระหว่าง DTE กับ DCE 40
รูปที่ 6.4 ระบุตำแหน่งขาและหน้าที่ของขั้วต่อ DB-9 41
รูปที่ 6.5 แสดงหน้าที่ของพอร์ตเมื่อคอนโทรลเลอร์ทำงานกับหน่วยความจำภายนอก 43
รูปที่ 6.6 โครงสร้างภายในหน่วยความจำของ MCS 51 44
รูปที่ 6.7 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 โหมด 0 และโหมด 1 46
รูปที่ 6.8 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 โหมด 1 ในโหมด 2 47
รูปที่ 6.9 วงจรการทำงานของตัวจับเวลา/ตัวนับที่ 0 ในโหมด 3 47
รูปที่ 6.10แสดงพอร์ตที่ใช้ในการติดต่อกับส่วนฮาร์ดแวร์ 51
รูปที่ 6.11 ลักษณะสัญญาณต่างๆ ในการทำงานแบบ PWM 54
รูปที่ 6.12 รูปแสดงส่วนของโปรแกรม DELPHI บริเวณที่แสดงบิตข้อมูลที่ใช้ส่ง 55
รูปที่ 6.13 รูปแสดงส่วนของโปรแกรม DELPHI ส่วนที่ใช้ปรับความเร็วสเตปปิ้งมอเตอร์ 56
รูปที่ 7.1 รูปแสดงการต่อวงจรสเตปปิ้งมอเตอร์ 61
รูปที่ 7.2 รูปแสดงการต่อวงจร SUPPLY เพื่อเป็นไฟเลี้ยงคงที่ให้กับส่วน DC มอเตอร์ 62
รูปที่ 7.3 รูปแสดงการต่อวงจรควบคุม DC MOTOR โดยใช้ IC เบอร์ LMD 18245 63
รูปที่ 7.4 รูปแสดงการต่อวงจรควบคุม DC MOTOR โดยใช้ IC เบอร์ LMD 18200 64

# บทที่ 1
# บทนำ

คำว่า "หุ่นยนต์" นี้ต้นกำเนิดจริงๆ แล้วเกิดขึ้นจากนิยายวิทยาศาสตร์แบบเพ้อฝันหรือที่เราเรียกว่า แฟนตาซี ( fantasy) ของชาวเชคโกสโลวาเกีย (CZECHOSLOVAKIA) ในปี 1920 โดยทั่วไปแล้วพอพูดถึงคำว่า "หุ่นยนต์" คนทั่วไปมักจะคิดถึงเครื่องยนต์ที่มีรูปร่างเหมือนมนุษย์สามารถที่จะคิดและทำงานได้ด้วยตัวเอง แต่ที่จริงแล้วตามคำนิยาม ROBOTICS หมายถึงเครื่องยนต์ที่ประกอบกันขึ้นมาเพื่อทำงานบางอย่างแทนมนุษย์ สามารถควบคุมได้โดยใช้โปรแกรมทางคอมพิวเตอร์ และสามารถที่จะเปลี่ยนการทำงานให้ทำงานอย่างอื่นๆได้(ตามคำนิยามของสมาคมหุ่นยนต์โรงงาน ( Robotics Industrial Association,RIA) และสถาบันเทคโนโลยีเกี่ยวกับหุ่นยนต์ของอเมริกา (Robotics Institute of Amarica,RIA)

จากคำนิยามนี้จะเห็นว่าหุ่นยนต์ก็คือเครื่องยนต์อัตโนมัติชนิดหนึ่งที่สามารถเปลี่ยนโปรแกรมการทำงานได้เท่านั้นเอง เครื่องยนต์อัตโนมัติ ( Automatics) นี้แบ่งออกได้เป็น 3 ชนิดคือ

1. Fixed Automation คือเครื่องยนต์อัตโนมัติที่ทำงานอยู่กับที่และทำงานอย่างเดิมตลอด
2. Programmable Automation คือเครื่องยนต์อัตโนมัติที่สามารถเปลี่ยนการทำงานได้หลายอย่างแต่ต้องรองานที่ทำในตอนนั้นให้เสร็จก่อนจึงสามารถเปลี่ยนได้
3. Flesible Automation คือเครื่องยนต์อัตโนมัติที่สามารถเปลี่ยนการทำงานได้และสามารถทำงานได้หลายๆอย่างได้ในวงรอบการทำงานเดียวกัน

ในรายงานฉบับนี้จะเน้นที่หุ่นยนต์ในโรงงานอุตสาหกรรม ( Industrial Robotics) ดังนั้นคำว่าหุ่นยนต์หรือ Robotics ที่กล่าวถึงจะหมายถึงหุ่นยนต์ในโรงงานอุตสาหกรรมเท่านั้น

# บทที่ 2

# แขนกล

## 2.1 ชนิดของหุ่นยนต์

หุ่นยนต์แบ่งตามลักษณะการเคลื่อนที่และลักษณะของแขนแบ่งได้เป็น 4 ชนิดใหญ่ๆ คือ

1. Poly Configuration.
2. Cylindrical Configuration.
3. Cartesian Configuration.
4. Articulated Confiuration.

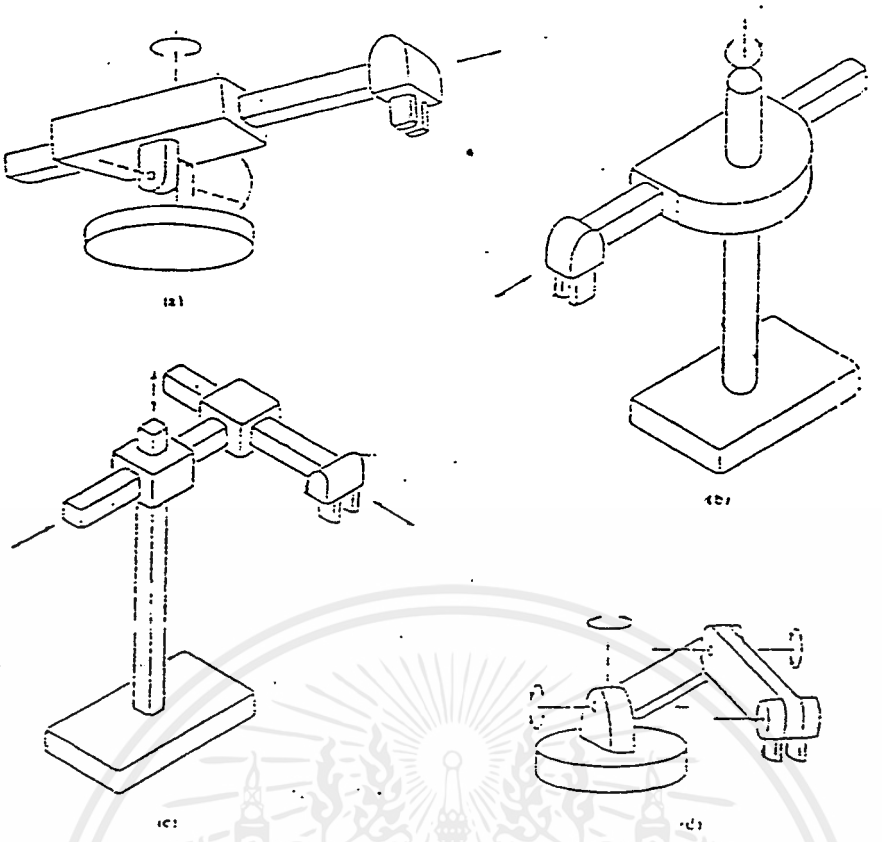
หุ่นยนต์ที่มีลักษณะเป็นแบบโพลาร์(Polar) นั้นลักษณะการเคลื่อนที่ของแขนจะสามารถยกขึ้นลงได้ในแนวดิ่งโดยยกทำมุมกับฐาน สามารถหมุนได้รอบตัว (รูป2.1a) พื้นที่การทำงานจะเป็นแบบทรงกลม ดังนั้นในบางครั้งจึงเรียกว่า "Spherical coordinate"

หุ่นยนต์ที่มีลักษณะเป็นแบบ ไซเลนดริคัล (Cylindrical) (รูป2.1b) เคลื่อนที่ขึ้นลงได้ตามแกนตั้งที่เป็นแกนหลัก และสามารถเคลื่อนที่ไปมาได้โดยใช้แกนนอน แกนดั้งสามารถที่จะหมุนได้ พื้นที่การทำงานจะเป็นแบบทรงกระบอก

หุ่นยนต์ที่มีลักษณะเป็นแบบ อาร์ติคูเลต(Articulated)(รูป2.1c) ลักษณะการเคลื่อนที่จะมีแกน 3 แกนเป็นเหมือนแกน X,Y,Z ดังนั้นในบางครั้งจึงเรียกว่าหุ่นยนต์เรคติลิเนียร์(Rectilinear Robot) พื้นที่การทำงานสามารถที่จะทำงานได้ในส่วนที่เป็นด้านของมือเพียงด้านเดียวเท่านั้นเพราะไม่มีการหมุนของฐาน

หุ่นยนต์ที่มีลักษณะเป็นอาร์ติคคูเลต หุ่นยนต์แบบนี้จะมีลักษณะใกล้เคียงกับแขนของมนุษย์มีข้อหมุนต่างๆ เหมือนกัน(รูป2.1d)ดังนั้นพื้นที่การทำงานจึงสามารถที่จะทำงานได้ในทุกตำแหน่งในระยะความยาวของแขน

ในโครงงานนี้ได้เลือกเอาแบบ อาร์ติคูเลต เป็นต้นแบบในการทำทั้งนี้เพราะว่ามีลักษณะเหมือนแขนของมนุษย์ ซึ่งคนส่วนใหญ่ก็มีความคิดอยู่แล้วว่า หุ่นยนต์ต้องมีลักษณะเหมือนมนุษย์ และอีกอย่างในการพัฒนาหุ่นยนต์ชนิด อาร์ติคูเลตนี้สามารถที่จะพัฒนาไปสู่การทำงานได้ดีกว่าชนิดอื่นๆ



รูปที่ 2.1ชนิดของหุ่นยนต์ (a)โพล่าร์(b)ไซเลนดริคัลคาร์ทีเชียน(d)อาร์ติคูเลต

ข้อดีข้อเสียของแต่ละชนิดนี้แตกต่างกันออกไป เพราะว่าลักษณะทางกายภาพแตกต่างกัน แต่ถ้ามองในแง่ของการทำงานที่เป็นแบบซ้ำๆ ที่เดิมตลอด ชนิดคาร์ทีเชียน จะสามารถทำงานได้ดีกว่า ( สามารถเคลื่อนที่ไปหาเป้าหมายโดยมีความผิดพลาดน้อยที่สุด) แต่ถ้ามองในแง่การเข้าถึงวัตถุ ชนิดที่เป็นแบบ โพล่าร์ และ อาร์ติคูเลต จะสามารถเข้าถึงวัตถุได้ดีกว่าชนิดอื่น และชนิดที่เป็นแบบไซเลนดริคัล จะมีข้อดีในแง่ที่สามารถยกวัตถุได้มากกว่าในงานทั่วไปแล้วใช้แบบโพล่าร์ และ ไซเลนดริคัลเพราะ 2 ชนิดนี้สามารถที่จะทำงานเป็นแบบ Load และ Unload โดยมีการเคลื่อนที่ของแขนออกไปในด้านข้างได้ดีกว่าชนิดอื่นๆ

## 2.2 มือของหุ่นยนต์ (Robot end effector)

มือจับของหุ่นยนต์คือตัวที่ทำหน้าที่ขั้นสุดท้ายของหุ่นยนต์มือจับของหุ่นยนต์สามารถแบ่งตามลักษณะของการทำงานได้ดังนี้

1. ทำงานเป็นตัวจับ (Gripper)
2. ทำหน้าที่เป็นเครื่องมือ (Tool)

มือของหุ่นยนต์ชนิดที่เป็นตัวจับมี หน้าที่โดยทั่วไปคือ การจับและการยกวัตถุสามารถแบ่งออกอีกเป็น 2 ชนิด คือ Single gripperและ Double Gripper . Single gripper หมายถึงมือจับที่มีการ

เคลื่อนที่ของนิ้วเพียงข้างเดียวอีกข้างหนึ่งอยู่กับที่ และ Double gripper มีการเคลื่อนที่ของนิ้วทั้ง 2 ข้างเข้าหากัน

มือของหุ่นยนต์ชนิดที่เป็นตัวจับ สามารถแบ่งออกเป็นหลายๆ ชนิดตามลักษณะของตัวควบคุมได้ดังนี้

2.2.1 เมคคานิคกริปเปอร์ (Mechanical gripper)

เมคคานิคกริปเปอร์ คือมือจับที่มีกลไกทางเมคคานิคเป็นตัวควบคุมนิ้วบางครั้งเรียกว่าจอว์ส (jaws)หน้าที่พื้นฐานของมือจับแบบเมคคานิค คือการส่งถ่ายกำลังจากมือไปสู่วัตถุต้นกำลังอาจใช้ไฟฟ้า,นิวเมติกส์,เชิงกลหรือไฮโดรลิค ลักษณะการจับของมือจับนี้สามารถที่จะออกแบบการจับได้ 2 อย่างคือ

1. ใช้ลักษณะทางกายภาพ เช่น ต้องการจับวัตถุเป็นทรงกลมตรงปลายมือก็ทำเป็นส่วนโค้งที่มีขนาดพอเหมาะกับวัตถุนั้น(รูป2.2)
2. ใช้ความเสียดทาน(รูป2.3)นิ้วไม่ได้ออกแบบให้มีลักษณะเหมือนกับรูปทรงของวัตถุแต่อาศัยแรงเสียดทานของนิ้วกับวัตถุโดยมีสมการของแรงเสียดทานคือ

$$\mu n_f F_g = w$$

เมื่อ $\mu$= สัมประสิทธิแรงเสียดทานของนิ้วมือกับวัตถุ

$n_f$= จำนวนนิ้วของมือจับ

$F_g$= แรงจับ

w= น้ำหนักของวัตถุ

สมการ2.1 นี้ใช้ เมื่อแรงโน้มถ่วงมีทิศทางขนานกับพื้นที่สัมผัส ถ้าทิศทางของแรงโน้มถ่วงเปลี่ยนไปจะใช้สมการ2.2 คือ

$$\mu n_f F_g = mg$$

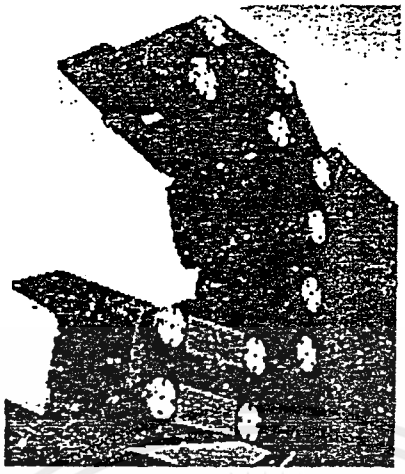
เมื่อ m= มวล

g คือgravity factor และความเร่ง

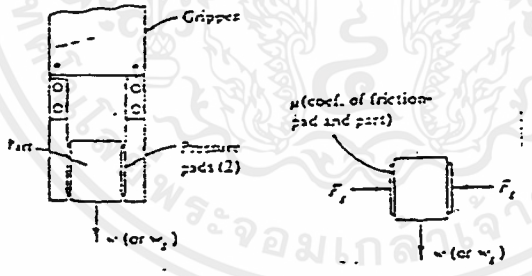
g=1 เมื่อความเร่งมีทิศทางตรงข้ามกับแรงโน้มถ่วง

g=2 เมื่อความ เร่งมีทิศทางในแนวราบ( ตั้งฉากกับแรงโน้มถ่วง)

g=3เมื่อความเร่งมีทิศทางเดียวกันกับแรงโน้มถ่วง



รูปที่2.2 มือจับที่ออกแบบนิ้วให้มีรูปทรงคล้ายกับวัตถุ




รูปที่2.3 มือจับที่อาศัยแรงเสียดทานของนิ้วกับวัตถุ

## 2.3 ความแม่นยำของการเคลื่อนที่

ความสำคัญอย่างหนึ่งของการสร้างหุ่นยนต์คือต้องการความแม่นยำในการทำงาน ความแม่นยำของหุ่นยนต์แต่ละตัวขึ้นกับตัวแปร 3 ประการคือ

1. สปาเชียนรีโซลูชั่น (Spatial resolution)
2. แอคคูเรซี(Accuracy)
3. รีพีททีบิลินี้ (Repeatability)

### 2.3.1 สปาเชียนรีโซลูชั่น

คือช่วงการเคลื่อนที่ที่มีระยะทางสั้นที่สุดที่หุ่นยนต์แต่ละตัวสามารถที่จะทำได้ สปาเชียนรีโซลูชั่นขึ้นอยู่กับองค์ประกอบสำคัญ 2 ประการคือ

2.3.1.1ระบบควบคุม(Control system) ระบบการควบคุมนี้จะรวมถึงการวัดสัญญาณป้อนกลับของหุ่นยนต์ด้วย ช่วงของการเคลื่อนที่ที่ระบบควบคุมสามารถที่จะทำได้ขึ้นอยู่กับหน่วยความจำหลักของเครื่องคอมพิวเตอร์เช่นเครื่องคอมพิวเตอร์ที่มีหน่วยความจำ 8 bit จะสามารถที่จะแบ่งช่วงการเคลื่อนที่ออกได้เป็น 256 ช่วง คือช่วงของการเคลื่อนที่ที่คอมพิวเตอร์แบ่งได้มีค่าเท่ากับ $2^n$ เมื่อ n คือหน่วยความจำหลักของเครื่อง

2.3.1.2ความคลาดเคลื่อนเชิงกล (Mechanical inaccuracy) ความคลาดเคลื่อนเชิงกลของหุ่นยนต์แต่ละตัวขึ้นอยู่กับลักษณะของข้อหมุน (joint) และ ข้อต่อ (link) และระบบต้นกำลังของหุ่นยนต์ตัวนั้นด้วย

### 2.3.2แอคคูเรซี

แอคคูเรซี คือ ตัวที่แสดงถึงความสามารถของหุ่นยนต์ในการที่เคลื่อนที่เข้าใกล้จุดเป้าหมายตามที่เราสั่ง แอคคูเรซี สามารถที่จะกำหนดให้อยู่ในเทอมของ สปาเชียนรีโซลูชั่น ได้ทั้งนี้เพราะว่าการเคลื่อนที่ให้เข้าใกล้จุดเป้าหมายก็ต้องขึ้นอยู่กับช่วงของการเคลื่อนที่ว่ามีความละเอียดมากน้อยเพียงใด ในการทำงานเราต้องวางจุดที่เราต้องการให้หุ่นยนต์ทำงานอยู่ระหว่างกลางของตำแหน่งการเคลื่อนที่ของหุ่นยนต์ทั้งนี้เพราะว่าความคลาดเคลื่อนเชิงกลมีผลต่อความแม่นยำของหุ่นยนต์

ความแม่นยำของ หุ่ นยนต์กำหนดให้เท่ากับครึ่งหนึ่งของระยะทางการเคลื่อนที่ที่สั้นที่สุดของหุ่นยนต์ที่สามารถทำได้ความแม่นยำของหุ่นยนต์ขึ้นอยู่กับองค์ประกอบเหล่านี้คือ

1. พื้นที่การ ทำงานของหุ่นยนต์ถ้าแขนทำงานในพื้นที่การทำงานจะมีความแม่นยำมากกว่าเมื่อแขนออกนอกพื้นที่การทำงาน
2. วงรอบการทำงาน ถ้าวงรอบการทำงานเป็นวงรอบที่แน่นอนความแม่นยำจะมีมากขึ้น

3. น้ำหนักที่ได้รับถ้าหุ่นยนต์ทำงานโดยก่ารรับน้ำหนักมากๆ ความแม่นยำจะลดลง

2.3.3 รีพีททีบิลิตี้

รีพีททีบิลิตี้ คือความสามารถของหุ่นยนต์ในการกลับมาทำงานซ้ำที่เดิมจากรูป2.1 จุดเป้าหมายคือจุด Tแต่เนื่องจากข้อจำกัดของหุ่นยนต์ หุ่นยนต์ไม่สามารุที่จะทำงานที่จุด T ได้ดังนั้นจึงทำงานที่จุด Pระยะห่างระหว่างจุด Pกับจุด T คือตัวแสดงถึงความแม่นยำของหุ่นยนต์แต่เมื่อสั่งให้หุ่นยนต์กลับมาทำงานที่จุด P

อีกครั้งหนึ่งหุ่นยนต์ไม่สามารถกลับมาทำงานที่จุด P ได้จะทำงานที่จุด R แทนระยะห่างระหว่างจุด P และจุด R คือค่าแสดงถึงรีพีททีบิลิตี้


รูปที่2.4 ความแม่นยำและความสามารถในการทำซ้ำของหุ่นยนต์

## 2.4 การเคลื่อนที่ของหุ่นยนต์

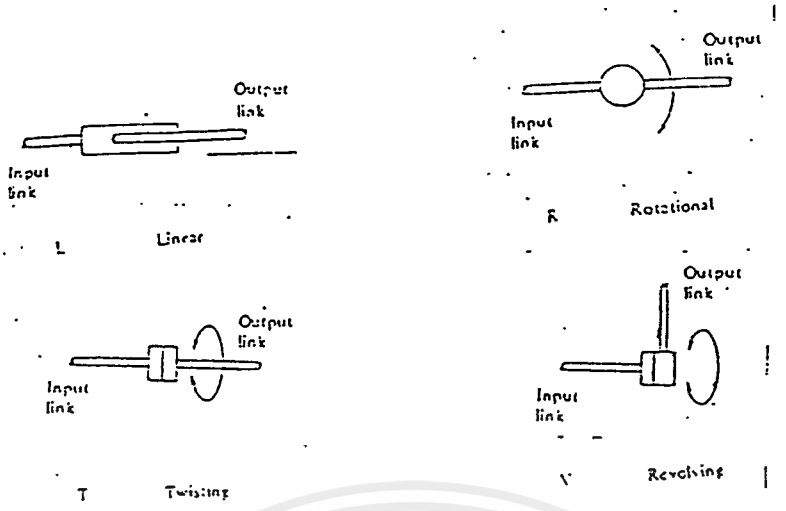
**2.4.1 การเคลื่อนที่ของหุ่นยนต์สามารถแบ่งการเคลื่อนที่ออกได้ดังนี้คือ**

**2.4.1.1 การเคลื่อนที่แบบเส้นตรง (Linear)**

**2.4.1.2 การเคลื่อนที่แบบหมุนรอบจุดหมุน (Rotational)**

**2.4.1.3 การเคลื่อนที่แบบบิดรอบจุดหมุน(Twisting)**

**2.4.1.4 การเคลื่อนที่แบบหมุนตั้งฉาก (Revolving)**



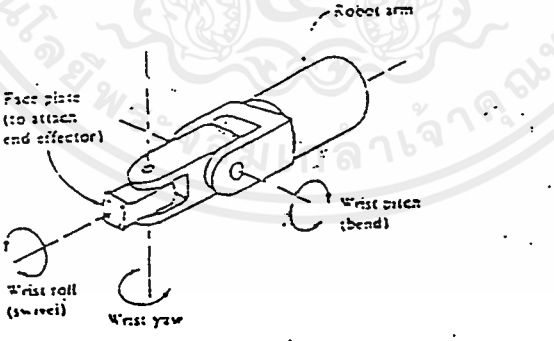

รูปที่2.5 ลักษณะการเคลื่อนที่ของแขนแบบต่างๆ

2.4.2 การเคลื่อนที่ของมือ

2.4.2.1 หมุน (Rool) บางครั้งเรียกว่า Swivasl ข้อมือหมุนรอบแกนของแขน

2.4.2.2 บิด (Pitch) บางครั้งเรียกว่า Bend ข้อมือจะยกขึ้นลงในแนวดิ่ง

2.4.2.3 ส่าย (Yaw) หมายถึงการบิดไปมาทางซ้ายและขวาของแกนมือ




รูปที่2.6 การเคลื่อนที่ของมือ

# บทที่ 3

# การทำงานของสเตปปิ้งมอเตอร์และการทำงานของ DC มอเตอร์

## 3.1 การทำงานของสเตปปิ้งมอเตอร์

สเตปปิ้งมอเตอร์มีความแตกต่างจากมอเตอร์ทั่วๆไป โดยเมื่อป้อนกำลังไฟฟ้าให้กับมัน จะหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งแตกต่างจากมอเตอร์ทั่วๆ ไปซึ่งจะหมุนทันทีและตลอดเวลา สเตปปิ้งมอเตอร์สามารถกำหนดตำแหน่งของการหมุนด้วยตัวเลขได้อย่างละเอียดโดยการใช้คอมพิวเตอร์เป็นตัวกำหนดและจะเก็บตัวเลขเหล่านั้นไว้

สเตปปิ้งมอเตอร์สามารถใช้งานในระบบเปิด ( open loop system ) นั้นก็คือมันทำงานได้โดยไม่ต้องการการป้อนกลับ ( feedback) แต่ทุกวิธีที่ต้องการกำหนดตำแหน่งได้อย่างถูกต้องจำเป็นที่ต้องการการป้อนกลับไปยังระบบให้รับรู้และอะไรจะเป็นตัวบอกได้ว่าตำแหน่งถูกต้องแล้วหรือเกิดการผิดพลาด ( error )

วิธีหนึ่งที่ใช้กันโดยทั่วไปกับสเตปปิ้งมอเตอร์ก็คือการใช้สวิตซ์ที่ติดตั้งไว้ที่ตำแหน่งที่ต้องการตรวจจับ ( limit switch ) เมื่อสเตปปิ้งมอเตอร์เริ่มหมุนและหมุนจนกระทั่งถึงตำแหน่งของสวิตซ์ตรวจจับสัญญาณก็จะถูกป้อนกลับเข้าสู่ระบบและทราบการทำงานของสเตปปิ้งมอเตอร์ได้ตลอดเวลา ซึ่งโดยปกติในวงจรคอนโทรลเลอร์จะมีการกำหนดจุดอ้างอิง ( reference point ) ไว้ด้วยเพื่อให้เริ่มทำงานและอ้างอิงตำแหน่งได้อย่างถูกต้อง

มอเตอร์ทั่วไปการที่จะทำให้เกิดการหมุนของโรเตอร์ ( rotor ) ได้ต้องมีการกระทำของสนามแม่เหล็กที่เกิดขึ้นระหว่างโรเตอร์และสเตเตอร์ ( stator ) ซึ่งขึ้นอยู่กับการจัดวางขั้วแม่เหล็ก ( pole ) การหมุนทำได้ทั้งแบบต่อเนื่องและกลับทิศทางไปมา โดยกระบวนการทางไฟสลับหรือการจัดวางแปรงถ่าน และการจัดแยกคอมมิวเตเตอร์ และทำการสวิตซ์ชิ่งกำลังไฟฟ้าให้เกิดแรงดึงดูดของแม่เหล็ก ( magnetic ) ที่ขั้วแม่เหล็กสร้างและหยุดสลับกัน ผลก็คือเกิดสนามแม่เหล็กหมุนขึ้นบนสเตเตอร์โดยการจ่ายกำลังไฟฟ้าทีละคู่ของขั้วแม่เหล็กในทิศทางตรงกันข้ามไปตลอดเวลา และเมื่อต้องการหยุดหมุนทำได้โดยหยุดการเกิดขั้วแม่เหล็กที่จุดหนึ่งโดยหยุดการสวิตชิ่งในลำดับต่อไปเสีย การหมุนกลับทิศทางก็ทำได้เช่นเดียวกับที่กล่าวมาแล้วเพียงแต่ทำการสวิตชิ่งกำลังไฟฟ้าให้เกิดสนามแม่เหล็กหมุนในทิศทางกลับกัน หรือกลับลำดับการสวิตซ์ของมัน

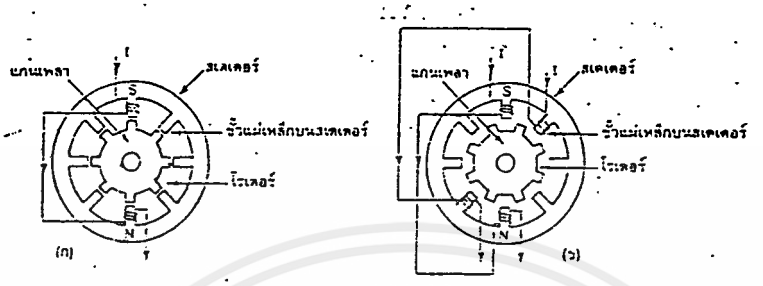
โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ประกอบขึ้นจากแผ่นเหล็กวงแหวนที่มีซี่ยื่นออกมา แต่ละซี่เหล่านั้นจะมีคอยล์พันสวมอยู่ ดังนั้นเมื่อป้อนกระแสไฟฟ้าผ่านคอยล์ทำให้เกิดสนามแม่เหล็ก

ไฟฟ้า( electromagnetic ) ขึ้น ด้านตรงข้ามของแต่ละขั้วแม่เหล็กจะได้รับกระแสไฟฟ้าในขณะเดียวกัน แต่ว่าจะไหลวนในทิศทางตรงกันข้ามทำให้เกิดสนามแม่เหล็กไฟฟ้าทิศทางตรงข้ามขึ้น ดังแสดงในรูปที่ 1 ( ก ) ดังนั้นถ้าเพิ่มจำนวนของขั้วแม่เหล็กมากขึ้นจะเพิ่มจำนวนของสเตปต่อวงรอบมากขึ้นตามไปด้วย

อย่างไรก็ตามผู้ใช้งานสามารถเพิ่มจำนวนของสเตปได้อีกวิธีหนึ่งโดยไม่ต้องปรับเปลี่ยนโครงสร้างภายใน โดยทำการจ่ายกำลังไฟฟ้าไปยังขั้วแม่เหล็ก 2 ขั้วที่อยู่ใกล้กันในเวลาเดียวกัน ซึ่งจะทำให้โรเตอร์หยุดหมุนอยู่ระหว่างกลางของ 2 ขั้วแม่เหล็กนั้นหรือเคลื่อนที่ไปครึ่งสเตปเท่านั้น และวิธีการนี้ยังช่วยให้เกิดแรงบิด ( torque ) มากขึ้นด้วย ดังแสดงในรูปที่ 1( ข )

สเตปปิ้งมอเตอร์โดยทั่วไปมีจำนวนของขั้วแม่เหล็กหรือจำนวนสเตปปิ้งต่อรอบเป็นจำนวนมาก ปกติอยู่ที่ประมาณ 100-400 สเตอต่อรอบ การมีจำนวนสเตปมาก ๆนี้ ไม่ได้เพิ่มที่จำนวนขั้วแม่เหล็กไฟฟ้าที่สเตเตอร์ แต่ทำได้โดยเพิ่มจำนวนซี่ขั้วแม่เหล็กที่โรเตอร์ จำนวนสเตปต่อรอบทั้งหมดจะได้จากการคูณจำนวนขั้วแม่เหล็กบนสเตเตอร์และจำนวนซี่ที่โรเตอร์ ดังเช่นถ้ามีขั้วแม่เหล็ก 3 ขั้วบนสเตเตอร์ และ 8 ซี่ขั้วแม่เหล็กบนโรเตอร์ สเตปปิ้งมอเตอร์ตัวนี้ จะทำงานที่ 24 สเตปต่อรอบหรือ หมุนเป็นมุม 15 องศา ต่อสเตป

การใช้วงจรดิจิตอลคอนโทรลเลอร์กำหนดการจ่ายกำลังไฟฟ้าเข้าสู่ขดลวดบนสเตเตอร์แบบซีเควนเชียลทำให้สามารถควบคุมการเคลื่อนที่ทุกสเตปได้ เช่นเดียวกับการควบคุมในวงจรดีซีเซอร์โว ( DC servo ) แต่การควบคุมด้วยดิจิตอลไม่จำเป็นต้องมีการป้อนกลับ การเคลื่อนที่ทุกสเตปได้จากการคำนวณจำนวนรอบหรือมุมในการหมุนที่ต้องการ แล้วจึงส่งข้อมูลที่ได้ไปควบคุมการหมุนของมอเตอร์ พิกัดในการทำงานอาทิความเร็ว , มุมในการเคลื่อนที่ , ตำแหน่งของเพลาถูกกำหนดจากข้อมูลที่ส่งมาควบคุมสเตปปิ้งมอเตอร์




รูปที่ 3.1(ก) แสดงสเตปปิ้งมอเตอร์ที่มีการต่อวงจรขดลวดภายในเพื่อกระตุ้นให้เกิดขั้วแม่เหล็กขึ้น 1 ขั้ว ในทิศทางตรงกันข้าม ส่วนขดลวดอื่นๆ จะไม่ถูกกระตุ้นเลย

(ข) แสดงการต่อวงจรขดลวดแบบกระตุ้นให้เกิดขั้วแม่เหล็กพร้อมกัน 2 ขั้วที่อยู่ใกล้กันทำให้โรเตอร์เคลื่อนที่มาอยู่ระหว่างขั้วแม่เหล็กทั้งสอง

### 3.1.1 การกระตุ้นและควบคุมการหมุนของสเตปปิ้งมอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเตปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้องด้วย แบ่งออกได้เป็น 3 รูปแบบ คือ แบบเวฟ , แบบ 2 เฟส และแบบครึ่งสเตป ทั้ง 3 แบบต่างก็มีข้อดีและข้อเสียแตกต่างกันออกไป

แบบเวฟเป็นแบบที่ง่ายที่สุดโดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งเรียงถัดกันไป ดังเช่น ขดที่ 1,2,3,4,1 หรือ 1,4,3,2,1 ซึ่งอยู่กับทิศทางที่ต้องการหมุนดังนั้นจึงมีขดลวดเพียงขดเดียวที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 3.1

ตารางที่ 3.1 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบเวฟ

| สเตปที่ | เฟสที่ 1 | เฟสที่ 2 | เฟสที่3 | เฟสที่4 |
|---|---|---|---|---|
| 1 | ทำงาน | — | — | — |
| 2 | — | ทำงาน | — | — |
| 3 | — | — | ทำงาน | — |
| 4 | — | — | — | ทำงาน |

แบบ 2 เฟสเป็นการกระตุ้นอีกรูปแบบหนึ่งซึ่งคล้ายกับแบบเวฟ แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกันและเรียงลำดับกันไปเช่นเดียวกับแบบเวฟ คือ ขดลวดที่ถูกกระตุ้น 12,23,34,41,12 หรือ 14,43,32,21,14 ขึ้นอยู่กับทิศทางการหมุน การเพิ่มจำนวนของขดลวดที่ถูกกระตุ้นนี้ทำให้เพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกันและต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียก็คือการกระตุ้นแบบนี้ต้องใช้แหล่งจ่ายกำลังไฟฟ้ามากขึ้นขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 3.2

ตารางที่ 3.2 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบ 2 เฟส

| สเตปที่ | เฟสที่ 1 | เฟสที่ 2 | เฟสที่3 | เฟสที่4 |
|---|---|---|---|---|
| 1 | ทำงาน | ทำงาน | — | — |
| 2 | — | ทำงาน | ทำงาน | — |
| 3 | — | — | ทำงาน | ทำงาน |
| 4 | ทำงาน | — | — | ทำงาน |

แบบครึ่งสเตปเป็นรูปแบบที่เกิดขึ้นจากการผสมผสานระหว่างการกระตุ้นแบบเวฟและแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเตปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปตามลำดับดังนี้ ขดลวดที่ถูกกระตุ้น 1,12,2,23,3,34,4,41,1
หรือในการหมุนอีกทิศทางหนึ่งจะได้เป็น 1,14,4,43,3,32,2,21,1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีกเพราะช่วงสเตปมีระยะสั้นลงและแต่ละสเตปเกิดแรงดึงจากขดลวด 2 ขด ที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีมากขึ้นแต่ต้องพึงระวังไว้อีกประการหนึ่งว่าเมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2 สเตปจึงจะได้เท่ากับ 1 สเตปเต็ม เหมือนกับในการควบคุม

2 แบบแรก สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้เทียบเท่ากับแบบ 2 เฟส จึงจะเพียงพอ ขั้นตอนการทำงานต่างๆ แสดงได้ดังตารางที่ 3.3

ตารางที่ 3.3 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบครึ่งสเตป

| สเตปที่ | เฟสที่ 1 | เฟสที่ 2 | เฟสที่3 | เฟสที่4 |
|---|---|---|---|---|
| 1 | ทำงาน | — | — | — |
| 2 | ทำงาน | ทำงาน | — | — |
| 3 | — | ทำงาน | — | — |
| 4 | — | ทำงาน | ทำงาน | — |
| 5 | — | — | ทำงาน | — |
| 6 | — | — | ทำงาน | ทำงาน |
| 7 | — | — | — | ทำงาน |
| 8 | ทำงาน | — | — | ทำงาน |

