ปีการศึกษา 2539

การควบคุมระยะทางของมอเตอร์

MOTOR FEEDING MEASUREMENT

โดย

นายเฉลิมพล ทองเปลว

นายประพัฒน์ สุขสวัสดิ์

อาจารย์ที่ปรึกษา

อาจารย์พิชิต ลำยอง

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมระยะทางของมอเตอร์

ผู้จัดทำ

1. นายเฉลิมพล ทองเปลว
2. นายประพัฒน์ สุขสวัสดิ์

อาจารย์ที่ปรึกษา

(อาจารย์พิชิต ลำยอง)

I

การควบคุมระยะทางของมอเตอร์

นายเฉลิมพล ทองเปลว
นายประพัฒน์ สุขสวัสดิ์
อาจารย์พิชิต ลำยอง อาจารย์ที่ปรึกษา
ปีการศึกษา 2539

บทคัดย่อ

โครงงานนี้เป็นลักษณะหนึ่งของการศึกษาด้านการควบคุมมอเตอร์ในลักษณะการควบคุมระยะทางหรือตำแหน่งโดยใช้ไมโครคอนโทรลเลอร์ซึ่งใช้เบอร์ 68HC11 มาใช้ในการประมวลผล เพื่อควบคุมระยะทางให้มอเตอร์หมุนไปเป็นระยะทางที่เราต้องการ โดยได้ข้อมูลในการประมวลผลเป็นสัญญาณพัลส์จากอุปกรณ์ตรวจสอบจำนวนรอบการหมุนของมอเตอร์ที่เรียกว่าเอนโค้ดเดอร์ (Encoder) เป็นข้อมูลป้อนกลับ ผลที่ได้จากการประมวลผลจะทำให้ไมโครคอนโทรลเลอร์ส่งสัญญาณควบคุมอุปกรณ์ต่าง ๆ เพื่อผลในการควบคุมระยะทางของมอเตอร์

II

MOTOR FEEDING MEASUREMENT

Chalermphol Tongphew
Prapat Suksawad
Pichit Lumyong Advisor
1997

ABSTRACT

This project is one in many fields about motor control studying which is distance or position control by using microcontroller 68HC11 for processing. The purpose is to control the distance of motor to rotate in the distance equal the satisfied distance. The data for processing is the pulse signal from the encoder which is the device for sensoring the number of motor rotation . The result from 68HC11 processing will make it input the control signal to another device to control the distance of motor.

สารบัญ

หน้า

สารบัญ

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ(ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

เนื่องจากในปัจจุบันในภาคอุตสาหกรรมมีการใช้มอเตอร์เป็นส่วนประกอบหลักของเครื่องจักรเป็นจำนวนมาก ดังนั้นการศึกษาด้านการควบคุมมอเตอร์จึงเป็นสิ่งสำคัญและมีประโยชน์อย่างมากต่อวิศวกรหรือบุคคลที่เกี่ยวข้อง ซึ่งการควบคุมมอเตอร์นั้นมีหลายลักษณะเช่น การควบคุมตำแหน่ง(Position control), การควบคุมความเร็ว(Speed control) เป็นต้น

Motor Feeding Measurement เป็นโครงงานหนึ่งซึ่งอาศัยหลักการควบคุมแบบป้อนกลับมาใช้ในการควบคุมมอเตอร์และอาศัยคุณสมบัติของไมโครคอนโทรลเลอร์มาช่วยในการประมวลผล โดยเป็นโครงงานที่ศึกษาด้านการควบคุมมอเตอร์เพื่อให้ได้ระยะทางหรือตำแหน่งของมอเตอร์ที่ต้องการ

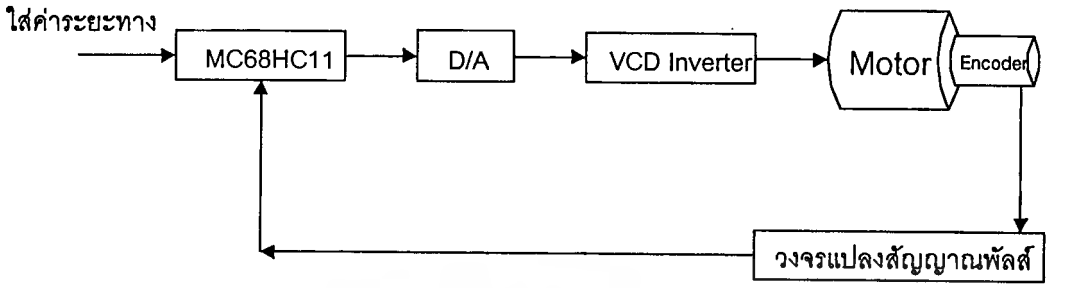
จุดประสงค์และวิธีการ

ลักษณะการทำงานเบื้องต้นคือการกำหนดระยะทางที่ต้องการและป้อนข้อมูลให้ไมโครคอนโทรลเลอร์ให้ควบคุมมอเตอร์ให้หมุนไปได้ระยะเท่ากับระยะที่ต้องการ โดยจะมีตัวตรวจสอบจำนวนรอบของการหมุนหรือระยะทางการหมุนของมอเตอร์คือตัวเอนโค้ดเดอร์

ตัวเอนโค้ดเดอร์จะเป็นตัวสร้างสัญญาณพัลส์(pulse)ที่ได้มาจากการตรวจสอบจำนวนรอบการหมุนของมอเตอร์ซึ่งจำนวนพัลส์จะสัมพันธ์กับจำนวนรอบการหมุนตามลักษณะของเอนโค้ดเดอร์ต่าง ๆ เช่น 1000 พัลส์ ต่อการหมุน 1 รอบเป็นต้น

สัญญาณพัลส์ที่สร้างได้จะส่งให้ตัวนับ(counter)ของไมโครคอนโทรลเลอร์นับและคำนวณระยะทางได้ และตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณไปควบคุมเพื่อให้เกิดผลในการควบคุมแหล่งจ่ายไฟ(power supply) ของมอเตอร์ซึ่งจะเลือกใช้แหล่งจ่ายไฟแบบอินเวอร์เตอร์(inverter) มาเป็นแหล่งจ่ายไฟให้มอเตอร์ ให้หยุดจ่ายไฟหรือมีการเปลี่ยนแปลงความถี่เพื่อลดความเร็วของมอเตอร์ในเวลาที่เหมาะสมได้ ซึ่งถือว่าเป็นการควบคุมความเร็วอย่างหนึ่ง โดยหลักการเบื้องต้นของการควบคุมความเร็วมอเตอร์จะขอกล่าวในภาคผนวก

อนึ่งเครื่องอินเวอร์เตอร์ที่ใช้ในโครงงานเป็นเครื่องที่สามารถควบคุมได้จากสัญญาณอนาลอกจากภายนอกให้จ่ายค่าความถี่ไฟต่างๆสัมพันธ์อย่างเป็นสัดส่วนกันกับสัญญาณอนาลอกภายนอกค่านั้น



รูปที่ 1.1 แสดงบล็อกไดอะแกรมคร่าว ๆ ของโครงงาน

บทที่ 2
ทฤษฎีและการใช้งานเอนโค้ดเดอร์(Encoder)

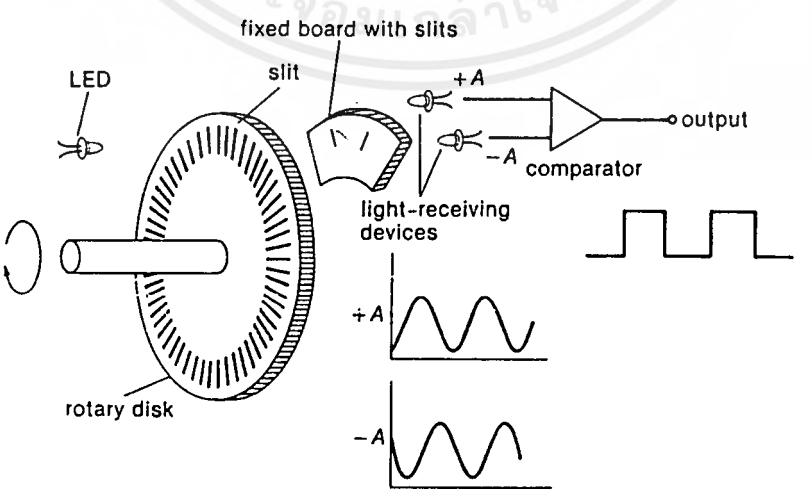
ในการตรวจสอบตำแหน่งในการควบคุมมอเตอร์จะไม่สามารถทำได้จนกว่าจะใส่ตัวตรวจสอบตำแหน่ง(position sensor) ร่วมกับส่วนควบคุมวงจรซึ่งตัวตรวจสอบเหล่านี้ได้แก่ตัวเอนโค้ดเดอร์ ซึ่งเอนโค้ดเดอร์มีหลายลักษณะโครงสร้างต่าง ๆ กันไป เช่น เอนโค้ดเดอร์ระบบใช้แสง(Optical encoder), เอนโค้ดเดอร์ระบบแม่เหล็ก(Magnetic encoder) ในโครงงานจะใช้เอนโค้ดเดอร์ระบบใช้แสง ดังนั้นจึงจะกล่าวถึงลักษณะของเอนโค้ดเดอร์ลักษณะนี้เท่านั้น

2.1 เอนโค้ดเดอร์ระบบใช้แสง

เอนโค้ดเดอร์ระบบนี้สามารถแบ่งออกเป็น2แบบคือ แบบอินครีเมนทอล(Incremental encoder)และแบบแอบโซลูท(Absolute encoder) โดยแบ่งตามลักษณะหน้าที่ของมันซึ่งเอนโค้ดเดอร์ระบบใช้แสงมีส่วนประกอบหลักดังนี้

1. แหล่งกำเนิดแสง(light source)
2. อุปกรณ์รับแสง(light receiving device)
3. จานหมุนที่มีช่อง(rotary disc with slits)

เอนโค้ดเดอร์แบบนี้จะกำเนิดพัลส์เป็นสัดส่วนกับมุมของการหมุนของจานหมุน จานหมุนจะอยู่ระหว่างแหล่งกำเนิดแสงและอุปกรณ์รับแสง ดังแสดงในรูปที่ 2.1

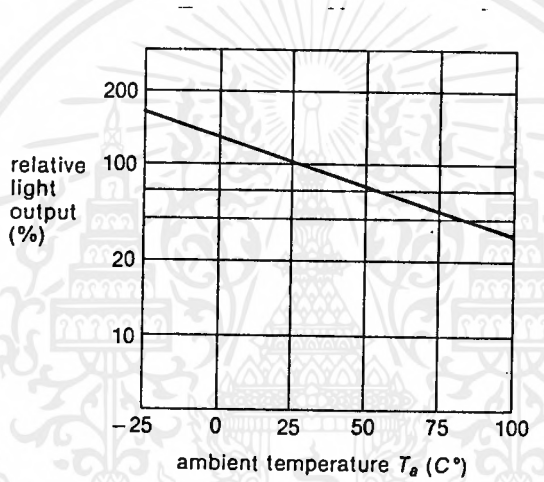


รูป 2.1 ลักษณะทางกายภาพของเอนโค้ดเดอร์ระบบใช้แสง

โดยจะใช้หลอด LED (light emitting diode) เป็นแหล่งกำเนิดแสงและประกอบกับแผ่นจานที่อยู่กับที่(fixed board) ที่มีช่องสำหรับให้แสงจากแหล่งกำเนิดแสงผ่านจานหมุนไปอย่างถูกต้องไปที่อุปกรณ์รับแสง

อุปกรณ์รับแสงสามารถตรวจจับแสง(รังสีอินฟาเรด)ที่เกิดจากหลอด LED หลังจากแสงได้ผ่านช่องของจานหมุนและช่องของจานที่อยู่กับที่ แสงจากหลอด LED ที่ใช้เป็นแหล่งกำเนิดแสงจะแปรค่าได้ตามอุณหภูมิ

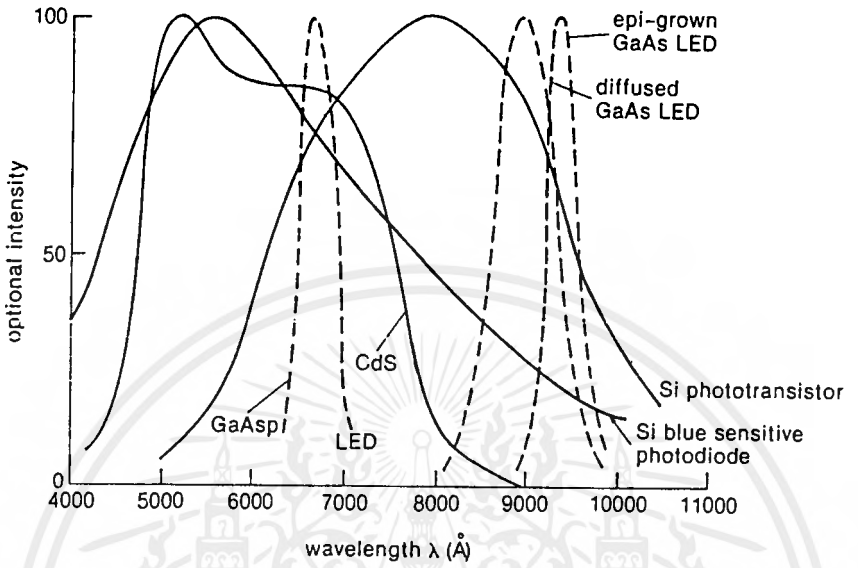
ความสัมพันธ์ระหว่างอุณหภูมิห้องกับแสงของ LED มีความสัมพันธ์ตามรูปที่ 2.2 โดยจะสังเกตเห็นว่าถ้าอุณหภูมิห้องเพิ่มขึ้น แสงจาก LED จะลดลง