## 3.2 การทำงานของ DC มอเตอร์

ในโครงงานนี้มีการใช้งาน DC มอเตอร์อยู่ 4 ตัวด้วยกัน คือ มอเตอร์ขนาด 20 volts 2.5 amp 80 rpm 2 ตัว , มอเตอร์ขนาด 24 volts 1 ตัว และ มอเตอร์ 12 volts ( มอเตอร์ปัดน้ำฝน ) ดังนั้นการควบคุม DC มอเตอร์จึงเป็นส่วนสำคัญอีกส่วนหนึ่งในโปรเจ็คนี้ หลักการทำงานในการคอนโทรลความเร็ว DC มอเตอร์ที่ใช้ทั่วไปจะเป็นแบบ PWM ( pulse width modulation )

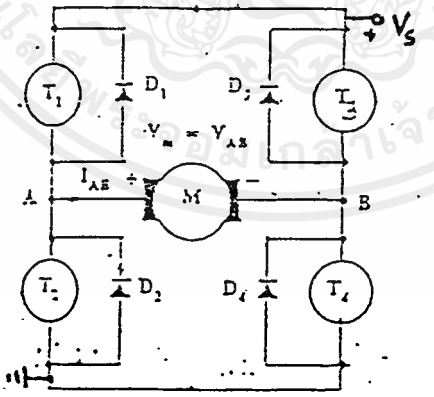
### 3.2.1 PULSE WIDTH MODULATION

ระบบ PWM ปกติจะใช้แหล่งจ่ายไฟกระแสตรงและ amplifier เป็นตัวสวิตว์ชิ่งซัพพลายโวลท์เดจให้ on และ off ที่ความถี่คงที่และมีช่วงเวลาของการ on ที่ปรับค่าได้ในหนึ่งคาบหรือที่เรียกว่า การปรับ duty cycle การปรับ duty cycle นั้นเป็นการปรับค่าเฉลี่ยของ voltage ที่จ่ายให้แก่โหลดได้ในที่นี้ก็คือ DC มอเตอร์นั้นเอง

เมื่อโหลดเพิ่มขึ้นความถี่ของ PWM amplifier จะคงที่แต่จะเปลี่ยนค่า duty cycle ตามโหลด สวิตช์ชิ่ง ampifier สามารถควบคุมความเร็วต่ำๆ โดยมีแรงบิดสูงอยู่โดยไม่สิ้นเปลืองพลังงานเหมือนพวก amplifier เชิงเส้น amplifier แบบ PWM สามารถทำงานได้ 3 ลักษณะ คือ

1. ไบโพล่าร์
2. ยูนิโพล่าร์
3. ลิมิตยูนิโพล่าร์

สำหรับแบบ ไบโพล่าร์เป็นแบบที่ง่ายที่สุดดูการทำงานตามรูปที่ 3.2




รูปที่ 3.2 แสดง PWM amplifier สำหรับมอเตอร์กระแสตรง

ซึ่งส่วนประกอบของรูป 3.1 นั้นจะประกอบอยู่ในส่วนของ IC เบอร์ LMD 18245 ที่เลือกใช้ในการคอนโทรล DC มอเตอร์ในโครงงาน

โดยที่เราจะกำหนดให้มีความถี่การสวิตซ์ให้เป็น fs โดยให้ $t_{on}$ เกิดขึ้นในช่วงแรกของคาบและ $t_{off}$ เกิดขึ้นในช่วงหลัง

$t_{on}$ เมื่อ $0 \leq t \leq t_1$

$t_{off}$ เมื่อ $t_1 \leq t \leq t_f$

ไบโพล่าร์จะมี $T_1$ และ $T_4$ นำกระแสระหว่าง เฟส on และส่วน $T_2$ และ $T_3$ จะนำกระแสขณะเฟส off จะได้ function ตกคร่อมมอเตอร์เป็น

$$V_{mOTOR} = V_{AB} = \begin{cases} V_S ; 0 < t < t_1 \\ V_S ; t_1 < t < t_f \end{cases}$$

ยูนิโพล่าร์จะลดจำนวนทรานซิสเตอร์ในการสวิตซ์ลง การสวิตซ์ขึ้นกับ $V_{in}$ เป็นบวก หรือ ลบ เมื่อ $V_{in}$ เป็น บวก $T_4$ จะนำกระแสตลอดคาบในขณะที่ $T_1$ นำกระแสในช่วงเฟส on และ $T_2$ จะนำกระแสในช่วงเฟส off เมื่อ $V_{in}$ เป็นลบ $T_2$ จะนำกระแสตลอดโดยมี $T_3$ และ $T_4$ สลับกันทำงาน ถ้า $V_{in}$ เป็นบวกจะได้

$$V_{in} = \begin{cases} V_S \text{ เมื่อ } t_{on} \\ 0 \text{ เมื่อ } t_{off} \end{cases}$$

การแสดงค่าในทางลบจะเหมือนกันเพียงแต่ เป็นลบเท่านั้น

จากลักษณะของ 2 แบบ ที่กล่าวมานั้นเป็นประโยชน์เหมือนกันซึ่งในแต่ละกรณีจะมีทรานซิสเตอร์คู่หนึ่ง $T_1$, $T_2$ หรือ $T_3$, $T_4$ จะหยุดนำกระแสขณะที่อีกคู่หนึ่งนำกระแส ซึ่งมีเวลาเก็บสะสมและเวลาที่ปล่อยออกของทรานซิสเตอร์เกิดขึ้นและมันอาจเป็นไปได้ที่ทรานซิสเตอร์ทั้งหมดนำกระแสในเวลาเดียวกัน ซึ่งจะทำให้เกิดการลัดวงจรของแหล่งจ่ายไฟ เราจำเป็นต้องหลีกเลี่ยงสภาวะดังกล่าวซึ่งสามารถทำได้โดยการสร้างช่วงเวลาหน่วง deray time ระหว่างการหยุดและนำกระแสของทรานซิสเตอร์และด้วยเหตุผลดังกล่าวความถี่ของการสวิตซ์จะถูกจำกัดในวงที่แคบลง

ลิมิตยูนิโพล่าร์ จะแสดงให้เห็นคือมีความจำเป็นต้องมีช่วงเวลาหน่วงซึ่งการสวิตซ์ขึ้นกับค่า $V_{in}$ เมื่อ $V_{in}$ เป็น บวก $T_4$ จะนำกระแสตลอด โดย $T_1$ จะเป็นสวิตซ์ on ในช่วงเฟส on ดังนั้นในช่วงเฟส on ทั้ง $T_1$ และ $T_4$ จะon ยังผลแก่โวลท์เตจของมอเตอร์ $V_m$ คือ

$$V_{motor} = V_S \text{ เมื่อ } t_{on}$$

ระหว่างเฟส off จะมี $T_4$ นำกระแสเพียงตัวเดียวเป็นผลให้ $V_{in}$ ขึ้นกับ $I_{AB}$ ตราบใดที่ $I_{AB} > 0$ ซึ่งเป็นสภาวะปกติเมื่อ $V_{AB} > 0$ กระแส $I_{AB}$ จะไหลผ่าน $D_2$ และ $T_4$ เป็นผลให้ $V_A = 0$ และ

$$V_{motor} = V_{AB} \text{ เมื่อ } t_{off} \text{ และ } I_{AB} > 0$$

ในกรณีที่ $I_{AB}$ เป็นลบกระแสจะไหลผ่าน $D_1$ และ $D_4$ เป็นผลให้ $V_A = V_S$ และ

$$V_{motor} = V_{AB} = V_S \text{ เมื่อ } t_{off} \text{ และ } I_{AB} < 0$$

ซึ่งจะเกิดขึ้นภายหลังเปลี่ยนขั้ว $V_{in}$

ในที่สุด ถ้าเราสามารถทำให้ $I_{AB} = 0$ (เข้าใกล้ศูนย์จนถือว่าเป็นศูนย์ )จะทำให้ ทั้ง $D_1$และ $D_4$ ไม่นำกระแสและโวสท์เตจ Vin จะอยู่ระหว่างค่าศูนย์และ Vs ดังต่อไปนี้

$$0 < V_{motor} < V_S \text{ เมื่อ } t_{off} \text{ และ } I_{AB} = 0$$

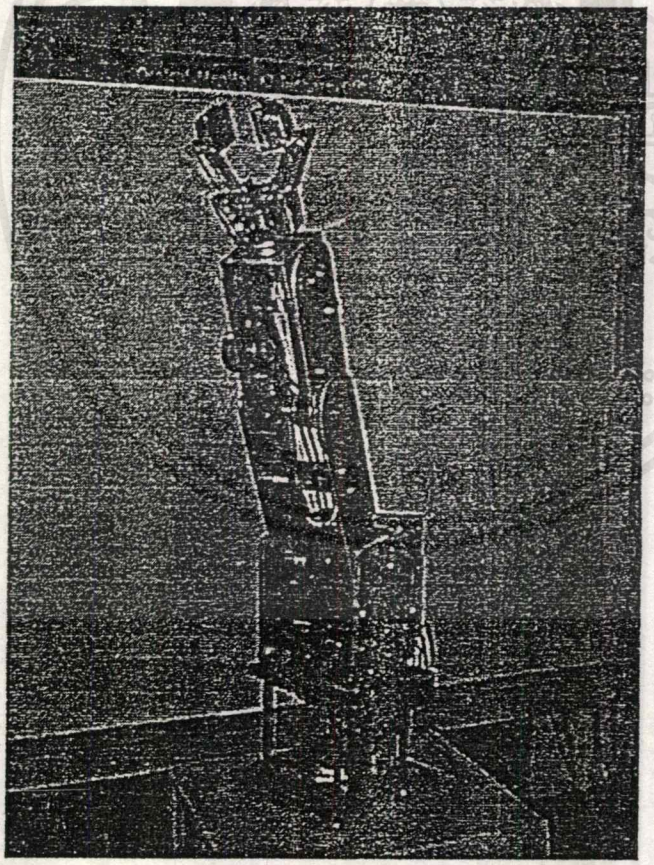
อย่างไรก็ตาม ถ้า $I_{AB} > 0$ เป็นสภาวะปกติ เมื่อ $V_{in} > 0$ แบบยูนิโพล่าร์และแบบลิมิตยูนิโพล่าร์จะแสดงคุณสมบัติคล้ายกันมาก ซึ่งเราสามารถสรุป mode การทำงานและผลของโวลท์เตจ

# บทที่ 4
# การประกอบแขนกล

แขนกลที่ประกอบขึ้นเป็นแขนกลแบบอาร์ติคูเลต ซึ่งมีการเคลื่อนที่ 5 ดีกรีออฟฟรีดอม (ไม่รวมgripper) ใช้ DC มอเตอร์ 4 ตัว และสเตปปิ้งมอเตอร์ 2 ตัวในการ drive ในแต่ละส่วนของแขนกล

วัสดุที่ใช้ในการประกอบแขนกลนี้ใช้อลูมิเนียมและทองเหลืองเพื่อความแข็งแรงของโครงสร้างของแขนกลเอง และขนาดในแต่ละส่วนได้ออกแบบตามความเหมาะสมของโครงสร้าง

การ desige แขนกลเน้นที่มอเตอร์ต้นกำลังทั้งหมดซึ่งจะอยู่ตรงบริเวณส่วนฐานของแขนกลเพื่อไม่ให้เกิด moment ที่บริเวณแขนท่อนบนของแขนกลขึ้น และ ช่วยลด load ที่จะเกิดขึ้นกับตัว motor โดยการส่งกำลังระหว่าง motor กับข้อต่อส่วนต่างๆ ของแขนจะใช้สายพานและเฟืองสายพานเป็นตัวช่วยในการส่งกำลังไปยังส่วนต่างๆ ของข้อต่อแต่ละที่



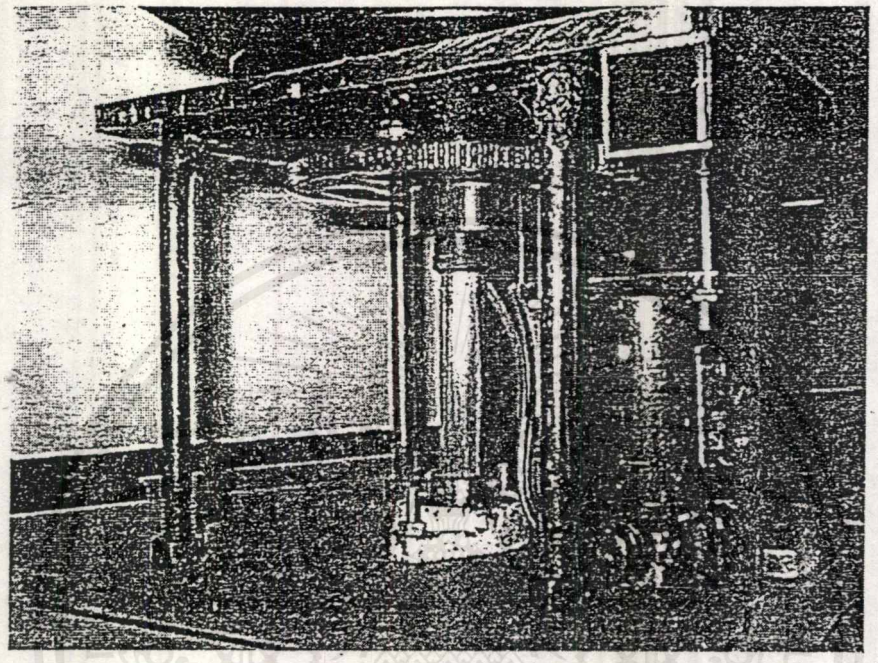
รูปที่ 4.1 แสดงโครงสร้างของแขนกล

จากรูปที่ 4.1 แสดงโครงสร้างของแขนกลซึ่งประกอบด้วยส่วนต่างๆ ดังนี้

1. ส่วนฐาน ( base part )
2. ส่วนชุดเฟืองที่หัวไหล่ (shoulder part)
3. ส่วนข้อศอก (elbow part)
4. ส่วนข้อมือ ( hand - connecter part )
5. ส่วน gripper (gripper part )

โดยทั้ง 5 ส่วนจะต้องทำงานสัมพันธ์กันเพื่อที่จะทำให้แขนกลทำงานได้อย่างมีประสิทธิภาพ

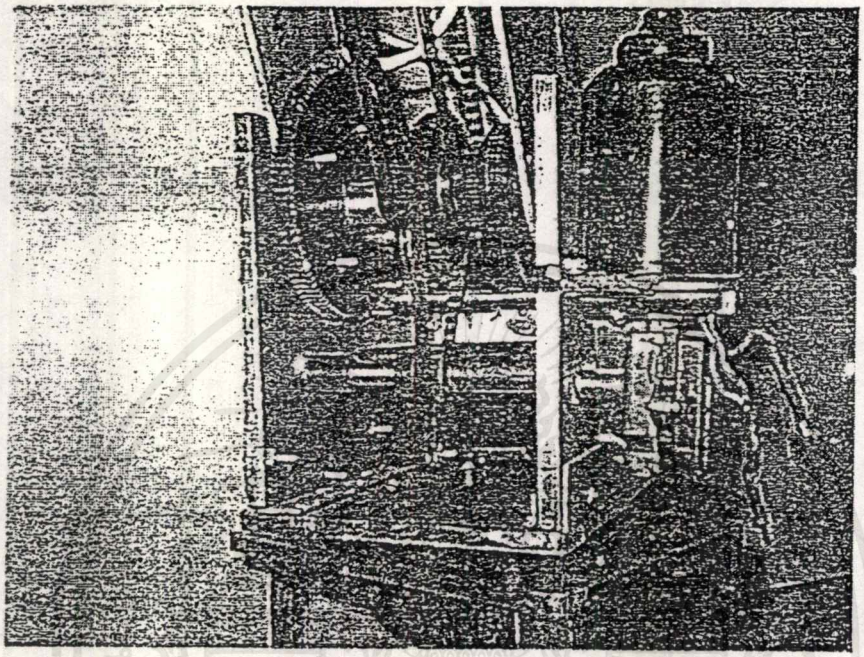
## 4.1 ส่วนฐาน ( base part )



รูปที่ 4.2 ภาพแสดงโครงสร้างส่วนฐานของแขนกล

จากรูปที่ 4.2 จะประกอบด้วย DC มอเตอร์ 1 ตัวซึ่งมีขนาด 20 โวลต์ 2.5 แอมป์ 80 RPM DC มอเตอร์ตัวนี้จะมีเฟืองขนาดเล็กติดอยู่เพื่อทำการทดกำลังไปยังเฟืองตัวใหญ่เพื่อทำหน้าที่ในการหมุนแขนกลทั้งชุดโดยสามารถหมุนได้ 360 องศา ซึ่งส่วนฐานนี้ต้องสามารถรับน้ำหนักของแขนกลทั้งชุดได้

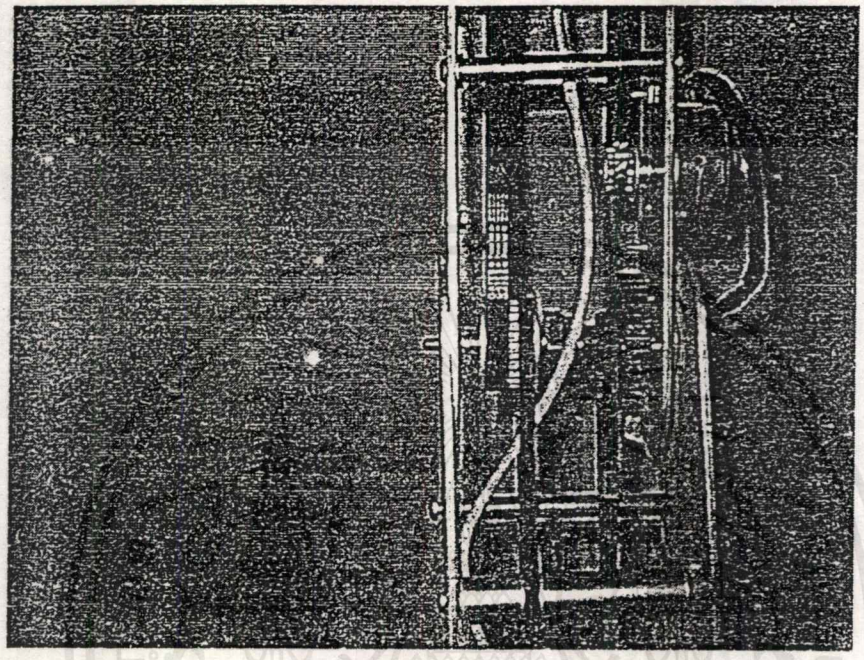
4.2 ส่วนเฟืองที่หัวไหล่ ( shoulder part )



รูปที่ 4.3 ภาพแสดงส่วนเฟืองที่หัวไหล่

**จากรูปที่ 4.3 ประกอบด้วย DC MOTOR 2 ตัวซึ่งประกอบด้วยขนาดต่างๆ ดังนี้ DC MOTOR ขนาด 20 โวลท์ 2.5 แอมป์ 80 rpm ซึ่งมีการทดเฟือง 3 ชุดเพื่อทำหน้าที่ในการขับแขนท่อนบน(แขนส่วนที่ติดอยู่กับส่วนหัวไหล่) ในลักษณะขึ้นลง มอเตอร์กระแสตรงตัวต่อมามีขนาด 12 โวลท์ ทำหน้าที่ในการขับส่วนข้อศอกและประกอบด้วยสเตปปิ้งมอเตอร์ 1 ตัวขนาด 5.4 โวลท์ 1.5 amp/phase 1.8 deg/step ทำหน้าที่ในการขับส่วนข้อมือ ให้สามารถเคลื่อนที่ขึ้นลงได้**

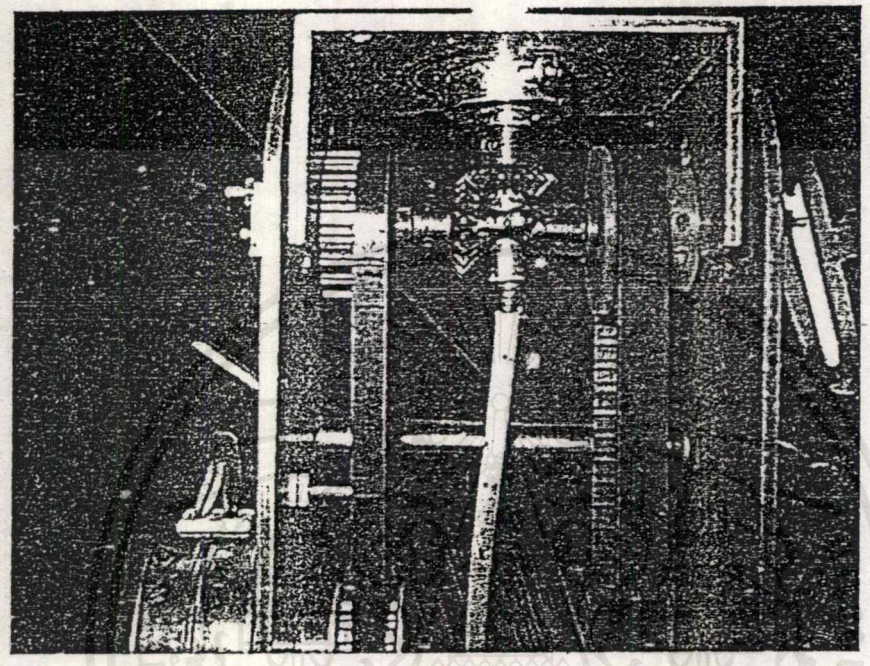
## 4.3 ส่วนข้อศอก ( elbow part )



รูปที่ 4.4 ภาพแสดงส่วนข้อศอก ( elbow part )

จากรูปที่ 4.4 ประกอบด้วยชุดเฟือง 2 ชุด ชุดแรกทำหน้าที่ในการหมุนข้อศอกขึ้นลง ส่วนชุดที่สองเป็นชุดเฟืองที่ทำหน้าที่ส่งผ่านสายพาน โดยชุดเฟืองชุดนี้จะไม่ล็อคติดอยู่กับเพลาบริเวณข้อต่อที่ข้อศอก ซึ่งสายผ่านชุดนี่จะส่งผ่านกำลังไปยังชุดเฟืองบริเวณข้อมือ

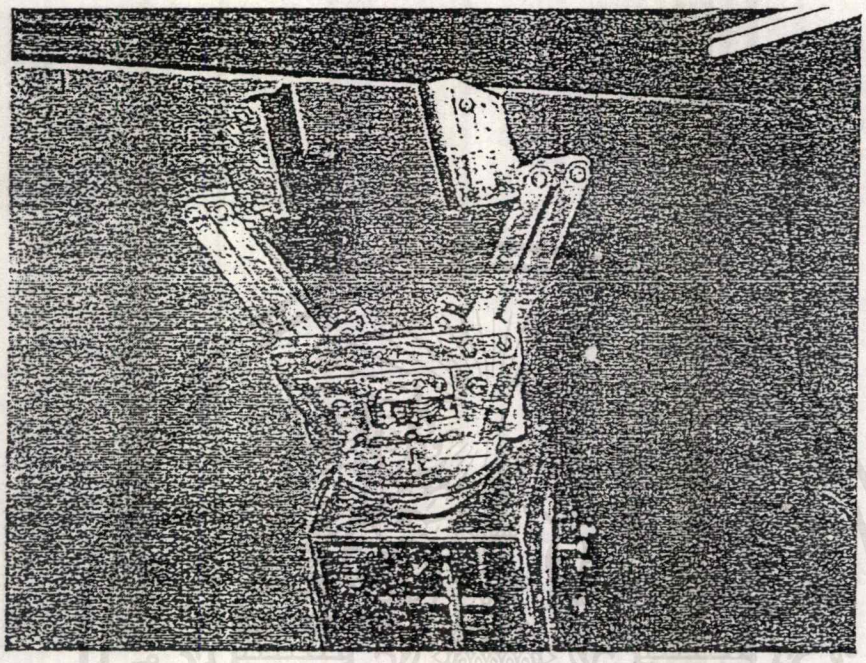
## 4.4 ส่วนข้อมือ ( hand connecter part )



รูปที่ 4.5 ภาพแสดงส่วนข้อมือ

จากรูปที่ 4.5 จะประกอบด้วย สเตปปิ้งมอเตอร์ 1 ตัว และ สายพานเฟือง 2 ชุด ซึ่งสเตปปิ้งมอเตอร์มีขนาด 12 volts 30 ohm/coil จะส่งกำลังผ่านสายพานเฟืองชุดแรกเพื่อทำหน้าที่ในการหมุนส่วนข้อมือไปทางซ้ายและทางขวา และสายพานเฟืองชุดที่ 2 ทำหน้าที่รับกำลังจากสายพานเฟืองส่วนข้อศอกไปหมุนส่วนข้อมือในการขึ้นลง

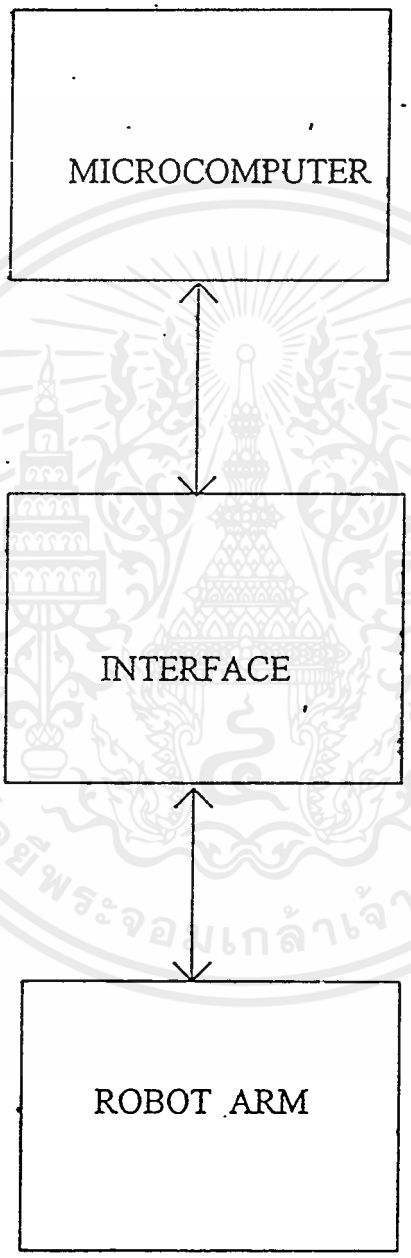
4.5 ส่วน gripper ( gripper part )



รูปที่ 4.6 ภาพแสดงส่วน gripper

**จากรูปที่ 4.6 ประกอบด้วยส่วนฐานใช้ในการหมุนส่วนของมือจับได้ 360 องศา และมีมอเตอร์กระแสตรง 1 ตัวใช้ในการบีบจับวัตถุ โดยจะใช้ IC เบอร์ LMD 18245 ซึ่งมีความสามารถในการจำกัดกระแสได้ในการควบคุมการทำงานของมอเตอร์กระแสตรงตัวนี้**

# บทที่ 5
# โครงสร้างของโครงงาน




รูปที่ 5.1 บล็อกไดอะแกรมของแขนกล

ไมโครคอมพิวเตอร์ (Microcomputer) ใช้สำหรับส่งค่าข้อมูลที่ใช้ในการควบคุมแขนกลไปยังระบบอินเตอร์เฟส (Interface) และรับค่าฟีดแบค (feedback) จากระบบอินเตอร์เฟส กลับเข้ามาและแสดงผลการทำงานของแขนกล

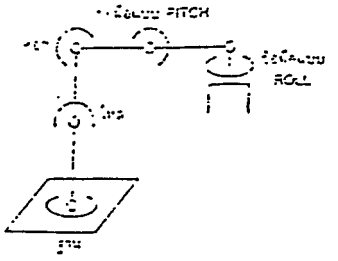
แขนกล ( Robot Arm) เคลื่อนที่โดยใช้สเตปปิ้งมอเตอร์ 1 ตัว และ D Cมอเตอร์ 4 ตัว ใช้ควบคุมการเคลื่อนไหวของจุดหมุนต่างๆ 5 จุดและสเตปปิ้งมอเตอร์อีก 1 ตัวใช้สำหรับการทำงานของมือจับ (gripper)

## 5.1 อัลกอริทึมในการควบคุมแขนกล

กลวิธีและขั้นตอนในการควบคุมแขนกลทำงามตามจินตนาการ

สิ่งที่บ่งบอกถึงประสิทธิภาพในการทำงานของแขนกล คือความแม่นยำและเที่ยงตรงซึ่งการเคลื่อนไหวขณะการทำงานของแขนกลจะประกอบด้วยการเคลื่อนไหวของฐาน,ไหล่,ข้อศอก,ข้อมือเป็นต้น การเคลื่อนไหวตามส่วนต่างๆ นี้จะส่งผลให้ตำแหน่งของมือเคลื่อนที่ไปตามตำแหน่งต่างๆ ที่ปฏิบัติงาน ฉะนั้น ถ้าส่วนต่างๆ นี้มีการควบคุมที่ไม่ดีพอจะส่งผลถึงความแม่นยำและความเที่ยงตรงในการทำงานนั้นลดน้อยลง

ดังนั้นมือของแขนกลจะเคลื่อนที่ไปตามตำแหน่งใดๆ ได้นั้นขึ้นอยู่กับข้อต่อต่างๆ จะเคลื่อนไปในตำแหน่งที่เหมาะสมและสอดคล้องกันด้วย ซึ่งการควบคุมก็ต้องควบคุมตำแหน่งของข้อต่อต่างๆ ให้เหมาะสมกันซึ่งสามารถแสดงได้ด้วยสมการคณิตศาสตร์ สำหรับสมการที่กล่าวถึงในที่นี้มี สองลักษณะด้วยกันคือ สมการ direct kinematic และสมการ inverse kinematic ซึ่งสมการทั้งสองลักษณะใช้แสดงความสัมพันธ์ของข้อต่อต่างๆ และตำแหน่งปลายมือ ก่อนที่จะกล่าวถึงสมการของแขนกลมาทราบถึงศัพท์บางคำที่ใช้กันบ่อย ในแขนกลหรือแขนหุ่นยนต์ คำแรกคือดีกรีออฟฟรีดอม (degree of freedom) หรือ DOF หมายถึงจำนวนข้อต่อหรือจุดต่อที่เคลื่อนที่เป็นอิสระต่อกัน ส่วนอีกคำคือก้าน (link) หมายถึงสิ่งที่เชื่อมต่อระหว่างข้อต่อกับข้อต่อ ตัวอย่างเช่น แขนกลชนิดอาร์ติคูเลตที่แสดงในรูปที่ 5.2 เป็นไดอะแกรมแสดงแขนกลที่มีการเคลื่อนที่แบบ 5 ดพกรีออฟฟรีดอม

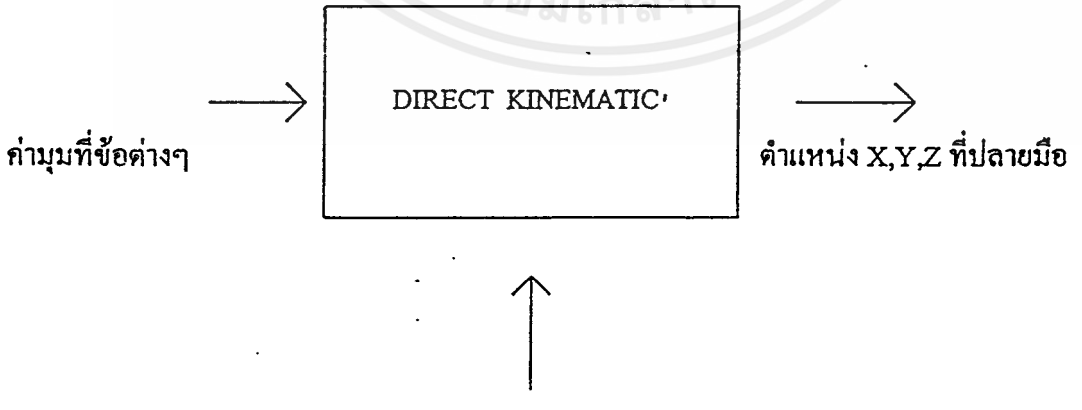


รูปที่5.2 ไดอะแกรมการทำงานของแขนกลในรูปแบบ 5 ดรีกรีออฟฟรีดอม

โดยทั่วไปในการควบคุมแขนกล ให้เคลื่อนที่ไปตามต้องการนั้น จะต้องมีตำแหน่งเริ่มต้นของแขน โดยเป็นตำแหน่งอ้างอิงในการเคลื่อนที่ เรียกว่าตำแหน่ง โฮม (home) ซึ่งก็ขึ้นอยู่กับชนิดของแขนกล และผู้ควบคุมว่าต้องการให้ตำแหน่งนี้อยู่ในลักษณะใด สำหรับบทความในตอนนี้จะอ้างอิงตำแหน่ง โฮม ในลักษณะดังรูปที่ 5.2 เป็นหลักรวมทั้งชนิดของแขนกลที่ใช้ในการอ้างอิงสมการหรือสูตรต่างๆ เป็นชนิดอาร์ติคูเลต แบบ 5 ดีกรีออฟฟรีดอม

## 5.2 สมการ Direct kinematic

ความหมายของสมการ Direct kinematic คือสมการของแขนกลที่แสดงความสัมพันธ์ของการเคลื่อนที่ของข้อต่อต่างๆ โดยผลลัพธ์ของสมการคือ ตำแหน่งปลายมือในพื้นที่ 3 มิติ (ตำแหน่ง X,Y,Z) บล็อกไดอะแกรมแสดงการควบคุมแสดงดังรูปที่ 5.3

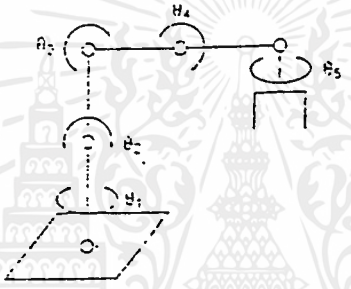



รูปที่ 5.3 บล็อกไดอะแกรมของสมการ direct kinematic

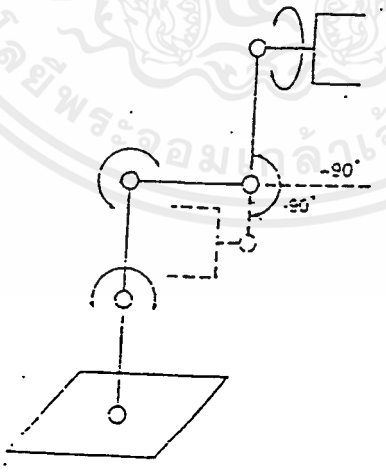
จากบล็อกไดอะแกรมจะพบว่าส่วนที่เป็นอินพุทของสมการ คือค่าของมุมที่ข้อต่อต่างๆ ซึ่งจะขึ้นอยู่กับแขนกลว่ามีกี่ข้อต่อ ส่วนค่าพารามิเตอร์ที่เกี่ยวข้อง จะเป็นค่าคงที่ประจำตัวแขนกล เช่น ความยาวของแต่ละก้านจากข้อต่อหนึ่ง เป็นต้น และเมื่อผ่านสมการนี้ผลลัพธ์ที่ได้ออกมาจะเป็นตำแหน่งของปลายมือที่แขนกลเคลื่อนที่ไปบนพื้นที่ 3 มิติ (X,Y,Z)

5.2.1 การอ้างอิงมุมของแขนกล

สำหรับค่าของมุมของแต่ละข้อต่อที่ป้อนให้กับสมการ Direct kinematic จะถูกอ้างอิงจากตำแหน่งโฮม ดังนั้น ในการป้อนค่ามุมของแต่ละข้อต่อจำเป็นต้องรู้ขนาดและทิศทางของมุมที่จะป้อนด้วยโดยปกติจะกำหนดให้ที่ตำแหน่งโฮม ของทุกข้อต่อมีมุมเป็น 0 องศา ตัวอย่างการอ้างอิงมุมดังรูปที่ 5.4

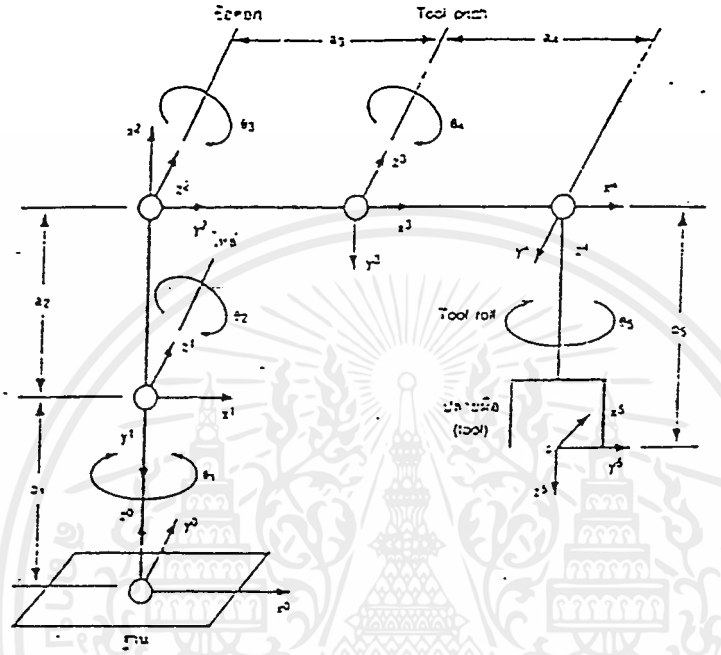


รูปที่ 5.4 ตำแหน่งโฮมของแขนกล




รูปที่ 5.5 แสดงตัวอย่างในการอ้างอิงมุม

ซึ่งเป็นลักษณะตำแหน่งโฮม อีกรูปแบบหนึ่ง ค่าของมุมของทุกข้อต่อในตำแหน่งนี้จะเป็น 0 องศา $\theta_1, \theta_2, \theta_3, \theta_4$ และ $\theta_5$ กำหนดให้เป็น 0องศา สมมติต้องการเคลื่อนเฉพาะไป +90 องศา รูปแขนกลที่ได้จะเป็นดังรูปที่ 5.5



รูปที่ 5.6 ตำแหน่งด้านของแขนกลชนิดอาร์ติคูเลต แบบ 5 ดีกรีออฟฟรีดอม

จากรูปที่ 5.5 ค่าของมุมอื่นๆ ยังคงเป็น 0 องศามีเฉพาะค่า $\theta_4$ เท่านั้นที่เปลี่ยนแปลง ซึ่งถ้าเป็นค่าลบมุมก็จะตรงข้ามกับทิศทางบวกโดยเริ่มนับจากตำแหน่งที่กำหนดให้เป็นตำแหน่งอ้างอิงดังนั้นมุมที่ข้อต่ออื่นๆ ก็จะมีวิธีการอ้างอิงมุมในลักษณะเหมือนกัน

ในรูปที่ 5.6 จะแสดงทิศทางของเวกเตอร์ในแต่ละข้อต่อรวมทั้งแสดงค่าพารามิเตอร์ต่างๆที่เกี่ยวข้องซึ่งสามารถที่จะอธิบายแยกเป็นส่วนๆ ดังนี้

$\theta_1$ คือ มุมในการกวาดแขนไปมารอบฐาน

$\theta_2$ คือ ในการหมุนไหล่ขึ้นลง

$\theta_3$ คือ มุมในการหมุนข้อศอก

$\theta_4$ คือ มุมในการหมุนข้อมือแบบหักขึ้นลง (toll pitch)

$\theta_5$ คือ มุมในการหมุนข้อมือในลักษณะหน้ามือไปหลังมือ (toll roll)

การหมุนข้อมือ มีการหมุนอยู่ใน 3 ลักษณะ คือ YAW ,PITCH และ ROLL สำหรับแขนกลที่ใช้อ้างอิงนี้เป็นแบบ 5 ดีกรีออฟฟรีดอม โดยการตัดการเคลื่อนที่ของข้อมือแบบ YAW ไปดังนั้น ถ้าแขนกลชนิดอาร์ติคูเลต แบบ 6 ดีกรีออฟฟรีดอม ก็จะมีการเคลื่อนที่แบบ YAW นี้ด้วย ซึ่งจะทำให้แขนกลมีลักษณะการเคลื่อนที่ของข้อต่อทุกข้อคล้ายกับแขนของมนุษย์มากที่สุด

สำหรับความจำเป็นของการมีจำนวน ดีกรีออฟฟรีดอม เนื่องจากดีกรีออฟฟรีดอมเป็นการเคลื่อนที่อิสระของแต่ละข้อต่อถ้ามีจำนวนดีกรีออฟฟรีดอมมากแขนกลก็สามารถเคลื่อนไหวได้ ซับซ้อนขึ้น แต่การสร้างและการควบคุมก็จะซับซ้อนขึ้นเป็นเงาตามตัวด้วย

สำหรับเวกเตอร์ $X_0,Y_0,Z_0$ จนถึง $X_5,Y_5,Z_5$ เป็นเวกเตอร์ที่แสดงแนวการเคลื่อนที่ ส่วนค่าพารามิเตอร์ที่เป็นค่าคงที่ของแขนกลมีดังนี้

$d_1$ คือระยะระหว่างฐานไปยังจุดหมุนที่ไหล่

$a_2$ คือระยะห่างจากจุดหมุนที่ไล่ไปยังจุดหมุนที่ศอก

$a_3$ คือระยะห่างจากจุดหมุนที่ศอกไปยังจุดหมุนที่ข้อมือหมุนแบบ PITCH

$a_4$ คือระยะห่างจากจุดหมุนที่ข้อมือแบบPITCH ไปยังจุดหมุนข้อมือหมุนแบบ ROLL

$d_5$ คือระยะห่างจากจุดหมุนที่ข้อมือหมุนแบบ ROLL ไปยังจุดปลายมือ

จากนั้นทำการป้อนค่ามุม $\theta_1,\theta_2,\theta_3,\theta_4,\theta_5$ และค่าพารามิเตอร์ $d_1,a_2,a_3,a_4,d_5$ ให้สมการ direct kinematic จะได้ตำแหน่ง P(X,Y,Z) ในพื้นที่ 3 มิติ ที่มือไปวางอยู่เนื่องจากที่มาสมการได้มาจากการวิเคราะห์ทางด้านเวกเตอร์ ฉะนั้นถ้าผู้ใดสนใจ สามารถค้นคว้าเพิ่มเติมได้จากหนังสืออ้างอิงโดยสมการมีดังนี้

$$X = C_1(a_2C_2 + a_3C_2 + a_4C_{234} - d_5S_{234})$$

$$Y = S_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234})$$

$$Z = d_1 - a_2S_2 + a_3S_{23} - a_4S_{234} - d_5C_{234}$$

จากสมการมีการใช้สัญญลักษณ์เพื่อย่อความหมายให้สมการดูง่ายขึ้นความหมายของตัวย่อเหล่านี้คือ

$S_k$ คือ sink

$C_k$ คือ cosk

$S_{ijk}$ คือ sin (I+j+k)

$C_{ijk}$ คือ cos(I+j+k) คือมุมค่าใดๆ

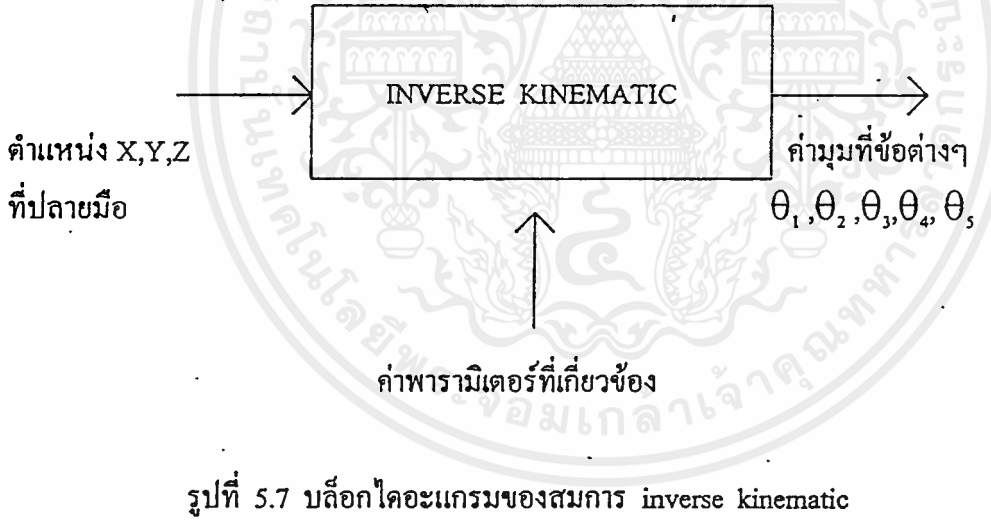
ตัวอย่างเช่น $C_1 = \cos\theta_1$ หรือ $S_{234} = (\theta_2 + \theta_3 + \theta_4)$ เป็นต้น

สำหรับแขนกลชนิดกลชนิดอาร์ติคูเลตอื่นๆ ที่ไม่ใช้แขนกลชนิดแบบ 5 ดีกรีออฟฟรีดอมก็มีในสมการเช่นกัน

## 5.3 สมการ Inverse kinematic

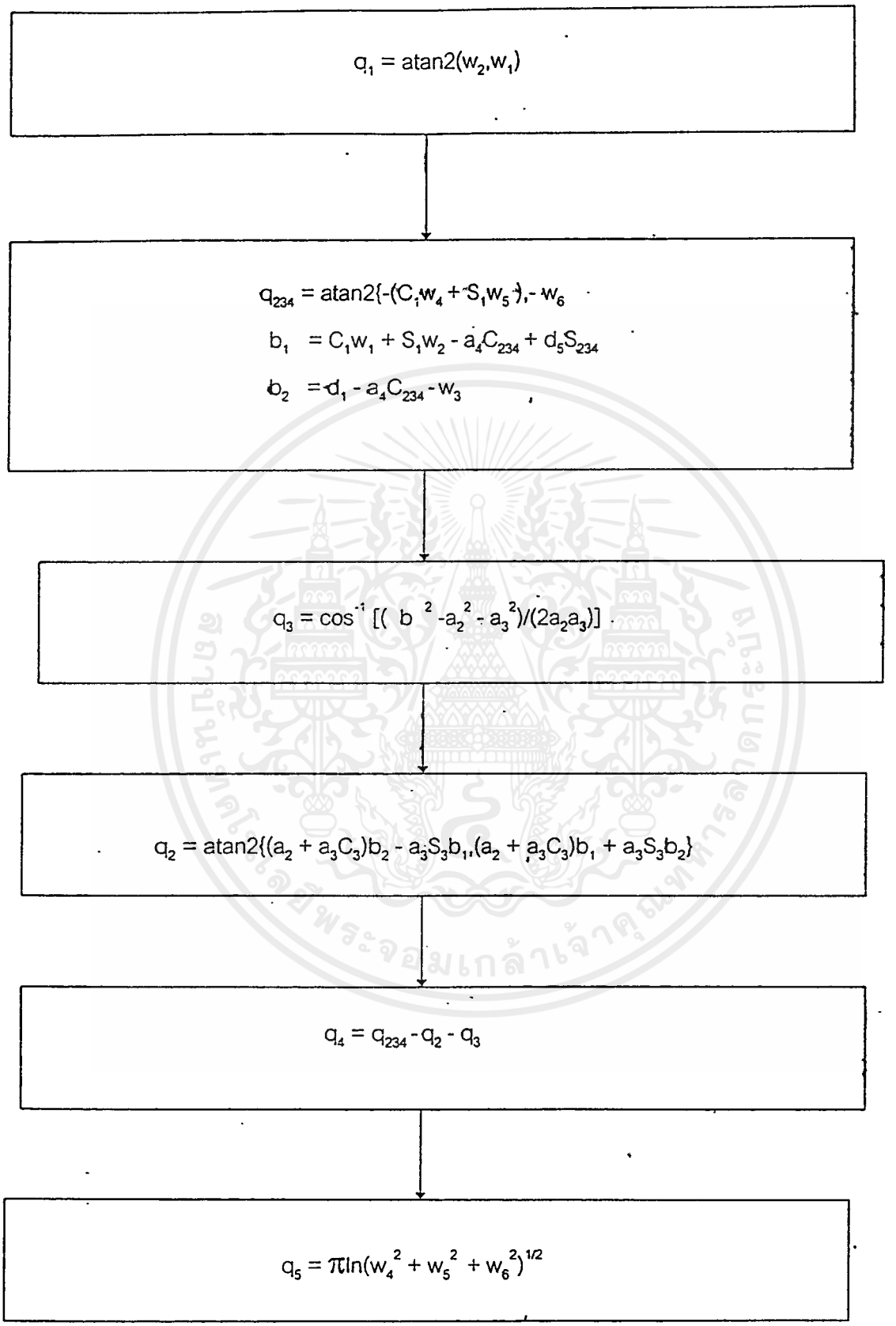
จากที่ผ่านมาพอทราบความหมายของสมการกันไปแล้วในส่วนต่อมาจะได้กล่าวถึงสมการในอีกลักษณะหนึ่ง ซึ่งอาจกล่าวได้ว่าเป็นหัวใจในการควบคุมแขนกลอีกแบบหนึ่งสมการชุดนี้คือสมการ inverse kinematic

ความหมายของสมการจะมีความหมายที่ตรงข้ามกับสมการฉือจากเดิมป้อนค่าของมุมที่ข้อต่อต่างๆ เข้าไปในสมการผลลัพธ์ที่ได้จะเป็นตำแหน่ง X,Y,Z ที่ปลายมือเคลื่อนไป แต่สมการนี้จะเปลี่ยนเป็นการป้อนตำแหน่ง X,Y,Z ของปลายมือเข้าไปแล้วจะได้ผลลัพธ์เป็นมุมของข้อต่อว่าต้องเคลื่อนไปกี่องศาจากตำแหน่งโฮม ปลายมือจึงจะไปอยู่ตำแหน่งที่ต้องการ บล็อกไดอะแกรมแสดงการทำงานจะแสดงดังในรูปที่ 5.7




รูปที่ 5.7 บล็อกไดอะแกรมของสมการ inverse kinematic

จากรูปที่ 5.7 โดยค่าพารามิเตอร์ต่างๆ ยังคงใช้ค่าเหมือนเดิมทุกประการจากความหมายของสมการ inverse kinematic เมื่อป้อนตำแหน่ง X,Y,Z ที่ปลายมือและค่าพารามิเตอร์ที่เกี่ยวข้องผลลัพธ์จะได้เป็นมุมของข้อต่อต่างๆ คือ $\theta_1$ ถึง $\theta_5$ โดยสามารถเขียนเป็นอัลกอริทึมของสมการได้ดังรูปที่ 5.8




รูปที่ 5.8 แสดงอัลกอริทึมของสมการ inverse kinematic

จากรูปที่ 5.8 แสดงลำดับการหาค่ามุมตามข้อต่อต่างๆ ซึ่งสามารถอธิบายความหมายของค่าตัวแปรต่างๆ ดังนี้

$q_1$ ซึ่ง $\theta_1$ เป็นมุมในการหมุนฐาน

atan2 คืออาร์คแทน 4 ควอแดนท์

$q_2$ คือ $\theta_2$ เป็นมุมที่ไหล่

$q_3$ คือ $\theta_3$ เป็นมุมที่ศอก

$q_4$ คือ $\theta_4$ เป็นมุมการหมุนข้อมือคน แบบ PITCH

$q_5$ คือ $\theta_5$ เป็นมุมการหมุนข้อมือแบบ PITCH โดยอ้างอิงจากระนาบพื้น $(X_0,Y_0,Z_0)$

$w_1$ คือตำแหน่งปลายมือที่ X

$w_2$ คือตำแหน่งปลายมือที่ Y

$w_3$ คือตำแหน่งปลายมือที่ Z

$w_4,w_5,w_6$ คือค่าในการหมุนข้อมือ

$C_1$ คือ $\cos\theta_1$

$S_1$ คือ $\sin\theta_1$

$C_3$ คือ $\cos\theta_3$

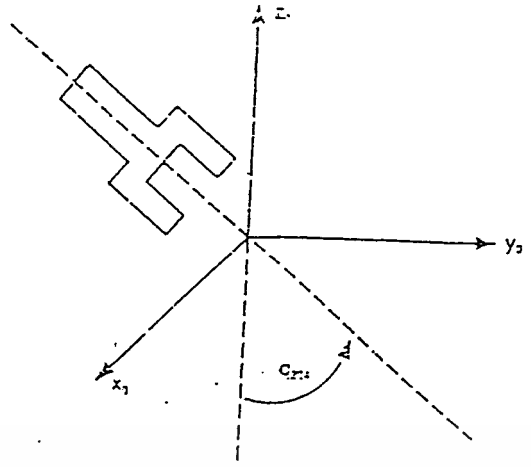
$S_3$ คือ $\sin\theta_3$

$C_{23}$ คือ cos ของมุม $q_{234}$

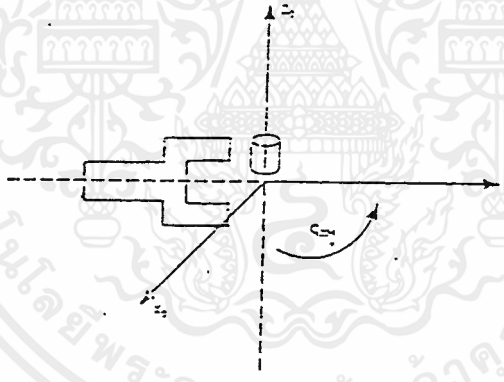
$S_{234}$ คือ sin ของมุม $q_{234}$

$\| b\| = b_1^2 + b_2^2$

สำหรับ $q_{234}$ นั้นสามารถอธิบายรายละเอียดโดยรูปที่ 5.9 ประกอบจากความหมายของ $q_{234}$ เป็นมุมการหมุนข้อมือแบบ PITCH โดยอ้างอิงจากระนาบพื้น $X_0,Y_0,Z_0$ ในการวัดมุมจะวัดเปรียบเทียบกับแกน $Z_0$



รูปที่ 5.9 แสดงการอ้างอิง $q_{234}$



รูปที่ 5.10 ตัวอย่างการอ้างอิง $q_{234}$

สังเกตได้ว่า $q_{234}$ เป็นมุมที่มือกระทำกับระนาบพื้น ในที่นี้คือ รูปแบบในการวางมือในการเคลื่อนที่ไปยังตำแหน่งที่ต้องการ ซึ่งที่มาของ $q_{234}$ มาจากเวกเตอร์ $w_4$,$w_5$,$w_6$ ตัวอย่างเช่น ต้องการเคลื่อนมือไปจับวัตถุทางด้านข้างแสดงดังรูปที่ 5.10 มุมซึ่งเป็นมุมในการวางมือเมื่อเทียบกับระนาบพื้นของวัตถุจะมีค่าเป็น -90 องศา

โดยทั่วไปค่าอินพุทที่ป้อนให้กับอัลกอริทึมของสมการ inverse kinematic จะเป็นค่า $w_1$ ,$w_2$, $q_{234}$ และ $q_5$ ( ตำแหน่งของวัตถุ ,มุมในการวางมือ ,มุมในการไมุนข้อมือแบบ ROLL ) ตามลำดับ ดังนั้นในการทำงานจริงๆ จะพบว่าอัลกอริทึมของสมการ inverse kinematic ในรูปที่ 5.8 จะมีบางส่วนที่สามารถตัดออกไปได้เพื่อลดความซับซ้อนของสมการ เนื่องจากในการควบคุมแขนกลโดยใช้อัลกอริทึมของสมการ inverse kinematic ค่า $q_{234}$ ซึ่งเป็นมุมในการวางมือเมื่อเทียบกับระนาบของวัตถุและมุมในการหมุนข้อมือแบบ ROLL ผู้ควบคุมจะเป็นผู้กำหนดลงไปเองโดยไม่ต้องผ่านอัลกอริทึมของสมการ inverse kinematic ซึ่งสมการส่วนที่สามารถตัดออกไปได้คือ

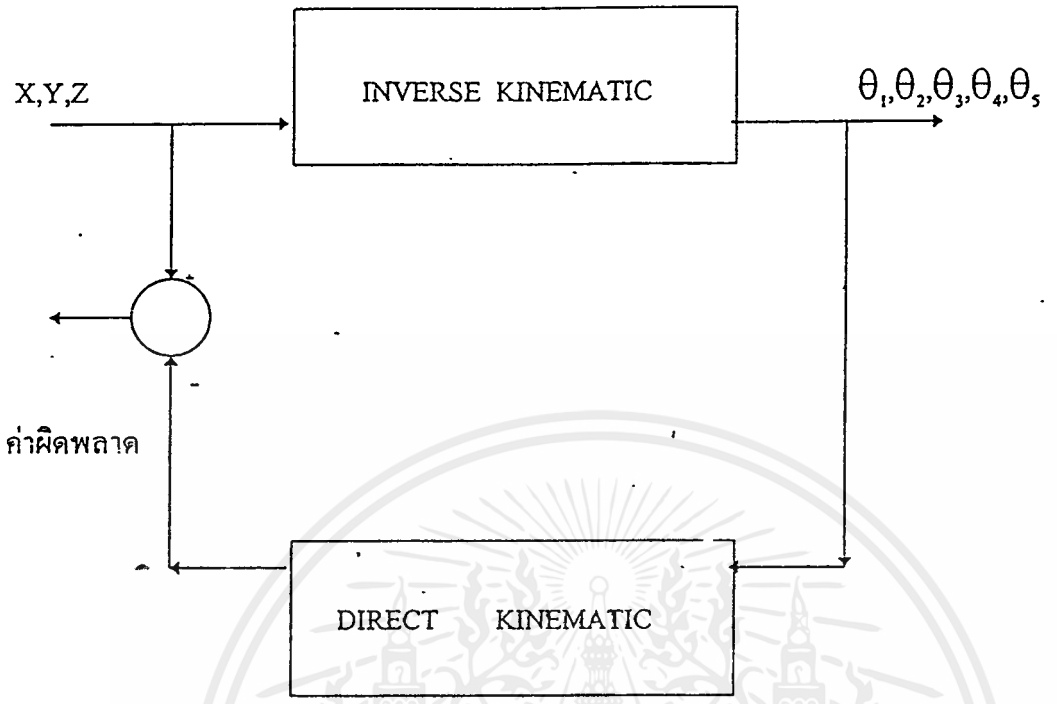
$$q_{234} = \text{atan2}(-(C_1 w_4 + S_1 w_5), -w_6 \text{ และ } q_5 = \pi \ln(w_4^2 + w_5^2 + w_6^2)^{1/2}$$

เอาท์พุทที่ได้จะเหลือเพียงมุม $\theta_1$, $\theta_2$ , $\theta_3$ และ $\theta_4$ นอกจากอัลกอริทึมของสมการ inverse kinematic ของแขนกลชนิดอาร์ติคูเลต แบบ 5 ดีกรีออฟฟรีดอม นี้แล้วยังมีของแขนกลชนิดอื่นๆ อีก รวมทั้งที่มาของสมการนี้ด้วย สามารถค้นคว้าเพิ่มเติมได้จากหนังสืออ้างอิงได้เช่นกัน

## 5.4 การประยุกต์ใช้งาน

สำหรับการประยุกต์ใช้งานอัลกอริทึมของสมการ inverse kinematic อันดับแรกที่เห็นได้ชัดคือ ควบคุมการจับวัตถุที่ตำแหน่ง X,Y,Z ต้นทางให้ไปวางที่ตำแหน่ง X,Y,Z ปลายทาง โดยคอมพิวเตอร์จะเป็นตัวคำนวณค่าของมุมที่ข้อต่อต่างๆ เมื่อคำนวณเสร็จ คอมพิวเตอร์จะส่งค่ามุมต่างๆ ไปให้กับชุดขับเคลื่อน ชุดขับเคลื่อนก็จะทำการขับเคลื่อนข้อต่างๆ ไปตามมุมที่ได้มาจากอัลกอริทึมของสมการ inverse kinematic ผลลัพธ์ที่ปลายมือจะอยู่ที่ตำแหน่งที่ต้องการ จากนั้นก็เป็นหน้าที่ของผู้เขียนโปรแกรมควบคุมว่าจะให้ทำอย่างไรต่อไป

สำหรับสมการ direct kinematic ที่ใช้งานอยู่บางส่วนใช้ในการตรวจสอบตำแหน่งในการเขียนโปรแกรมจำลองหรือซิมูเลต การทำงานของแขนกล โดยจะใช้สมการ direct kinematic เป็นชุดป้อนกลับแสดงดังรูปที่ 5.11




รูปที่ 5.11 บล็อกไดอะแกรมในการตรวจสอบตำแหน่งด้วยสมการ direct kinematic

จากบล็อกไดอะแกรมค่าเอาท์พุทของอัลกอริทึมของสมการ inverse kinematic จะถูกสมการ direct kinematic นำมาเป็นอินพุทเพื่อป้อนกลับไปหาค่าผิดพลาดในการเคลื่อนที่ ค่าผิดพลาดที่ได้จะถูกส่งไปยังโปรแกรมอีกชุดหนึ่ง ซึ่งทำหน้าที่เป็นตัวควบคุม ถ้าเปรียบเทียบแขนกลจริงๆ กับโปรแกรมจำลองการทำงานนั้น สมการ direct kinematic ก็คือตัวตรวจจับนั้นเอง

# บทที่ 6
# การควบคุมและโปรแกรมการควบคุม

ในการทำงานของ ROBOT ARM ในเทอมนี้จะเป็นการทำงานโดยทำการส่งข้อมูลที่ใช้ในการควบคุม ROBOT ARM โดยจะเป็นการส่งแบบอนุกรม จากส่วน PC ไปยังส่วนประมวลผล(8031) และ ทำการ OUT ค่าที่ได้จากการประมวลผลออกทาง OUT PUT ของ 8255 เพื่อไปควบคุมให้แขนกลทำงานตามคำสั่ง โดยจะใช้ IC เบอร์ 18245 และ 18200 ในการควบคุมการทำงานของแขนกลในส่วนที่เป็น DC มอเตอร์ ส่วนสเตปปิ้งมอเตอร์ก็ทำงานตามปกติ

ในการควบคุมการทำงานสามารถแบ่งได้เป็น 3 ส่วน

1 ส่วน SERIAL PORT หรือ การส่งข้อมูลแบบอนุกรม

2. ส่วนประมวลผล(8031) และ OUT PUT ที่ใช้ในการควบคุม

3. ส่วนที่เป็นวงจร drive motor ใช้ IC เบอร์ 18245 และ 18200

4. ส่วนโปรแกรมการควบคุม

ในที่นี้จะไม่กล่าวถึงการทำงานของสเตปปิ้งมอเตอร์เพราะได้กล่าวไว้แล้วในบทที่ 3

## 6.1 ส่วน SERIAL PORT

ความต้องการพื้นฐานในการส่งข้อมูลก็คือ เราต้องการสื่อสารข้อมูลที่มีอยู่หรือใช้ข้อมูลที่มีอยู่ร่วมกัน เพราะฉะนั้นก่อนที่เราจะรู้การทำงานจริงในส่วน SERIAL PORT เราก็ควรจะรู้ถึงทฤษฎีพื้นฐานบางอย่างเพื่อความเข้าใจที่ถูกต้องร่วมกันก่อน ซึ่งจะขอกล่าวตามลำดับดังต่อไปนี้

### 6.1.1 การสื่อสารข้อมูล

การสื่อสาร (communication) คือการกระทำ หรือ กระบวนการในการแลกเปลี่ยนข้อมูล (exchanging information) หรือการใช้ข้อมูลร่วมกัน (sharing information) ในการสื่อสารนั้นต้องประกอบด้วย 3 สิ่งด้วยกันคือ ตัวส่ง (sender), ตัวรับ (reciever), และตัวกลาง (medium)

การสื่อสารข้อมูล คือ กระบวนการแลกเปลี่ยน หรือ การใช้ร่วมกัน ของข้อมูลที่ผ่านการเข้ารหัส (encode information) ระหว่างอุปกรณ์ 2 เครื่องหรือมากกว่านั้น สำหรับการสื่อสารข้อมูลนั้นตัวส่งและตัวรับจะเป็นเครื่องจักรเท่านั้น

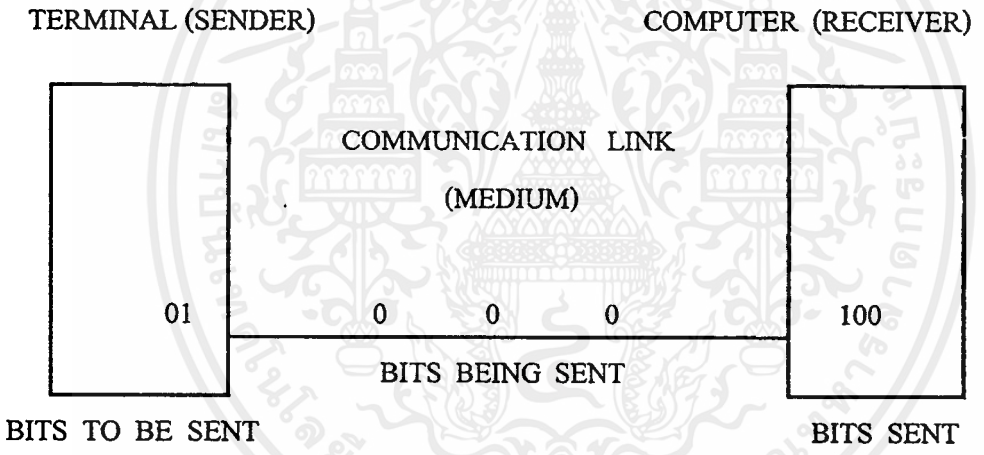
ข้อมูลที่ผ่านการเข้ารหัส หมายถึง การที่ข้อมูลถูกส่งเป็นลำดับของสัญญาณไฟฟ้า เช่น เมื่อคอมพิวเตอร์ปฏิบัติการรับการกดแป้นอักษร D จากแป้นพิมพ์ ลำดับของสัญญาณไฟฟ้าก็จะถูกส่งมาที่คอมพิวเตอร์ ถ้าสัญญาณไฟฟ้านั้น แสดงในรูป ลอจิก 1 และ 0 อักษร D ก็จะสามารถแทนได้ด้วย "01000100" รหัสเลขฐานสองนี้จะถูกส่งจากแป้นพิมพ์ (ตัวส่ง) ผ่านสายสัญญาณ (ตัวกลาง) มายังคอมพิวเตอร์ (ตัวรับ) คอมพิวเตอร์จะทำการแปล รหัสเลขฐานสองให้เป็นอักษร D ซึ่งสัญญาณ

ไฟฟ้า หรือ รหัสเลขฐานสองแบบนี้ เราจะเรียกว่า การเข้ารหัสข้อมูล และสามารถส่งได้ทั้งแบบอนาลอก และ แบบดิจิตอล แต่ไม่ว่าจะเป็นแบบใด ตัวส่งและตัวรับ ล้วนเป็นอุปกรณ์อิเล็กทรอนิกส์ทั้งสิ้น

ระวัง การสับสนระหว่าง การสื่อสารข้อมูล กับ การส่งข้อมูลแบบ ดิจิตอล เพราะการสื่อสารข้อมูลจะหมายถึง การส่งข้อมูลที่ผ่านการเข้ารหัส ระหว่างอุปกรณ์อิเล็กทรอนิกส์ ในรูปของ อนาลอก หรือ ดิจิตอลก็ได้ แต่การส่งข้อมูลแบบดิจิตอลนั้น ข้อมูลไม่ว่าจะเป็น ภาพ หรือ เสียง จะต้องถูกแปลงเป็นสัญญาณดิจิตอลก่อนทำการส่ง แต่ส่วนใหญ่ในปัจจุบันนี้จะนิยมการส่งข้อมูลเป็นแบบดิจิตอลมากขึ้น

**6.1.2 การส่งข้อมูลแบบอนุกรม**

การส่งข้อมูลแบบอนุกรมจะใช้สายเพียง 2-3 เส้น แล ที่เวลาหนึ่งๆ จะมีการส่งข้อมูลเพียง 1 บิตเท่านั้น ดังแสดงในรูปที่ 6.1




รูปที่ 6.1 การส่งข้อมูลแบบอนุกรม

เมื่อเราพิจารณา เปรียบเทียบกับการส่งข้อมูลแบบขนาน แล้วก็จะเห็นได้ว่า ถึงแม้การส่งแบบขนานนั้นจะรวดเร็วกว่าเพราะสามารถที่จะส่งข้อมูลได้พร้อมกัน หลายๆ บิต แต่การที่ต้องใช้สายจำนวนมากนั้นจะทำให้เกิดความยุ่งยาก รุงรัง ราคาแพง และอ่อนไหวต่อสัญญาณรบกวน ดังนั้นส่วนใหญ่แล้วจึงนิยมใช้การส่งแบบอนุกรม ทั้งนี้เนื่องจาก IC ที่ใช้ในการแปลงข้อมูลจากแบบขนานให้เป็นอนุกรม เพื่อส่งออก และรับข้อมูลแบบอนุกรม มาแปลงให้เป็นขนานเพื่อประมวลผลนั้นสามารถหาได้ง่ายมาก ดังนั้น การสื่อสารข้อมูลที่ใช้ระหว่างเครื่องคอมพิวเตอร์ด้วยกัน หรือ กับอุปกรณ์ภายนอก จึงนิยมใช้แบบอนุกรมมากกว่า

**6.1.3 อสมวาร กับ สมวาร ( Asynchronous Versus Synchronous )**

วิธีการรับ-ส่งข้อมูลแบบอนุกรมนั้น จะมี 2 วิธีด้วยกัน คือ การส่งข้อมูลแบบอสมวาร และแบบสมวาร เครื่องส่งและเครื่องรับนั้น จะต้องใช้วิธีการรับ-ส่งข้อมูลแบบเดียวกัน สำหรับแบบอสมวาร เครื่องรับจะสามารถตรวจสอบ ตัวเริ่มและสิ้นสุดได้ และในแบบสมวาร จะสามารถตรวจสอบกลุ่มของตัวเริ่มและสิ้นสุดได้

**6.1.3.1 การส่งข้อมูลแบบอสมวาร**

การส่งแบบอสมการ จะใช้ได้กับการส่งข้อมุล ด้วยอัตราที่ไม่เร็วนัก (น้อยกว่า 19,200 บิทต่อ วินาที {bps})และใช้กับอุปกรณ์ที่ราคาไม่แพง วิธีนี้เป็นที่นิยมเพราะวงจรที่ใช้เป็นแบบพื้นฐาน ไม่ซับซ้อน ทำให้ราคาถูก แต่ก็มีโอกาสเกิดความผิดพลาดของข้อมูล จากการคลาดเคลื่อนของเวลาได้ เนื่องจากเครื่องส่งและเครื่องรับ ไม่เข้าจังหวะกัน ดังนั้น จึงมักออกแบบให้ ตัวรับทำการเข้าจังหวะทุกๆ ครั้งที่มีตัวอักษรเข้ามา โดยใช้บิทเริ่มต้น (start bit ) เป็นตัวบอกหรือจะทำการแบ่งส่วนออกเป็น 4 ส่วน ในแต่ละชุดข้อมูลประกอบด้วย บิทเริ่มต้น (start bit ) ,บิทข้อมูล ( data bit ) , บิทแสดงภาวะเสมอมูล ( parity bit ) , และ บิทสิ้นสุด (stop bit ) ถึงแม้ว่า บิทแสดงภาวะเสมอมูล จะเป็นส่วนเสริมขึ้นมา แต่ก็เป็นที่นิยมใช้ในระบบทั่วไป

แม้ว่าการส่งข้อมูลแบบอสมการ จะง่ายในการออกแบบ การสร้างและการใช้งาน แต่ก็ให้ประสิทธิภาพในการส่งที่ต่ำ เพราะว่าอย่างน้อยที่สุด จะต้องส่งบิทเริ่มต้น และบิทสุดท้าย รวมไปกับข้อมูลด้วยเสมอ

**6.1.3.2 การส่งข้อมูลแบบสมวาร**

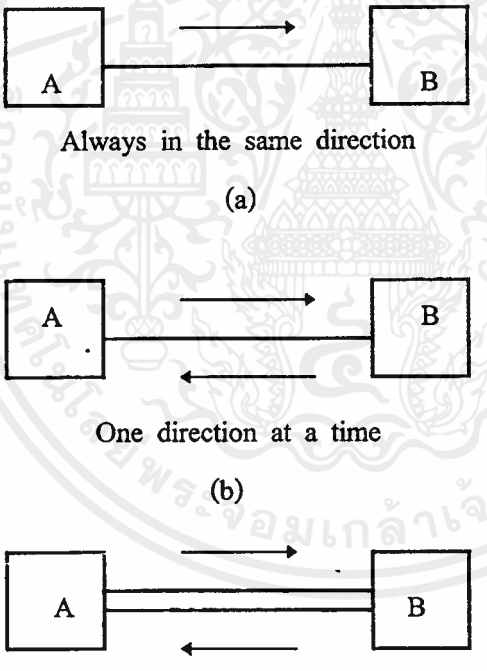
เป็นการส่งที่มีประสิทธิภาพสูงกว่า การส่งข้อมูลแบบสมวารนั้น ไม่จำเป็นต้องมีบิทเริ่มต้นและบิทสิ้นสุดในทุกๆ ชุดของข้อมูลแต่ละตัว แต่จะมีการส่งข้อมูลชุดใหญ่ไปเพียงครั้งเดียว