รูปที่ 2.2 ลักษณะกราฟความสัมพันธ์ของอุณหภูมิกับแสงจากหลอด LED

วิธีการส่วนมากที่ใช้เพื่อชดเชยส่วนที่ลดลงคือการวางอุปกรณ์รับแสง 2 ตัวและทำให้มันลดสัญญาณเอาท์พุทที่มีเฟสต่างกัน180องศาสัญญาณจะถูกประมวลผลโดยตัวเปรียบเทียบ(comparator) เพื่อให้เกิดสัญญาณคลื่นรูปสี่เหลี่ยม(rectangular wave) ที่คงที่

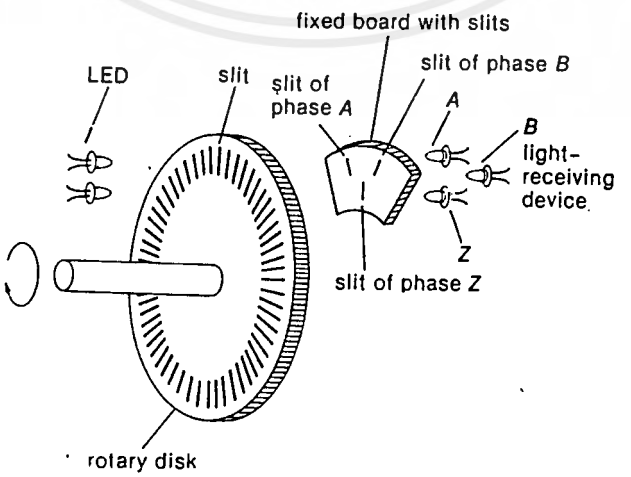
ลักษณะความยาวคลื่นของอุปกรณ์กำเนิดแสง อุปกรณ์รับแสง และอุปกรณ์อื่นที่เกี่ยวข้องที่ใช้ในเอนโค้ดเดอร์ระบบใช้แสงแสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 ลักษณะความยาวคลื่นของแหล่งกำเนิดแสงและอุปกรณ์รับแสง

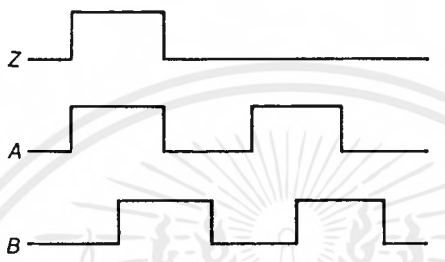
2.2 อินครีเมนทอล เอนโค้ดเดอร์(Incremental encoder)

ลักษณะของเอนโค้ดเดอร์แบบนี้แสดงดังรูปที่ 2.4 เป็นโครงสร้างของการตรวจจับตำแหน่งมุมที่สัมพันธ์กับเอาท์พุท A และ B ของเอนโค้ดเดอร์และมีสัญญาณ Z ของเฟสศูนย์



รูปที่ 2.4 โครงสร้างของเอนโค้ดเดอร์แบบอินครีเมนทอล

แสงที่สร้างจากหลอด LED จะผ่านช่องของจานหมุนและผ่านแต่ละช่องของ A, B และ Z ของจานที่อยู่กับที่ ดังนั้นแสงจะถูกตรวจจับโดยอุปกรณ์รับแสง A, B และ Z ช่อง A และ B บนจานอยู่กับที่มีเฟสต่างกัน 90 องศา และเอาท์พุทจะมีลักษณะเป็นคลื่นรูปสี่เหลี่ยมโดยมีเฟสต่างกัน 90 องศาด้วย ดังรูปที่ 2.5 เป็นรูปแสดงเอาท์พุทสุดท้ายของอุปกรณ์รับแสง



รูปที่ 2.5 รูปแสดงลักษณะคลื่นเอาท์พุทของเอนโค้ดเดอร์แบบอินครีเมนทอล

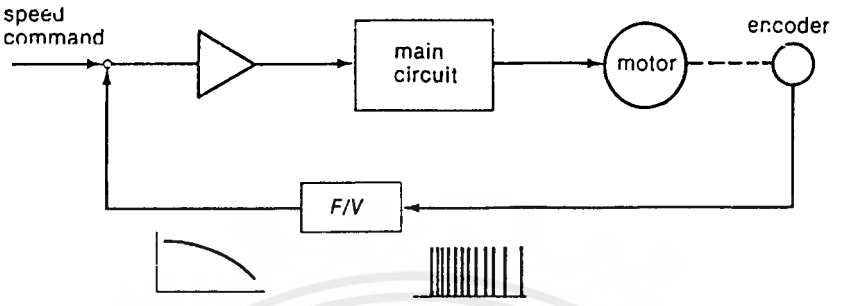
เอนโค้ดเดอร์แบบอินครีเมนทอลมีลักษณะโครงสร้างง่าย ๆ และราคาถูก และง่ายต่อการส่งสัญญาณเพราะมีสายไฟเพียง 2-3 สายสำหรับสายเอาท์พุท

พัลส์ที่ได้ของเอนโค้ดเดอร์ไม่ได้แสดงค่าที่แท้จริงของตำแหน่งการหมุนของแกน แต่จำนวนพัลส์เป็นสัดส่วนกับมุมของแกน ค่าแท้จริงของตำแหน่งการหมุนของแกนจะสามารถหาได้จากผลของการคำนวณพัลส์ของเอนโค้ดเดอร์จากตัวนับของไมโครคอนโทรลเลอร์

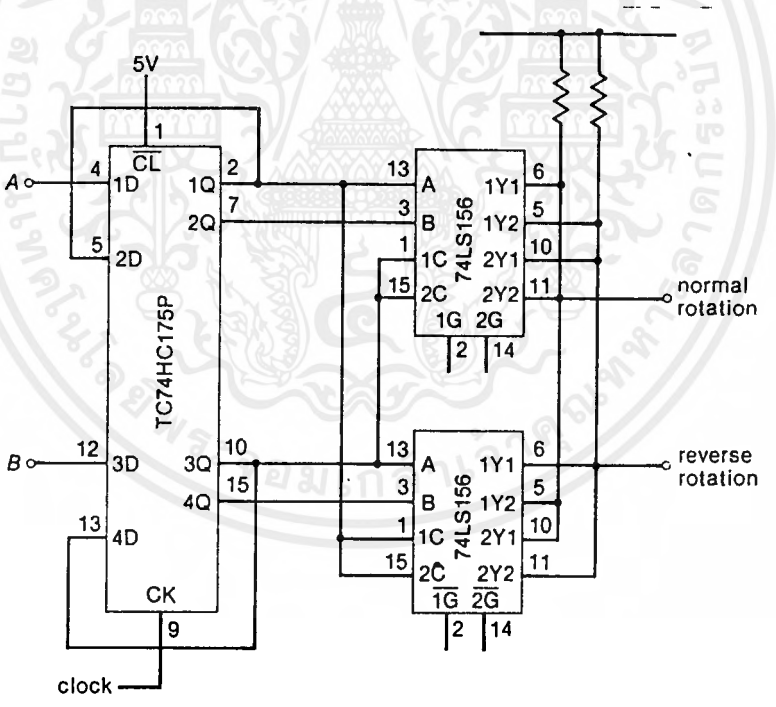
2.3 การตรวจจับความเร็วในการหมุนโดยใช้เอนโค้ดเดอร์แบบอินครีเมนทอล

เอนโค้ดเดอร์แบบนี้ไม่มีหน้าที่อะไรอื่นพิเศษนอกจากการผลิตขบวนพัลส์ ดังนั้นจำนวนพัลส์เอาท์พุทของเอนโค้ดเดอร์ต้องแปลงเป็นสัญญาณอนาลอก โดยเป็นสัดส่วนกับความถี่พัลส์โดยวิธีแปลงความถี่เป็นโวลต์เตจ(F/V converter) ดังแสดงในรูปที่ 2.6 ซึ่งทำให้สามารถให้สัญญาณอนาลอกสำหรับตรวจจับความเร็วในการหมุนของมอเตอร์ได้

ในกรณีจำนวนพัลส์น้อยมากจนเกิดปัญหาบางส่วนในทางปฏิบัติ จำนวนพัลส์สามารถเพิ่มได้โดยวิธีความถี่คูณ 4 (frequency multiplication by four) ดังแสดงในรูป 2.7 และแปลงเป็นสัญญาณอนาลอกโดยวิธีการแปลงความถี่เป็นโวลต์เตจ



รูปที่ 2.6 การตรวจจับความเร็ว

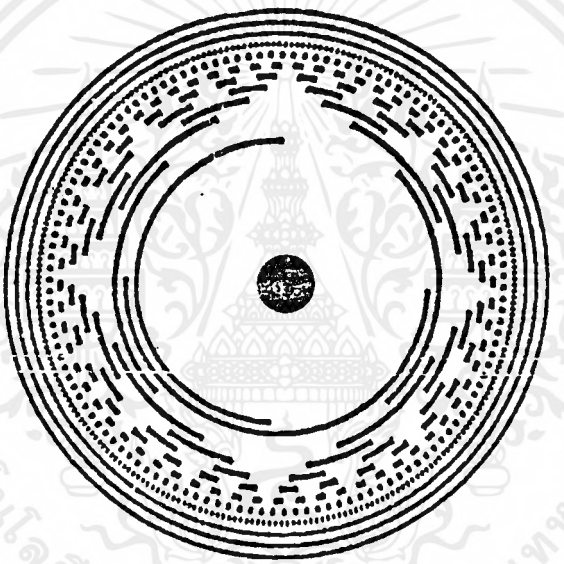


รูปที่ 2.7 วงจรความถี่คูณ 4

2.4 เอนโค้ดเดอร์แบบแอบโซลูท (Absolute encoder)

ลักษณะทางกายภาพของเอนโค้ดเดอร์แบบนี้จะคล้ายกับเอนโค้ดเดอร์แบบอินครีเมนทอล โดยรูปแบบของจานหมุนที่มีรูเป็นไปดังรูปที่ 2.10 แต่เอาท์พุทของเอนโค้ดเดอร์แบบนี้ไม่ใช่ลักษณะคลื่นรูปสี่เหลี่ยม แต่จะเป็นลักษณะบิต 0, 1 โดยจำนวนบิตสัมพันธ์กับจำนวนช่วงที่เป็นชั้น ๆ ที่มีจุดศูนย์กลางร่วมกันบนจานหมุนเดียวกัน ชั้นที่อยู่นอกสุดช่องจะเล็กที่สุด

ค่าความคลาดเคลื่อนในการตรวจจับตำแหน่งของเอนโค้ดเดอร์แบบแอบโซลูทจะมีค่าน้อยกว่าเอนโค้ดเดอร์แบบอินครีเมลทอล หรือแสดงว่ามีความแม่นยำมากนั่นเอง

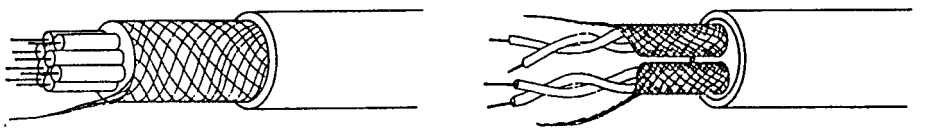


รูปที่ 2.8 แสดงจานหมุนของเอนโค้ดเดอร์แบบแอบโซลูท

โดยในโครงงานที่ศึกษาจะใช้เอนโค้ดเดอร์แบบอินครีเมนทอลซึ่งแสดงลักษณะไว้ในภาคผนวก

2.5 ลักษณะสายเอาท์พุทของเอนโค้ดเดอร์

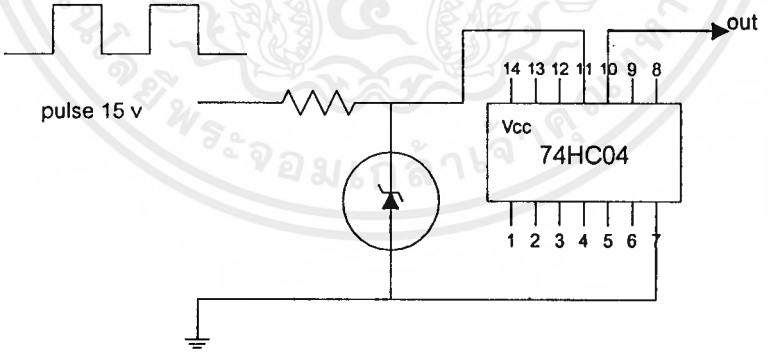
สายส่งสัญญาณควรเลือกพิจารณาเกี่ยวกับการป้องกันสัญญาณรบกวน อาจใช้สายชีลด์หรือสาย twisted pair with shield ก็ได้ ดังรูป 2.9



รูปที่ 2.9 ลักษณะสายชีลด์

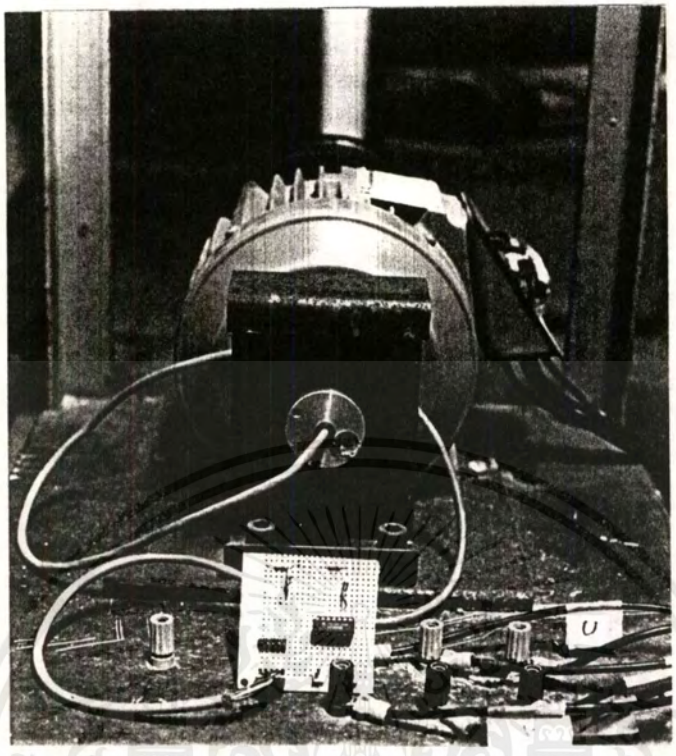
จากที่กล่าวมาแล้วว่า เอาท์พุทของเอนโค้ดเดอร์ที่ได้จะมีลักษณะเป็นขบวนพัลส์นั้น แต่ขนาดของสัญญาณพัลส์ที่ได้จะมีขนาดตามค่าไฟเลี้ยงที่ป้อนให้กับตัวเอนโค้ดเดอร์ ซึ่งค่าไฟเลี้ยงของเอนโค้ดเดอร์แต่ละแบบก็ขึ้นกับลักษณะเฉพาะของเอนโค้ดเดอร์แต่ละตัว

ในโครงงานใช้เอนโค้ดเดอร์ที่มีลักษณะตามภาคผนวก ซึ่งจะทำการจ่ายไฟเลี้ยงให้กับมันขนาด 15 โวลต์ ดังนั้นขนาดสัญญาณพัลส์เอาท์พุทที่ได้ จะมีขนาด 15 โวลต์ด้วย แต่เนื่องจากเราจะนำสัญญาณพัลส์นี้ไปเข้าตัวนับของตัวไมโครคอนโทรลเลอร์ ทำให้ต้องทำการลดขนาดสัญญาณพัลส์ก่อน ซึ่งได้ทำการต่อวงจรดังรูป 2.10



รูปที่ 2.10 วงจรปรับขนาดสัญญาณพัลส์

จากรูปวงจรลักษณะกราฟที่ได้นั้นก็จะเป็นสัญญาณพัลส์ขนาด 5 โวลต์พร้อมที่จะเข้าไปที่ตัวไมโครคอนโทรลเลอร์ได้



รูปที่ 2.11 แสดงการคัปปลิ้งระหว่างเอนโค้ดเดอร์กับมอเตอร์ที่ใช้ในโครงงาน

บทที่ 3
ทฤษฎีและการใช้งานไมโครคอนโทรลเลอร์ 68HC11 เบื้องต้น

จากที่ได้กล่าวมาแล้วว่าในโครงงานนี้ได้ใช้ตัวไมโครคอนโทรลเลอร์เบอร์ 68HC11 เป็นตัวประมวลผลหลัก ดังนั้นเนื้อหาในบทนี้จะขอกล่าวถึงคุณสมบัติ,โหมดการใช้งานแบบต่าง ๆ ของไมโครคอนโทรลเลอร์ตัวนี้อย่างคร่าว ๆ

3.1 คุณสมบัติทางฮาร์ดแวร์และซอฟแวร์ของ MCS 68HC11

3.1.1 คุณสมบัติทางฮาร์ดแวร์

- ซีพียูขนาด 8 บิต
- มีหน่วยความจำROMภายในสำหรับเก็บโปรแกรมขนาด 8 กิโลไบต์
- มีหน่วยความจำEEPROMภายในขนาด 512 ไบต์
- มีหน่วยความจำRAMภายในขนาด 256 ไบต์
- มีTimer(ตัวตั้งเวลา)และCounter(ตัวนับ)ขนาด 8 บิต
- มีวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล 8 บิต 8 ช่อง
- มีวงจรพัลส์แอกคิวมูเลเตอร์ขนาด 8 บิต
- มีส่วนติดต่อสื่อสารข้อมูลผ่านทางพอร์ตอนุกรม
- มีวงจรเชื่อมต่ออุปกรณ์อนุกรม
- มีวงจรเรียลไทม์อินเตอร์รัปต์
- มีระบบวอตช์ด็อก
- สามารถขยายตัวตั้งเวลาเป็นระบบ 16 บิตได้
- บรรจุในตัวถังแบบ PLCC ขนาด 52 ขา
- มีระบบอินเตอร์รัปต์ 2 ระดับ
- สามารถติดต่อหน่วยความจำภายนอกได้ 64 กิโลไบต์

3.1.2 คุณสมบัติทางซอฟแวร์

- มีชุดคำสั่งเพิ่มเติมมากกว่าเบอร์ 6800 และ 6801 จึงสามารถใช้ชุดคำสั่งชุดเดียวกับ 6800 หรือ 6801 ได้
- สามารถทำการหารเลข 16 บิตโดยได้ผลลัพธ์เป็นตัวเลข 16 บิต และเศษขนาด 16 บิต
- สามารถประมวลผลข้อมูลละเอียดถึงระดับบิต
- มีโหมดการทำงาน WAIT และโหมด STOP เพื่อประหยัดพลังงาน

68HC11 เป็นชิปไมโครคอนโทรลเลอร์ ที่ได้รับการผลิตโดยใช้เทคโนโลยี HCMOS (High density CMOS) จึงทำให้มีความเร็วในการทำงานสูง ในขณะที่กินกำลังงานไฟฟ้าต่ำ

3.2 สถาปัตยกรรมของ 68HC11

ไมโครคอนโทรลเลอร์ 68HC11 สามารถแสดงด้วยบล็อกไดอะแกรมดังรูปที่ 3.1 ซึ่งภายในชิปประกอบด้วยส่วนหลัก ๆ 6 ส่วนคือ

- ซีพียู
- หน่วยความจำ
- แอกคิวมูเลเตอร์-วอตช์ด็อก-ตัวตั้งเวลาหลัก
- พอร์ตอินพุต เอาท์พุต
- วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
- สโตรบ/แฮนด์เชก

ซีพียู ในส่วนนี้มีส่วนประกอบย่อยอีก 3 ส่วนคือ ส่วนควบคุมโหมดการทำงานของชิป (MODE) ส่วนกำเนิดสัญญาณนาฬิกา(CLK) และส่วนลอจิกอินเตอร์รัปต์(INT) โดยทั้งสามส่วนนี้จะได้รับการควบคุมการทำงานจากแก่นซีพียู

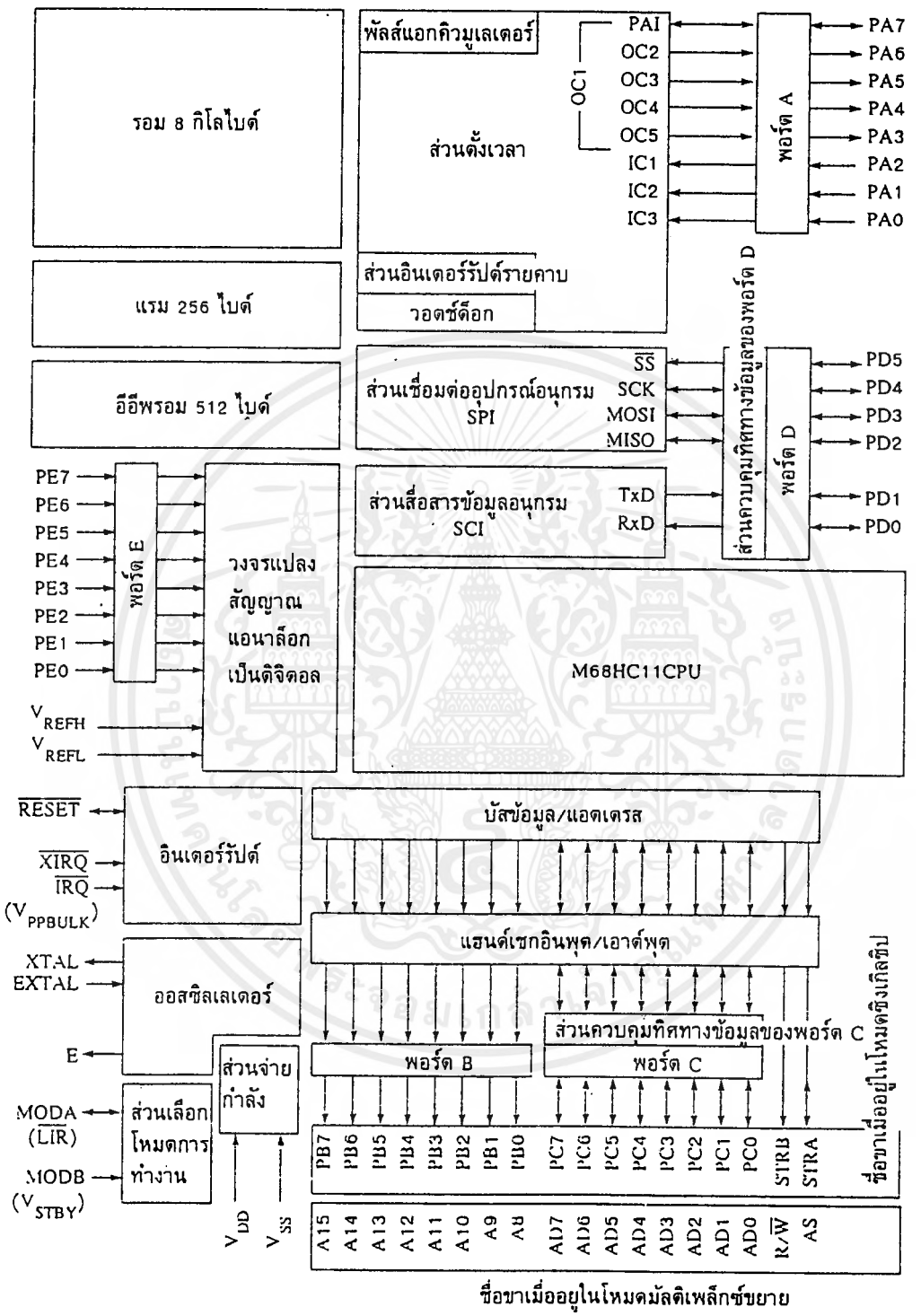
หน่วยความจำ ภายในชิป MC68HC11A8 จะมีหน่วยความจำครบทั้ง 3 แบบคือ ROM RAM และ EEPROM ซึ่งการต่อและเรียกใช้หน่วยความจำเบอร์นี้จะขึ้นอยู่กับส่วนซีพียูเป็นหลัก

แอกคิวมูเลเตอร์-วอตช์ดอก-ตัวตั้งเวลาหลัก ในส่วนนี้จะมีการติดต่อกับพอร์ต A โดยใช้7พอร์ต A เป็นทางผ่านของข้อมูล พัลส์แอกคิวมูเลเตอร์จะใช้พอร์ต PA7 ในขณะที่ส่วนตัวตั้งเวลาหลักจะใช้ PA3-PA6 ในชิปยังมีวงจรวอตช์ด็อก เพื่อช่วยให้ชิปสามารถทำงานได้อย่างต่อเนื่องแม้ว่าจะมีรีเซตระบบอยู่บ่อย ๆ ก็ตาม