จากที่ได้กล่าวไปแล้วว่า การส่งแบบอสมวาร จะใช้บิทเริ่มต้นเป็นตัวบอกว่า มีข้อมูลเข้ามาเพื่อให้ภาครับเข้าจังหวะได้ตรงกัน แต่สำหรับการส่งแบบสมวารนั้น เครื่องรับจะสามารถระบุข้อมูลใหม่ได้ เหมือนกับระบุข้อมูลแต่ละบิทข้างในกรอบนั้น ซึ่งมีด้วยกัน 2 แบบ คือ แบบกรอบที่ใช้ตัวอักษรในการกำหนด( Character oriented frame ) และกรอบที่ใช้บิทในการกำหนด ( Bit oriented frame ) ซึ่งแบบที่ใช้ตัวอักษรนั้นจะเริ่มด้วย ตัวอักษรพิเศษ สำหรับเข้าจังหวะ (special synchronization[SYN] character ) หนึ่งตัวหรือมากกว่า ซึ่งตัวอักษรเข้าจังหวะนี้จะประกอบด้วยเลขฐานสอง ที่เหมือนกัน และเปลี่ยนแปลงไปตาม ข้อความสำหรับควบคุม (control information) ข้อมูล อักษรควบคุม (control information) และตัวอักษรที่จะเสริมเข้าไปด้วย อาจจะเป็นจำนวนข้อมูลที่ส่งในตอนต้นของข้อมูลทั้งหมด เพื่อให้ภาครับรับตามจำนวนนี้ก็ได้

ส่วนแบบที่ใช้บิทในการกำหนดนั้น จะมีรูปแบบพิเศษของบิทในตอนเริ่มต้นและสิ้นสุดของแต่ละกรอบข้อมูลประกอบด้วยข้อมูล 8 บิท ที่เราเรียกว่า แฟลก (flag) ถ้าดูคร่าวๆ จะเห็นได้ว่าการส่งทั้ง 2 แบบ นี้ คล้ายกัน คือ ต่างก็มี ข้อความพิเศษสำหรับแสดงกรอบข้อมูล มิฉะนั้นอาจเกิดการเลื่อน (drifting) ระหว่างเครื่องส่งกับเครื่องรับ เนื่องจากสัญญาณนาฬิกาของ 2 เครื่อง ไม่ตรงกันได้ ทางแก้หนึ่งก็คือ เพิ่มเส้นสัญญาณนาฬิกาในการส่งข้อมูลด้วย เพื่อให้มั่นใจได้ว่า สัญญาณนาฬิกาของทั้ง 2 ตรงกัน แต่ในทางปฏิบัติแล้ว คงเป็นไปไม่ได้ ถ้าเราต้องการส่งข้อมูลผ่านสายโทรศัพท์ ดังนั้น ระบบส่วนใหญ่จึงทำการเข้ารหัสสัญญาณนาฬิกา เพื่อเป็นส่วนสำหรับการเข้าจังหวะ ก็จะทำให้เครื่องส่งและเครื่องรับ ทำงานสัมพันธ์กันได้อย่างถูกต้อง

### 6.1.4 รูปแบบการสื่อสาร

โดยทั่วไปแล้ว จะมี 3 แบบด้วยกันคือ แบบส่งทางเดียว (simplx) , แบบเลือกส่งหรือรับอย่างใดอย่างหนึ่ง (half-duplex) และแบบส่ง-รับได้พร้อมกัน (full-duplex)




รูปที่ 6.2 แสดงรูปแบบการสื่อสาร (a) Simplex (b) Half-duplex © Full-duplex

จากรูป 6.2 (a) ข้อมูลจะถูกส่งจาก จุด A ไปยังจุด B โดยที่ไม่มีเส้นทางย้อนกลับ ดังนั้นจุด B จะไม่มีทางส่งสีญญาณมายังจุด A ได้เลย

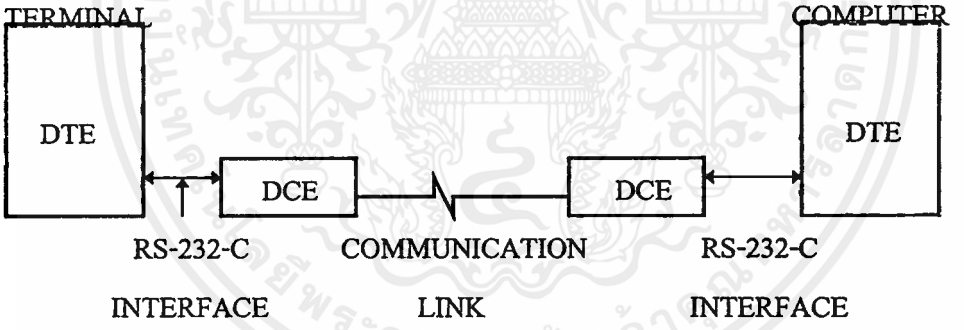
รูป 6.2 (b) เป็นแบบเลือกส่งอย่างใดอย่างหนึ่ง จะยอมให้ส่งข้อมูลจากจุด A ไป B หรือ จากจุด B ไป A ก็ได้ แต่ว่าต้องไม่ใช่ในขณะเดียวกัน ในการใช้งานนั้นช่องสัญญาณจะต้องมีการวนกลับ (turned around) ซึ่งเป็นการไม่สะดวกมากในหลายๆ งานเพราะอุปกรณ์ที่จุด A อาจจะหยุดส่งก่อนที่ อุปกรณ์ที่จุด B พร้อมจะส่งก็ได้

รูปที่ 6.2© ซึ่งเป็นแบบสองทางนั้น จะสามารถส่งข้อมูลจาก A ไป B หรือจาก B ไป A ในขณะเดียวกันได้

### 6.1.5 มาตราฐาน RS-232

ในการส่งข้อมูลแบบอนุกรมในครั้งนี้ได้ใช้มาตราฐาน RS-232 ตัวอักษร RS แทน "Recommended Standard" 232 แทน หมายเลขของมาตรฐาน

การเชื่อมต่อตามมาตราฐาน RS-232 นั้นจะหมายถึง การเชื่อมต่อระหว่าง อุปกรณ์ข้อมูลปลายทาง (Data terminal Equipment [DTE] ) กับอุปกรณ์สื่อสารข้อมูล (Data Communication Equipment [DCE] (แต่ในปัจจุบันตัวย่อ DCE จะแทน data circuit termination equipment)) โดยใช้การแลกเปลี่ยนข้อมูลแบบอนุกรม ซึ่งโครงสร้างคร่าวๆแสดงดังรูป 6.3




รูปที่ 6.3 การเชื่อมต่อระหว่าง DTE กับ DCE

คำจำกัดความของ DCE และ DTE พอสังเขป

DCE : เป็นอุปกรณ์ที่มีฟังก์ชั่นการทำงานต่างๆ ที่ทำให้เกิดการเชื่อมต่อ ทำให้การเชื่อมต่อยังดำรงต่อไป และยุติการเชื่อมต่อ นอกจากนี้ยังใช้เปลี่ยนลักษณะของสัญญาณและสร้างรหัสสัญญาณต่างๆ ที่จำเป็นต้องใช้ในการสื่อสารข้อมูลระหว่าง DTE และ Data circuit โดย DCE อาจเป็นส่วนใดส่วนหนึ่งของคอมพิวเตอร์หรือไม่ก็ได้

DTE : 1. เป็นอุปกรณ์ที่ประกอบไปด้วยตัวส่งข้อมูล (data source) หรือเป็นทั้งตัวส่งและตัวรับข้อมูลก็ได้

2. เป็นอุปกรณ์ที่ประกอบด้วยส่วนทำงานต่อไปนี้ ส่วนควบคุม (control logic) , ส่วนคงค่า (buffer store) และอุปกรณ์อินพุทหรือเอาท์พุทจำนวนหนึ่งตัวหรือมากกว่าก็ได้ หรือรวมเครื่องคอมพิวเตอร์เข้าไปด้วยก็ได้ DTE อาจจะรวมส่วนควบคุมความผิดพลาด (error control ) , ส่วนเข้าจังหวะ (synchronization) และความสามารถในการบ่งหรือระบุว่าต้องการเกี่ยวข้องกับอุปกรณ์ตัวใด(STATION IDENTIFICATION CAPACITY) เข้าไปด้วยก็ได้

**6.1.6 การใช้งาน**

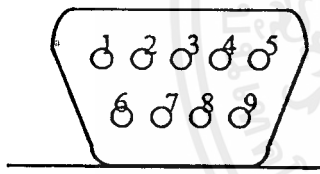
ในแต่ละวงจรใช้ลวดนำในการนำสัญญาณ 1 เส้น และมีสายกราวน์รวมของทุกวงจรเพียงเส้นดียว

อัตราเร็วในการส่งข้อมูลมีค่า เท่ากับ 9,600 bps

ระยะทางที่ใช้สูงสุดที่ใช้ในการส่งข้อมูลมีค่า < 15m

**6.1.7 ขั้วต่อแบบย่อ (miniature connects)**

ขั้วต่อแบบย่อที่ใช้ใน project นี้ถือว่าเป็น ขั้วต่อแบบย่อ คือเป็นขั้วต่อแบบ DB-9 ที่ใช้เพราะสามารถใช้งานได้ง่าย ใช้พื้นที่น้อย และสามารถตอบสนองความต้องการของไมโครคอมพิวเตอร์ได้อย่างเพียงพอ รูปที่ 6.5 แสดงรูปร่างและระบุตำแหน่งขาต่างๆ ของขั้วต่อทั้ง 2 แบบ




DB9P

| PIN NUMBER | DESIGNATION |
|---|---|
| 1 | RECIVED LINE SIGNAL DETECTOR |
| 2 | TRANSMITTED DATA |
| 3 | RECEIVED DATA |
| 4 | DTE READY |
| 5 | SIGNAL GROUND |
| 6 | DCE READY |
| 7 | RTS READY FOR RECEIVING |
| 8 | CTS |
| 9 | RI |

**รูปที่ 6.5 ระบุตำแหน่งขาและหน้าที่ของ ขั้วต่อ DB-9**

## 6.2 ส่วนประมวลผล(8031) และ output ที่ใช้ในการควบคุม

### 6.2.1ไมโครคอนโทรลเลอร์แบบชิพเดี่ยว เบอร์ 8031

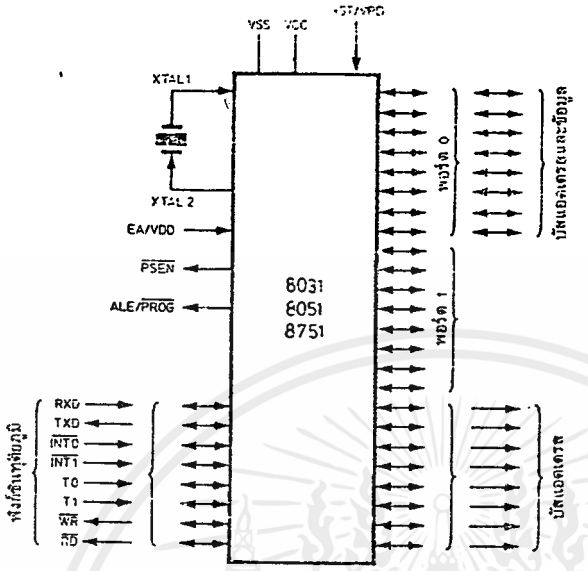
ในปัจจุบันมีไมโครคอนโทรลเลอร์เอ็มซีเอส 51 (ตระกูล 51) ซึ่งเป็นไมโครคอมพิวเตอร์แบบชิพเดี่ยว (ไม่ต้องต่อกับอุปกรณ์ภายนอกก็สามารถทำงานได้) 8031 ก็ถือเป็นไมโครคอนโทรลเลอร์ตระกูล 51 ด้วย มีความสะดวกในการใช้งานและเขียนโปรแกรมควบคุมด้วยภาษาเบสิกได้ โดยไม่ต้องศึกษาการทำงานของวงจรเหมือนกับภาษาแอสแซมบลี้

MCS 51 นี้เป็นอุปกรณ์ที่ออกแบบมาสนองความต้องการของผู้ใช้คือมีสายอินพุทและเอาท์พุตภายในตัวเองพอร์ตของอินพุตและเอาท์พุทบัฟเฟอร์อินเตอร์เฟส และสายควบคุมอื่นๆ ที่ใช้สำหรับแยกข้อมูลกับแอดเดรสและยังมีชุดคำสั่งเพิ่มขึ้นเป็นพิเศษเพื่อจัดการข้อมูลแถมท้ายด้วยวงจรตั้งเวลากับวงจรนับด้วย (ปกติวงจรนับจะสามารถทำงานเป็นวงจรตั้งเวลาได้ด้วย จึงเรียกควบคู่กันไป คือ วงจรตั้งเวลา/ วงจรนับ)

คุณสมบัติของไมโครคอนโทรลเลอร์ MCS 51

- ซีพียู 8 บิต ที่ควบคุมได้ง่าย
- เพิ่มการทำงานลอจิกครั้งละ 1 บิต
- สายอินพุตและเอาท์พุตมีจำนวน 32 เส้น ใช้เลือกแอดเดรสแยกต่างหากจากกันได้
- มีแรมบรรจุไว้ภายในขนาด 128 ไบต์หรือ 256 ไบต์
- วงจรตั้งเวลา/ วงจรนับมีขนาด 2,3 หรือ 16 บิต
- กำหนดเป็น UART (Universal S ynchronous Asynchronous Receiver Transmitter ) ที่รับส่งข้อมูลอนุกรมได้ สองทิศทาง
- อินเตอร์รัพต์ แบ่งออกเป็น 2 ระดับจาก 5 หรือ 6 แหล่ง
- มีสัญญาณนาฬิกาอยู่ภายในตัว
- มีหน่วยความจำสำหรับเก็บข้อมูลภายในขนาด 4 หรือ 8 กิโลไบต์(อีพรอม 8751และ 8752)
- มีแอดเดรสของหน่วยความจำสำหรับเก็บโปรแกรม จำนวนทั้งหมด 64 กิโลไบต์
- มีแอดเดรสของหน่วยความจำสำหรับเก็บข้อมูล จำนวนทั้งหมด 64 กิโลไบต์
- คำสั่งทั้งหมดมี 111 คำสั่ง (64 รอบ)
- ทำงานด้วยเลขฐานสิบ และฐานสิบหก
- ตัวแปลภาษาเบสิกมีขนาด 8 กิโลไบต์(8025 AH-BASIC)
- ควบคุมอีพรอมภายในตัวด้วยภาษาเบสิก(8052 AH-BASIC)

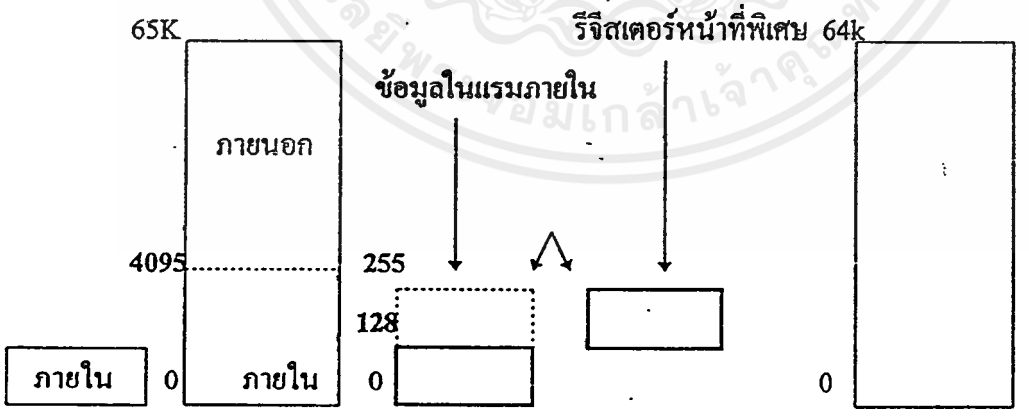
- มีคำสั่งเฉพาะของภาษาเบสิกที่ใช้สำหรับอินพุตและเอาท์พุต วงจรนับ และอินเตอร์เฟสแบบอนุกรม(8052 AH-BASIC)




รูปที่ 6.6 แสดงหน้าที่ของพอร์ตเมื่อคอนโทรลเลอร์ทำงานกับ หน่วยความจำภายนอก

จากรูปที่ 6.6 ไมโครคอนโทรลเลอร์เบอร์ 8031 AH ที่ใช้แทน 8051 ได้โดยไม่ต้องส่งให้โรงงานโปรแกรมให้ เพราะเราจะเขียนและทดสอบโปรแกรมด้วยหน่วยความจำภายนอกแทน(ไม่มีรอมภายใน) และข้อมูลในหน่วยความจำภายในมีขนาด 128x8RAM ส่วนตั้งเวลา/นับ เท่ากับ 2x16-Bit และใช้อินเตอร์รัพต์หมายเลข 5

ภายในหน่วยความจำ




โปรแกรมเคาน์เตอร์ | หน่วยความเก็บโปรแกรม | หน่วยความจำเก็บข้อมูลที่อยู่ภายใน | หน่วยความจำเก็บข้อมูลที่อยู่ภายนอก

**INTERNAL** หน่วยความจำภายใน

**EXTERNAL** หน่วยความจำภายนอก

**PROGRAM COUNTER** วงจรนับลำดับโปรแกรม

**PROGRAM MEMORY** หน่วยความจำสำหรับเก็บโปรแกรม

**INTERNAL DATA RAM** แรมภายในที่เก็บข้อมูล

**SPECIAL FUNCTION REGISTERS** รีจีสเตอร์หน้าที่พิเศษ

**INTERNAL DATA MEMORY** หน่วยความจำภายในสำหรับเก็บข้อมูล

**EXTERNAL DATA MEMORY** หน่วยความจำภายนอกสำหรับเก็บข้อมูล

**รูปที่ 6.6 โครงสร้างภายในหน่วยความจำของ MCS 51**

จากรูปที่ 6.6 เป็นหน่วยความจำภายในตัวเอ็มซีเอส 51 หน่วยความจำนี้แบ่งได้ 2 กลุ่ม คือ หน่วยความจำสำหรับเก็บโปรแกรม และหน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำแรกมีแอดเดรสที่ต่ำกว่า 4 หรือ 6 กิโลไบต์บรรจุอยู่ในรอม ส่วน MCS 51 ที่ไม่มีรอมภายในจะใช้หน่วยความจำภายนอกซึ่งอาจเป็นรอม , แรม หรือ อีพรอมแทนก็ได้

MCS 51 จะอ่านหน่วยความจำสำหรับเก็บโปรแกรม เข้ามาเป็นภาษาเครื่องตามลำดับ ส่วนหน่วยความจำสำหรับเก็บข้อมูล จะใช้เป็นที่เก็บตัวแปร,การคำนวณหาผลลัพธ์ทันที, การจัดการกับข้อมูลที่มีขนาด 16 บิต (word)

หน่วยความจำสำหรับเก็บข้อมูลใช้ร่วมกับหน่วยความจำภายนอกได้ถึง 640 กิโลไบต์ ซึ่งเลือกใช้รอมหรือ แรมก็ได้ และยังมีรีจิสเตอร์พิเศษที่ใช้หน่วยความจำภายนอกของแรมได้ 128 หรือ 256 กิโลไบต์ ส่วนของไมโครคอนโทลเลอร์เบอร์ 8031 นั้นจะประกอบด้วยส่วนหน่วยความจำสำหรับเก็บข้อมูลแต่จะไม่มีหน่วยความจำเก็บโปรแกรม ซึ่งเป็นข้อแตกต่างจาก MCS-51 ส่วนหน้าที่ในส่วนอื่นๆ จะเหมือนกับ MCS-51

**รีจิสเตอร์ภายใน MCS**

MCS มีรีจิสเตอร์ที่อำนวยความสะดวกในการใช้งานตามคำสั่งต่างๆ ประกอบด้วยแอคคิวมูเลเตอร์ รีจีสเตอร์สภานะ สแต็กพอยน์เตอร์ ดาต้าพอยน์เตอร์(2x8 บิต หรือ 1x6 บิต) พอร์ตหมายเลขศูนย์ถึงพอร์ตหมายเลขสามรีจิสเตอร์แบบคู่ ซึ่งใช้ส่งและรับข้อมูลชนิดอนุกรม รีจิสเตอร์ 16 บิต ที่เป็นวงจรตั้งเวลาและวงจรนับ รีจิสเตอร์ซึ่งจองไว้สำหรับใช้สำหรับนับตัวที่ 3 รีจิสเตอร์คำสั่ง สำหรับหน้าที่พิเศษ (เช่น การอินเตอร์รัพต์ RTC : Read Time Clock ) และอินพุต,เอาท์พุตแบบอนุกรม

**เมื่อทำงานเป็นวงจรตั้งเวลา**

รีจิสเตอร์ที่ทำหน้าที่ตั้งเวลาจะเพิ่มค่าขึ้นหนึ่งของทุกๆ รอบคำสั่งของเครื่อง และจะนับด้วยอัตราสูงสุดที่ 1/12 ของความเร็วสัญญาณนาฬิกาของโปรแกรมเซสเซอร์

**เมื่อทำงานเป็นวงจรนับ**

รีจิสเตอร์วงจรนับจะเพิ่มขึ้นหนึ่งเมื่อมีสัญญาณป้อนให้อินพุต T0,T1 หรือ T2 (T2 มีเฉพาะ 8052) เป็นขอบสัญญาณขาลง อัตราการนับสูงสุดคือ 1/25 ของความเร็วสัญญาณนาฬิกาของโปรเซสเซอร์

วงจรนับและวงจรตั้งเวลา 0 และ 1 มีวิธีโปรแกรมให้ทำงานได้ต่างกันถึง 4 แบบ รวมทั้งการทำงานเป็น 8 บิต และการบรรจุค่าพรีเซตหนึ่งค่าได้อย่างอัตโนมัติ

วงจรนับและวงจรตั้งเวลาที่ 1 เลือกโปรแกรมให้ทำหน้าที่ เป็นตัวกำเนิดสัญญาณของอัตราการส่งบิตออกไปอย่างอนุกรม สำหรับใช้ในการอินเตอร์เฟสได้

วงจรนับและวงจรตั้งเวลาที่ 2 (เฉพาะ 8052 เท่านั้น ) มีการทำงานย่อยๆ อีก 3 ชนิด

1. วงจรนับ 16 บิตที่สามารถโหลดค่ากลับคืนเองอย่างอัตโนมัติ
2. วงจรนับที่จองไว้ ชนิด 16 บิต
3. วงจรกำเนิดสัญญาณของการส่งบิตเพื่อใช้อินเตอร์เฟส

**พอร์ตชนิดอนุกรมอยู่ภายใน**

ไมโครคอนโทรลเลอร์ทั้งหมดในตระกูล MCS 51 เป็นชิพที่มีอินเตอร์เฟสอนุกรมชนิดสองทิศทาง ทำให้รับและส่งข้อมูลพร้อมกัน ตัวรับข้อมูลชนิดอะซิงโครนัส (asynchronous receiver) มีบัฟเฟอร์สำหรับข้อมูลเป็นพิเศษเพื่อเพิ่มความเร็วในการสื่อสาร

พอร์ตชนิดอนุกรมนั้นเราสามารถเลือกโปรแกรม เพื่อเลือกใช้การทำงานแบบใดแบบหนึ่งใน 4 แบบ ด้วยการใช้โปรแกรมควบคุมอัตราการส่งข้อมูลและรูปแบบของข้อมูล

อัตราการส่งข้อมูลที่เลือกใช้ได้จะสูงถึง 19,200 บิต / วินาที ด้วยความเร็วของสัญญาณนาฬิกา 1 เมกะเฮิรตซ์ สำหรับใช้ในระบบการสื่อสารของโปรเซสเซอร์หลายตัวร่วมกัน เราจะเลือกความเร็วของสัญญาณนาฬิกาด้วยวงจรนับและวงจรตั้งเวลา

**อินเตอร์รัพต์และชุดคำสั่ง**

8051 รับการอินเตอร์รัพต์ได้ 5 แหล่ง ส่วน 8052 รับอินเตอร์รัพต์จากอุปกรณ์อื่นๆ ได้ถึง 6 แหล่ง คือ ขา INTO และขา INT1 (กำหนดให้ใช้ระดับพัลส์หรือขอบขาพัลส์ก็ได้) วงจรนับและวงจรตั้งเวลาที่ 0 และ 1 (สำหรับ 8052 เพิ่มวงจรตั้งเวลา / วงจรนับตัวที่ 2) และสุดท้ายจากพอร์ตอนุกรม เราสามารถกำหนด ลำดับความสำคัญของอินเตอร์รัพต์ได้ 2 ระดับ โดยไม่ต้องอาศัยวงจรภายนอกช่วย แต่ละแหล่งของอินเตอร์รัพต์ 5 หรือ 6 แหล่งนั้นจะกำหนดเป็นเวกเตอร์เฉพาะ (ตัวชี้แอดเดรส)

ดังนั้นเมื่อมีอินเตอร์รัพต์เข้ามาแล้วตัวโปรเซสเซอร์จะกระโดดไปที่ส่วนของโปรแกรมที่ทำงานตามวัตถุประสงค์ของอินเตอร์รัพต์นั้น หลังจากเก็บข้อมูลต่างๆ ของโปรแกรมเคาน์เตอร์ลงในสแต็ก ชุดคำสั่งทั้งหมดของตัวคอนโทลเลอร์ในตระกูล MCS

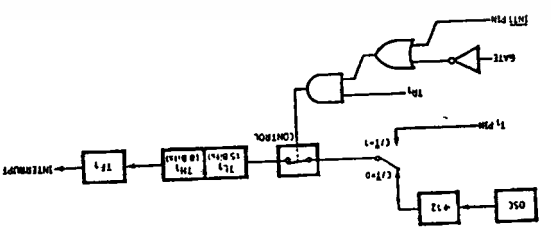
**ตัวจับเวลา/ ตัวนับ (Timer/Counter)**

ในเบอร์ 8031 นี้ จะมีตัวจับเวลา/ตัวนับขนาด 16 บิต จำนวน 2 ตัว คือ ไทเมอร์/เคาน์เตอร์ 0 และไทเมอร์/เคาร์เตอร์ 1 โดยแต่ละตัวสามารถที่จะกำหนดให้ทำงานเป็นตัวจับเวลาหรือตัวนับได้โดยการเซตหรือเคลียร์บิต C/T ที่ตัวรีจิสเตอร์ควบคุม TMOD ซึ่งอยู่ในกลุ่มรีจิสเตอร์ฟังก์ชันพิเศษ ในการกำหนดให้ทำงานเป็นตัวจับเวลา ตัวรีจิสเตอร์ TH1 และ TL1 ซึ่งทำหน้าที่เป็นตัวเก็บค่าจำนวนพัลส์ที่เข้ามาจะเพิ่มค่าทุกๆ แมชีนไซเกิล โดยแต่ละแมชีนไซเกิลจะประกอบด้วย 12 คาบออสซิลเลเตอร์ ดังนั้นอัตราการนับแต่ละครั้งจะใช้เวลาเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์ ซึ่งส่วนใหญ่จะใช้ในงานอินเตอร์รัพต์ RTC (Real Time Clock)

และถ้าให้ทำงานเป็นตัวนับ รีจิสเตอร์ตัวนับจะเพิ่มค่าทุกครั้งที่มีการเปลี่ยนแปลงสถานะจาก "1" เป็น "0" ที่ขา T0 หรือ T1 โดยอัตราความถี่สูงสุดที่สามารถนับได้ต้องไม่เกิน 1/24 ของความถี่ออสซิลเลเตอร์

ตัวจับเวลา/ตัวนับ สามารถโปรแกรมให้มันทำงานได้ต่างกันถึง 4 โหมด โดยการตั้งค่าในรีจิสเตอร์ TMOD ซึ่งจะกล่าวถึงการทำงานของแต่ละโหมดดังนี้

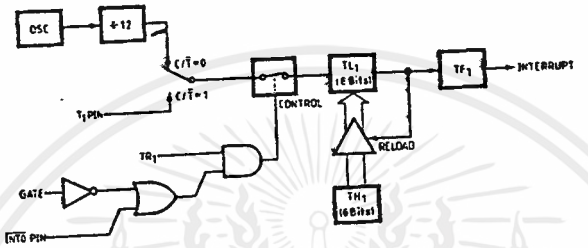
**โหมด 0** รีจิสเตอร์ตัวนับจะถูกกำหนดให้มี 13 บิต ประกอบด้วยรีจิสเตอร์ TH1 8 บิต และ TL1 อีก 5 บิตอันดับต่ำ ซึ่งสามารถกำหนดให้เป็นตัวจับเวลา หรือ ตัวนับได้โดยเซตหรือเคลียร์ที่บิต C/T ในตัวรีจิสเตอร์ TMOD การทำงานของรีจิสเตอร์ตัวนับจะนับขึ้นครั้งละ 1 เมื่อมีสัญญาณเข้ามา 1 ลูกและเมื่อนับจนเป็น "1" หมดทุกบิต ก็จะกลับมาเป็น "0" หมดทุกบิตใหม่ ซึ่งจะเป็นการเกิดโอเวอร์โฟลว์ (Overflow) ไปทดแฟลกอินเตอร์รัพต์ TF1 ให้เป็น "1" ลักษณะวงจรเป็น ดังรูปที่ 6.7



รูปที่ 6.7 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 0 และโหมด 1

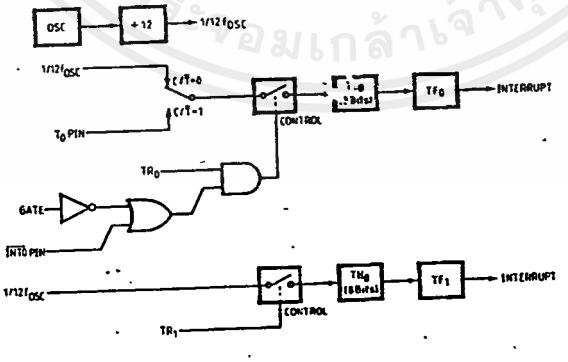
**โหมด 1** การทำงานจะเหมือนกับโหมด 0 ทุกอย่างยกเว้นรีจิสเตอร์ตัวนับจะเป็นขนาด 16 บิต

**โหมด 2** จะใช้รีจิสเตอร์TL1 เป็นตัวนับเพียงตัวเดียวและเมื่อ TL 1 นับจนเป็น "1" หมดทุกบิต ก็จะมีการโหลดค่าจากรีจิสเตอร์ TH1 เข้าไปไว้ใน TL1 โดยอัตโนมัติและทำการทดแฟลกอินเตอร์รัพต์ TF1 ให้เป็น "1" ค่าใน TH1 นี้เราสามารถตั้งค่าได้ด้วยซอฟต์แวร์ ลักษณะ วงจรแสดงในรูปที่ 6.9




รูปที่ 6.9 แสดงวงจรการทำงานของตัวจับเวลา/ ตัวนับ ที่ 1 ในโหมด 2

**โหมด 3** เป็นการเพิ่มตัวจับเวลาขึ้นอีก 1 ตัว แต่จะเป็นขนาด 8 บิตทั้งคู่ ซึ่งลักษณะการทำงานอื่นๆ จะเหมือนกับโหมด 0 การทำงานแสดงในรูปที่ 6.10




รูปที่ 6.10 วงจรการทำงานของตัวจับเวลา/ตัวนับที่ 0 ในโหมด 3

สำหรับรายละเอียดบิตควบคุมในรีจิสเตอร์ TMOD และ รีจิสเตอร์ TCON มีดังต่อไปนี้

แสดงบิตควบคุมที่อยู่ในรีจิสเตอร์ TMOD

| TIMER/COUNTER1 | | | | TIMER/COUNTER 0 | | | |
|---|---|---|---|---|---|---|---|
| GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |

GATE : เซตเป็น "1" จะเป็นการอีนาเบิลตัวจับเวลา/ตัวนับให้ถูกควบคุมด้วยขา INTx ต้องมีสถานะสูงและบิต TRX ใน TCON ต้องเซตเป็น "1" จึงจะเริ่มทำงาน แต่ถ้า GATE เป็น "0" ตัวจับเวลา/ ตัวนับจะถูกควบคุมให้เริ่มทำงานด้วยบิต TRX เท่านั้น

C/T : เป็นบิตควบคุมในการเลือกทำงานเป็นตัวจับเวลาหรือตัวนับ ถ้าเป็น "0" จะทำงาน เป็นตัวจับเวลา ถ้าเป็น "1" ทำงานเป็นตัวนับ

M1,,M0: เป็นตัวเลือกโหมดการทำงานดังนี้

| M1 | M0 | โหมดการทำงาน |
|---|---|---|
| 0 | 0 | โหมด 0 |
| 0 | 1 | โหมด 1 |
| 1 | 0 | โหมด 2 |
| 1 | 1 | โหมด 3 |

แสดงบิตควบคุมที่อยู่ภายในรีจิสเตอร์ TCON

| $TF_1$ | $TR_1$ | $TF_0$ | $TR_0$ | $IE_1$ | $IT_1$ | $IE_0$ | $IT_0$ |
|---|---|---|---|---|---|---|---|

TFx : เป็นแฟลกอินเตอร์รัพต์จะถูกเซตด้วยฮาร์ดแวร์เมื่อเกิดโอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับ และ จะเคลียร์ตัวเองโดยอัตโนมัติเมื่อทำงานอินเตอร์รัพต์นั้นเสร็จเรียบร้อยแล้ว

TRx : เป็นบิตควบคุมให้ตัวจับเวลา/ตัวนับเริ่มทำงานโดยการเซตให้เป็น "1" และให้หยุด ทำงานด้วยการเคลียร์ให้เป็น "0" โดยซอฟต์แวร์

IEx : เป็นแฟลกอินเตอร์รัพต์จากสัญญาณภายนอก เซตด้วยฮาร์ดแวร์เมื่อมีสัญญาณขอการอินเตอร์รัพต์ปรากฏที่ขา INTx และจะเคลียร์ตัวเองโดยอัตโนมัติเมื่อกระโดดไปทำงานบริการอินเตอร์รัพต์ที่ขอมาเรียบร้อยแล้ว

ITx : เป็นบิตควบคุมรูปแบบสัญญาณอินเตอร์รัพต์ภายนอกจะเซต/เคลียร์ด้วยซอฟแวร์ โดยถ้าเซตเป็น "1" จะถูกอินเตอร์รัพต์ด้วยสัญญาณขอบขาลงและ ถ้าเคลียร์เป็น "0" จะถูก

อินเตอร์รัพต์ด้วยสัญญาณของขาลงและถ้าเคลียร์เป็น "0" จะถูกอินเตอร์รัพต์ด้วยสัญญาณระดับแรงดันต่ำ

หมายเหตุ x หมายถึงเลข 1 หรือเลข 0

**** ซึ่งในที่นี้ใช้โหมด 2 รีจิสเตอร์ TMOD ถูกเซตให้มีค่าเท่ากับ B7H**

**พอร์ตอนุกรม**

MCS -51 จะมีพอร์ตอนุกรมเป็นแบบฟูลดูเพล็กซ์ (full duplex) สามารถที่จะส่งและรับข้อมูลได้พร้อมกันเพราะมีบัฟเฟอร์ 2 ตัว ใช้ในการรับตัวหนึ่งและส่งตัวหนึ่ง โดยโครงสร้างของรีจิสเตอร์บัฟเฟอร์ทั้ง 2 ตัวนี้จะแยกกันแต่การติดต่อจะใช้ชื่อเดียวกันคือ SBUF พอร์ตอนุกรมของ MCS-51 สามารถที่จะโปรแกรมให้ทำงานได้แตกต่างกัน 4 โหมด

**โหมด 0** ข้อมูลจะเข้าและออกทางขา RXD โดยการเลื่อนสัญญาณนาฬิกาออกที่ขา TXD ข้อมูลจะเป็น 8 บิต โดยจะส่งบิตนัยสำคัญต่ำ (LSB) ก่อน อัตราบ็อดจะคงที่ที่ 1/12 ของความถี่ออสซิลเลเตอร์

**โหมด 1** เป็นการรับ/ส่งข้อมูลขนาด 10 บิต โดยการส่งออกทางขา TXD และรับเข้าทางขา RXD รูปแบบบิตจะประกอบด้วย 1 บิตสตาร์ตเป็น "0" , 8 บิตข้อมูล และ 1 สตอ๊อปบิตเป็น "1" อัตราบ๊อด (baud rate ) แปรผันได้ตามการตั้งตัวจับเวลาตัวที่ 1 โดยมีสูตรดังนี้

$$\text{อัตราบ๊อด} = \frac{2^{SMOD}}{32} \times \frac{\text{ความถี่ออสซิลเลเตอร์}}{12\text{x}(256\text{-TH1})}$$