พอร์ตอินพุท เอาท์พุท จะมีพอร์ตอินพุท เอาท์พุทด้วยกัน 5 พอร์ตดังนี้ พอร์ต A เป็นทั้งพอร์ตอินพุทและเอาท์พุท โดยมีการทำงานแยกกันคือ PA7 เป็นพอร์ตที่สามารถส่งผ่านข้อมูลได้2 ทิศทาง ในขณะที่ PA3-PA6 เป็นพอร์ตเอาท์พุทของตัวตั้งเวลาหลัก และPA0-PA2 เป็นพอร์ตอินพุท พอร์ต B เป็นพอร์ตเอาท์พุท ในขณะที่พอร์ต C เป็นพอร์ตอินพุท เอาท์พุท สามารถส่งข้อมูลได้ 2 ทิศทาง พอร์ต D เป็นพอร์ตที่ใช้ในการถ่ายทอดสัญญาณ จากส่วนเชื่อมต่ออุปกรณ์อนุกรมและส่วนสื่อสารข้อมูลอนุกรม พอร์ต E เป็นพอร์ตอินพุทสำหรับวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล

วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล เป็นวงจรที่ช่วยเสริมให้ชิปไมโครคอนโทรลเลอร์มีประสิทธิภาพมากขึ้น มีขนาด 8 บิต

สโตรบ/แฮนด์เชก ในส่วนนี้จะทำงานร่วมกับส่วนขยายบัสด้วย ทั้งนี้เพื่อให้ซีพียูสามารถทำงานได้กับแอดเดรสถึง 16 บิต ที่ส่วนนี้จะมีสัญญาณที่สำคัญ ๆ อยู่ 2 สัญญาณ คือ STRB/R/W และ STRA/AS โดยทั้งสองสัญญาณคือ สัญญาณสโตรบเพื่อให้ชิปทำการอ่านเขียนข้อมูลได้และที่ส่วนนี้ยังมีวงจรแฮนด์เชก เพื่อตรวจสอบความพร้อมในการรับข้อมูลของชิปกับอุปกรณ์ภายนอก



รูปที่ 3.1 บล็อกไดอะแกรมภายในของ MC68HC11

3.2.1 โหมดการทำงานของ 68HC11

68HC11 มีโหมดการทำงาน 4 โหมด ดังนี้

1. โหมดซิงเกิลชิป (Single chip Operating Mode)

2. โหมดมัลติเพล็กซ์ขยาย (Expanded Multiplexed Operating Mode)

3. โหมดบูตสแตร์ปพิเศษ (Special Bootstrap Operating Mode)

4. โหมดทดสอบพิเศษ (Special test Operating Mode)

ซึ่งในโครงงานจะใช้โหมดมัลติเพล็กซ์ขยาย ดังนั้นจะขอกล่าวเฉพาะโหมดนี้

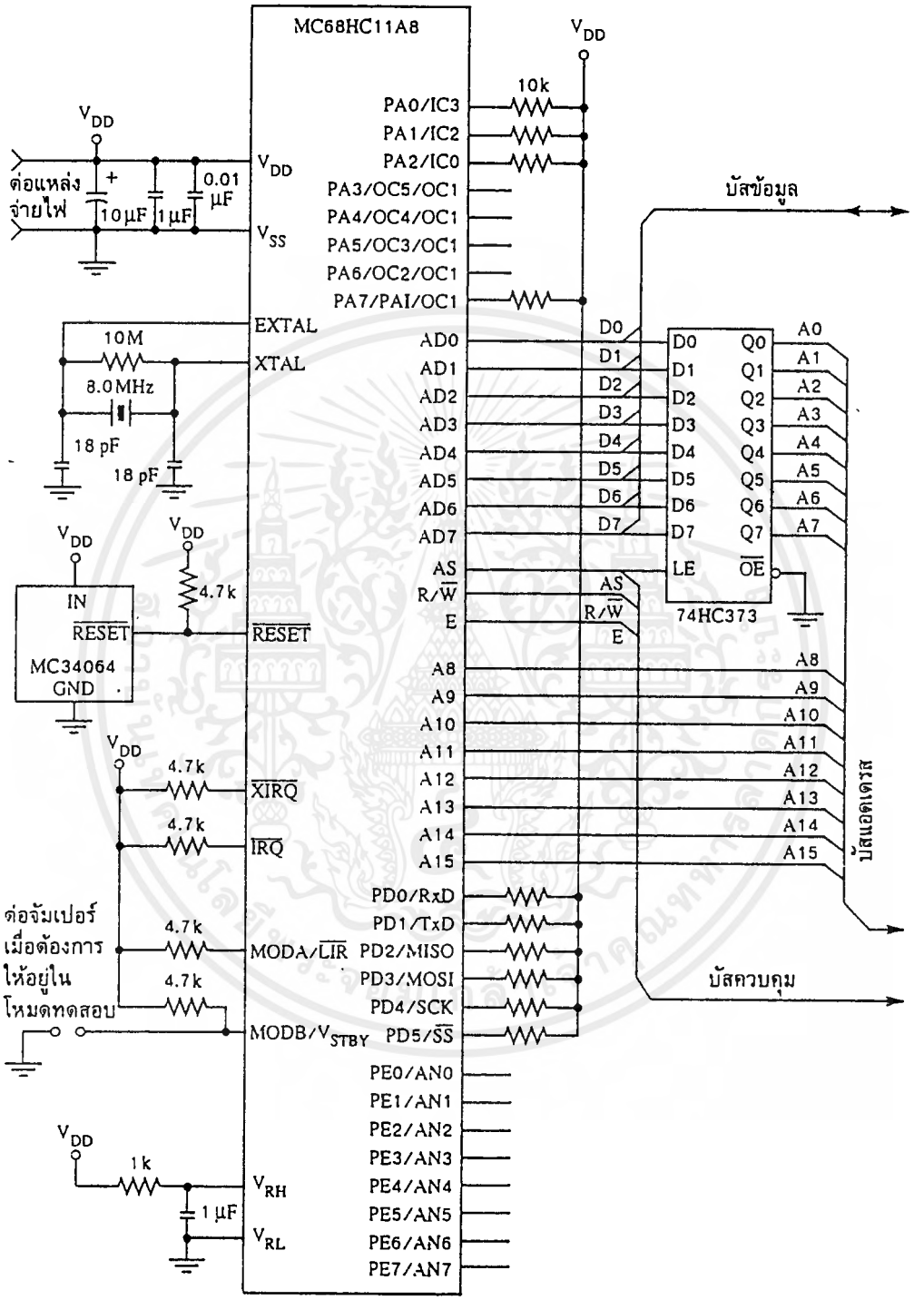
โหมดมัลติเพล็กซ์ขยาย

การทำงานของ 68HC11 ในโหมดนี้โปรแกรมการควบคุมการทำงานของซีพียูจะอยู่ในหน่วยความจำภายนอก โดยที่พอร์ต B และ C ของ68HC11 ทำหน้าที่เป็นบัสแอดเดรสและข้อมูลสังเกตจากบล็อกไดอะแกรมภายในของ 68HC11 ในรูปที่ 3.1 พอร์ต B ทำหน้าที่เป็นบัสแอดเดรสไบต์สูง ส่วนพอร์ต C ทำหน้าที่เป็นบัสแอดเดรสไบต์ต่ำและบัสข้อมูล

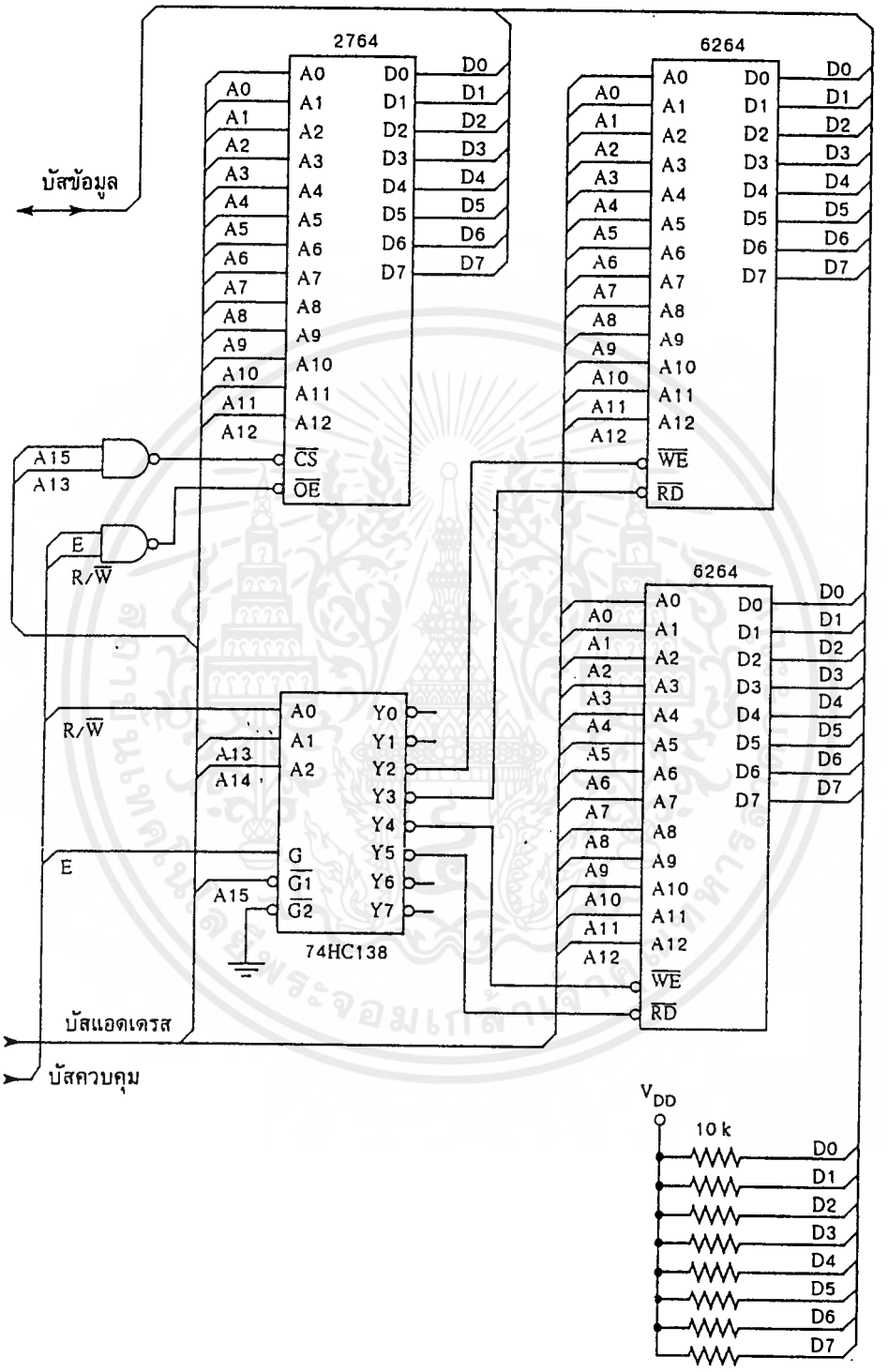
การที่พอร์ตC จะสามารถเป็นทั้งบัสแอดเดรสและบัสข้อมูลได้นั้น ต้องใช้กระบวนการที่เรียกว่า การมัลติเพล็กซ์ การแยกสัญญาณแอดเดรสและข้อมูลมาใช้งาน จะต้องต่อวงจรเพื่อทำการดีมัลติเพล็กซ์ภายนอก

สัญญาณควบคุมที่นำมาใช้ดีมัลติเพล็กซ์สัญญาณแอดเดรสและข้อมูล ได้แก่ สัญญาณ STROBE B/R/W (STRB/R/W) และสัญญาณ STROBE A/ADDRESS (STRA/AS)

ในโหมดนี้ชิป 68HC11 จะสามารถติดต่อกับหน่วยความจำภายนอกเพิ่มได้สูงสุดถึง 64 กิโลไบต์ แต่ก็มีข้อเสียคือต้องใช้อุปกรณ์ต่อเพิ่มมากไม่ว่าจะเป็นหน่วยความจำชิปพอร์ตอินพุทเอาท์พุททำให้ค่าใช้จ่ายของระบบสูงขึ้น แต่อย่างไรก็ตาม โหมดนี้เป็นโหมดที่นิยมใช้งานมากที่สุดเพราะสามารถต่อขยายหน่วยความจำและพอร์ตได้มากมายนั่นเอง รูปที่ 3.2 และ 3.3 เป็นการต่อ 68HC11 ในโหมดมัลติเพล็กซ์ขยาย