**โหมด 2** เป็นการรับส่งข้อมูลขนาด 11 บิต เข้าทางขา RXD และส่งออกทางขา TXD ประกอบด้วย 1 บิต สตาร์ตมีค่า "0" , 9 บิตข้อมูล และ บิตสต๊อป โดยการรับข้อมูลบิตที่ 9 จะถูกนำมาเก็บที่บิต RB8 ใน SCON ส่วนการส่งจะต้องใส่บิตที่ 9 ไว้ใน TB8 ของ SCON ส่วนการส่งจะต้องเลือกได้ 2 อัตราคือ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ขึ้นอยู่กับการเซตบิต SMOD ในรีจิสเตอร์ PCON

**โหมด 3** จะเหมือนกับโหมด 2 ทุกอย่าง ยกเว้นอัตราบอด จะแปรผันตามการตั้งตัวจับเวลา 1 ซึ่งจะใช้สูตรเดียวกับโหมด 1

แสดงบิตควบคุมที่อยู่ในรีจิสเตอร์ SCON

| SM0 | SM1 | SM2 | REN | TB8 | RB8 | T1 | R1 |
|---|---|---|---|---|---|---|---|

SM0 ,SM1 : เป็นตัวกำหนดโหมดการใช้งานของพอร์ตอนุกรมดังนี้

| SM0 | SM1 | โหมด |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

SM2 : ควบคุมอีนาเบิลการใช้โปรเซสเซอร์หลายตัวในการสื่อสารซึ่งกันและกัน

REN : ตัวอีนาเบิลอนุกรมการรับเมื่อเซตเป็น "1" และถ้าเป็น "0" เป็นการดิสเอเบิล

TB8 : เป็นตัวเก็บข้อมูลบิตที่ 9 ที่จะส่งในโหมด 2 และ 3

RB8 : เป็นตัวรับข้อมูลบิตที่ 9 ในโหมด 2 และ 3 ส่วนในโหมด 1 จะเป็นสต๊อปบิต

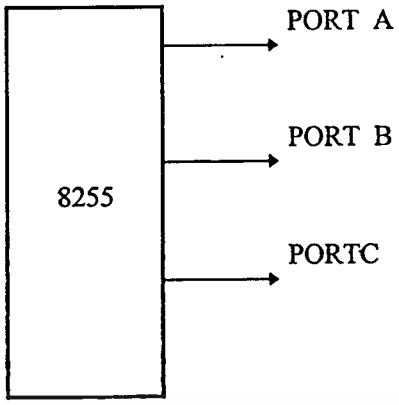
T1 : เป็นแฟลกอินเตอร์รัพต์การเซตด้วยฮาร์ดแวร์ที่ปลายช่วงของบิตที่ 8 ในโหมด 0 หรือที่จุดเริ่มต้นของบิตสต๊อปในโหมดอื่น ในการส่งแบบอนุกรมของทุกโหมดจะต้องเคลียร์บิตนี้ด้วยโปรแกรมหลังการส่ง

R1 : เป็นแฟลกอินเตอร์รัพต์การรับ เซตด้วยฮาร์ดแวร์ที่ปลายช่วงของบิตที่ 8 ในโหมด 0 หรือจุดครึ่งของช่วงบิตสต๊อปในโหมดอื่น ในการรับแบบอนุกรมจะต้องเคลียร์บิตนี้ด้วยโปรแกรมหลังการรับทุกครั้ง

****ซึ่งในที่นี้ใช้โหมด 0 ในการทำงานและ รีจิสเตอร์ SCON ถูกเซตค่าให้มีค่าเท่ากับ 10H**

### 6.2.2 IC ขยายพอร์ตเบอร์ 8255

ทำหน้าที่ในการติต่อระหว่างไมโครคอนโทลเลอร์เบอร์ 8031 กับ ส่วน ฮาร์ดแวร์ ซึ่งจะมีพอร์ตที่ใช้ในการติดต่อกับส่วนฮาร์ดแวร์ต่างๆ ดังแสดงในรูปที่ 6.11




รูปที่ 6.10 แสดงพอร์ตที่ใช้ในการติดต่อกับส่วนฮาร์ดแวร์

จากรูป PORT A จะประกอบด้วย bit ทั้งหมด 8 bit แต่และbit ทำงานดังต่อไปนี้

A0 : DIRECTION BASE

A1 : DIRECTION SHOULDER

A2 : DIRECTION ELBOW

A3 : DIRECTION GRIPPER

A4 : BREAK BASE

A5 : BREAK SHOULDER

A6 : BREAK ELBOW

A7 : BREAK GRIPPER

PORT B จะประกอบด้วย บิททั้งหมด 8 บิท แต่ละบิททำงานดังต่อไปนี้

B0 : HOME BASE

B1 : HOME SHOULDER

B2 : HOME ELBOW

B3 : HOME HANDUP-DOWN

B4 : HOME HANDRIGHT-LEFT

B5 : PULSE BASE

B6 : PULSE SHOULDER

B7 : PULSE ELBOW

PORT C จะประกอบด้วยบิททั้งหมด 8 บิท แต่ละบิททำงานดังต่อไปนี้

C0 : PHASE A

C1 : PHASE B

C2 : PWM

C3 : SELECT STEP

C4 : HIGHT/LOW VOLTAGE

C5 : BUZZER

C6 : _____

C7 : _____

## 6.3 ส่วนที่เป็นวงจร drive motor ใช้ IC เบอร์ 18245 และ 18200

ในโปรเจคครั้งนี้จะใช้ IC เบอร์ 18245 ในการควบคุมการ drive motor ในส่วน gripper ซึ่งเป็น DC MOTOR และใช้ IC เบอร์ 18200 ในการ drive motor ในส่วนอื่นๆ ที่เป็น DC MOTOR

### 6.3.1 วงจร DRIVE MOTOR LMD 18245

รายละเอียดของ IC เบอร์ LMD 18245 มีขาสำคัญๆ ดังต่อไปนี้

ขาที่ 1 เป็นขาที่ใช้ต่อเข้ากับมอเตอร์

ขาที่ 4 เป็นขาที่กำหนดการลิมิตกระแสที่จ่ายให้กับมอเตอร์(M4)

ขาที่ 5 เป็นขากราวด์

ขาที่ 6 เป็นขาที่กำหนดค่าลิมิตกระแสที่จ่ายให้กับมอเตอร์(M3)

ขาที่ 7 เป็นขาที่กำหนดค่าลิมิตกระแสที่จ่ายให้กับมอเตอร์(M2)

ขาที่ 8 เป็นขาที่กำหนดค่าลิมิตกระแสที่จ่ายให้กับมอเตอร์(M1)

ขาที่ 9 เป็นขาที่เป็น VCC ของชุด DMOS H-BRIDGE ภายในตัว LMD 18245

ขาที่ 10 เป็นขาที่ใช้ในการ break มอเตอร์

ขาที่ 14 เป็นขา voltage reference ของชุด DAC (M1-M4)

ขาที่ 15 เป็นขาที่ใช้ต่อกับมอเตอร์

**ส่วนที่ได้นำมาใช้ในการทำงานอีกส่วนหนึ่งของ LMD 18245 อีกส่วนหนึ่งก็คือ ส่วนทำหน้าที่ในการกำหนดค่า LIMIT CURRENT ที่จ่ายให้กับ ดีซีมอเตอรื**

ในการกำหนดค่าบลิมิตกระแสของ LMD 18245 สามารถกำหนดได้จากขา M1-M4 โดยคำนวณจากสูตร

$$\text{LIMIT CURRENT} = VDAC_{REF} * D/16$$

โดยในโปรเจคนี้จะใช้ DC มอเตอร์ที่มีการกำหนดค่ากระแสไม่ให้เกินค่า 2 amp โดยในการกำหนดค่า D จาก M1-M4 ที่มีค่าเท่ากับ 7 จะได้ค่า LIMIT CURRENT เท่ากับ 2.1875 amp และถ้ากำหนดค่า D ที่มีค่าเท่ากับ 6 จะได้ค่า LIMIT CURRENT เท่ากับ 1.875 amp ดังนั้นเราจึงเลือกค่า LIMIT CURRENT อยู่ที่ 1.875 amp ได้ค่า D เท่ากับ 6 ดังนั้นเราจึงต่อขา M1 และ M4 ลงกราวด์และขา M2,M3 ขึ้นกับ 5 volts

**6.3.2 วงจร DRIVE MOTOR LMD 18200**

รายละเอียดของ IC เบอร์ LMD 18200 มีขาที่สำคัญๆ ดังต่อไปนี้

ขาที่ 1 เป็นขาที่ใช้ต่อกับ C-BOOTSTRAP

ขาที่ 2 เป็นขาที่ใช้ต่อเข้ากับมอเตอร์

ขาที่ 3 เป็นขาที่ใช้ในการกำหนดทิศทาง ( ขา DIRECTION )

ขาที่ 4 เป็นขาที่ใช้ CONTROL BREAK

ขาที่ 5 เป็นขาที่ใช้ CONTROL PWM

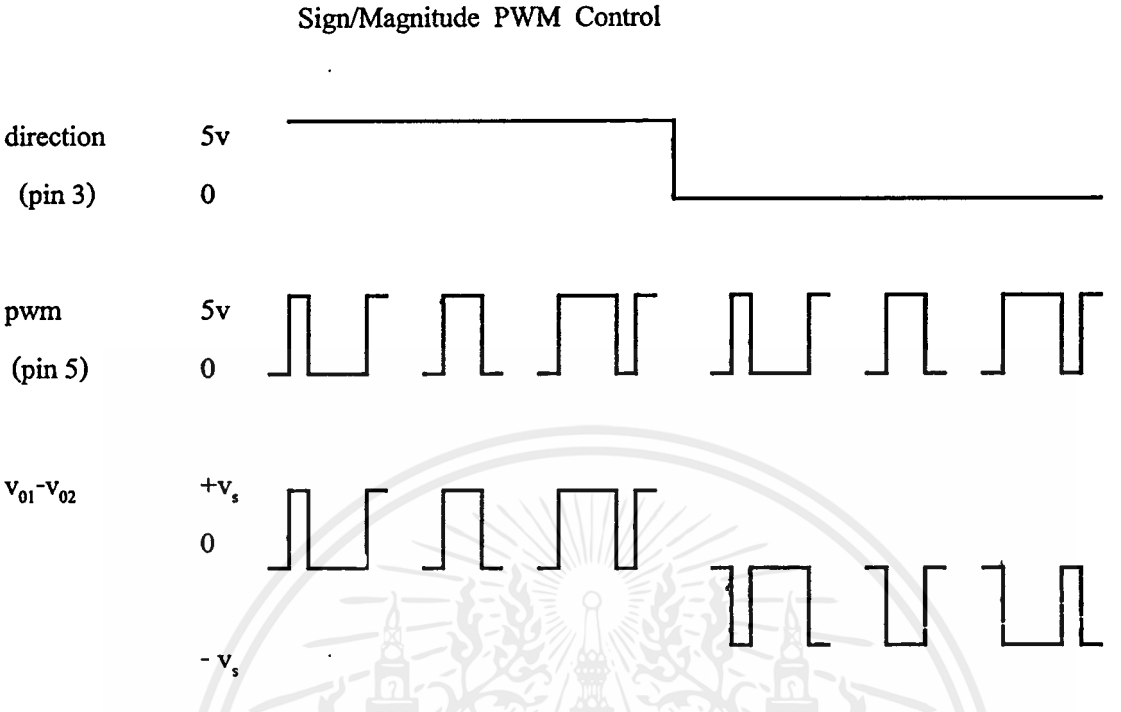
ขาที่ 6 เป็นขาที่เป็น $V_S$ Power Supply

ขาที่ 7 เป็นขากราวด์

ขาที่ 10 เป็นขาที่ใช้ต่อเข้ากับมอเตอร์

ขาที่ 11 เป็นขาที่ใช้ต่อกับ C-BOOTSTRAP

อีกส่วนหนึ่งที่ได้นำมาใช้ในการทำงานก็คือส่วน PWM ซึ่งจะใช้การทำงานแบบ Sign/magnitude PWM ลักษณะสัญญาณต่างๆแสดงในรูปที่ 6.11




รูปที่ 6.11 ลักษณะสัญญาณต่างๆ ในการทำงานแบบ Sign/Magnitude PWM

ลักษณะการทำงานของ PWM ในลักษณะนี้จะประกอบด้วย สัญญาณ direction(sign) และสัญญาณ amplitude (magnitude) ที่แตกต่างกันซึ่งจะขึ้นอยู่กับการป้อนแรงดันที่ output ที่แตกต่างกัน(ที่ขา direction ขา3 )ทำให้มอเตอร์หมุนในทิศทางที่แตกต่างกันคือทางซ้ายและทางขวาเพราะฉะนั้นอัตราเร็วในการหมุนจึงขึ้นอยู่อยู่กับ % ของ duty cycle ดังแสดงในรูป (motor speed ) แบ่งเป็น slow ,meduim, fast (ที่ขา 5 ขา PWM)

## 6.4 ส่วนโปรแกรมการควบคุม

ในส่วนของโปรแกรมการควบคุมของแขนกลนี้จะแบ่งเป็นส่วนหลักๆ ได้ 2 ส่วน คือ

1. ส่วนโปรแกรมแอสแซมบลี้ (โปรแกรม 8051)
2. ส่วนโปรแกรม DELPHI

### 6.4.1 ส่วนโปรแกรมแอสแซมบลี้ (โปรแกรม 8051)

เป็นส่วนหลักในการรับคำสั่งในการควบคุมแขนกลจากคอมพิวเตอร์ PC ซึ่งในการรับคำสั่งจะรับคำสั่งจาก serial port ในการรับคำสั่งแต่ละครั้งนั้นจะรับคำสั่งมาทีละ 2 byte ซึ่งในแต่ละไบต์จะมีหน้าที่ในการควบคุม output ของ 8255 ซึ่งติดต่ออยู่กับไมโครคอนโทลเลอร์ 8051 ในการควบคุม motor ในแต่ละส่วนบนแขนกล ข้อมูลบนพอร์ตเหล่านี้มีผลต่อการเลือก drive motor ในชุดที่เราต้องการ ในส่วนหลักของโปรแกรมเราจะเขียนโปรแกรมในลักษณะของการรับ interrupt ของ serial port ในการรับข้อมูลและการติดต่อระหว่าง PC ของ โครงงานนั้น ในขั้นนี้เราจะติดต่อในลักษณะของ simplex

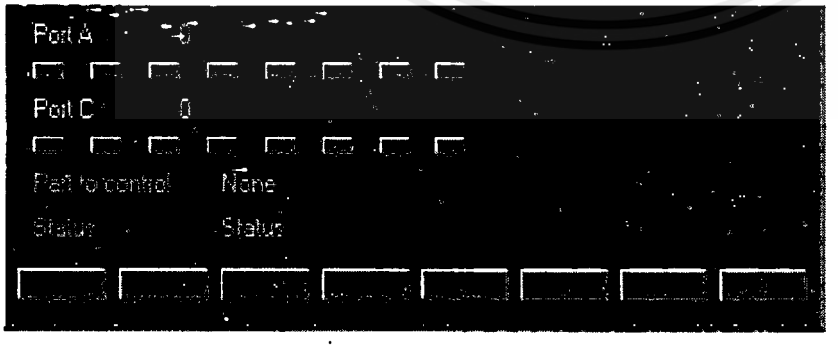
### 6.4.2 ส่วนโปรแกรม DELPHI

เป็นส่วนที่ใช้ทำหน้าที่ส่งคำสั่งไปควบคุมแขนกลโดยผ่านไมโครคอนโทรลเลอร์ 8051 ซึ่งในส่วนของการควบคุมจะแบ่งเป็น 2 ส่วนก็คือ

1. ส่วนที่ใช้ในการคอนโทรล DC MOTOR
2. ส่วนที่ใช้ในการคอนโทรล STEPPING MOTOR

#### 6.4.2.1 ส่วนที่ใช้ในการคอนโทรล DC MOTOR

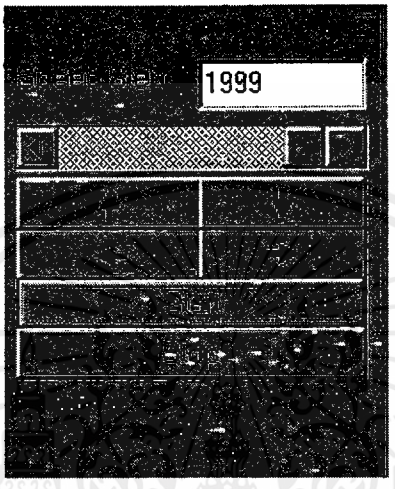
จะเป็นส่วนที่ใช้ในการ check ว่าเราจะควบคุม DC MOTOR ในส่วนใด และ ในทิศทางใด ก็จะส่งเป็นคำสั่งออกมาเป็นไบต์ที่หนึ่ง และ ในส่วนของการควบคุมแรงดันที่จะใช้ในการจ่ายให้กับให้กับ DC MOTOR จะอยู่ในบางบิตของคำสั่งควบคุมในไบต์ที่สอง



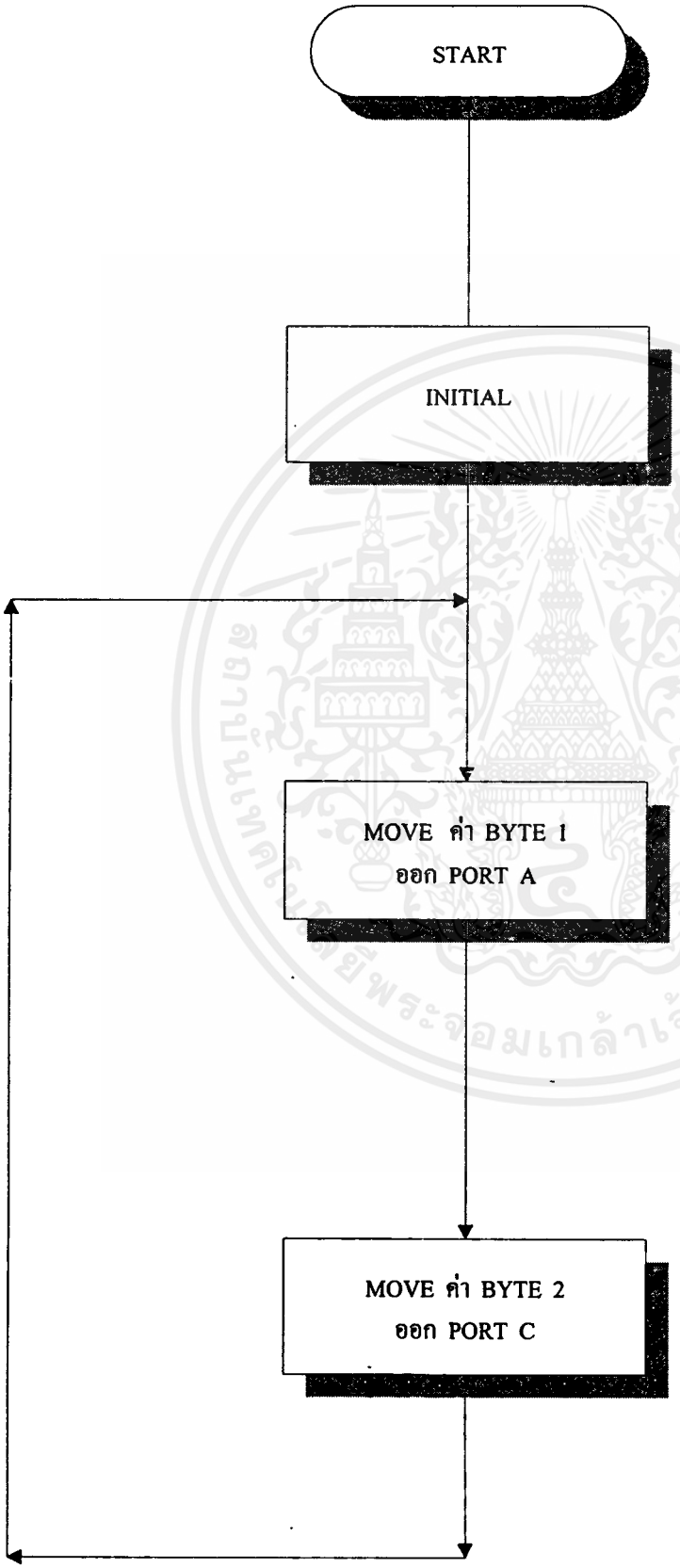

รูปที่ 6.12 รูปแสดงส่วนของโปรแกรม DELPHI บริเวณที่แสดงบิตข้อมูลที่ใช้ส่ง

**6.4.2.2 ส่วนที่ใช้ในการคอนโทรลสเตปปิ้งมอเตอร์**

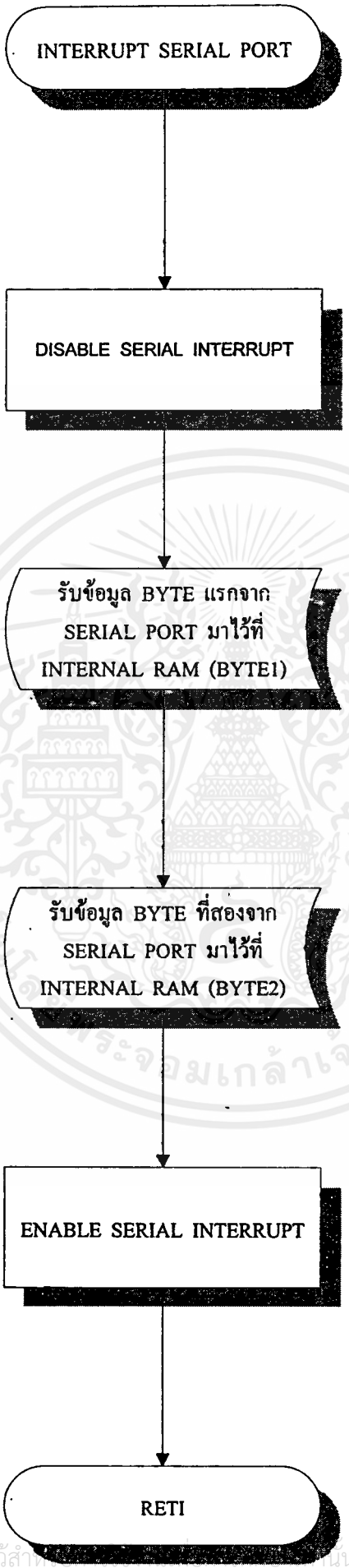
เป็นส่วนที่ใช้ในการคอนโทรลสเตปปิ้งมอเตอร์ซึ่งมีอยู่ในตัวแขนกล 2 ตัว ในโปรแกรมส่วนนี้จะเป็นส่วนที่ส่งสัญญาณกำเนิดเฟสในการคอนโทรลโดยการตั้ง timer ในการส่งเฟสเป็นระยะเวลาที่คงที่ซึ่งความเร็วในการเปลี่ยนแต่ละครั้งนั้นเราสามารถปรับเปลี่ยนได้ด้วย speedbar component ซึ่งผลของการเปลี่ยน speedbar component ที่ตำแหน่งต่างๆ จะทำให้ความเร็วในการหมุนของสเตปปิ้งมอเตอร์เปลี่ยนไป



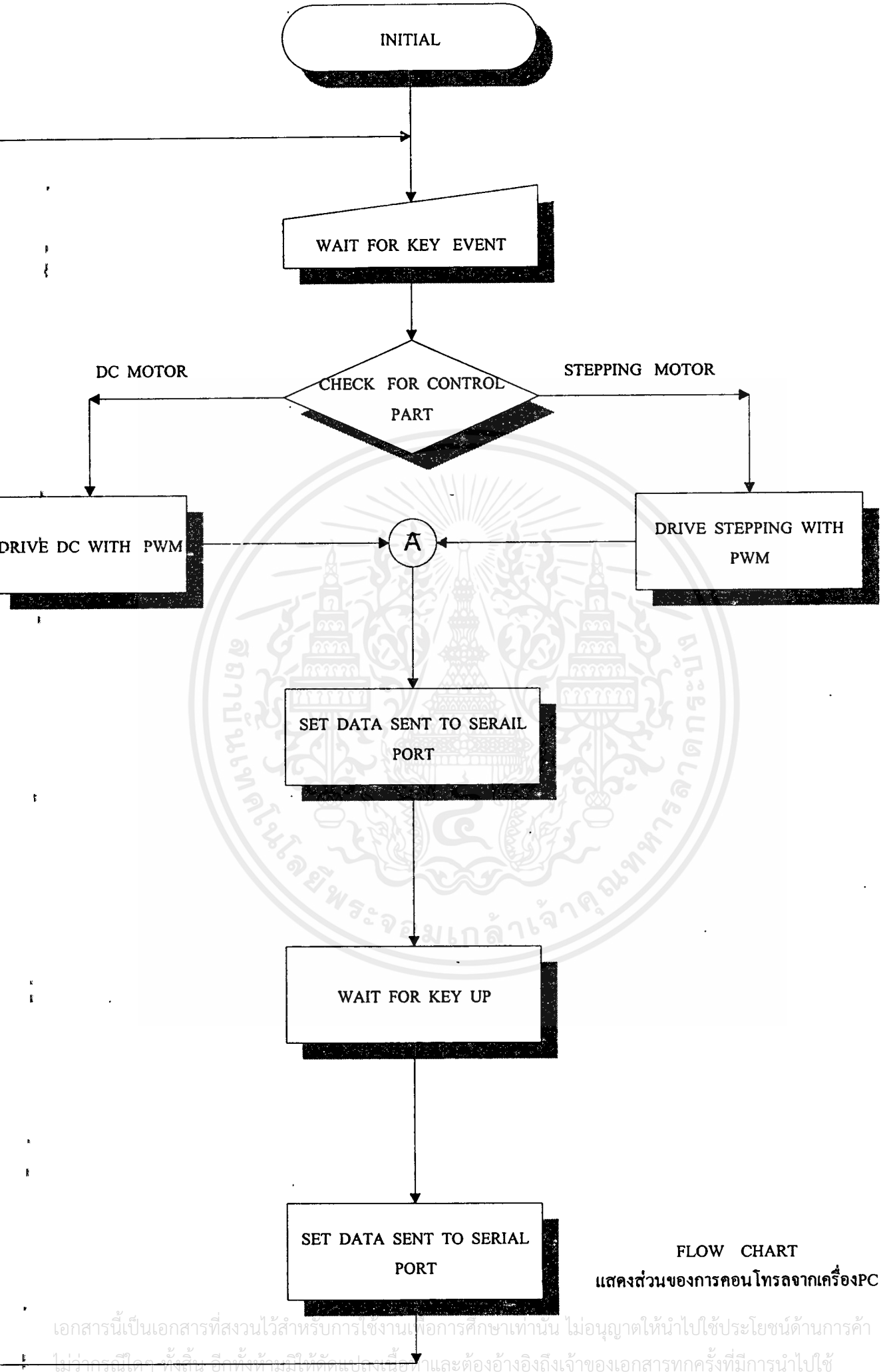

รูปที่ 6.13 รูปแสดงส่วนของโปรแกรม DELPHI ส่วนที่ใช้ปรับความเร็วสเตปปิ้งมอเตอร์




FLOWCHART แสดงส่วนของ MAIN PROGRAM ซึ่งเป็นส่วนของการ OUTPORT




FLOW CHART แสดงส่วนของ INTERRUPT ROUTINE




FLOW CHART
แสดงส่วนของการคอนโทรลจากเครื่องPC

# บทที่ 7

# วงจรและส่วนที่ใช้ในการควบคุม

## 7.1 วงจรสเตปปิ้งมอเตอร์ (STEPPING MOTOR)

เป็นส่วนที่ใช้ในการควบคุมการเคลื่อนที่ในส่วนที่เป็น สเตปปิ้งมอเตอร์ซึ่งมีอยู่ 2 ส่วน คือ ส่วนข้อมือที่ใช้ในการหมุนข้อมือหมุนขึ้นลง และ ส่วนข้อมือที่ใช้ในการหมุนข้อมือไปทางซ้ายและขวา(หมุนรอบข้อมือ ได้ 360 องศา) และเราสามารถเลือกการทำงานได้ว่าจะให้ทำงานในส่วนใด(เนื่องจากมีวงจรสเตปปิ้งมอเตอร์ 2 วงจร) โดยจะใช้คำสั่งคำสั่งที่เป็นลอจิก 1 หรือ ลอจิก 0 ในการเลือก โดยจะผ่านพอร์ต C ของ 8255(ขา select) มายังส่วนวงจรเพื่อทำการเลือกส่วนในการทำงาน ซึ่งการทำงานของสเตปปิ้งมอเตอร์ที่ใช้ในครั้งนี้นั้นจะเป็นแบบ 2 เฟส ดังได้อธิบายไว้แล้วในบทที่ 3 ซึ่งจะทำให้มอเตอร์มีกำลังมากขึ้น รูปที่ 7.1 แสดงวงจรส่วนสเตปปิ้งมอเตอร์

จากรูปที่ 7.1 สามารถอธิบายการทำงานของวงจรสเตปปิ้งมอเตอร์ ได้ดังนี้ คือ เรารับข้อมูลจาก พอร์ตของ 8255 มาเป็นลอจิก 0 หรือ ลอจิก 1 จำนวน 2 bit เพื่อทำการ drive stepping motor ให้ทำงานโดยจะผ่าน IC เบอร์ 74LS138 ซึ่งจะได้ output จำนวน 4 bit ออกมาแต่ยังไม่สามารถนำไป drive motor ได้ จึงต้องผ่านส่วนที่เป็นวงจรเกทก่อนเพื่อทำการแปลงข้อมูลให้เป็นข้อมูลที่สามารถทำให้มอเตอร์ทำงานได้

ลักษณะการทำงานตั้งแต่เมื่อ input(ลอจิก 0 หรือ 1 จำนวน 2 bit ) ได้เข้ามาและผ่าน 74LS138 ( output 4 bit ),วงจรเกทจนได้output ( output 4 bit )ออกมาเป็นดังต่อไปนี้

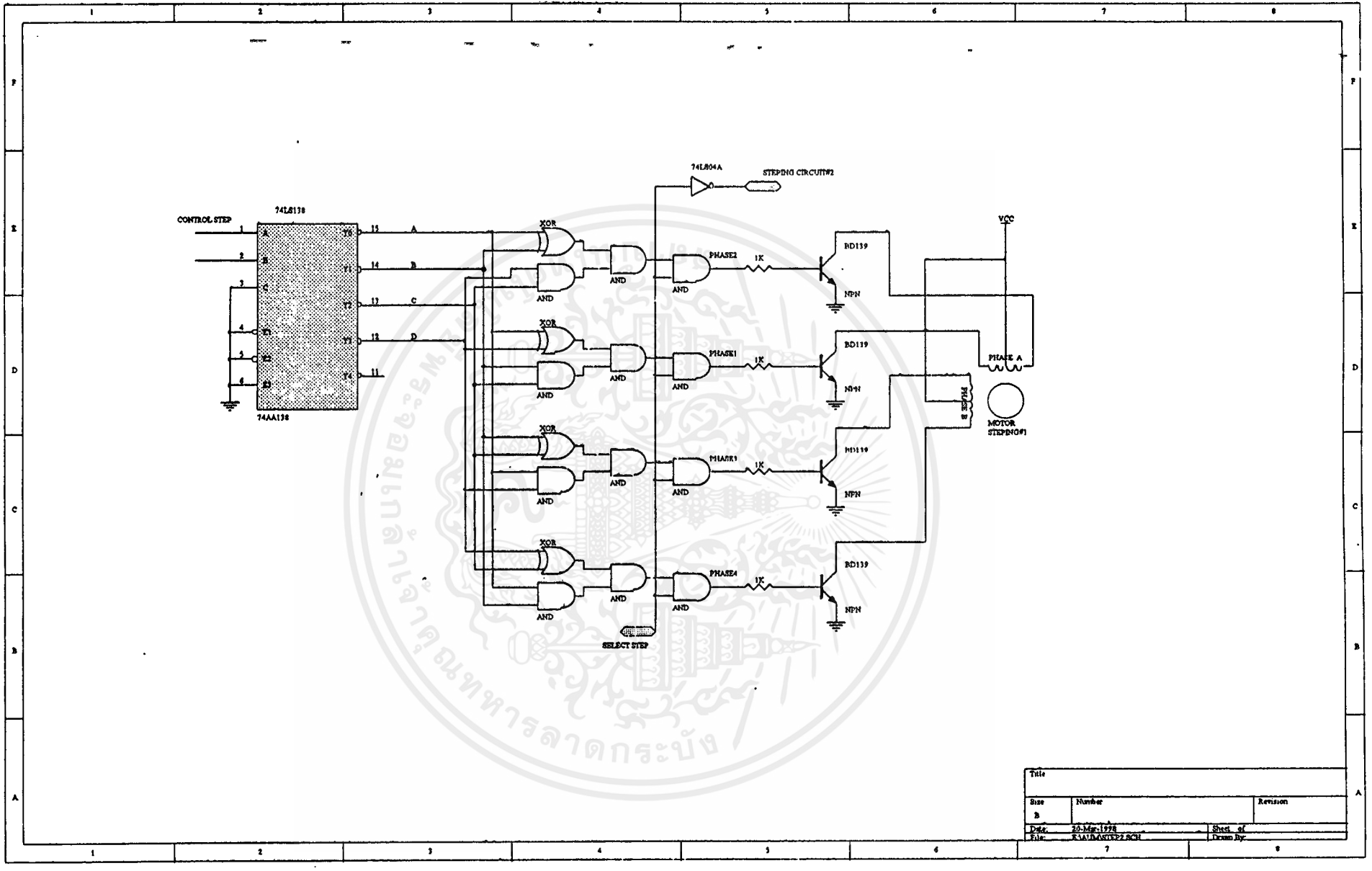
| B | A | A | B | C | D | Y1 | Y2 | Y3 | Y4 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

$Y1 = CD\ (A \oplus B)$

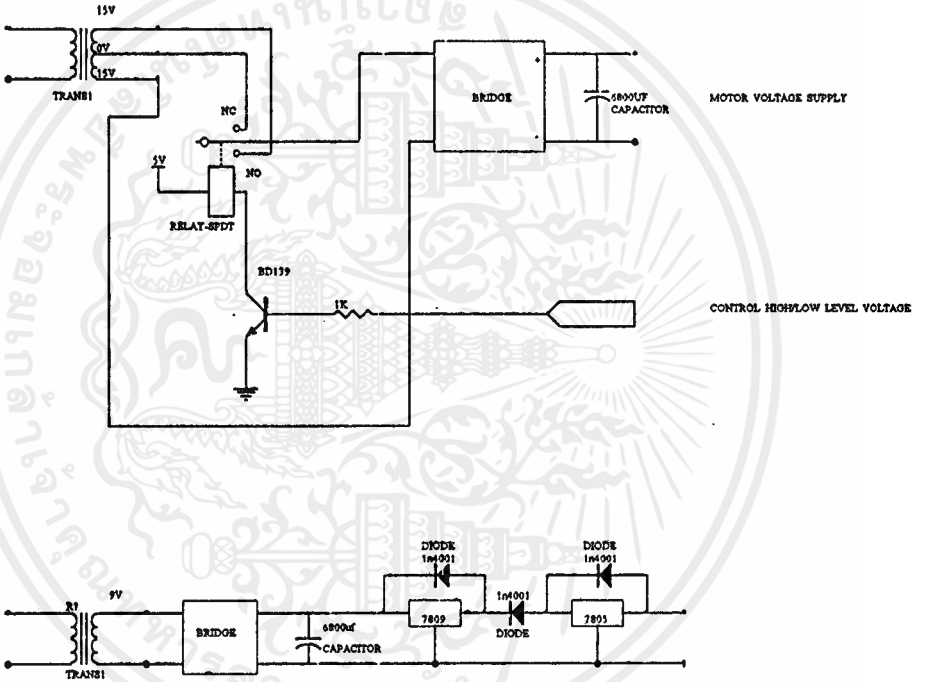
$Y2 = AD\ (B \oplus C)$

$Y3 = AB\ (C \oplus D)$

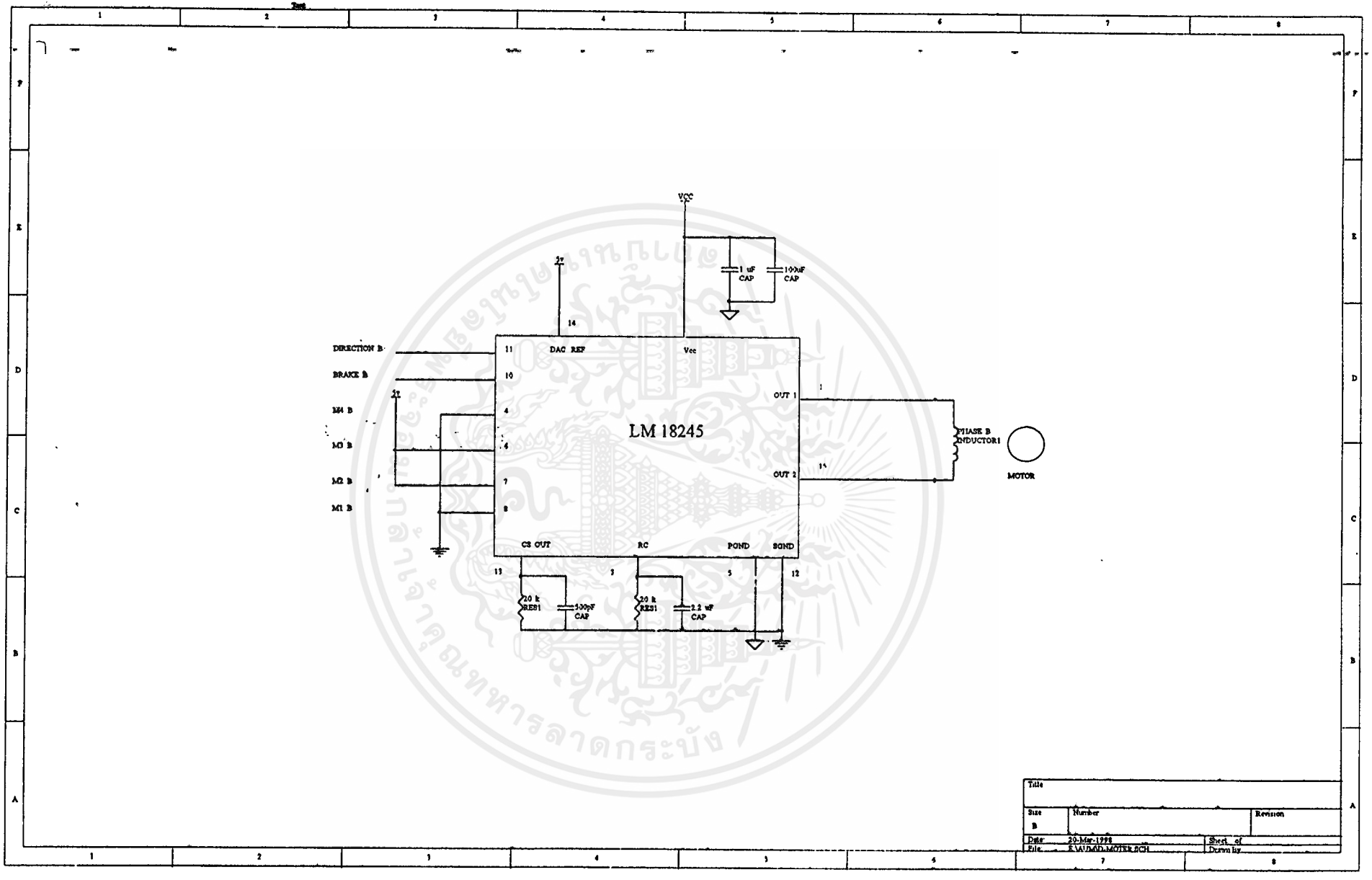
$Y4 = CB\ (A \oplus D)$



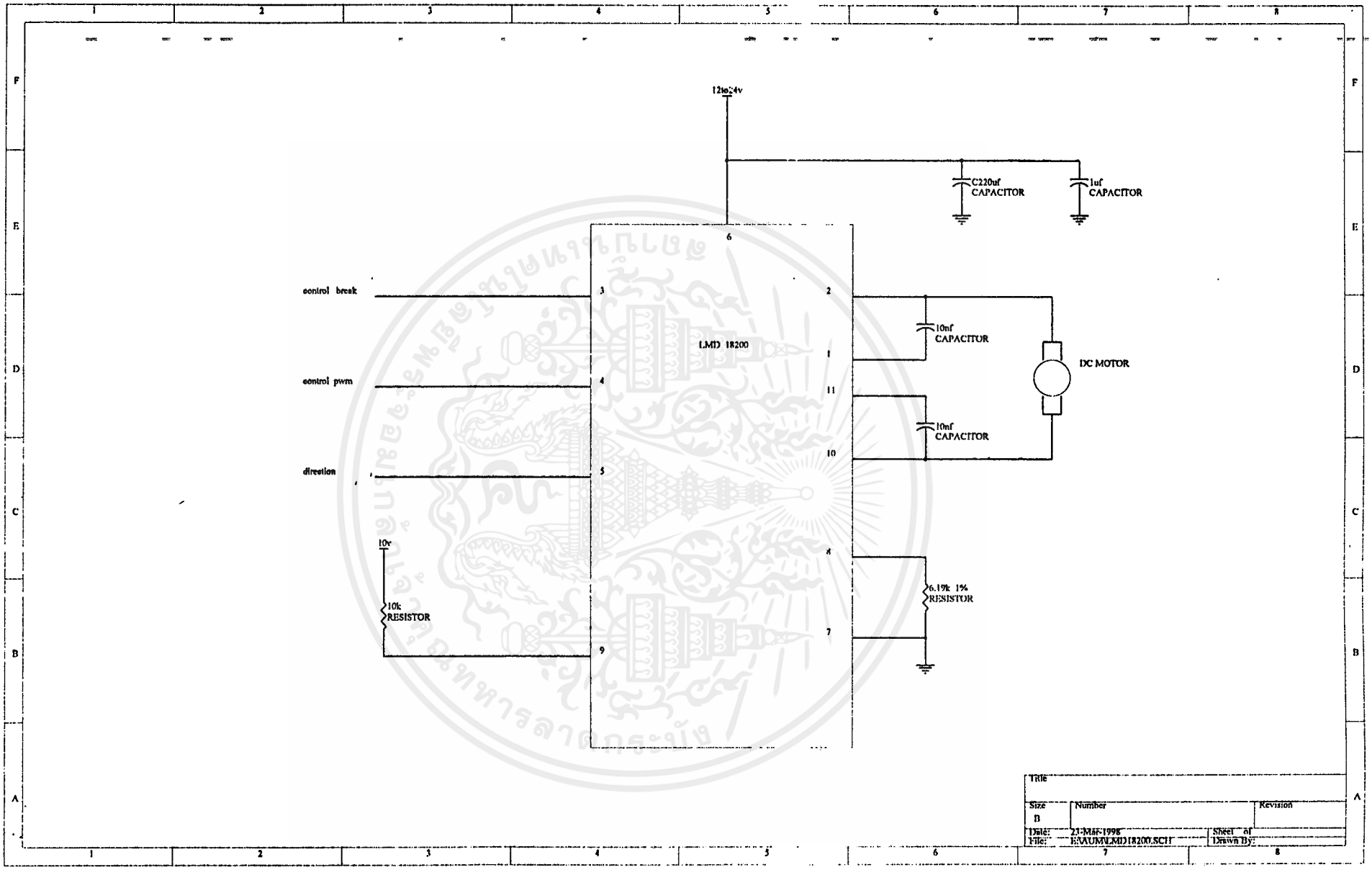

รูปที่ 7.1 รูปแสดงการต่อวงจรควบคุมสเตปปิ้งมอเตอร์




รูปที่ 7.2 รูปแสดงการต่อวงจร SUPPLY เพื่อเป็นไฟเลี้ยงคงที่ให้กับส่วน DC มอเตอร์




รูปที่ 7.3 รูปแสดงการต่อวงจรควบคุม DC MOTOR โดยใช้ IC เบอร์ LMD 18245




รูปที่ 7.4 รูปแสดงการต่อวงจรควบคุม DC MOTOR โดยใช้ IC เบอร์ LMD 18200

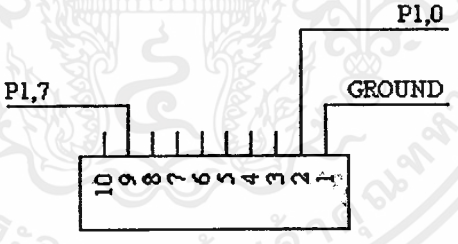
## 7.2 วงจรในส่วนมอเตอร์กระแสตรง (DC MOTOR)

ในส่วน DC MOTOR นี้จะประกอบด้วย 2 ส่วนหลักๆ ด้วยกันคือ ส่วนวงจร SUPPLY ที่ใช้เป็นตัวป้อนไฟคงที่เลี้ยงวงจร drive dc motor แสดงในรูปที่ 7.2 และ ส่วนที่สองก็คือส่วน dc motor ซึ่งส่วนใช้ dc motor เป็นส่วนควบคุมในการเคลื่อนที่นั้นจะมีอยู่ทั้งหมด 4 ส่วน คือ ส่วน base , ส่วน shoulder , ส่วน elbow ซึ่งใน 3 ส่วนนี้จะใช้ IC เบอร์ LMD 18200 ในการควบคุมการเคลื่อนที่เนื่องจากมีส่วนที่สามารถควบคุมความเร็วในการเคลื่อนที่ได้ ( ส่วน PWM) และ ส่วนสุดท้ายคือ ส่วน gripper ( ที่เป็นตัวจับวัตถุ) จะใช้ IC เบอร์ LMD 18245 ในการควบคุมการเคลื่อนที่เนื่องจากมีส่วนที่สามารถ ลิมิตกระแสได้ซึ่งจะใช้เป็นส่วนในการควบคุมการจับวัตถุโดยถ้ากระแสเกินที่เราลิมิตเอาไว้แล้วมันจะทำการลิมิตกระแสคงที่ไว้เท่ากับค่ากระแสที่ลิมิตไว้ วงจรของส่วน drive dc motor ของ IC เบอร์ LMD 18245 และ IC เบอร์ LMD 18200 แสดงไว้ในภาคผนวก และรูป ส่วนวงจรของ IC เบอร์ LMD 18245 และ LMD18200 แสดงในรูปที่ 7.3 และ 7.4 ตามลำดับ

## 7.3 ส่วนที่ใช้ในการเชื่อมต่อกับส่วนต่างๆ( CONNECTER BOARD)

จะประกอบด้วยส่วน connecter ต่าง ๆดังต่อไปนี้

### 7.3.1 ส่วน CONNECT LED




CONNECTOR LED

ขาที่ 1 เป็นขา ground

ขาที่ 2 เป็นขาติดต่อกับ led ตัวที่ 1

ขาที่ 3 เป็นขาติดต่อกับ led ตัวที่ 2

ขาที่ 4 เป็นขาติดต่อกับ led ตัวที่ 3

ขาที่ 5 เป็นขาติดต่อกับ led ตัวที่ 4

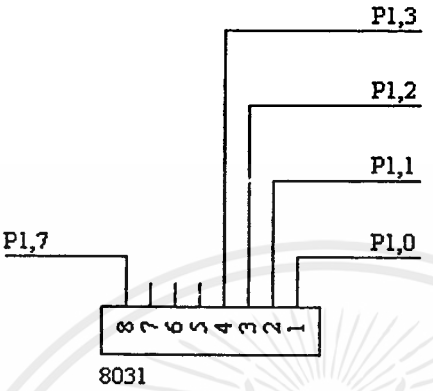
ขาที่ 6 เป็นขาติดต่อกับ led ตัวที่ 5

ขาที่ 7 เป็นขาติดต่อกับ led ตัวที่ 6

ขาที่ 8 เป็นขาติดต่อกับ led ตัวที่ 7

ขาที่ 9 เป็นขาติดต่อกับ led ตัวที่ 8

**7.3.2 ส่วน CONNECT 8031**




ส่วน connect ส่วนนี้จะนำไปต่อเข้ากับ IC เบอร์ 74LS 244 แล้วนำไปต่อเข้ากับส่วน connect led เพื่อติดต่อกับ led แต่ละตัว

ขาที่ 1 ติดต่อกับ connect led ขาที่2

ขาที่ 2 ติดต่อกับ connect led ขาที่3

ขาที่ 3 ติดต่อกับ connect led ขาที่4

ขาที่ 4 ติดต่อกับ connect led ขาที่5

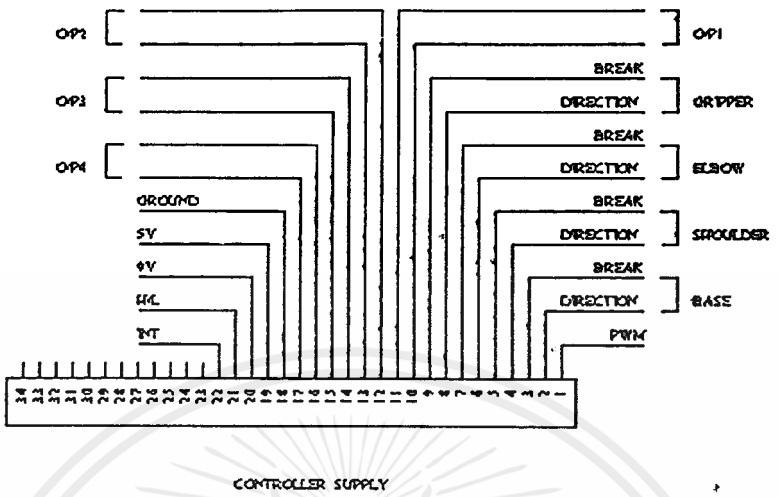
ขาที่ 5 ติดต่อกับ connect led ขาที่6

ขาที่ 6 ติดต่อกับ connect led ขาที่7

ขาที่ 7 ติดต่อกับ connect led ขาที่8

ขาที่ 8 ติดต่อกับ connect led ขาที่9

### 7.3.3 ส่วน CONNECT SUPPLY AND DC MOTOR




ขาที่ 1 เป็นขาที่ติดติดต่อกับส่วน PWM ในส่วน dc motor

ขาที่ 2 เป็นขาที่ติดต่อกับส่วน direction base

ขาที่ 3 เป็นขาที่ติดต่อกับส่วน break base

ขาที่ 4 เป็นขาที่ติดต่อกับส่วน direction shoulder

ขาที่ 5 เป็นขาที่ติดต่อกับส่วน break shoulder

ขาที่ 6 เป็นขาที่ติดต่อกับส่วน direction elbow

ขาที่ 7 เป็นขาที่ติดต่อกับส่วน break elbow

ขาที่ 8 เป็นขาที่ติดต่อกับส่วน direction gripper

ขาที่ 9 เป็นขาที่ติดต่อกับส่วน break gripper

ขาที่ 10และ 11เป็นขาที่ติดต่อกับส่วน output base ที่ต่อกับมอเตอร์ในส่วนฐาน

ขาที่ 12 และ 13 เป็นขาที่ติดต่อกับส่วน output shoulder ที่ต่อกับมอเตอร์ในส่วนหัวไหล่

ขาที่ 14และ 15 เป็นขาที่ติดต่อกับส่วน output elbow ที่ติดต่อกับมอเตอร์ในส่วนข้อศอก

ขาที่ 16 และ 17 เป็นขาที่ติดต่อกับส่วน output gripper ที่ติดต่อกับมอเตอร์ในส่วนมือจับ

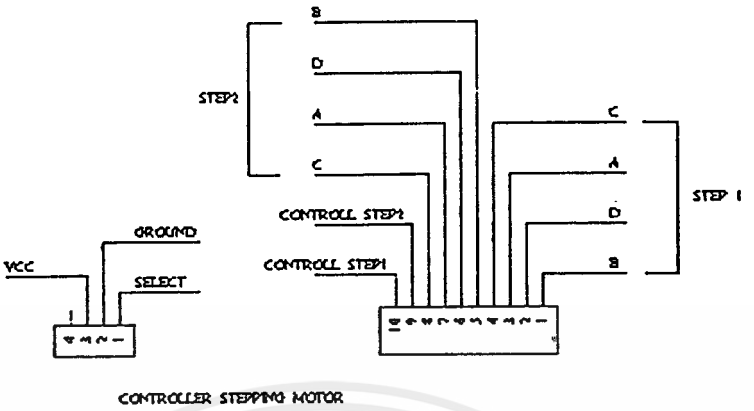
ขาที่ 18 เป็นขา ground

ขาที่ 19 เป็นขาไฟเลี้ยงคงที่ 5 v

ขาที่ 20 เป็นขาไฟเลี้ยงคงที่ 9 v

ขาที่ 21 เป็นขา H/L เป็นขาเลือก voltage ที่จะป้อนให้ส่วนควบคุม dc motor ซึ่งที่ high เท่ากับ 12 v และ ที่ low เท่ากับ 6 v

### 7.3.4 ส่วน CONNECT STEPPING MOTOR



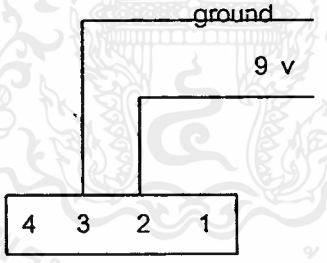

CONTROLLER STEPPING MOTOR

ขาที่ 1 ถึง 4 เป็นขา output stepping motor ในส่วนหมุนข้อมือ ซ้าย-ขวา

ขาที่ 5 ถึง 8 เป็นขา output stepping motor ในส่วนหมุนข้อมือ ขึ้น-ลง

ขาที่ 9 และ 10 เป็นขา input ที่ติดต่อกับส่วนวงจรควบคุม stepping motor ซึ่งจะรับข้อมูลเป็นลอจิก

### 7.3.5 ส่วน CONNECT SUPPLY MICROCONTROLLER




ขาที่ 2 เป็นขาไฟเลี้ยงบอร์ด ไมโครคอนโทรลเลอร์ 9 v

ขาที่ 3 เป็นขา ground

### 7.3.6 ส่วน CONNECT ROBOT ARM

| 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |

ขาที่ 1 ถึง ขาที่ 6 เป็นส่วนติดต่อกับส่วน stepping motor ที่ใช้หมุนข้อมือไปทางซ้ายและขวา

ขาที่ 7 ถึง ขาที่ 8 เป็นส่วนติดต่อกับส่วน elbow

ขาที่ 9 ถึง ขาที่ 10 เป็นส่วนติดต่อกับส่วน base

ขาที่ 11 ถึง ขาที่ 12 เป็นส่วนติดต่อกับส่วน gripper

ขาที่ 13 และขาที่ 25 เป็นส่วนติดต่อกับส่วน shoulder

ขาที่ 19 ถึง ขาที่ 24 เป็นส่วนติดต่อกับส่วน stepping motor ที่ใช้หมุนข้อมือขึ้น - ลง

### 7.3.7 ส่วน CONNECT SENSOR

| 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |

ขาที่ 1 เป็นขาไฟเลี้ยงวงจร 5 volts

ขาที่ 2 เป็นขา ground

ขาที่ 3 เป็นขา home base

ขาที่ 4 เป็นขา home shoulder

ขาที่ 5 เป็นขา home elbow

ขาที่ 6 เป็นขา home hand connector up- down

ขาที่ 7 เป็นขา home hand connector left-right

ขาที่ 8 เป็นขา pulse base

ขาที่ 9 เป็นขา pulse shoulder

ขาที่ 10 เป็นขา pulse elbow

# บทที่ 8

## ผลการทดลอง

### 8.1 ด้าน mechanic

ส่วนของฐานแขนกลสามารถหมุนได้ 360 องศาและสามารถรับน้ำหนักของแขนกลทั้งท่อนได้เนื่องจากมอเตอร์กระแสตรงที่ใช้มีกำลังในการขับพอที่จะทำให้ส่วนฐานสามารถที่จะหมุนได้ และเนื่องจากมีการใช้เฟืองเป็นตัวทดซึ่งที่ส่วนฐานจะใช้เฟืองซึ่งมีขนาดใหญ่และทดกำลังจากเฟืองซึ่งมีซี่เฟืองน้อยไปยังซี่เฟืองมากทำให้มีกำลังในการหมุนเพียงพอ

ส่วนของหัวไหล่สามารถเคลื่อนที่ขึ้นลงได้ตามปกติโดยไม่ติดขัด โดยการใช้มอเตอร์กระแสตรงในการขับให้ส่วนหัวไหล่สามารถเคลื่อนที่ได้และมีแรงพอที่จะรับโหลดในท่อนบนได้และใช้เฟืองเป็นตัวทดกำลังเช่นเดียวกัน

ส่วนของ ข้อศอกสามารถเคลื่อนที่ขึ้นลงได้แต่ถ้าเคลื่อนลงต่ำมากจะไม่สามารถยกวัตถุขึ้นได้ เนื่องจากเฟืองที่ใช้ทดกำลังมีแรงไม่พอทำให้เมื่อยกแขนขึ้นจะเกิดการ slip ของซี่สายพาน ซึ่งควรจะหาสายพานที่มีขนาดใหญ่มาเปลี่ยนหรือออกแบบให้แขนท่อนบนมีน้ำหนักที่น้อยลง

ส่วนข้อมือในส่วนนี้จะใช้ stepping motor ในการหมุนข้อมือไปทางซ้ายและขวาและหมุนข้อมือขึ้นลงซึ่งจากการทดลองก็สามารถหมุนได้แต่จะมีการติดขัดในบางจังหวะเนื่องจากการออกแบบทางด้าน mechanic และกำลังของสเตปปิ้งมอเตอร์ซึ่งยังน้อยอยู่ไม่สามารถที่จะหมุนได้อย่างต่อเนื่อง

ส่วนมือจับ สามารถจับวัตถุได้ตามต้องการโดยสามารถอ้ามือจับและบีบจับวัตถุได้ โดยในส่วนนี้ได้ใช้ มอเตอร์กระแสตรงในการควบคุมการเคลื่อนที่โดยใช้ ic เบอร์ LMD 18245 เป็นตัวควบคุมในการกำจัดกระแสเพื่อใช้ในการบีบจับวัตถุ

### 8.2 ด้าน HARDWARE

ประกอบด้วย 3 ส่วนคือ

1.ส่วนข้อมือใช้สเตปปิ้งมอเตอร์ควบคุมในการหมุนข้อมือไปทางซ้ายและขวาและหมุนข้อมือขึ้นลง

จากการทดลองในส่วนนี้ก็พบว่าสามารถหมุนข้อมือได้ตามปกติแต่ก็มีติดขัดบ้างซึ่งไม่ได้เกิดจากส่วนวงจรสเตปปิ้งมอเตอร์และวงจรนี้ก็สามารถทำงานได้ดีตามต้องการโดยเมื่อป้อนลอจิกเข้าไปตามที่อธิบายไว้ในบทที่ 7 ก็พบว่าสามารถให้ผลลัพธ์ที่ถูกต้องคือสามารถขับให้มอเตอร์หมุนไปในทิศทางที่ถูกต้องตามที่ได้คำนวณไว้

2.ส่วนมือจับใช้มอเตอร์กระแสตรงและ IC เบอร์ LMD 18245 ในการควบคุมการเคลื่อนที่ของส่วนมือจับและการจับวัตถุ

จากการทดลองวงจรซึ่งใช้ LMD 18245 ในการควบคุมการบีบจับวัตถุนั้นสามารถทำให้มือจับสามารถทำการจับวัตถุได้เป็นอย่างดีสามารถทำงานตามที่โปรแกรมไว้ได้อย่างถูกต้อง

3.ส่วนที่เคลื่อนที่โดยใช้ IC เบอร์ LMD 18200ในการควบคุมการเคลื่อนที่

ซึ่งส่วนที่ใช้ควบคุมได้แก่ส่วนฐาน ส่วนหัวไหล่ ส่วนข้อศอก ซึ่งวงจรที่ใช้ LMD 18200 ในการควบคุมการเคลื่อนที่ของมอเตอร์ได้ผลตามต้องการคือสามารถบังคับให้มอเตอร์หมุนในทิศทางที่ต้องการได้ตามคำสั่งไม่มีปัญหาใดๆ

**8.3 ด้าน SOFTWARE**

ได้ใช้โปรแกรมการควบคุมโดยใช้โปรแกรมแอสแซมบลี้และโปรแกรมเคลไฟล์ในการควบคุมแขนกลให้ทำงานตามต้องการ โดยในขณะทำการทดลองควบคุมแขนกลนั้นพบว่าไม่มีข้อมผิดพลาดเกิดขึ้นเมื่อทำการสั่งก็สามารถเคลื่อนที่ในส่วนต่างๆ ได้ตามคำสั่งที่สั่งลงไป แต่เนื่องจากยังมีประสบการณ์ในการเขียนโปรแกรมทางด้านนี้น้อยจึงยังไม่สามารถเขียนโปรแกรมในลักษณะของ full duplex ได้

# บทที่ 9

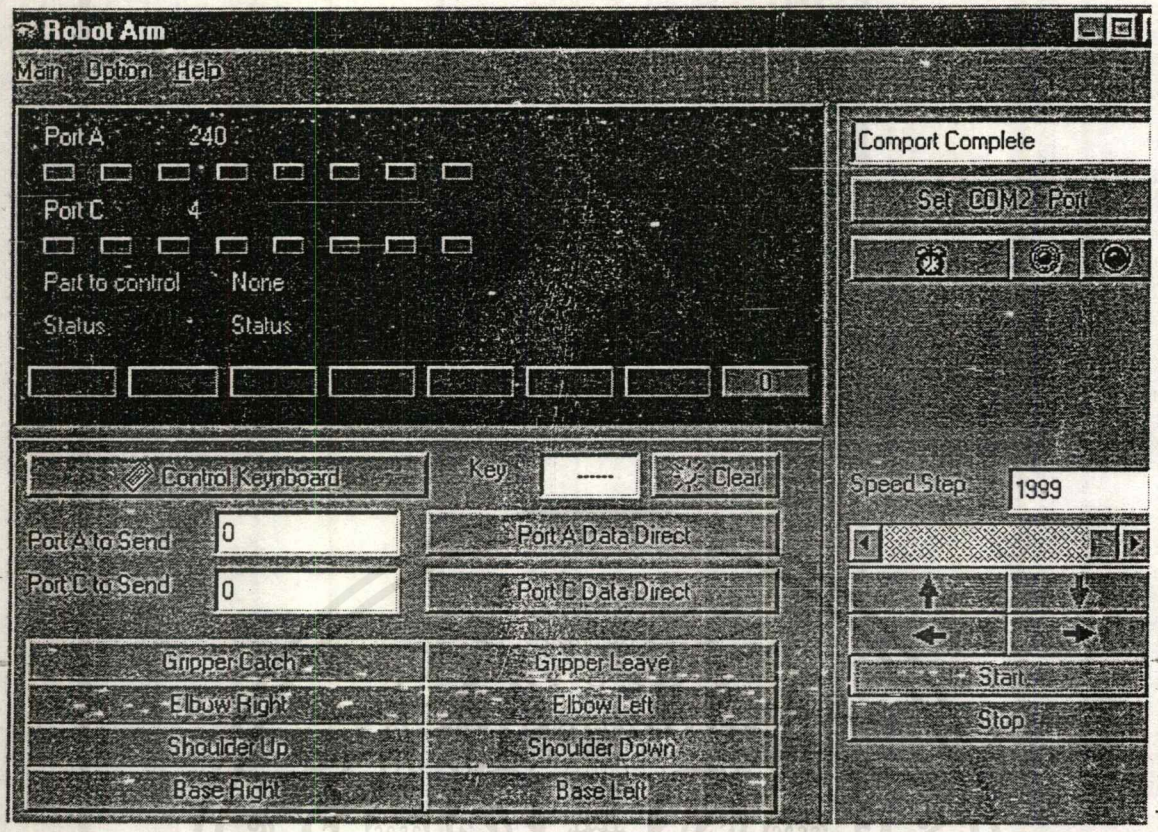
# สรุปและวิจารณ์

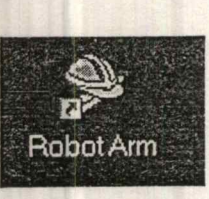
การทำปริญญานิพนธ์ในครั้งนี้นั้นก็เพื่อต้องการที่จะศึกษาการทำงานของแขนกลเบื้องต้นซึ่งสามารถนำความรู้ที่ได้จากการศึกษานี้เป็นใช้ในการประยุกต์ใช้กับการควบคุมการทำงานของเครื่องจักรกลที่มีลักษณะคล้ายกันได้บ้างโดยนำความรู้ในด้านการส่งข้อมูลทั้งแบบอนุกรมและขนานได้ ซึ่งเป็นการเขียนโปรแกรมในรูปของการ interface กับ hardware และการควบคุมการขับเคลื่อนของทั้งสเตปปิ้งมอเตอร์และมอเตอร์กระแสตรงได้โดยใช้ความสามารถของ IC เบอร์ LMD18245 และ 18200 ซึ่งจากการศึกษาในครั้งนี้ค้นพบว่าควรทำการป้อนกลับข้อมูลเพื่อควบคุมตำแหน่งในการจับวัตถุได้ถูกต้องและแม่นยำมากขึ้น

วิจารณ์

เนื่องจากในปัจจุบันนี้ความเจริญก้าวหน้าทางเทคโนโลยีได้มีการพัฒนาอย่างรวดเร็วซึ่งจะเห็นว่าในปัจจุบันนี้นั้นการใช้หุ่นยนต์เข้ามาทำงานแทนมนุษย์นั้นมีมากขึ้น ฉะนั้นกลุ่มของข้าพเจ้าจึงคิดว่าในการศึกษาการควบคุมการทำงานของแขนกลนั้นคงจะพอเป็นประโยชน์กับผู้ที่สนใจศึกษาในด้านนี้บ้างไม่มากก็น้อย

# ภาคผนวก ก






```
unit Urobot;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, ExtCtrls, Menus, Buttons;

type
 Tmain = class(TForm)
  MainMenu1: TMainMenu;
  Main1: TMenuItem;
  Exit1: TMenuItem;
  Option1: TMenuItem;
  Testport1: TMenuItem;
  Help1: TMenuItem;
  Panel1: TPanel;
  monitor: TPanel;
  Label1: TLabel;
  a7: TPanel;
  a6: TPanel;
  a5: TPanel;
  a4: TPanel;
  a3: TPanel;
  a2: TPanel;
  a1: TPanel;
  a0: TPanel;
  Label2: TLabel;
  c7: TPanel;
  c6: TPanel;
  c5: TPanel;
  c4: TPanel;
  c3: TPanel;
  c2: TPanel;
```

```
            Application.MessageBox('GetCommTimeOuts','Error Initialize',MB_OK);
            halt;
        end;
        with CommTimeOuts do
        begin
        WriteTotalTimeoutMultiplier:=10;
        WriteTotalTimeoutConstant:=35;
        end;
        If SetCommTimeouts(Comport,CommTimeouts)=False then
        begin
            MessageBeep(MB_ICONEXCLAMATION);
            Application.MessageBox('SetCommTimeOuts','Error Initialize',MB_OK);
        end;
       i:=inbuffer[1];
       Edit1.text:='Comport Complete';
       send;
       display_a;
       display_c;
end;
procedure Tmain.Outserial(COM:THandle;Data:DWORD);
const dwtowrite:DWORD=5;
var  dwwritten:DWORD;
   bcheck  :BOOL;

begin   outbuffer[1]:=data;
       dwwritten:=0;
       bcheck:=WriteFile(COM,outbuffer,1,dwwritten,nil);
       IF (bCheck=False) then
       begin
        MessageBeep(MB_ICONEXCLAMATION);
        Application.MessageBox('Can''t Out to Comport','Error',MB_OK);
       end;
```