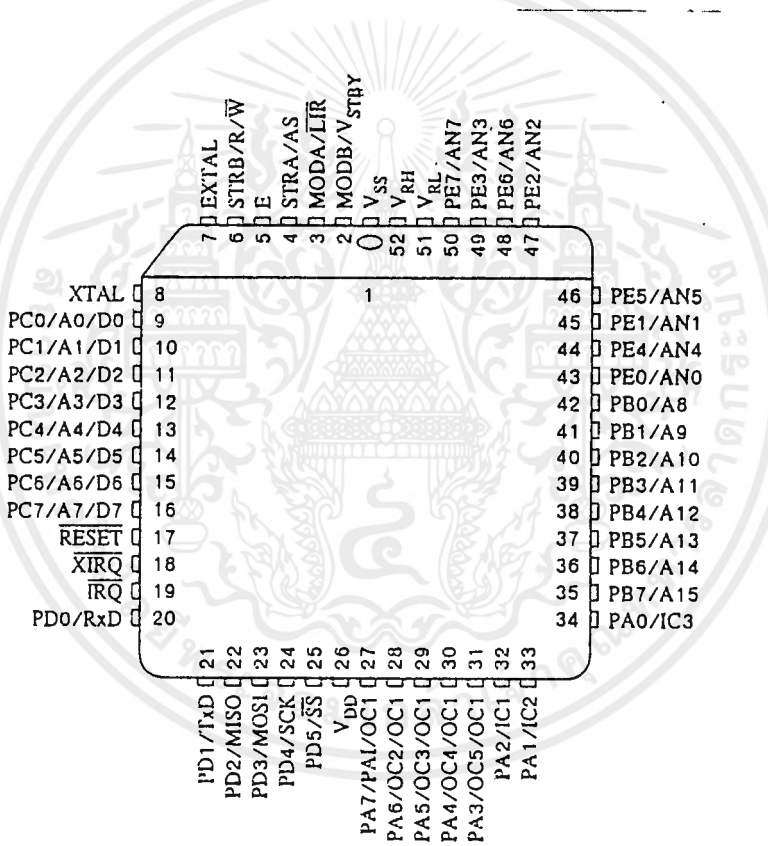
รูปที่ 3.2 ตัวอย่างวงจรการต่อ 68HC11 เมื่อทำงานในโหมดมัลติเพล็กซ์ขยาย



รูปที่ 3.3 ตัวอย่างวงจรการต่อ 68HC11 เมื่อทำงานในโหมดมัลติเพล็กซ์ขยาย(ต่อ)

3.2.2 รายละเอียดสัญญาณและการจัดขาของ 68HC11

การจัดขาของ 68HC11 นั้นมีด้วยกันอยู่หลายลักษณะ อยู่ในตัวถังต่าง ๆ กันไป เช่น เป็นตัวถังแบบ DIP ขนาด 48 ขา ส่วนรูปที่ 3.4 เป็นตัวถังแบบ PLCC ขนาด 52 ขา ซึ่งเป็นแบบที่ใช้ในโครงงานด้วย นอกจากนี้แล้วยังมีแบบตัวถัง QFP ขนาด 64 ขา อีกด้วย



รูปที่ 3.4 แสดงการจัดขา 68HC11 แบบ PLCC 52 ขา

รายละเอียดของขาสัญญาณต่าง ๆ มีดังนี้

ขาไฟเลี้ยงและกราวด์ (V_{DD} , V_{SS})
เป็นขาที่ใช้สำหรับจ่ายแรงดันเพื่อให้ 68HC11 ทำงานโดยขา V_{DD} ต้องเป็นแรงดัน +5 โวลต์ ในขณะที่ขา V_{SS} เป็นกราวด์

ขา RESET
เป็นขาอินพุทรับสัญญาณเพื่อให้ 68HC11เริ่มต้นทำงานใหม่อันอาจเกิดจากสภาวะที่ซีพียูเพิ่งได้รับไฟเลี้ยงให้เริ่มทำงาน หรือเกิดจากการทำงานภายในวงจรเกิดความผิดพลาด

ขาXTAL , EXTAL
ทั้ง 2 ขา นี้ถูกจัดไว้ให้ต่อกับคริสตอล หรือวงจรกำเนิดสัญญาณนาฬิกา เพื่อควบคุมวงจรกำเนิดสัญญาณนาฬิกาในชิป ความถี่ที่ปรากฏอยู่ที่ขานี้มีค่าสูงกว่าที่ขา E อยู่ประมาณ 4 เท่า

ขา E
เป็นขาเอาท์พุทของวงจรกำเนิดสัญญาณนาฬิกาภายในชิป สามารถใช้เป็นฐานเวลาอ้างอิงได้โดย ความถี่ที่ออกจากขานี้จะมีค่า 1/4 เท่าของความถี่อินพุทที่ขา XTAL และ EXTAL เมื่อใดที่ขา E นี้มีสภาวะลอจิกเป็น “0” หมายความว่า ขณะนี้ซีพียูกำลังทำการประมวลผลภายในอยู่ ถ้าหากเป็น “1” หมายถึง ขณะนี้ข้อมูลได้รับการเรียกใช้จากซีพียูถ้าหากไมโครคอนโทรลเลอร์อยู่ในโหมด STOP สัญญาณที่ขา E จะไม่มีออกมาหรือเกิดสภาวะ High impedance

ขา IRQ (Interrupt Request)
เป็นขาอินพุทสำหรับเรียกการอินเตอร์รัปต์แบบอะซิงโครนัสของชิป 68HC11 โดยการรับสัญญาณเพื่ออินเตอร์รัปต์นี้สามารถจะเลือกได้ว่า จะรับสัญญาณขอบขาลง(Negative edge) หรือจะรับการทริกเป็นแบบระดับสัญญาณ โดยปกติจะทำงานที่ระดับลอจิก “0” เมื่อต่อใช้งานปกติต้องต่อตัวต้านทานค่า 4.7 กิโลโอห์มพูลอัปเข้ากับไฟเลี้ยง

ขา XIRQ (Non-Maskable Interrupt)

เป็นขาอินพุทสำหรับตอบรับการอินเตอร์รัปต์แบบ นอน-มาสเคเบิลหลังจากที่มีการรีเซตซีพียู สามารถรับการอินเตอร์รัปต์ด้วยการทริกแบบระดับสัญญาณ โดยทำงานที่ลอจิก "0" และต้องต่อตัวต้านทานพูลอัปกับไฟเลี้ยงเข้าที่ขา XIRQ นี้ด้วย ในขณะที่ไม่มีการส่งสัญญาณอินเตอร์รัปต์

ขา MODA/LIRและMODB/V_{STBY}(Mod A/load instruction register และ ModeB/Standby voltage) หลังจากที่มีการรีเซตไมโครคอนโทรลเลอร์ ขานี้(ทั้ง MODA และ MODB) จะถูกใช้ให้เป็นขาสำหรับเลือกโหมดการทำงานของ 68HC11 ซึ่งเลือกได้เพียง 1 โหมดจาก 4 โหมดตามที่ได้กล่าวมาแล้ว โดยเลือกโหมดตามตารางที่ 3.1

MODA	MODB	โหมดการทำงาน
1	0	ซิงเกิลชิป
1	1	มัลติเพล็กซ์ขยาย
0	0	บูตสแตรปพิเศษ
0	1	ทดสอบพิเศษ

ตารางที่ 3.1 การเลือกโหมดการทำงานของ 68HC11

ขา VRL และ VRH (A/D Converter Reference Voltage)

เป็นขาที่ใช้สำหรับต่อแรงดันอ้างอิงสำหรับวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลภายในชิป 68HC11

ขา STRB/R/W (Strobe B และ Read/Write)

ขานี้จะแสดงตนเป็นขา R/W ใช้ในการควบคุมทิศทางของการถ่ายเทข้อมูลกับบัสข้อมูลภายนอกชิป ถ้าขานี้เป็นลอจิก "0" จะหมายความว่า ขณะนี้ซีพียูกำลังทำการเขียนข้อมูลลงไปในบัสข้อมูลภายนอก แต่ถ้าเป็นลอจิก "1" จะเป็นการแสดงว่า ขณะที่กำลังอ่านข้อมูลเข้ามาเพื่อทำการประมวลผล

ขา STRA/AS (Strobe A และ Address Strobe)

เป็นขาอินพุทโดยรับสัญญาณเป็นแบบขอบขาของสัญญาณ ขานี้จะเปลี่ยนลักษณะการทำงานเป็นขาเอาท์พุท ASโดยใช้ในวงจรดีมัลติเพล็กซ์สัญญาณข้อมูลกับสัญญาณแอดเดรสที่พอร์ต C

ขาสัญญาณของพอร์ต

พอร์ต A, D, E จะเป็นพอร์ตที่ไม่ขึ้นกับโหมดการทำงานของ 68HC11นั่นคือ ไม่ว่า 68HC11จะทำงานในโหมดใดก็ตาม พอร์ตทั้ง 3 ยังคงมีลักษณะสัญญาณและการทำงานเช่นเดิมขณะที่พอร์ต B โหมดมัลติเพล็กซ์ขยายจะกลายเป็นขาแอดเดรสไบต์สูง ส่วนพอร์ต C จะใช้เป็นบัสมัลติเพล็กซ์ระหว่างแอดเดรส 8 บิตต่ำกับบัสข้อมูล

3.2.3 รีจิสเตอร์ของซีพียู

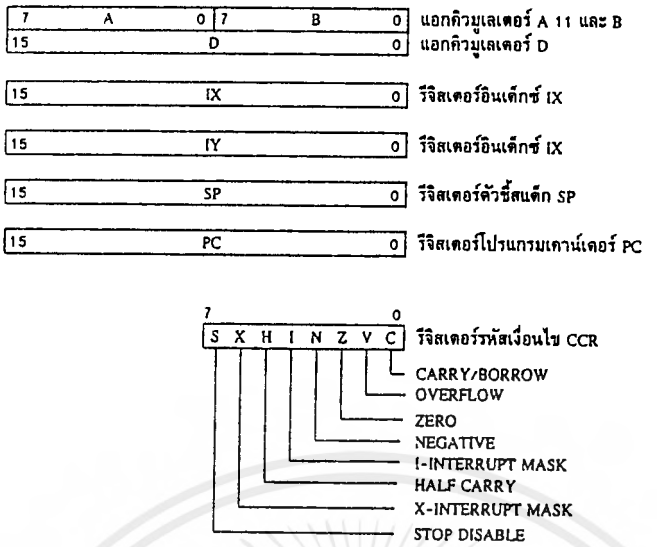
ซีพียูของไมโครคอนโทรลเลอร์ 68HC11 จะมีรีจิสเตอร์ใช้งานอยู่ 7 ตัว อันได้แก่

1. แอกคิวมูเลเตอร์ A และ B
2. แอกคิวมูเลเตอร์ D
3. รีจิสเตอร์อินเด็กซ์ IX
4. รีจิสเตอร์อินเด็กซ์ IY
5. รีจิสเตอร์ตัวชี้สแต็ก(Stack Pointer : SP)
6. รีจิสเตอร์โปรแกรมเคาน์เตอร์(Program Counter : PC)
7. รีจิสเตอร์รหัสเงื่อนไข(Condition Code Register : CCR)

โดยมีลักษณะของรูปแบบการจัดจำนวนบิตและความหมายดังในรูปที่ 3.5

แอกคิวมูเลเตอร์ A และ B

เป็นรีจิสเตอร์ขนาด 8 บิต ที่ใช้ในการเก็บค่าโอเปอแรนด์และผลของการคำนวณทางคณิตศาสตร์หรือผลจากการจัดการข้อมูลโดยตัวซีพียู ในการประมวลผลทางคณิตศาสตร์หรือลอจิกจะต้องนำข้อมูลเหล่านั้นมาเก็บในรีจิสเตอร์ทั้ง 2 ตัวนี้ จึงจะสามารถประมวลผลได้



รูปที่ 3.5 การจัดจำนวนบิตและความหมายของรีจิสเตอร์ของซีพียู

แอกคิวมูเลเตอร์ D

เป็นรีจิสเตอร์ขนาด 16 บิต ใช้ในการประมวลผลและเก็บค่าจากการคำนวณทางคณิตศาสตร์และลอจิก แอกคิวมูเลเตอร์ D ก็เกิดจากการรวมกันของแอกคิวมูเลเตอร์ A และ B จึงทำให้มีขนาด 16 บิต

รีจิสเตอร์อินเด็กซ์ IX

เป็นรีจิสเตอร์ขนาด 16 บิตใช้ในการชี้ตำแหน่งแอดเดรส เพื่อเข้าไปจัดการประมวลผลกับข้อมูลในแอดเดรสนั้น ๆ นอกจากนั้น IX สามารถใช้เป็นตัวนับหรือรีจิสเตอร์เก็บข้อมูลชั่วคราวได้ด้วย

รีจิสเตอร์อินเด็กซ์ IY

เป็นรีจิสเตอร์ขนาด 16 บิตที่มีหน้าที่เหมือน IX แต่จะแตกต่างกันตรงที่ในทุกคำสั่งที่ต้องเกี่ยวข้องกับ IY