c1: TPanel;
c0: TPanel;
Edit1: TEdit;
Button1: TButton;
data_a: TLabel;
data_c: TLabel;
Label4: TLabel;
status: TLabel;
dattime7: TPanel;
dattime6: TPanel;
dattime5: TPanel;
dattime4: TPanel;
dattime3: TPanel;
dattime2: TPanel;
dattime1: TPanel;
dattime0: TPanel;
Panel27: TPanel;
BitBtn1: TBitBtn;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
Button5: TButton;
Button6: TButton;
BitBtn4: TBitBtn;
BitBtn5: TBitBtn;
speedstep: TScrollBar;
BitBtn6: TBitBtn;
BitBtn7: TBitBtn;
Edit2: TEdit;
Label5: TLabel;
Timer1: TTimer;
BitBtn8: TBitBtn;
BitBtn9: TBitBtn;
Edit3: TEdit;
BitBtn10: TBitBtn;

```
    Button7: TButton;
    Button8: TButton;
    directa: TEdit;
    Label6: TLabel;
    Label7: TLabel;
    Button9: TButton;
    Button10: TButton;
    Button11: TButton;
    Button12: TButton;
    Button13: TButton;
    Button14: TButton;
    Label3: TLabel;
    Part: TLabel;
    Label8: TLabel;
    directc: TEdit;
    Button2: TButton;
    procedure Outserial(COM:THandle;Data:DWORD);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure BitBtn1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn8KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure BitBtn8KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure BitBtn9Click(Sender: TObject);
    procedure BitBtn10Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button7MouseDown(Sender: TObject; Button: TMouseButton;
```

```
    Shift: TShiftState; X, Y: Integer);
  procedure Button9MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button11MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button13MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button8MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button10MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button12MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button14MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button7MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button9MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button11MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button13MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button8MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button10MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button12MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button14MouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Button5Click(Sender: TObject);
  procedure Button6Click(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
```

```
    procedure speedstepChange(Sender: TObject);
    procedure BitBtn6MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn7MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn4MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn5MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn6MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn7MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn4MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure BitBtn5MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  main: Tmain;
  Comport:THandle;

implementation
VAR
  ComDCB:TDCB;
  Comname:PChar;
  Commtimeouts:TCommtimeOuts;
  outbuffer:array [1..5]of integer;
  inbuffer:array[1..5]of integer;
```

```
  porta,portc:integer;
  temp:integer;
  time:integer;
  LEDon,LEDoff:integer;
  lighton:integer;
  delaytime:integer;
  tempstep:integer;
{$R *.DFM}
procedure Display_a;
begin   temp:= porta and $01;
        if temp<>0 then main.a0.color:=LEDon
          else main.a0.color:=LEDoff;
        temp:= porta and $02;
        if temp<> 0 then main.a1.color:=LEDon
          else main.a1.color:=LEDoff;
        temp:= porta and $04;
        if temp<> 0 then main.a2.color:=LEDon
          else main.a2.color:=LEDoff;
        temp:= porta and $08;
        if temp<> 0 then main.a3.color:=LEDon
          else main.a3.color:=LEDoff;
        temp:= porta and $10;
        if temp<> 0 then main.a4.color:=LEDon
          else main.a4.color:=LEDoff;
        temp:= porta and $20;
        if temp<> 0 then main.a5.color:=LEDon
          else main.a5.color:=LEDoff;
        temp:= porta and $40;
        if temp<> 0 then main.a6.color:=LEDon
          else main.a6.color:=LEDoff;
        temp:= porta and $80;
        if temp<> 0 then main.a7.color:=LEDon
          else main.a7.color:=LEDoff;
        main.data_a.caption:=inttostr(porta);
```

```
end;
procedure Display_c;
begin    temp:= portc and $01;
         if temp<> 0 then main.c0.color:=LEDon
          else main.c0.color:=LEDoff;
         temp:= portc and $02;
         if temp<> 0 then main.c1.color:=LEDon
          else main.c1.color:=LEDoff;
         temp:= portc and $04;
         if temp<> 0 then main.c2.color:=LEDon
          else main.c2.color:=LEDoff;
         temp:= portc and $08;
         if temp<> 0 then main.c3.color:=LEDon
          else main.c3.color:=LEDoff;
         temp:= portc and $10;
         if temp<> 0 then main.c4.color:=LEDon
          else main.c4.color:=LEDoff;
         temp:= portc and $20;
         if temp<> 0 then main.c5.color:=LEDon
          else main.c5.color:=LEDoff;
         temp:= portc and $40;
         if temp<> 0 then main.c6.color:=LEDon
          else main.c6.color:=LEDoff;
         temp:= portc and $80;
         if temp<> 0 then main.c7.color:=LEDon
          else main.c7.color:=LEDoff;
         main.data_c.caption:=inttostr(portc);
end;

procedure send;
begin   main.Outserial(comport,porta);
      display_a;
      main.outserial(comport,portc);
      display_c;
```

```
end;
procedure forw;
begin   main.dattime0.caption:='0';
        main.dattime1.caption:='1';
        main.dattime2.caption:='2';
        main.dattime3.caption:='3';
        main.dattime4.caption:='0';
        main.dattime5.caption:='1';
        main.dattime6.caption:='2';
        main.dattime7.caption:='3';
end;
procedure rew;
begin   main.dattime0.caption:='3';
        main.dattime1.caption:='2';
        main.dattime2.caption:='1';
        main.dattime3.caption:='0';
        main.dattime4.caption:='3';
        main.dattime5.caption:='2';
        main.dattime6.caption:='1';
        main.dattime7.caption:='0';
end;
procedure clrporta;
begin     porta:=$F0; //break all dc motor
          send;
          main.part.caption:=' None ';
          main.Status.caption:=' Stop ';
end;
procedure Tmain.Button1Click(Sender: TObject);
var i:integer;
   bresult:BOOL;
   ByteRead:DWORD;
   lap:array [1..5]of integer;
```

```
begin for i:=1 to 5 do
    begin
     outbuffer[i]:=0;
    end;
    Comport:=CreateFile(Comname,GENERIC_WRITE ,FILE_SHARE_WRITE,nil,
                    OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0);
    If Comport<=0 then
    begin
     MessageBeep(MB_ICONEXCLAMATION);
     Application.MessageBox('CreateFile','Error Initialize',MB_OK);
     halt;
    end;
    If GetCommState(Comport,ComDCB)=False then
    begin
      MessageBeep(MB_ICONEXCLAMATION);
      Application.MessageBox('GetCommState','Error initialize',MB_OK);
      halt;
    end;
    with ComDCB do
    begin
     BaudRate:=CBR_9600;
     ByteSize:=8;
     StopBits:=0; // 1 STOP BIT
     Parity:=0; //no parity
    end;
    If SetCommState(Comport,ComDCB)=False then
    begin
       MessageBeep(MB_ICONEXCLAMATION);
       Application.MessageBox('SetCommstate','Error Initialize',MB_OK);
       halt;
     end;
    If GetCommTimeOuts(Comport,CommTimeOuts)=False then
    begin
       MessageBeep(MB_ICONEXCLAMATION);
```

```
end;
procedure Tmain.FormCreate(Sender: TObject);
var i:integer;
begin    Comname:='COM2';
        porta:=$F0;
        portc:=$14;//SET HIGH VOLTAGE PWM=1
        LEDon:=$0000FF;
        LEDoff:=$00404080;
        a0.color:=Ledoff;
        a1.color:=ledoff;
        a2.color:=ledoff;
        a3.color:=ledoff;
        a4.color:=ledoff;
        a5.color:=ledoff;
        a6.color:=ledoff;
        a7.color:=ledoff;
        c0.color:=ledoff;
        c1.color:=ledoff;
        c2.color:=ledoff;
        c3.color:=ledoff;
        c4.color:=ledoff;
        c5.color:=ledoff;
        c6.color:=ledoff;
        c7.color:=ledoff;
        timer1.enabled:=false;
        time:=$01;
        delaytime:=1;


end;

procedure Tmain.Exit1Click(Sender: TObject);
begin    close;
```

```
end;

procedure Tmain.BitBtn1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   portc:=portc or $20;
        send;

end;

procedure Tmain.BitBtn1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   portc:=portc and $DF;
        send;

end;

procedure Tmain.BitBtn2Click(Sender: TObject);
begin   portc:=portc or $10;
      send;

end;

procedure Tmain.BitBtn3Click(Sender: TObject);
begin  portc:=portc and $EF;
      send;

end;

procedure Tmain.BitBtn8KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var numkey:integer;
begin      numkey:=ord(key);
          Edit3.text:=inttostr(numkey);
          case (numkey) of
```

```
97,65: begin //key a
    porta:=porta and $77;
    send;
    Part.caption:=' Gripper ';
    Status.caption:=' Catch ';
    end;
115,83:begin //key s
    porta:=porta and $BB;
    send;
    Part.caption:=' Elbow ';
    Status.caption:= ' Right ';
    end;
100,68:begin //key d
    porta:=porta and $DD;
    send;
    Part.caption:='Shoulder';
    Status.caption:=' Up ';
    end;
102,70:begin //key f
    porta:=porta and $EE;
    send;
    Part.caption:=' Base ';
    status.caption:=' Right ';
    end;
106,74:begin //key j
    porta:=porta and $7F;
    porta:=porta or $08;
    send;
    Part.caption:=' Gripper ';
    status.caption:=' Leave ';
    end;
107,75:begin //key k
    porta:=porta and $BF;
    porta:=porta or $04;
```

```
                send;
                Part.caption:=' Elbow';
                status.caption:=' Left ';
                end;
            108,76:begin //key l
                porta:=porta and $DF;
                porta:=porta or $02;
                send;
                part.caption:='Shoulder';
                status.caption:='Down';
                end;
            59,186 :begin //key ;
                porta:=porta and $EF;
                porta:=porta or $01;
                send;
                part.caption:='Base';
                status.caption:='Left';
                end;
            103,71:begin  // key g
                timer1.enabled:=true;
                end;
            end;//case


end;

procedure Tmain.BitBtn8KeyUp(Sender: TObject; var Key: Word;
 Shift: TShiftState);
begin     porta:=$F0; //break all dc motor
          send;
          part.caption:=' None ';
          Status.caption:=' Stop ';
end;
```

```
procedure Tmain.BitBtn9Click(Sender: TObject);
begin   porta:=$F0;
        send;

end;

procedure Tmain.BitBtn10Click(Sender: TObject);
begin   porta:=strtoint(directa.text);
        send;

end;

procedure Tmain.Button2Click(Sender: TObject);
begin   portc:=strtoint(directc.text);
        send;

end;

procedure Tmain.Button7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   porta:=porta and $77;
        send;
        Part.caption:=' Gripper ';
        Status.caption:=' Catch ';

end;

procedure Tmain.Button9MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   porta:=porta and $BB;
        send;
        Part.caption:=' Elbow ';
        Status.caption:= ' Right ';
```

```
end;

procedure Tmain.Button11MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin    porta:=porta and $DD;
        send;
        Part.caption:='Shoulder';
        Status.caption:=' Up ';

end;

procedure Tmain.Button13MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   porta:=porta and $EE;
     send;
       Part.caption:=' Base ';
       status.caption:=' Right ';

end;

procedure Tmain.Button8MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   porta:=porta and $7F;
       porta:=porta or $08;
       send;
       Part.caption:=' Gripper ';
       status.caption:=' Leave ';

end;

procedure Tmain.Button10MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin    porta:=porta and $BF;
        porta:=porta or $04;
```

```
        send;
        Part.caption:=' Elbow';
        status.caption:=' Left ';
    end;

    procedure Tmain.Button12MouseDown(Sender: TObject; Button: TMouseButton;
     Shift: TShiftState; X, Y: Integer);
    begin  porta:=porta and $DF;
        porta:=porta or $02;
        send;
        part.caption:='Shoulder';
        status.caption:='Down';

    end;

    procedure Tmain.Button14MouseDown(Sender: TObject; Button: TMouseButton;
     Shift: TShiftState; X, Y: Integer);
    begin  porta:=porta and $EF;
        porta:=porta or $01;
        send;
        part.caption:='Base';
        status.caption:='Left';

    end;

    procedure Tmain.Button7MouseUp(Sender: TObject; Button: TMouseButton;
     Shift: TShiftState; X, Y: Integer);
    begin  clrporta;

    end;

    procedure Tmain.Button9MouseUp(Sender: TObject; Button: TMouseButton;
     Shift: TShiftState; X, Y: Integer);
    begin  clrporta;
```

```
end;

procedure Tmain.Button11MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   clrporta;

end;

procedure Tmain.Button13MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   clrporta;

end;

procedure Tmain.Button8MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin     clrporta;

end;

procedure Tmain.Button10MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   clrporta;

end;

procedure Tmain.Button12MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   clrporta;

end;

procedure Tmain.Button14MouseUp(Sender: TObject; Button: TMouseButton;
```

```
  Shift: TShiftState; X, Y: Integer);
begin   clrporta;

end;

procedure Tmain.Button5Click(Sender: TObject);
begin    timer1.enabled:=true;
        portc:=portc and $EF; //stop high voltage
                        //choose low voltage for step
end;

procedure Tmain.Button6Click(Sender: TObject);
begin  timer1.enabled:=false;

end;

procedure Tmain.Timer1Timer(Sender: TObject);
begin    case time of
        1:time:=2;
        2:time:=3;
        3:time:=4;
        4:time:=5;
        5:time:=6;
        6:time:=7;
        7:time:=8;
        8:time:=1;
        end;//case
        case time of
        1:begin tempstep:=strtoint(dattime0.caption);
            dattime0.color:=$00d7a2;
            dattime7.color:=clgreen;
         end;
        2:begin tempstep:=strtoint(dattime1.caption);
            dattime1.color:=$00d7a2;
```

```
                dattime0.color:=clgreen;
              end;
            3:begin tempstep:=strtoint(dattime2.caption);
                dattime2.color:=$00d7a2;
                dattime1.color:=clgreen;
              end;
            4:begin tempstep:=strtoint(dattime3.caption);
                dattime3.color:=$00d7a2;
                dattime2.color:=clgreen;
              end;
            5:begin tempstep:=strtoint(dattime4.caption);
                dattime4.color:=$00d7a2;
                dattime3.color:=clgreen;
              end;
            6:begin tempstep:=strtoint(dattime5.caption);
                dattime5.color:=$00d7a2;
                dattime4.color:=clgreen;
              end;
            7:begin tempstep:=strtoint(dattime6.caption);
                dattime6.color:=$00d7a2;
                dattime5.color:=clgreen;
              end;
            8:begin tempstep:=strtoint(dattime7.caption);
                dattime7.color:=$00d7a2;
                dattime6.color:=clgreen;
              end;
            end;
          portc:=portc and $FC;
          portc:=portc or  tempstep;
          send;


end;
```

```
procedure Tmain.speedstepChange(Sender: TObject);
begin   timer1.interval:=2000-speedstep.position;
        Edit2.text:=inttostr(speedstep.position);
end;

procedure Tmain.BitBtn6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin   forw;    //set direction up
        portc:=portc or $08; //choose step motor1
        portc:=portc or $40; //enable voltage step

end;

procedure Tmain.BitBtn7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin     rew; //    set direction down
          portc:=portc or $08;   //choose step motor1
          portc:=portc or $40;   //enable voltage step
end;

procedure Tmain.BitBtn4MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin     forw;          //set direction
          portc:=portc and $F7;//choose step step2
          portc:=portc or $40; //enable step voltage
end;

procedure Tmain.BitBtn5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin     rew;       // set direction right
          portc:=portc and $F7;    //choose step motor2
          portc:=portc or $40; //enable step voltage
end;
```

```
procedure Tmain.BitBtn6MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin     portc:=portc and $BF;  //disable stepping motor voltage


end;

procedure Tmain.BitBtn7MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin    portc:=portc and $BF;

end;

procedure Tmain.BitBtn4MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin      portc:=portc and $BF;

end;

procedure Tmain.BitBtn5MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin      portc:=portc and $BF;

end;

end.
```

# ภาคผนวก ข

## Absolute Maximum Ratings (Note 1)

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

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

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

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

## Operating Conditions (Note 1)

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

## Electrical Characteristics

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

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

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

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

## Electrical Characteristics

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

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

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

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

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

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

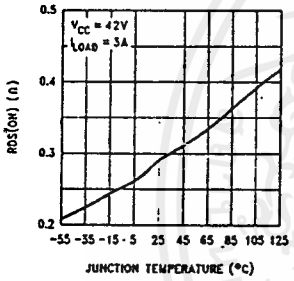
**Note 6:** Asymmetric turn OFF and ON delay times and switching times ensure a switch turns OFF before the other switch in the same half H-bridge begins to turn ON (preventing momentary short circuits between the power supply and ground). The transitional period during which both switches are OFF is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

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

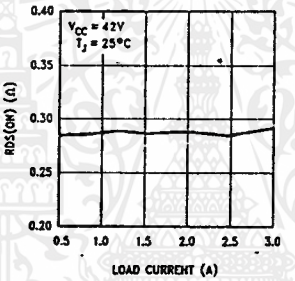
**Note 8:** Turn OFF delay, $t_{DELAY}$, is defined as the time from the voltage at the output of the current sense amplifier reaching the DAC output voltage to the lower DMOS switch beginning to turn OFF. With $V_{CC}$ = 32V, DIRECTION high, and 200Ω connected between OUT1 and $V_{CC}$, the voltage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

## Typical Performance Characteristics

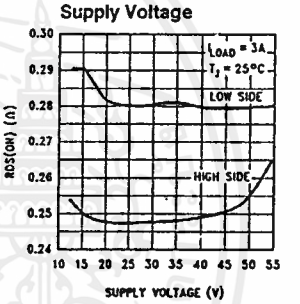
RDS(ON) vs Temperature



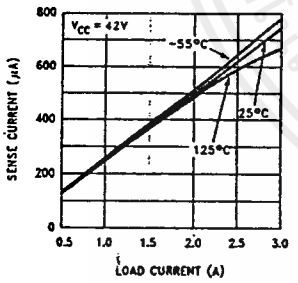

RDS(ON) vs Load Current



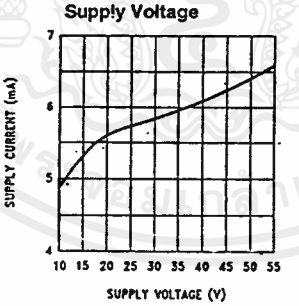

RDS(ON) vs Supply Voltage



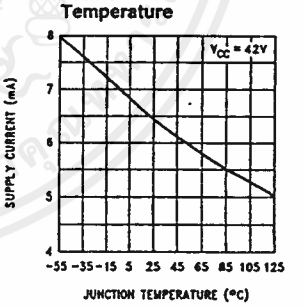

Current Sense Output vs Load Current




Supply Current vs Supply Voltage

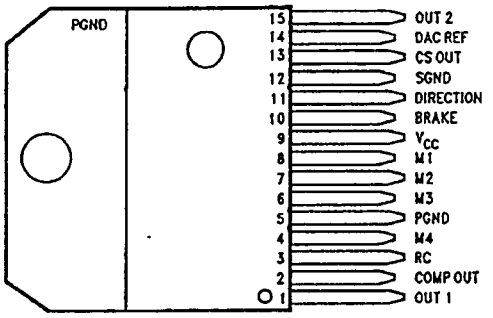



Supply Current vs Temperature




TL/H/11878–27

## Connection Diagram




**Top View**

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

## Pinout Descriptions (See Functional Block and Connection Diagrams)

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

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

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

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

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

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

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

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

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

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

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

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

**TABLE I. Switch Control Logic Truth Table**

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

X = don't care

MONO is the output of the monostable.

## Functional Descriptions

### TYPICAL OPERATION OF A CHOPPER AMPLIFIER

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

## Functional Descriptions (Continued)

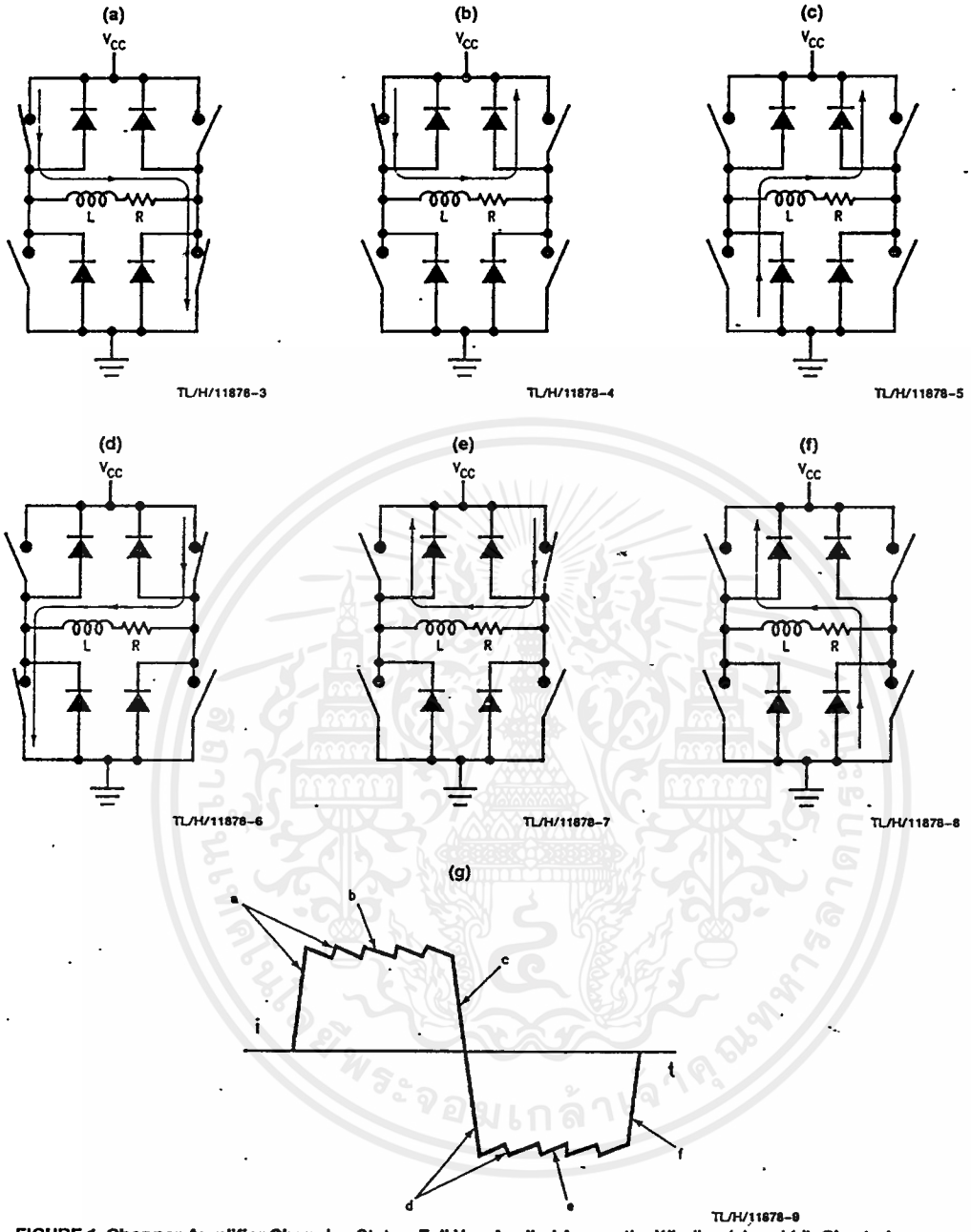



**FIGURE 1. Chopper Amplifier Chopping States: Full $V_{CC}$ Applied Across the Winding (a) and (d), Shorted Winding (b) and (e), Winding Current Decays During a Change in the Direction of the Winding Current (c) and (f), and the Chopped Winding Current (g)**

## Functional Descriptions (Continued)

### THE LMD18245 CHOPPER AMPLIFIER

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

### THE H-BRIDGE

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

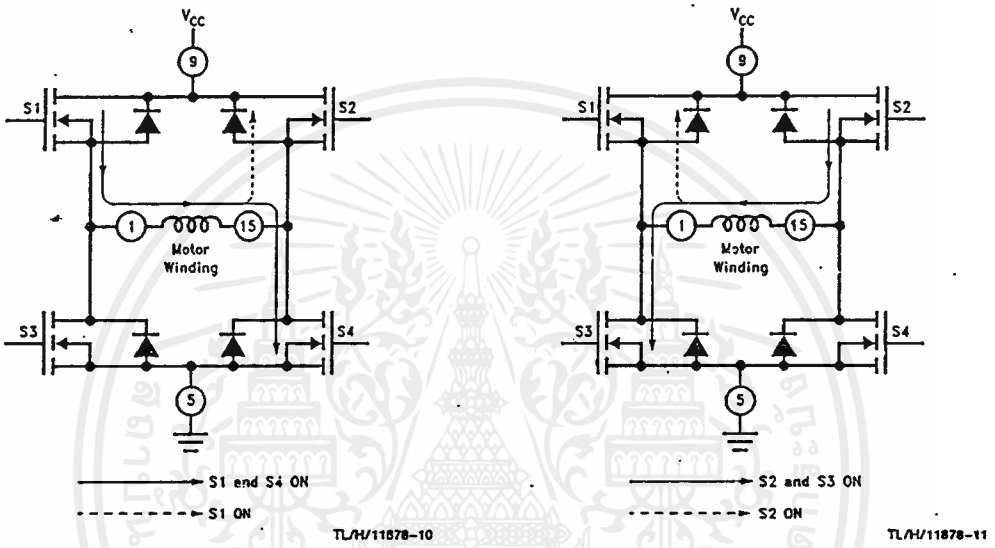



**FIGURE 2. The DMOS H-Bridge**

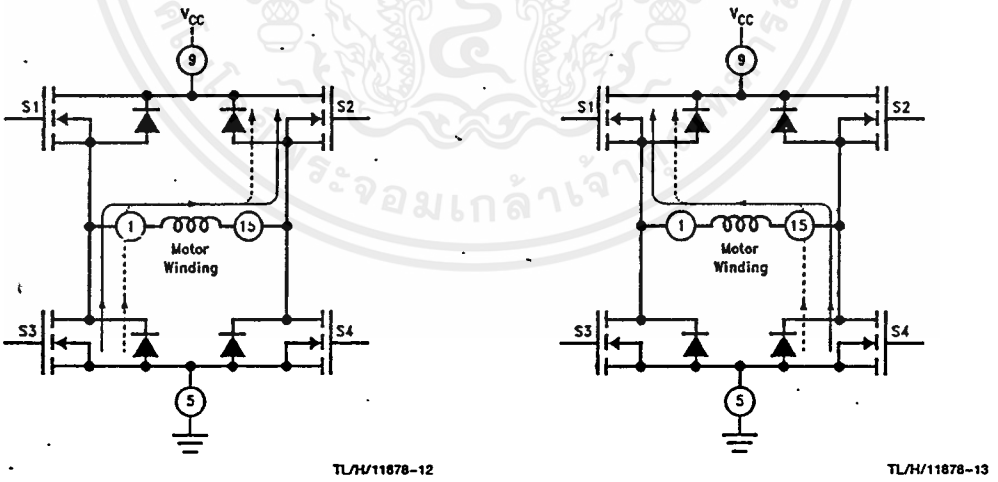



**FIGURE 3. Decay Paths for Initial Winding Current During a Change in the Direction of the Winding Current**

## Applications Information

### POWER SUPPLY BYPASSING

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

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

### OVERCURRENT PROTECTION

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

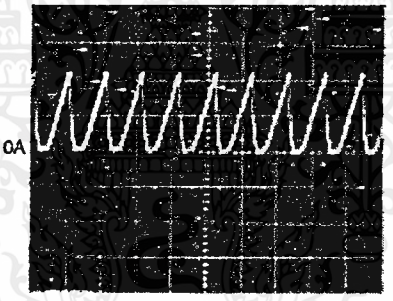
In the case of a locked rotor, the inductance of the winding tends to limit the rate of change of the fault current to a value easily handled by the protection circuitry. In the case of a low inductance short from either output to ground or between outputs, the fault current could surge past the 12A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

### THERMAL SHUTDOWN

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

### UNDERVOLTAGE LOCKOUT

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




TL/H/11878–15

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

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

## The Typical Application

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

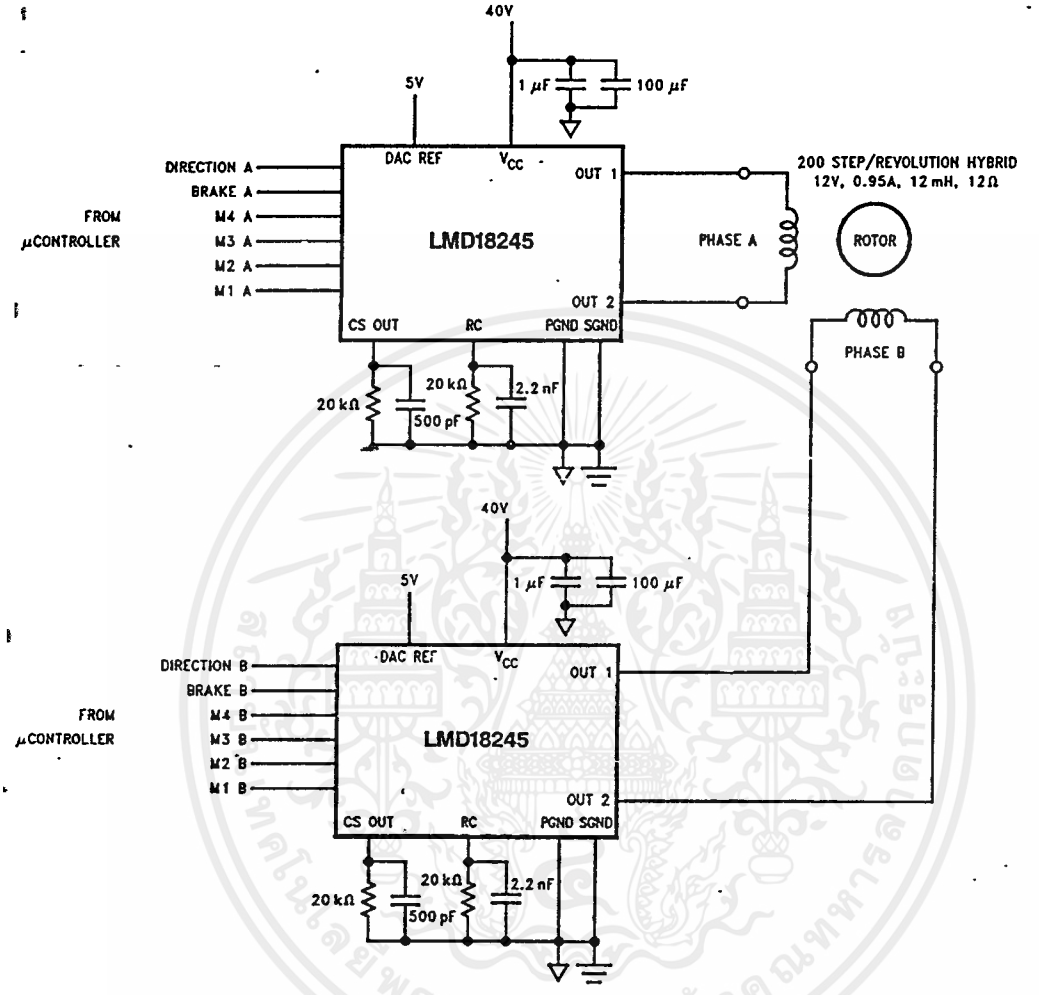



FIGURE 6. Typical Application Circuit for Driving Bipolar Stepper Motors

## The Typical Application (Continued)

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

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

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

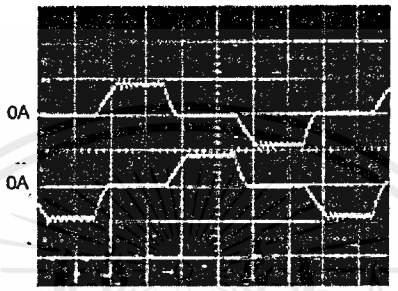
where A represents winding A energized with current in one direction and A* represents winding A energized with current in the opposite direction. The motor takes one full step each time one winding is de-energized and the other is energized. To make the motor step in the opposite direction, the order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

### TWO-PHASE-ON FULL STEP DRIVE

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

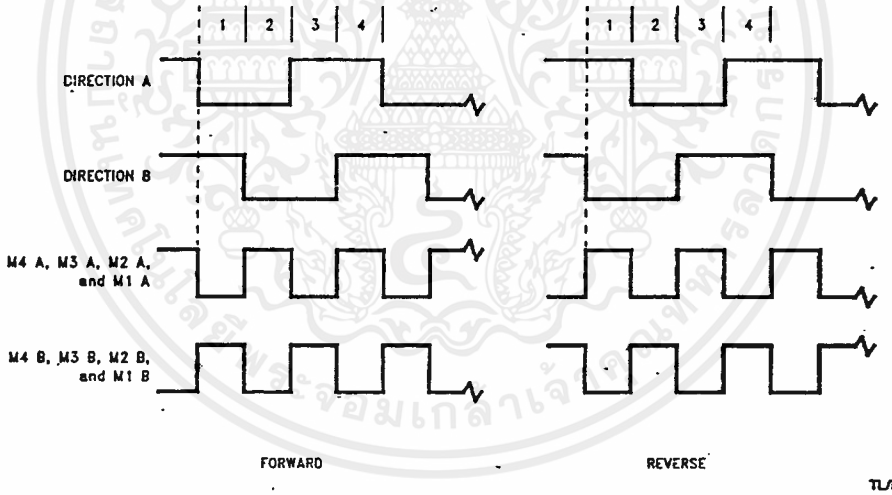
$$AB \rightarrow A^*B \rightarrow A^*B^* \rightarrow AB^* \rightarrow AB \rightarrow \ldots,$$

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




TL/H/11678–17

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

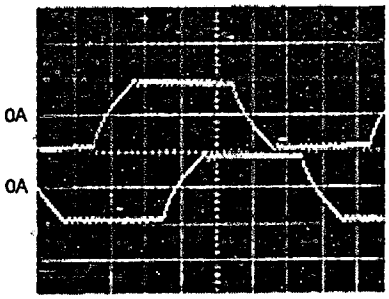



TL/H/11678–18

BRAKE A = BRAKE B = 0

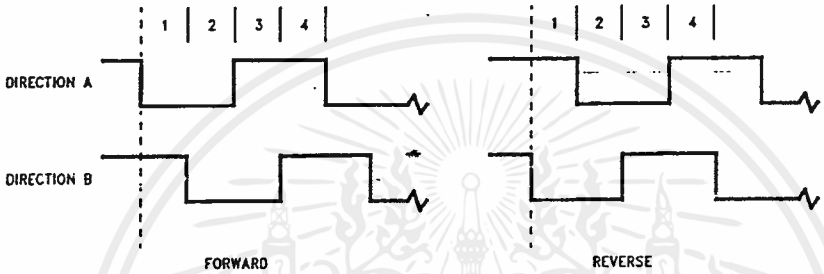
**FIGURE 7. Winding Currents and Digital Control Signals for One-Phase-On Drive (Wave Drive)**

## The Typical Application (Continued)




TL/H/11878–19

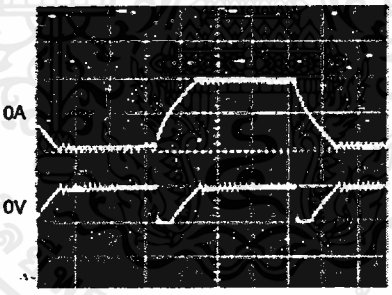
Top Trace: Phase A Winding Current at 1A/div
Bottom Trace: Phase B Winding Current at 1A/div
Horizontal: 1 ms/div
*500 steps/second




TL/H/11878–20

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

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




TL/H/11878–21

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

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

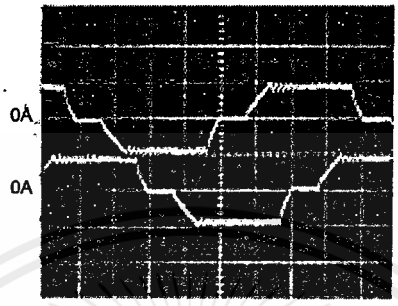
## The Typical Application (Continued)

### HALF STEP DRIVE WITHOUT TORQUE COMPENSATION

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

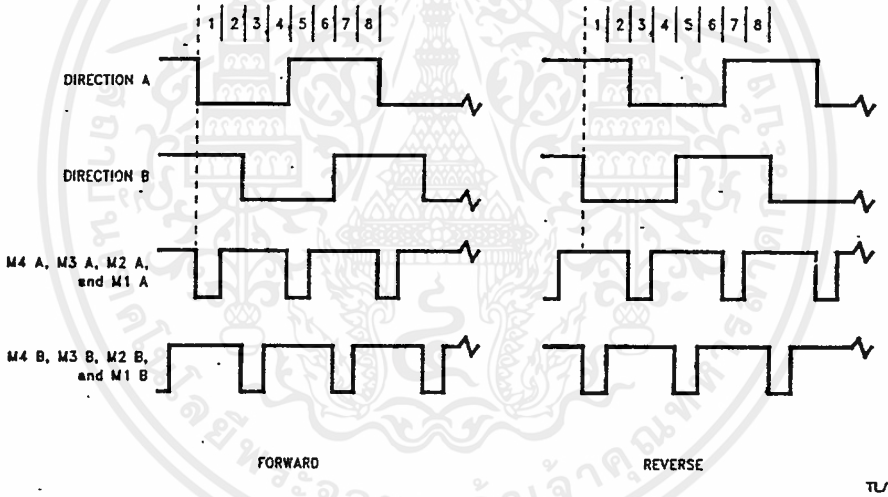
$$A \rightarrow AB \rightarrow B \rightarrow A^*B \rightarrow A^* \rightarrow A^*B^* \rightarrow B^* \rightarrow AB^* \rightarrow A \rightarrow \ldots$$

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




TL/H/11878–22

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




TL/H/11878–23

BRAKE A = BRAKE B = 0

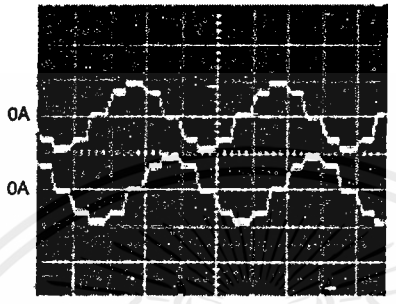
**FIGURE 10. Winding Currents and Digital Control Signals for Half Step Drive without Torque Compensation**

## The Typical Application (Continued)

### HALF STEP DRIVE WITH TORQUE COMPENSATION

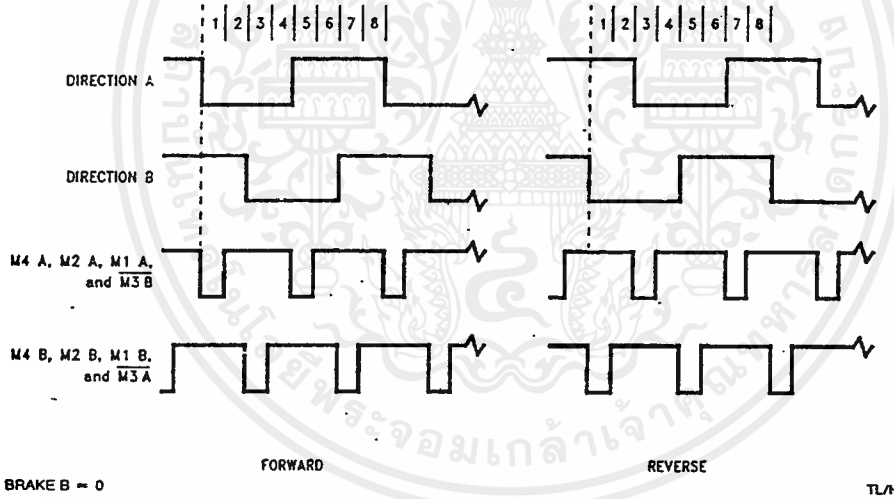
To make the motor take half steps, the windings can also be energized with sinusoidal currents *(Figure 11)*. Controlling the winding currents in the fashion shown doubles the step resolution without the significant torque ripple of the prior drive technique. The motor takes one half step each time the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

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




TL/H/11878–24

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




BRAKE A = BRAKE B = 0

TL/H/11878–25

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

## The Typical Application (Continued)

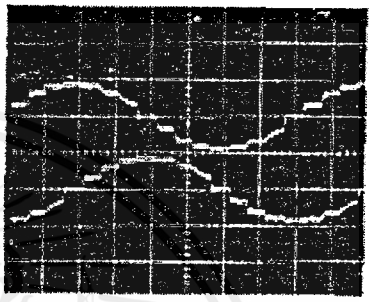
90° ELECTRICAL/FULL STEP ÷ 2 MICROSTEPS/FULL STEP = 45° ELECTRICAL/MICROSTEP

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

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

### QUARTER STEP DRIVE WITH TORQUE COMPENSATION

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




TL/H/11878–26

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

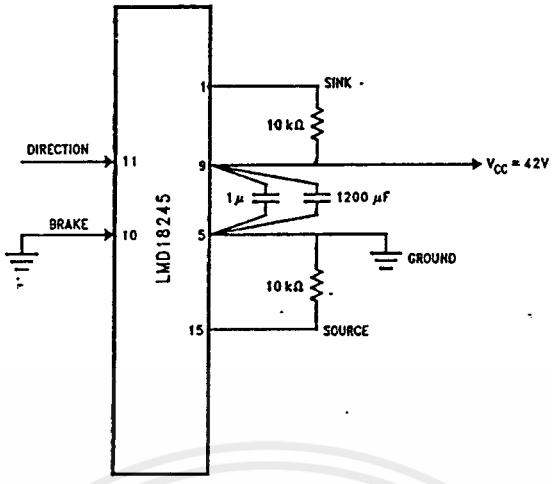
90° ELECTRICAL/FULL STEP ÷ 4 MICROSTEPS/FULL STEP = 22.5° ELECTRICAL/MICROSTEP

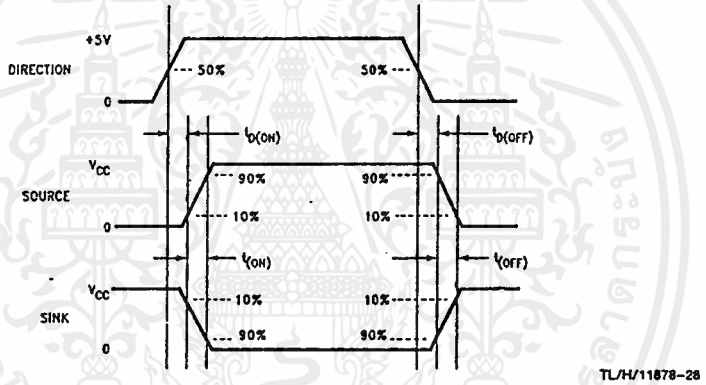
| | α | \|cos(α)\| | D A | DIRECTION A | \|sin(α)\| | D B | DIRECTION B |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 15 | 1 | 0 | 0 | 1 |
| | 22.5 | 0.924 | 14 | 1 | 0.383 | 6 | 1 |
| \| | 45 | 0.707 | 11 | 1 | 0.707 | 11 | 1 |
| FORWARD | 67.5 | 0.383 | 6 | 1 | 0.924 | 14 | 1 |
| ↓ | 90 | 0 | 0 |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