รีจิสเตอร์ตัวชี้สแต็ก

เป็นรีจิสเตอร์ขนาด 16 บิต ใช้เก็บแอดเดรสบนสแต็ก(stack) โดยสแต็กใน 68HC11 นี้จะมีลักษณะการเก็บข้อมูลเข้าและนำข้อมูลออกมาเป็น LIFO (Last-In-First-Out) หรือข้อมูลที่เข้าไปเก็บในสแต็กหลังสุด เมื่อเรียกออกมาจะถูกเรียกออกมาก่อน

รีจิสเตอร์โปรแกรมเคาน์เตอร์

เป็นรีจิสเตอร์ 16 บิต ใช้เก็บค่าของแอดเดรสของคำสั่งถัดไป ที่ซีพียูจะไปทำการเอ็กซคิวต์

รีจิสเตอร์รหัสเงื่อนไข

เป็นรีจิสเตอร์ขนาด 8 บิต ในแต่บิตจะแสดงความหมายของผลจากการกระทำคำสั่งที่เพิ่งจะเอ็กซีคิวต์ไปของซีพียู แต่ละบิตของรีจิสเตอร์นี้เป็นอิสระต่อกัน จึงสามารถตรวจสอบสถานะได้โดยใช้โปรแกรม และยังสามารถนำผลการตรวจสอบนั้นไปดำเนินการต่อได้เลย
รายละเอียดของแต่ละบิตในริจิสเตอร์รหัสเงื่อนไขมีดังนี้

บิต Carry/Borrow (C) : บิตนี้จะเซตเมื่อซีพียูทำการประมวลผลทางคณิตศาสตร์แล้วเกิดการทดค่า(carry) หรือยืมค่า(borrow) บิตนี้สามารถที่จะใช้งานร่วมกับคำสั่งการหมุน(rotate) และเลื่อน(shift)ข้อมูลได้ หรือที่เรียกบิตนี้อีกอย่างหนึ่งว่า บิตทด

บิต Over Flow (V) : บิตนี้จะเซตเป็น "1" เมื่อซีพียูกระทำคำสั่งคณิตศาสตร์แล้วเกิดค่าที่เกินออกมา นอกเหนือจากเงื่อนไขดังกล่าวบิตนี้จะเป็น "0"

บิต Zero (Z) : บิตนี้จะเซตเมื่อผลของการกระทำทางคณิตศาสตร์ หรือลอจิกหรือการประมวลผลข้อมูลแล้วทำให้เกิดเป็นค่าศูนย์ขึ้นมา

บิต Negative (N) : ถ้าหากผลของการกระทำทางคณิตศาสตร์หรือลอจิกหรือการประมวลผลข้อมูลแล้วทำให้เกิดค่าลบบิตนี้จะเซต ผลลัพธ์ที่เป็นลบสามารถสังเกตได้จากบิตที่มีนัยสำคัญสูงสุด(MSB) มีค่าเป็น "1"

บิต I-Interrupt (I) : สามารถเซตบิตนี้ให้เป็น "1" ได้ 2 วิธีคือ โดยวิธีทางฮาร์ดแวร์และโดยการใช้คำสั่งที่ใช้ในการดิสเอเบิลการอินเตอร์รัปต์แบบมาสเคเบิลทั้งภายในและภายนอก

บิต Half Carry (H) : จะเซตเป็น "1" เมื่อซีพียูกระทำคำสั่งทางคณิตศาสตร์ เช่น ADD,ABA,ADC แล้วเกิดการทดข้ามจากบิตที่3 มายังบิตที่4

บิต X-Interrupt Mask (X) : บิตนี้จะถูกเซตด้วยวิธีการทางฮาร์ดแวร์เท่านั้น โดยการป้อนสัญญาณเข้าที่ขา RESET และ XIRQ และจะรีเซตบิตนี้ด้วยการใช้คำสั่ง TAP และ RTI เท่านั้น

บิต Stop Disable (S) : บิตนี้จะเซตเมื่อต้องการดิสเอเบิลคำสั่ง STOP และถ้ารีเซตบิตนี้ก็จะเป็นการอีนาเบิลคำสั่ง STOP บิตนี้ถูกควบคุมโดยโปรแกรม และเมื่อบิตนี้ถูกเซตจะทำให้คำสั่ง STOP มีผลเช่นเดียวกับคำสั่ง NOP

3.3 การอ้างแอดเดรสและชุดคำสั่งของ 68HC11

การอ้างแอดเดรสของ 68HC11 มีกระบวนการอ้างแอดเดรส(Addressing) ทั้งสิ้น 6 โหมดด้วยกัน ประกอบด้วย

1. การอ้างแอดเดรสแบบทันทีทันใด(Immediate Addressing)

เป็นการติดต่อเพื่อจัดการข้อมูล ซึ่งเป็นค่าใด ๆ โดยตรงในทันทีทันใด จำนวนไบต์ของคำสั่งในการอ้างแอดเดรสแบบนี้จะมีขนาดตั้งแต่ 2-4 ไบต์ ขึ้นอยู่กับขนาดของรีจิสเตอร์ที่ต้องเกี่ยวข้องด้วย รูปแบบของคำสั่งที่มีการอ้างแอดเดรสแบบนี้ หลังจากคำสั่งแล้วต้องตามด้วยเครื่องหมาย #

2. การอ้างแอดเดรสแบบโดยตรง(Direct Addressing)

เป็นการติดต่อเพื่อประมวลผลข้อมูลขนาด 8 บิต ที่อยู่ในความจำRAM ภายในชิป ซึ่งมีอยู่ 256 ไบต์ โดยมีแอดเดรสตั้งแต่ $0000-$00FF ดังนั้นค่าโอเปอแรนด์ที่ตามหลังออปโค้ดคำสั่งก็คือค่าแอดเดรสของRAM นั่นเอง

3.การอ้างแอดเดรสแบบขยาย(Extended Addressing)

ข้อมูลในไบต์ที่2 และ3 ที่ตามหลังออปโค้ดจะเก็บค่าแอดเดรสจริงของหน่วยความจำที่ต้องการนำข้อมูลในหน่วยความจำออกมาประมวลผลหรือจัดเก็บข้อมูลลงไปใหม่ เมื่อใช้การอ้างแอดเดรสแบบนี้คำสั่งจะมีขนาด 3-4 ไบต์โดยเป็นออปโค้ดไบต์ที่1 หรือ 2 (กรณีถ้ามีขนาด 4 ไบต์) ส่วน 2 ไบต์หลังจะเป็นค่าแอดเดรสที่อ้างถึงในหน่วยความจำ

4. การอ้างแอดเดรสแบบอินเด็กซ์(Index Addressing)

จะมีการนำรีจิสเตอร์ชี้ข้อมูล ทั้ง2ตัว คือ IX และ IY มาใช้ในการคำนวณค่าแอดเดรสที่ต้องการเรียกใช้ข้อมูล ในการอ้างแอดเดรสแบบนี้ สามารถที่จะใช้หน่วยความจำที่ตำแหน่งใดก็ได้ในจำนวน 64 กิโลไบต์เป็นจุดอ้างอิง

5. การอ้างแอดเดรสแบบอินเฮียเรนต์(Inherent Addressing)

การอ้างแอดเดรสแบบนี้จะไม่ยุ่งเกี่ยวกับข้อมูลในหน่วยความจำแต่อย่างใด แต่จะเข้าไปจัดการในรีจิสเตอร์แทน ดังนั้นขนาดของคำสั่งในการอ้างแอดเดรสแบบนี้จะมีเพียง 1-2 ไบต์โดยเป็นออปโค้ดทั้งสิ้นไม่มีโอเปอแรนด์

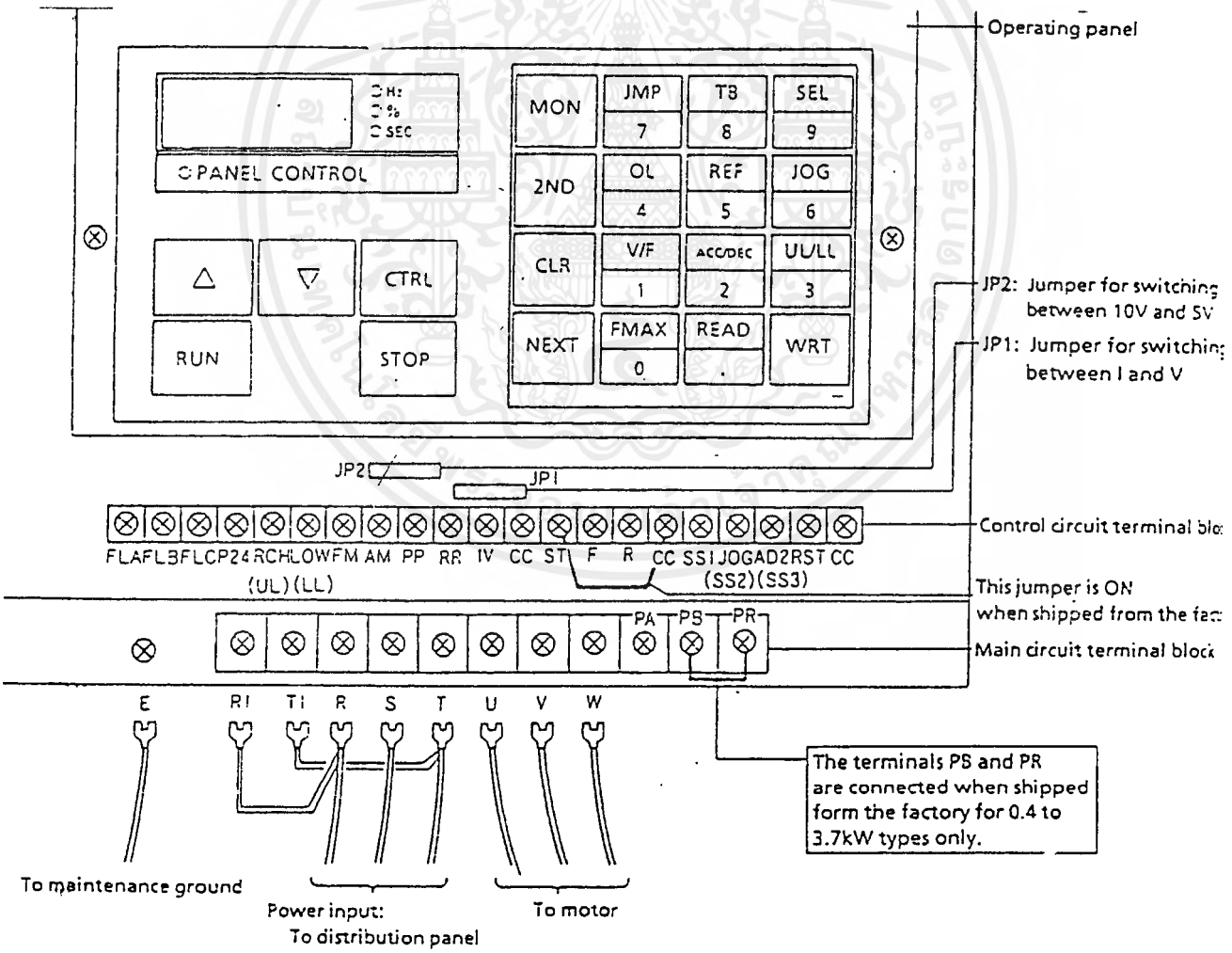
6. การอ้างแอดเดรสแบบสัมพัทธ์(Relative Addressing)

การอ้างแอดเดรสแบบนี้จะใช้ในชุดคำสั่งกระโดด (Jump and Branch Instruction) ถ้าหากเงื่อนไขในการกระโดดเป็นจริง ค่าออฟเซตขนาด 8 บิต ที่อยู่ตามหลังออปโค้ดก็จะถูกบวกเข้าไปในรีจิสเตอร์โปรแกรมเคาน์เตอร์ กำหนดแอดเดรสต่อไปที่จะข้ามไปทำงานของซีพียู

บทที่ 4

การใช้งานเครื่อง VCD อินเวอร์เตอร์

เครื่องอินเวอร์เตอร์ที่ใช้ในโครงงานเพื่อใช้เป็นแหล่งจ่ายไฟให้กับมอเตอร์นั้น เราใช้เครื่องในตระกูล VCD Series ที่ผลิตโดยบริษัท MIKI PULLEY เครื่องนี้สามารถควบคุมได้ทั้งการสั่งที่ตัวเครื่องโดยตรงโดยการกดฟังก์ชันและค่าความถี่ต่าง ๆ ที่ต้องการ หรือจะสามารถควบคุมเครื่องให้จ่ายความถี่มอเตอร์ที่ต้องการได้ โดยอินพุทสัญญาณควบคุมจากภายนอก ซึ่งในโครงงานเราจะใช้วิธีหลัง คือการใช้สัญญาณอนาลอกที่ได้จากวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอกมาเป็นสัญญาณควบคุมจากภายนอก ซึ่งรายละเอียดของเครื่อง VCD อินเวอร์เตอร์ที่ใช้ในโครงงานจะขอกล่าวไว้ในภาคผนวก