BRAKE A = BRAKE B = 0

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

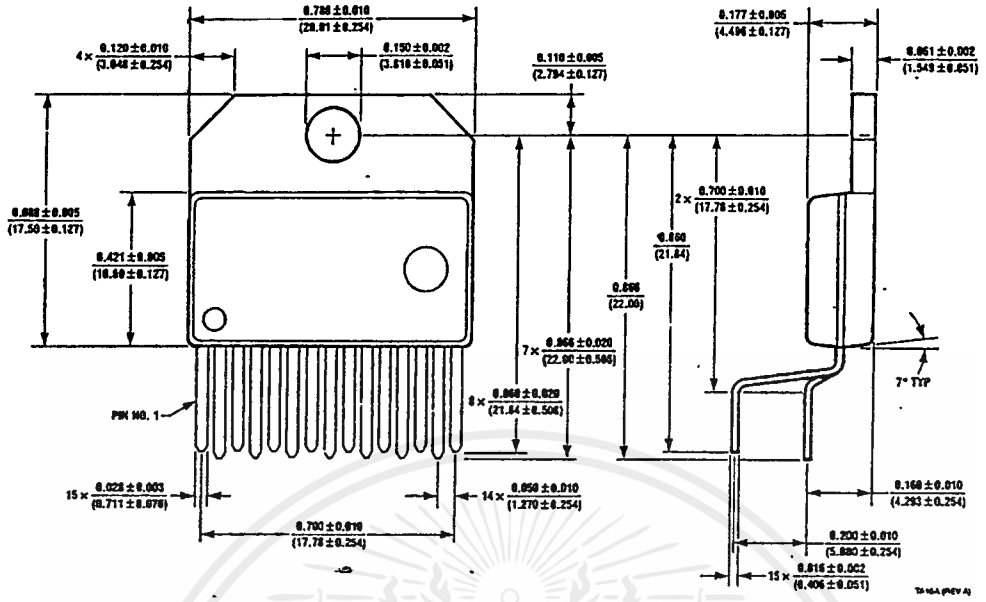
## Test Circuit and Switching Time Definitions




TL/H/11878–28

## Physical Dimensions inches (millimeters) unless otherwise noted




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

### LIFE SUPPORT POLICY

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

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

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

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

http://www.national.com

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

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

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

## Absolute Maximum Ratings (Note 1)

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

| | |
|---|---|
| Total Supply Voltage ($V_S$, Pin 6) | 60V |
| Voltage at Pins 3, 4, 5, 8 and 9 | 12V |
| Voltage at Bootstrap Pins (Pins 1 and 11) | $V_{OUT}$ + 16V |
| Peak Output Current (200 ms) | 6A |
| Continuous Output Current (Note 2) | 3A |
| Power Dissipation (Note 3) | 25W |
| Power Dissipation ($T_A$ = 25°C, Free Air) | 3W |
| Junction Temperature, $T_{J(max)}$ | 150°C |
| ESD Susceptibility (Note 4) | 1500V |
| Storage Temperature, $T_{STG}$ | −40°C to +150°C |
| Lead Temperature (Soldering, 10 sec.) | 300°C |

## Operating Ratings (Note 1)

| | |
|---|---|
| Junction Temperature, $T_J$ | −40°C to +125°C |
| $V_S$ Supply Voltage | +12V to +55V |

## Electrical Characteristics

The following specifications apply for $V_S$ = 42V, unless otherwise specified. **Boldface** limits apply over the entire operating temperature range, −40°C ≤ $T_J$ ≤ +125°C, all other limits are for $T_A$ = $T_J$ = 25°C. (Note 5)

| Symbol | Parameter | Conditions | Typ | Limit | Units |
|---|---|---|---|---|---|
| $R_{DS}(ON)$ | Switch ON Resistance | Output Current = 3A (Note 6) | 0.33 | 0.4/**0.6** | Ω (max) |
| $R_{DS}(ON)$ | Switch ON Resistance | Output Current = 6A (Note 6) | 0.33 | 0.4/**0.6** | Ω (max) |
| $V_{CLAMP}$ | Clamp Diode Forward Drop | Clamp Current = 3A (Note 6) | 1.2 | 1.5 | V (max) |
| $V_{IL}$ | Logic Low Input Voltage | Pins 3, 4, 5 | | **−0.1**<br>**0.8** | V (min)<br>V (max) |
| $I_{IL}$ | Logic Low Input Current | $V_{IN}$ = −0.1V, Pins = 3, 4, 5 | | **−10** | µA (max) |
| $V_{IH}$ | Logic High Input Voltage | Pins 3, 4, 5 | | **2**<br>**12** | V (min)<br>V (max) |
| $I_{IH}$ | Logic High Input Current | $V_{IN}$ = 12V, Pins = 3, 4, 5 | | **10** | µA (max) |
| | Current Sense Output | $I_{OUT}$ = 1A (Note 8) | 377 | 325/**300**<br>425/**450** | µA (min)<br>µA (max) |
| | Current Sense Linearity | 1A ≤ $I_{OUT}$ ≤ 3A (Note 7) | ±6 | ±9 | % |
| | Undervoltage Lockout | Outputs turn OFF | | 9<br>11 | V (min)<br>V (max) |
| $T_{JW}$ | Warning Flag Temperature | Pin 9 ≤ 0.8V, $I_L$ = 2 mA | 145 | | °C |
| $V_F(ON)$ | Flag Output Saturation Voltage | $T_J$ = $T_{JW}$, $I_L$ = 2 mA | 0.15 | | V |
| $I_F(OFF)$ | Flag Output Leakage | $V_F$ = 12V | 0.2 | 10 | µA (max) |
| $T_{JSD}$ | Shutdown Temperature | Outputs Turn OFF | 170 | | °C |
| $I_S$ | Quiescent Supply Current | All Logic Inputs Low | 13 | 25 | mA (max) |
| $t_{Don}$ | Output Turn-On Delay Time | Sourcing Outputs, $I_{OUT}$ = 3A<br>Sinking Outputs, $I_{OUT}$ = 3A | 300<br>300 | | ns<br>ns |
| $t_{on}$ | Output Turn-On Switching Time | Bootstrap Capacitor = 10 nF<br>Sourcing Outputs, $I_{OUT}$ = 3A<br>Sinking Outputs, $I_{OUT}$ = 3A | <br>100<br>80 | | <br>ns<br>ns |
| $t_{Doff}$ | Output Turn-Off Delay Times | Sourcing Outputs, $I_{OUT}$ = 3A<br>Sinking Outputs, $I_{OUT}$ = 3A | 200<br>200 | | ns<br>ns |
| $t_{off}$ | Output Turn-Off Switching Times | Bootstrap Capacitor = 10 nF<br>Sourcing Outputs, $I_{OUT}$ = 3A<br>Sinking Outputs, $I_{OUT}$ = 3A | <br>75<br>70 | | <br>ns<br>ns |
| $t_{pw}$ | Minimum Input Pulse Width | Pins 3, 4 and 5 | 1 | | µs |
| $t_{cpr}$ | Charge Pump Rise Time | No Bootstrap Capacitor | 20 | | µs |

## Electrical Characteristics Notes

**Note 1:** Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions.

**Note 2:** See Application Information for details regarding current limiting.

**Note 3:** The maximum power dissipation must be derated at elevated temperatures and is a function of $T_{J(max)}$, $\theta_{JA}$, and $T_A$. The maximum allowable power dissipation at any temperature is $P_{D(max)} = (T_{J(max)} - T_A)/\theta_{JA}$, or the number given in the Absolute Ratings, whichever is lower. The typical thermal resistance from junction to case ($\theta_{JC}$) is 1.0°C/W and from junction to ambient ($\theta_{JA}$) is 30°C/W. For guaranteed operation $T_{J(max)} = 125°C$.

**Note 4:** Human-body model, 100 pF discharged through a 1.5 kΩ resistor. Except Bootstrap pins (pins 1 and 11) which are protected to 1000V of ESD.

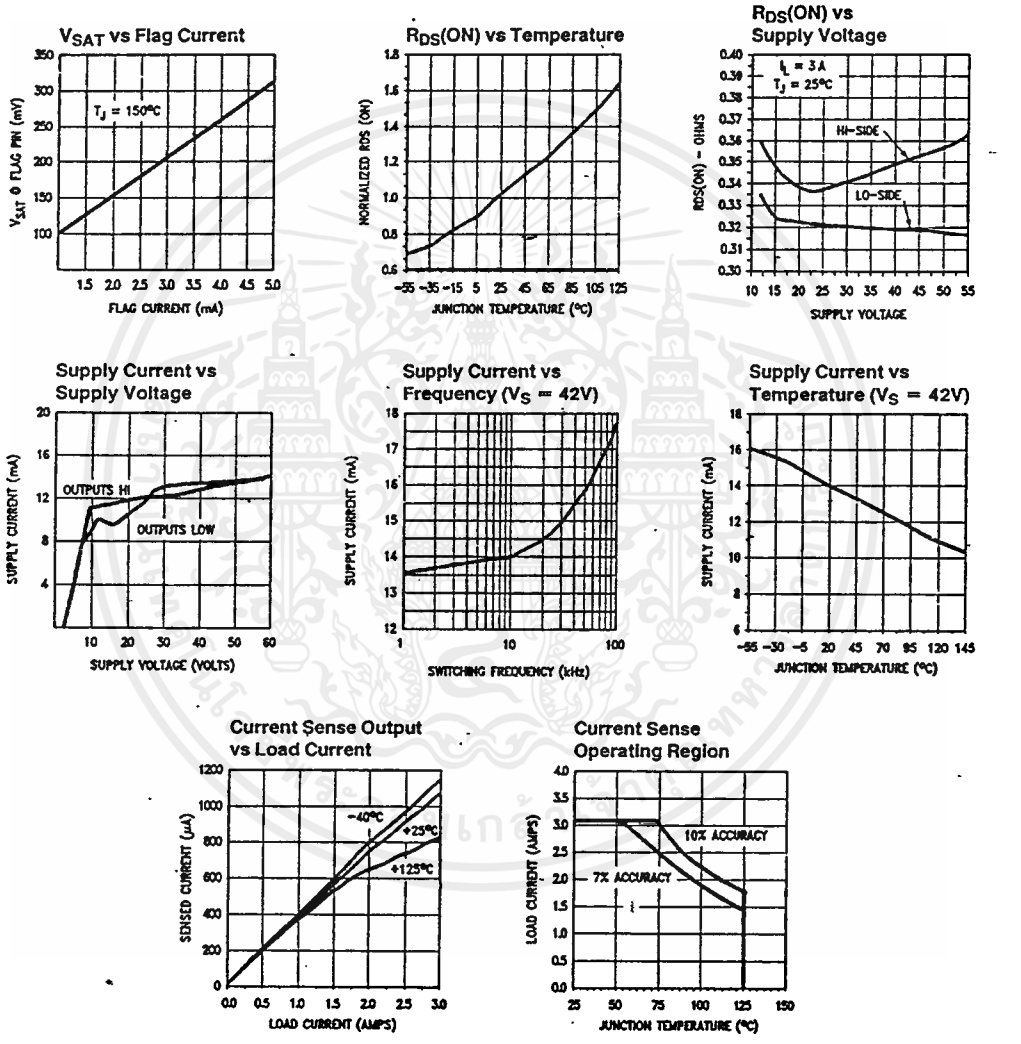
**Note 5:** All limits are 100% production tested at 25°C. Temperature extreme limits are guaranteed via correlation using accepted SQC (Statistical Quality Control) methods. All limits are used to calculate AOQL, (Average Outgoing Quality Level).

**Note 6:** Output currents are pulsed ($t_W < 2$ ms, Duty Cycle < 5%).

**Note 7:** Regulation is calculated relative to the current sense output value with a 1A load.

**Note 8:** Selections for tighter tolerance are available. Contact factory.

## Typical Performance Characteristics




TL/H/10568–3

## Pinout Description (See Connection Diagram)

**Pin 1, BOOTSTRAP 1 Input:** Bootstrap capacitor pin for half H-bridge number 1. The recommended capacitor (10 nF) is connected between pins 1 and 2.

**Pin 2, OUTPUT 1:** Half H-bridge number 1 output.

**Pin 3, DIRECTION Input:** See Table I. This input controls the direction of current flow between OUTPUT 1 and OUTPUT 2 (pins 2 and 10) and, therefore, the direction of rotation of a motor load.

**Pin 4, BRAKE Input:** See Table I. This input is used to brake a motor by effectively shorting its terminals. When braking is desired, this input is taken to a logic high level and it is also necessary to apply logic high to PWM input, pin 5. The drivers that short the motor are determined by the logic level at the DIRECTION input (Pin 3): with Pin 3 logic high, both current sourcing output transistors are ON; with Pin 3 logic low, both current sinking output transistors are ON. All output transistors can be turned OFF by applying a logic high to Pin 4 and a logic low to PWM input Pin 5; in this case only a small bias current (approximately −1.5 mA) exists at each output pin.

**Pin 5, PWM Input:** See Table I. How this input (and DIRECTION input, Pin 3) is used is determined by the format of the PWM Signal.

**Pin 6, $V_S$ Power Supply**

**Pin 7, GROUND Connection:** This pin is the ground return, and is internally connected to the mounting tab.

**Pin 8, CURRENT SENSE Output:** This pin provides the sourcing current sensing output signal, which is typically 377 μA/A.

**Pin 9, THERMAL FLAG Output:** This pin provides the thermal warning flag output signal. Pin 9 becomes active-low at 145°C (junction temperature). However the chip will not shut itself down until 170°C is reached at the junction.

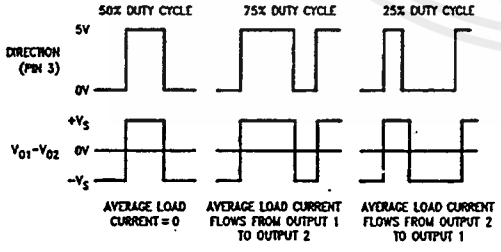
**Pin 10, OUTPUT 2:** Half H-bridge number 2 output.

**Pin 11, BOOTSTRAP 2 Input:** Bootstrap capacitor pin for Half H-bridge number 2. The recommended capacitor (10 nF) is connected between pins 10 and 11.

**TABLE I. Logic Truth Table**

| PWM | Dir | Brake | Active Output Drivers |
|---|---|---|---|
| H | H | L | Source 1, Sink 2 |
| H | L | L | Sink 1, Source 2 |
| L | X | L | Source 1, Source 2 |
| H | H | H | Source 1, Source 2 |
| H | L | H | Sink 1, Sink 2 |
| L | X | H | NONE |

**Locked Anti-Phase PWM Control**




TL/H/10568–4

## Application Information

### TYPES OF PWM SIGNALS

The LMD18200 readily interfaces with different forms of PWM signals. Use of the part with two of the more popular forms of PWM is described in the following paragraphs.

**Simple, locked anti-phase PWM** consists of a single, variable duty-cycle signal in which is encoded both direction and amplitude information. A 50% duty-cycle PWM signal represents zero drive, since the net value of voltage (integrated over one period) delivered to the load is zero. For the LMD18200, the PWM signal drives the direction input (pin 3) and the PWM input (pin 5) is tied to logic high.

**Sign/magnitude PWM** consists of separate direction (sign) and amplitude (magnitude) signals. The (absolute) magnitude signal is duty-cycle modulated, and the absence of a pulse signal (a continuous logic low level) represents zero drive. Current delivered to the load is proportional to pulse width. For the LMD18200, the DIRECTION input (pin 3) is driven by the sign signal and the PWM input (pin 5) is driven by the magnitude signal.

### USING THE CURRENT SENSE OUTPUT

The CURRENT SENSE output (pin 8) has a sensitivity of 377 μA per ampere of output current. For optimal accuracy and linearity of this signal, the value of voltage generating resistor between pin 8 and ground should be chosen to limit the maximum voltage developed at pin 8 to 5V, or less. The maximum voltage compliance is 12V.

It should be noted that the recirculating currents (free wheeling currents) are ignored by the current sense circuitry. Therefore, only the currents in the upper sourcing outputs are sensed.

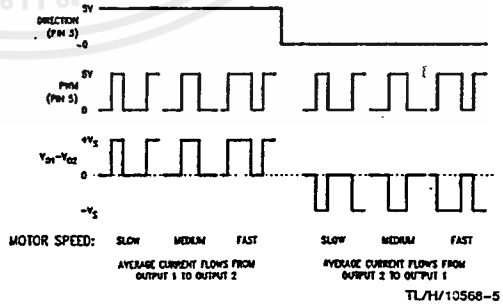
### USING THE THERMAL WARNING FLAG

The THERMAL FLAG output (pin 9) is an open collector transistor. This permits a wired OR connection of thermal warning flag outputs from multiple LMD18200's, and allows the user to set the logic high level of the output signal swing to match system requirements. This output typically drives the interrupt input of a system controller. The interrupt service routine would then be designed to take appropriate steps, such as reducing load currents or initiating an orderly system shutdown. The maximum voltage compliance on the flag pin is 12V.

### SUPPLY BYPASSING

During switching transitions the levels of fast current changes experienced may cause troublesome voltage transients across system stray inductance.

**Sign/Magnitude PWM Control**




TL/H/10568–5

## Application Information (Continued)

It is normally necessary to bypass the supply rail with a high quality capacitor(s) connected as close as possible to the $V_S$ Power Supply (Pin 6) and GROUND (Pin 7). A 1 $\mu$F high-frequency ceramic capacitor is recommended. Care should be taken to limit the transients on the supply pin below the Absolute Maximum Rating of the device. When operating the chip at supply voltages above 40V a voltage suppressor (transorb) such as P6KE62A is recommended from supply to ground. Typically the ceramic capacitor can be eliminated in the presence of the voltage suppressor. Note that when driving high load currents a greater amount of supply bypass capacitance (in general at least 100 $\mu$F per Amp of load current) is required to absorb the recirculating currents of the inductive loads.

### CURRENT LIMITING

Current limiting protection circuitry has been incorporated into the design of the LMD18200. With any power device it is important to consider the effects of the substantial surge currents through the device that may occur as a result of shorted loads. The protection circuitry monitors this increase in current (the threshold is set to approximately 10 Amps) and shuts off the power device as quickly as possible in the event of an overload condition. In a typical motor driving application the most common overload faults are caused by shorted motor windings and locked rotors. Under these conditions the inductance of the motor (as well as any series inductance in the $V_{CC}$ supply line) serves to reduce the magnitude of a current surge to a safe level for the LMD18200. Once the device is shut down, the control circuitry will periodically try to turn the power device back on. This feature allows the immediate return to normal operation in the event that the fault condition has been removed. While the fault remains however, the device will cycle in and out of thermal shutdown. This can create voltage transients on the $V_{CC}$ supply line and therefore proper supply bypassing techniques are required.

The most severe condition for any power device is a direct, hard-wired ("screwdriver") long term short from an output to ground. This condition can generate a surge of current through the power device on the order of 15 Amps and require the die and package to dissipate up to 500 Watts of power for the short time required for the protection circuitry to shut off the power device. This energy can be destructive, particularly at higher operating voltages (>30V) so some precautions are in order. Proper heat sink design is essential and it is normally necessary to heat sink the $V_{CC}$ supply pin (pin 6) with 1 square inch of copper on the PCB.

### INTERNAL CHARGE PUMP AND USE OF BOOTSTRAP CAPACITORS

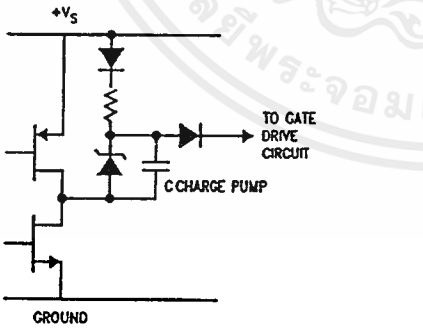
To turn on the high-side (sourcing) DMOS power devices, the gate of each device must be driven approximately 8V more positive than the supply voltage. To achieve this an internal charge pump is used to provide the gate drive voltage. As shown in *Figure 1*, an internal capacitor is alternately switched to ground and charged to about 14V, then switched to V supply thereby providing a gate drive voltage greater than V supply. This switching action is controlled by a continuously running internal 300 kHz oscillator. The rise time of this drive voltage is typically 20 $\mu$s which is suitable for operating frequencies up to 1 kHz.

For higher switching frequencies, the LMD18200 provides for the use of external bootstrap capacitors. The bootstrap principle is in essence a second charge pump whereby a large value capacitor is used which has enough energy to quickly charge the parasitic gate input capacitance of the power device resulting in much faster rise times. The switching action is accomplished by the power switches themselves *(Figure 2)*. External 10 nF capacitors, connected from the outputs to the bootstrap pins of each high-side switch provide typically less than 100 ns rise times allowing switching frequencies up to 500 kHz.

### INTERNAL PROTECTION DIODES

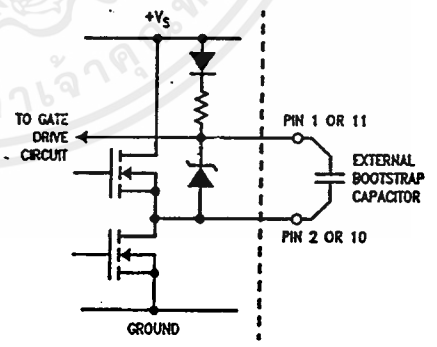
A major consideration when switching current through inductive loads is protection of the switching power devices from the large voltage transients that occur. Each of the four switches in the LMD18200 have a built-in protection diode to clamp transient voltages exceeding the positive supply or ground to a safe diode voltage drop across the switch.

The reverse recovery characteristics of these diodes, once the transient has subsided, is important. These diodes must come out of conduction quickly and the power switches must be able to conduct the additional reverse recovery current of the diodes. The reverse recovery time of the diodes protecting the sourcing power devices is typically only 70 ns with a reverse recovery current of 1A when tested with a full 6A of forward current through the diode. For the sinking devices the recovery time is typically 100 ns with 4A of reverse current under the same conditions.




TL/H/10568–6

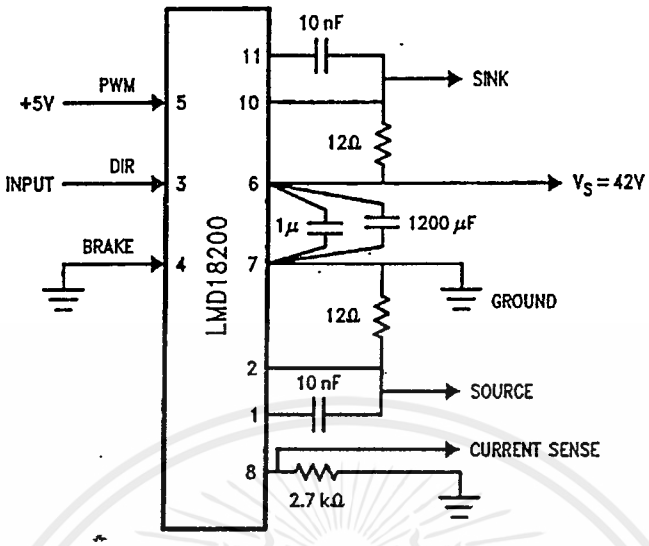
**FIGURE 1. Internal Charge Pump Circuitry**




TL/H/10568–7

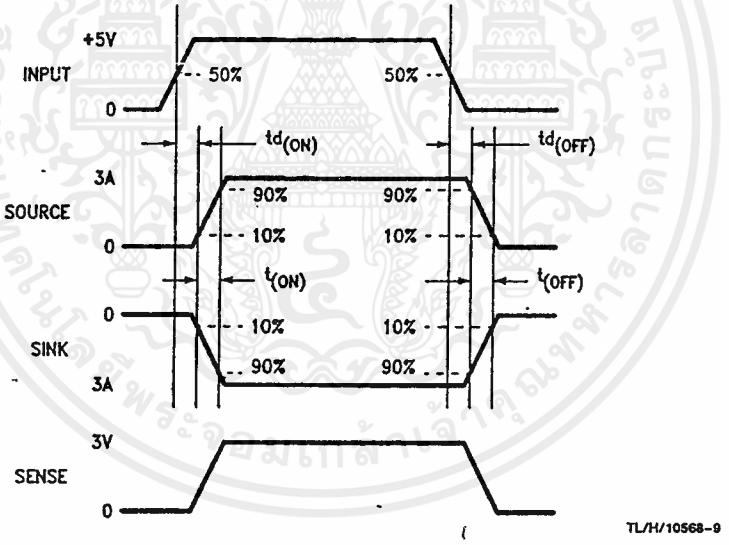
**FIGURE 2. Bootstrap Circuitry**

## Test Circuit




TL/H/10568–8

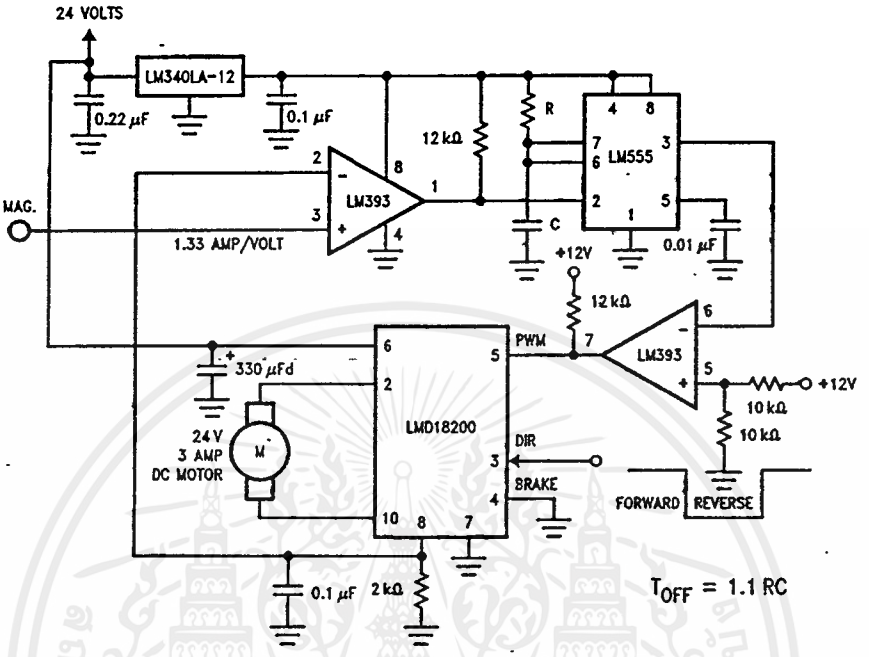
## Switching Time Definitions




TL/H/10568–9

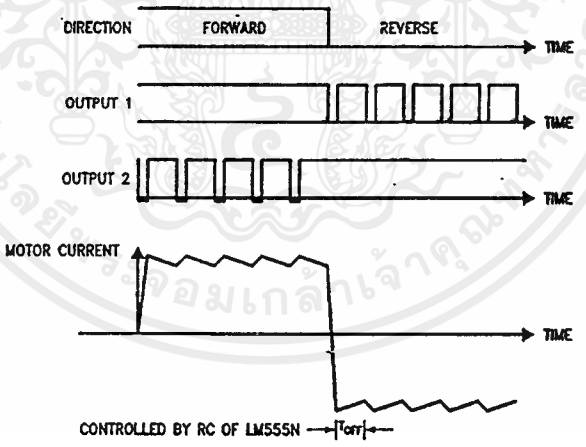
## Typical Applications

**Fixed Off-Time Control:** This circuit controls the current through the motor by applying an average voltage equal to zero to the motor terminals for a fixed period of time, whenever the current through the motor exceeds the commanded current. This action causes the motor current to vary slightly about an externally controlled average level. The duration of the Off-period is adjusted by the resistor and capacitor combination of the LM555. In this circuit the Sign/Magnitude mode of operation is implemented (see Types of PWM Signals).




TL/H/10568–10

## Switching Waveforms

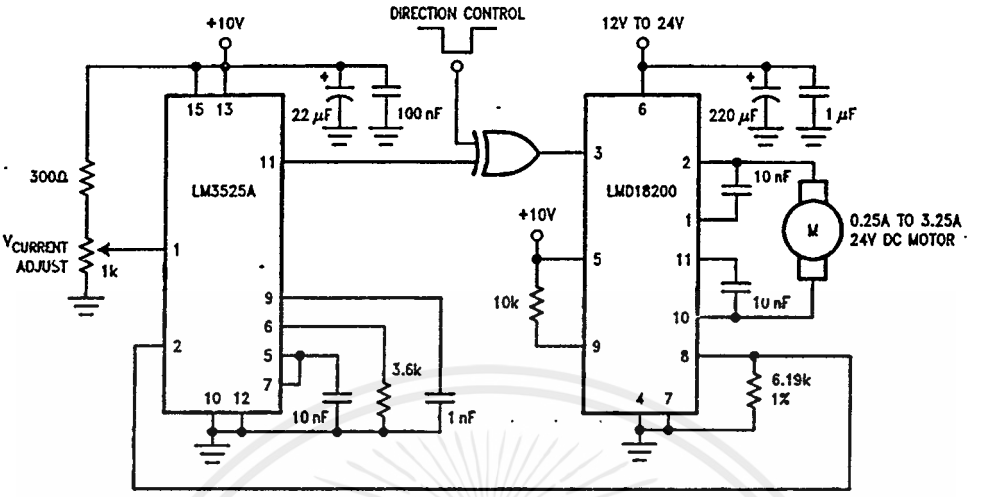



TL/H/10568–11

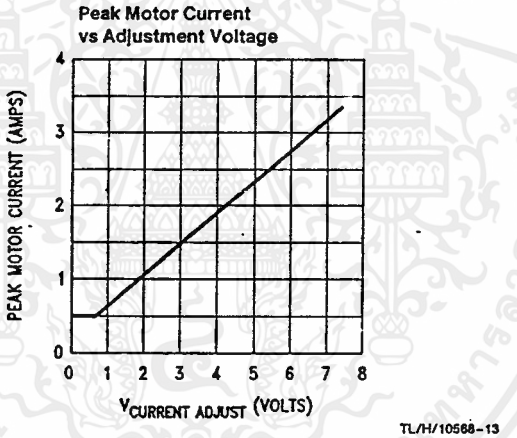
## Typical Applications (Continued)

**TORQUE REGULATION**

Locked Anti-Phase Control of a brushed DC motor. Current sense output of the LMD18200 provides load sensing. The LM3525A is a general purpose PWM controller.




TL/H/10568–12

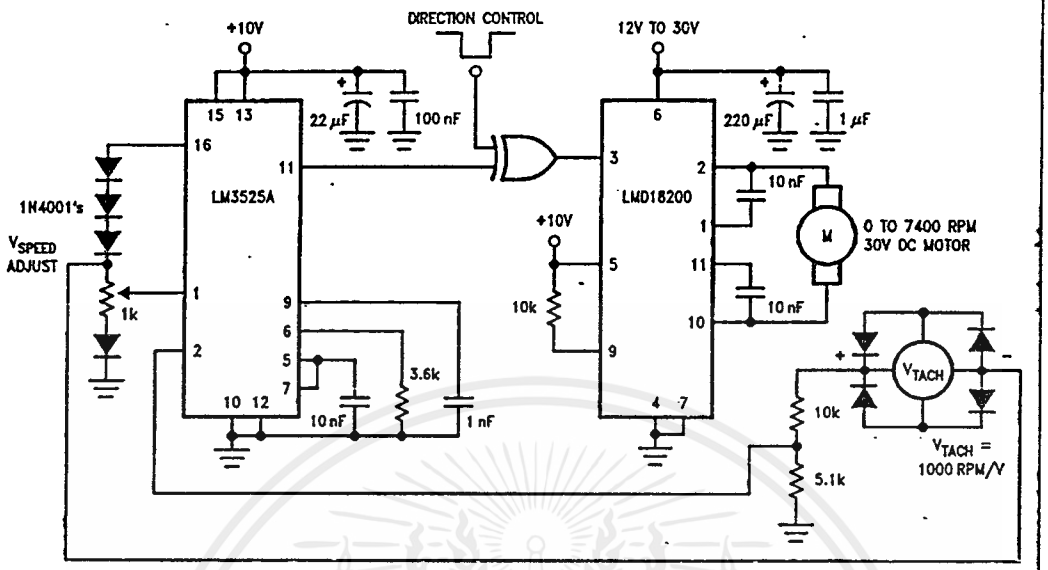



TL/H/10568–13

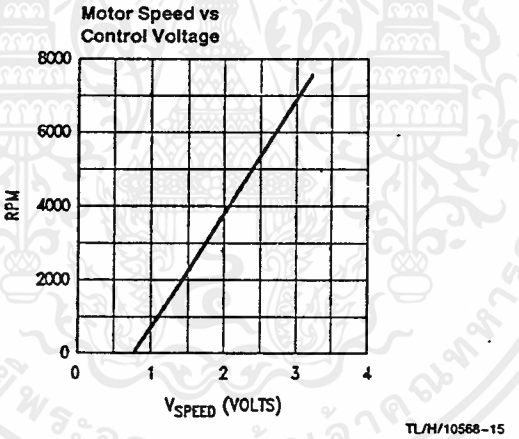
## Typical Applications (Continued)

**VELOCITY REGULATION**

Utilizes tachometer output from the motor to sense motor speed for a locked anti-phase control loop.

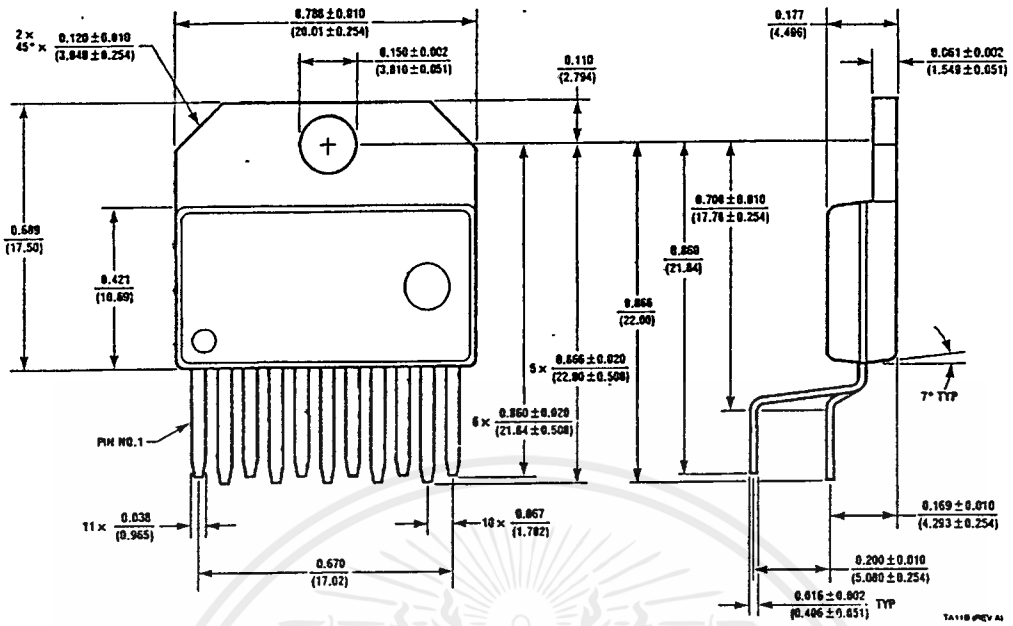



TL/H/10568–14




TL/H/10568–15

## Physical Dimensions inches (millimeters) unless otherwise noted




**11-Lead TO-220 Power Package (T)**
**Order Number LMD18200T**
**NS Package Number TA11B**

### LIFE SUPPORT POLICY

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

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

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

http://www.national.com

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

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

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

1 รายงานการประยุกต์วงจรอิเล็กทรอนิกส์ เรื่องคอมพิวเตอร์สื่อสารข้อมูลแบบ RS-232

2.K CARR-BRION,Moisure Sensors in Process Control,ELSEVIER SCIENCE Publisher

3. National Power IC's Databook