รูปที่ 4.1 แสดงหน้าปัดเครื่อง

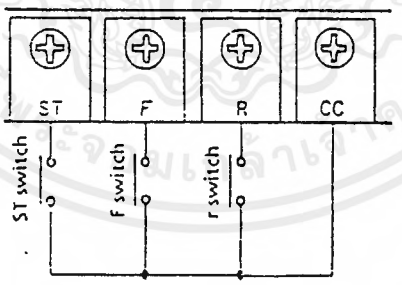
จากรูปที่ 4.1 จะเห็นว่าที่หน้าปัดของเครื่องจะมีคีย์บอร์ดให้เลือกใช้ในหลายฟังก์ชัน ซึ่งคีย์บอร์ดเหล่านี้จะเป็นฟังก์ชันและค่าต่าง ๆ ที่ใช้ในการสั่งที่ตัวเครื่องโดยตรง นอกจากนี้จะยังสังเกตเห็นช่องควบคุมจากสัญญาณภายนอก(Control circuit terminal block) ซึ่งช่องต่าง ๆ เหล่านี้เราจะใช้เป็นอินพุทในการควบคุมเครื่องอินเวอร์เตอร์นี้

4.1 การต่อเครื่องอินเวอร์เตอร์โดยควบคุมจากสัญญาณภายนอก

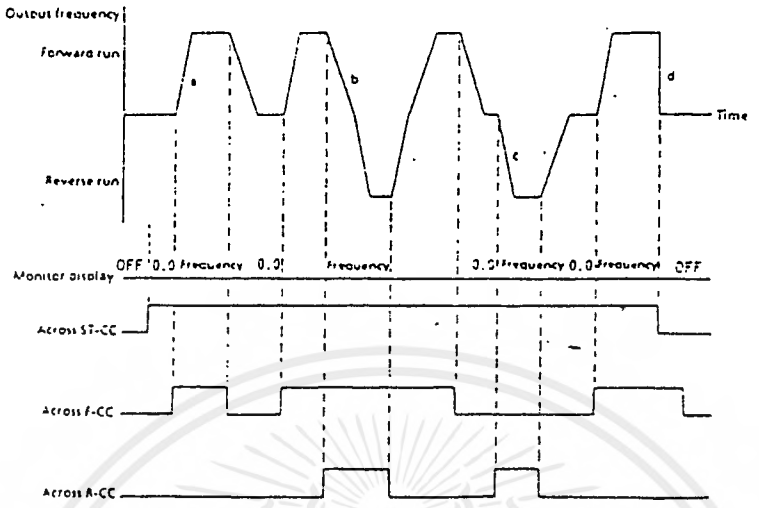
สัญญาณภายนอกที่ใช้ควบคุมอินเวอร์เตอร์นั้น ขึ้นกับว่าจะควบคุมที่ช่องไหน ซึ่งจะเห็นว่ามีช่องต่าง ๆ มากมายนั่นหมายความว่า เราสามารถควบคุมหรือรับค่าต่าง ๆ ของอินเวอร์เตอร์ได้หลายค่า แต่ที่เราใช้ในโครงงานจะมีช่องสำคัญอยู่ 2 ลักษณะ คือ ช่องควบคุมความถี่เอาท์พุทกับช่องควบคุมทิศทางการหมุน

4.2 การควบคุมทิศทางการหมุนจากสัญญาณควบคุมภายนอก

นอกจากเราจะสามารถสั่งให้เครื่องอินเวอร์เตอร์หมุนในทิศทางใดจากคีย์บอร์ดในตัวเครื่องตรง ๆ แล้ว เรายังสามารถต่อควบคุมจากภายนอกโดยใช้ลักษณะสวิตช์ดังรูปที่ 4.2



รูปที่ 4.2 สวิตช์ควบคุมทิศทางการหมุน



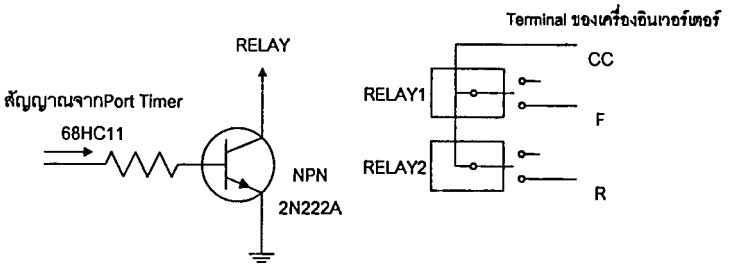
รูปที่ 4.3 Timing chart ของการหมุนไปข้างหน้าและหมุนไปข้างหลัง

ลักษณะการหมุนหรือหยุดขึ้นอยู่กับสถานะของสวิตช์ดังตารางที่ 4.1

TERMINAL			Action
ST	F	R	
OFF	ON/OFF	ON/OFF	output off
ON	OFF	OFF	หยุด
ON	OFF	ON	หมุนไปข้างหลัง
ON	ON	OFF	หมุนไปข้างหน้า
ON	ON	ON	หมุนไปข้างหลัง

ตารางที่ 4.1 แสดงสถานะของสวิตช์และผลลักษณะการหมุนที่ได้

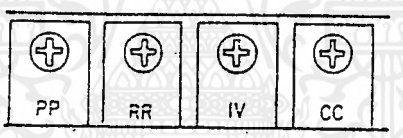
โดยในการต่อวงจรในโครงงานในการควบคุมทิศทางการหมุนของมอเตอร์เราจะต่อวงจรโดยใช้สวิตช์รีเลย์ดังรูปที่ 4.4



รูปที่ 4.4 รูปแสดงการต่อสวิตซ์วงจรรีเลย์ตัดต่อทิศทางการหมุน

4.3 การควบคุมความถี่เอาท์พุทด้วยสัญญาณควบคุมภายนอก

ความถี่เอาท์พุทของเครื่อง VCD อินเวอร์เตอร์นั้นสามารถควบคุมจากอุปกรณ์ภายนอก โดยใช้ช่องควบคุม PP, RR, IV และ CC ดังแสดงในรูปที่ 4.5

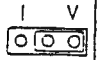
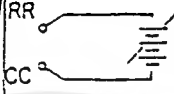
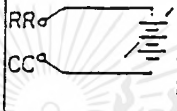
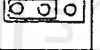
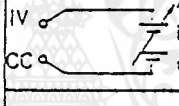
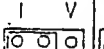
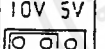
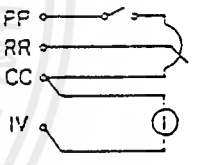


รูปที่ 4.5 ช่องควบคุมความถี่เอาท์พุทจากสัญญาณภายนอก

4.3.1 แบบของสัญญาณควบคุมความถี่เอาท์พุท

ความถี่เอาท์พุทของอินเวอร์เตอร์สามารถปรับเปลี่ยนได้โดยตั้ง Jumper JP1 และ JP2 ที่อยู่ที่บอร์ดควบคุมดังแสดงในรูปที่ 4.1

ลักษณะการผสมของค่า JP1 และ JP2 และช่องควบคุม PP, RR, IV และ CC เพื่อลักษณะสัญญาณควบคุมแบบต่าง ๆ แสดงดังตารางที่ 4.2

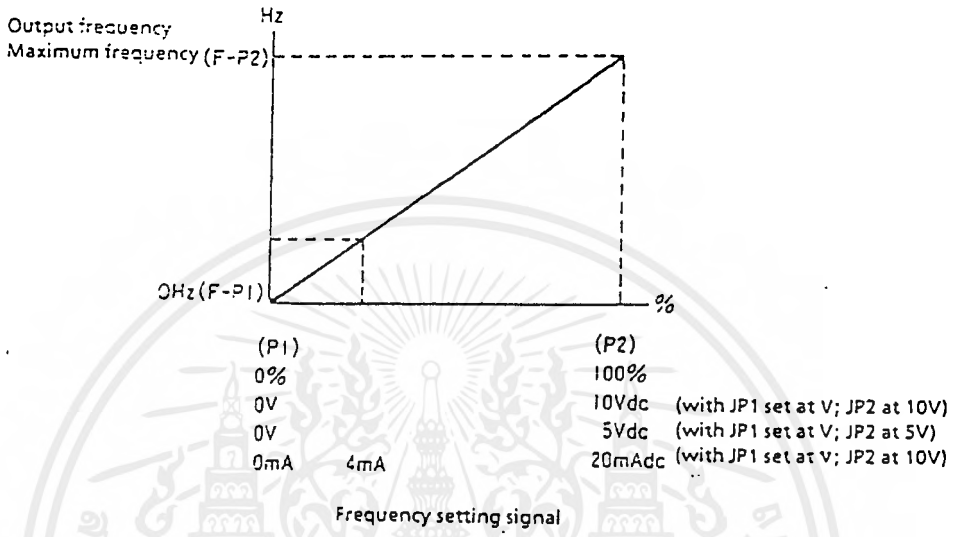
	JP1	JP2	Frequency setting mode	Function	Remarks
1			The frequency setting mode number 2, 3, 6, or 7 engages the panel control mode ("PANEL CONTROL" LED is lit) [Refer to Section 3.23 (page 61).]	Allows the setting of operating frequencies via the inverter's operating panel.	
2				 A 0V to 5Vdc signal is input across terminals RR-CC.	
3				 A 0V to 10Vdc signal is input across terminals RR-CC.	
4			The frequency setting mode number 1, 3, 5, or 7 engages the panel control mode ("PANEL CONTROL" LED is lit). [Refer to Section 3.23 (page 61).]	 A 3kΩ potiometer is connected to PP-RR-CC.	1kΩ ~ 10kΩ-rated potentiometer can be used.
5				 A0 to 5Vdc signal is input across terminals IV-CC.	
6				A0 to 20mA or 4 to 20mA signal is input across terminals IV-CC. 	 When the switch adjacent to terminal PP is turned on, the potentiometer is prioritized. Use a switch designed for weak currents for this circuit.

ตารางที่ 4.2 ลักษณะสัญญาณควบคุมความถี่เอาท์พุทที่สัมพันธ์กับJumperและช่องควบคุม

ซึ่งในโครงงานเราได้เลือกใช้แบบที่ 5 คือ JP1 = V , JP2 = 10V และต่อสัญญาณอนาลอกขนาด 0-5 Vdc คร่อมช่อง IV และ CC

4.3.2 การตั้งค่าความถี่เอาท์พุท

ค่าความถี่เอาท์พุทที่สัมพันธ์กับขนาดของสัญญาณอนาลอก 0-5 Vdc แสดงดังรูปที่ 4.6



รูปที่ 4.6 กราฟแสดงความสัมพันธ์ของสัญญาณควบคุมกับความถี่เอาท์พุท

โดยที่ค่า (P1) , (F-P1) , (P2) , (F-P2) สามารถตั้งค่าได้ดังนี้
(P1) ตั้งได้ในช่วง 0-100% และ (F-P1) ตั้งได้ในช่วง 0-ค่าความถี่สูงสุดที่ตั้งค่าไว้
(P2) ตั้งได้ในช่วง 0-100% และ (F-P2) ตั้งได้ในช่วง 0-ค่าความถี่สูงสุดที่ตั้งค่าไว้

ในโครงงานเราจะปรับค่าความถี่สูงสุดไว้ 50 Hz ซึ่งหมายความว่า (P1) และ (P2) สามารถตั้งค่าได้ในช่วง 0-5 V (F-P1) และ (F-P2) สามารถตั้งค่าได้ในช่วง 0-50 Hz

นอกจากที่กล่าวมาแล้ว ที่ตัวเครื่องอินเวอร์เตอร์ยังสามารถปรับตั้งค่าอื่น ๆ ได้ให้เหมาะสม เช่น การปรับตั้งค่าระดับความถี่สูงสุดและต่ำสุด ลักษณะแรงบิด เวลาเร่ง เวลาหน่วง รูปแบบ v/f เป็นต้น ซึ่งวิธีการปรับค่าต่าง ๆ สามารถศึกษาได้จากคู่มือของเครื่องได้โดยตรง

บทที่ 5

การสร้างโครงงาน "การควบคุมระยะทางมอเตอร์ด้วยวิธีการควบคุมแบบป้อนกลับ"

ขั้นตอนการดำเนินงานในการสร้างโครงงาน

ขั้นตอนการดำเนินงานในการสร้างโครงงานนี้ได้แบ่งขั้นตอนการทำงานออกเป็น 2 ส่วนหลัก ๆ คือ
- ส่วนวงจรต่าง ๆ ทั้งหมดหรือส่วนฮาร์ดแวร์
- ส่วนโปรแกรมต่าง ๆ ทั้งหมดหรือส่วนซอฟแวร์

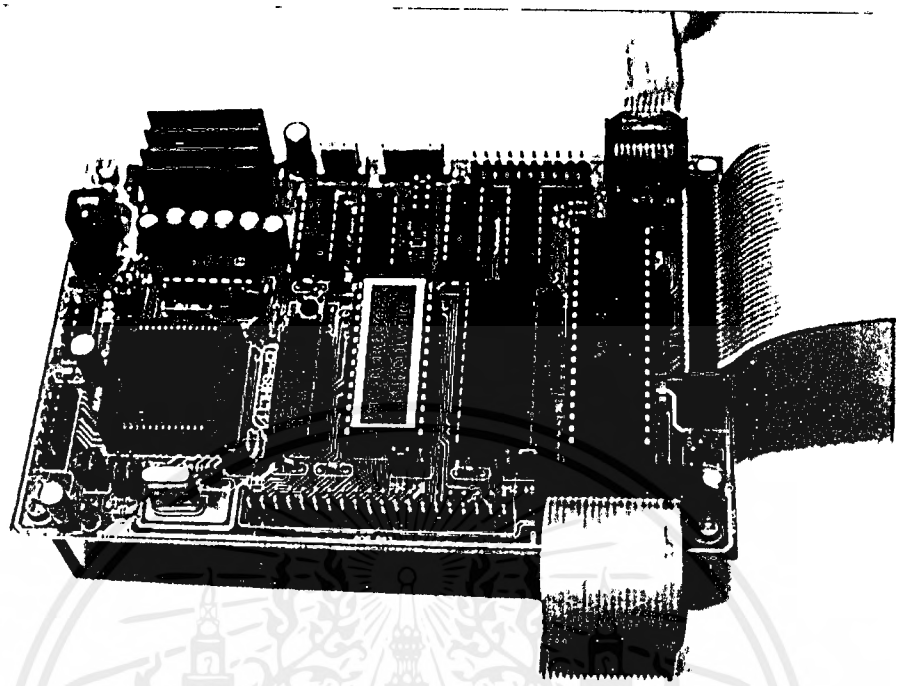
5.1 ส่วนฮาร์ดแวร์ของโครงงาน

ส่วนฮาร์ดแวร์ทั้งหมดประกอบด้วย

5.1.1 ส่วนของไมโครคอนโทรลเลอร์

ส่วนนี้จะใช้ควบคุมการทำงานทั้งหมด ได้มีการเอาแผงไมโครคอนโทรลเลอร์ซึ่งใช้หน่วยประมวลผล (Central Processing Unit) ของบริษัทโมโตโรล่าเบอร์ 68HC11 ซึ่งเป็นชิปตัวล่าสุดที่เป็นแบบซีมอสความเร็วสูง (high speed CMOS technology) ซึ่งทำงานได้รวดเร็วและมีการใช้พลังงานต่ำ ส่วนของไมโครคอนโทรลเลอร์ได้ใช้หน่วยความจำซึ่งมีทั้งแบบแรมและแบบอีอีพรอม ซึ่งแบบแรมใช้ในการเก็บข้อมูลในส่วนของระยะทางที่ต้องการและความเร็ว ส่วนอีอีพรอมใช้เก็บโปรแกรมที่ใช้ทำการควบคุมเครื่อง

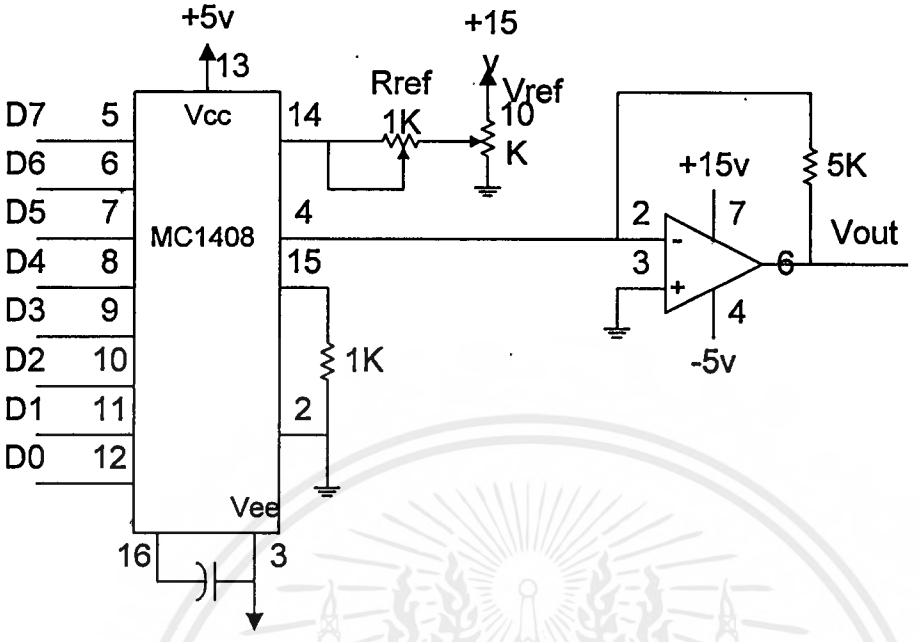
การทำงานในส่วนนี้เราได้นำเอาแผงไมโครคอนโทรลเลอร์ 68HC11 ดังรูปที่ 5.1 มาทำการเชื่อมต่อเข้ากับส่วนต่าง ๆ ของวงจรอื่นโดยทำการต่อพอร์ตต่าง ๆ ดังนี้
- ทำการต่อพอร์ตอินพุทเข้ากับคีย์บอร์ดเพื่อใช้ในการป้อนข้อมูลการทำงานของโครงงาน
- ทำการต่อพอร์ตเอาท์พุทเข้ากับส่วนแสดงผลแบบจอผลึกเหลว(Liquid Crytal Display)เพื่อใช้ในการแสดงผล
- ทำการต่อพอร์ตเอาท์พุทเข้ากับวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก เพื่อใช้ในการควบคุมเครื่องVCD อินเวอร์เตอร์ ในการควบคุมความเร็วของมอเตอร์
- ทำการต่อพอร์ตอินพุทเข้ากับเอนโค้ดเดอร์เพื่อใช้รับค่าระยะทางที่หมุนไปของมอเตอร์ในรูปสัญญาณพัลส์เพื่อนำไปประมวลผลป้อนกลับ



รูปที่ 5.1 รูปภาพแสดงวงจรไมโครคอนโทรลเลอร์ 68HC11

5.1.2 ส่วนวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก

ในส่วนนี้จะประกอบไปด้วยไอซีที่ใช้ในการแปลงสัญญาณเบอร์ MC1408 และมีตัวต้านทานแบบปรับค่าได้เพื่อใช้ในการปรับสัญญาณเอาท์พุทให้ได้ตามต้องการ วงจรนี้มีหน้าที่ในการรับค่าข้อมูลที่เป็นดิจิตอลที่มาจากวงจรไมโครคอนโทรลเลอร์แล้วจึงทำการแปลงค่าข้อมูลที่เป็นดิจิตอลที่ได้รับมานั้นให้เป็นสัญญาณอนาลอก โดยค่าที่ได้จะมีลักษณะเป็นไฟกระแสตรงขนาด 0-5 โวลต์หรือมากกว่าขึ้นกับการปรับค่าตัวต้านทานแบบปรับค่าได้ ซึ่งวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอกที่ใช้ในโครงงานมีการต่อวงจรดังรูปที่ 5.2



รูปที่ 5.2 แสดงวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก

จากรูปสามารถคำนวณค่าเอาท์พุทต่าง ๆ ที่สัมพันธ์กับลอจิกอินพุทได้ดังสมการ

$$V_{out} = V_{ref}/R_{ref} \times 5K \times \{D_7/2 + D_6/4 + D_5/8 + D_4/16 + D_3/32 + D_2/64 + D_1/128 + D_0/256\}$$

วงจร Unpolar D/A รหัสเลขฐานสองขนาด 8 บิต ถูกป้อนให้กับอินพุทของ DAC เอาท์พุทที่ได้จากไอซี MC1408 จะเป็นกระแสไฟฟ้าออกมาทางขา 4 ดังนั้นเราจึงต้องใช้ไอซีออปแอมป์มาเปลี่ยนกระแสไฟฟ้าที่ได้ให้อยู่ในรูปสัญญาณแรงดันไฟฟ้า

ซึ่งในโครงงานจะใช้ค่าไฟกระแสตรงขนาดสูงสุด 5 โวลต์เป็นเอาท์พุท นั่นคืออินพุทเป็นลอจิก1111 1111 และทำการปรับค่าตัวต้านทานแบบปรับค่าได้ให้ได้เอาท์พุทขนาด 5 โวลต์ตามต้องการ

ตามที่ได้กล่าวไปแล้วในบทที่ 4 ว่าสัญญาณอนาลอกขนาด 0-5 โวลต์จะเป็นสัญญาณควบคุมให้เครื่องVCD อินเวอร์เตอร์ให้จ่ายความถี่ไฟให้แก่มอเตอร์ 0-50 Hz อย่างเป็นสัดส่วนกันดังแสดงในตารางที่ 5.1

ค่าดิจิตอล								ค่าเอาท์พุท		
D7	D6	D5	D4	D3	D2	D1	D0	ความถี่	โวลต์	Syn Speed*
1	1	1	1	1	1	1	1	50	5	1500
0	1	1	1	1	1	1	1	25	2.5	750
0	0	1	1	1	1	1	1	12.5	1.25	375
.										
.										
0	0	0	0	0	0	0	0	0	0	0

* ค่าความเร็วซินโครนัสของมอเตอร์ = 120f/p เมื่อ p = จำนวนขั้วของมอเตอร์(4 ขั้ว)

ตารางที่ 5.1 แสดงความสัมพันธ์ของลอจิกอินพุทกับค่าเอาท์พุทที่ได้ต่าง ๆ

5.1.3 ส่วนของอุปกรณ์อินพุทและเอาท์พุท

ในส่วนนี้ประกอบด้วย คีย์บอร์ดขนาด 4×4 และจอผลึกเหลว(LCD) ขนาด 1 บรรทัดซึ่งเป็นส่วนที่ใช้ในการรับส่งข้อมูลของเครื่องในโครงงาน

นอกจากนี้ยังมีวงจรตรวจสอบ(sensor)จำนวนรอบที่มอเตอร์หมุนไปได้ที่ครัมของมอเตอร์ด้วยซึ่งใช้ตัว Opto Interrupterในตระกูล MOC70 Series มาเป็นตัวตรวจสอบแล้วต่อวงจรเข้ากับตัวนับในวงจรไมโครคอนโทรลเลอร์ 8051 ดังรูปที่ 5.3

5.1.4 ส่วนวงจรแปลงสัญญาณพัลส์จากเอนโค้ดเดอร์

อินพุทที่จะนำมาประมวลต่าง ๆในโครงงานที่สำคัญก็คือ พัลส์ที่ได้จากเอนโค้ดเดอร์ดังนั้นจึงจำเป็นจะต้องมีวงจรที่ทำการแปลงสัญญาณพัลส์ให้ได้ขนาดที่เหมาะสมที่จะสามารถนำไปใช้ประมวลผลในไมโครคอนโทรลเลอร์ 68HC11 ซึ่งการต่อวงจรได้กล่าวไว้ในบทที่2

5.1.5 ส่วนวงจรรีเลย์ตัดต่อทิศทางการหมุนของมอเตอร์

ในโครงงานเราสามารถกำหนดทิศทางการหมุนของมอเตอร์ได้จากคำสั่งที่คีย์บอร์ดที่ทำขึ้นโดยเมื่อไมโครคอนโทรลเลอร์รับรู้ว่าเราต้องการการหมุนในทิศทางไหนมันก็จะส่งสัญญาณจากพอร์ตTimer ไปตัดต่อวงจรโดยสัมพันธ์กับค่าเทอร์มินอลF , R ของเครื่องอินเวอร์เตอร์การต่อวงจรได้กล่าวไว้ในบทที่ 4

รูปที่5.3 แสดงตัวตรวจสอบจำนวนรอบของมอเตอร์ที่ครัมของมอเตอร์

5.2 ส่วนซอฟแวร์ของโครงงาน

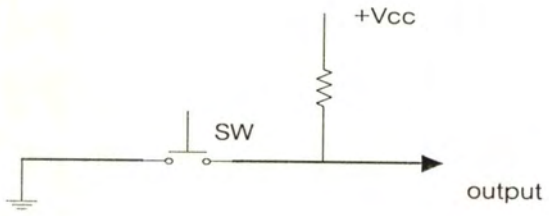
ส่วนซอฟแวร์ของโครงงานหรือโปรแกรมต่าง ๆ ที่เขียนขึ้นในโครงงานมีโปรแกรมต่าง ๆ ดังนี้

5.2.1 โปรแกรมการอ่านค่าและจัดการเกี่ยวกับคีย์บอร์ด

โปรแกรมส่วนนี้จะเกี่ยวข้องกับการรับค่าจากคีย์บอร์ดและแสดงผลที่จอ LCD

การรับข้อมูลจากคีย์บอร์ด

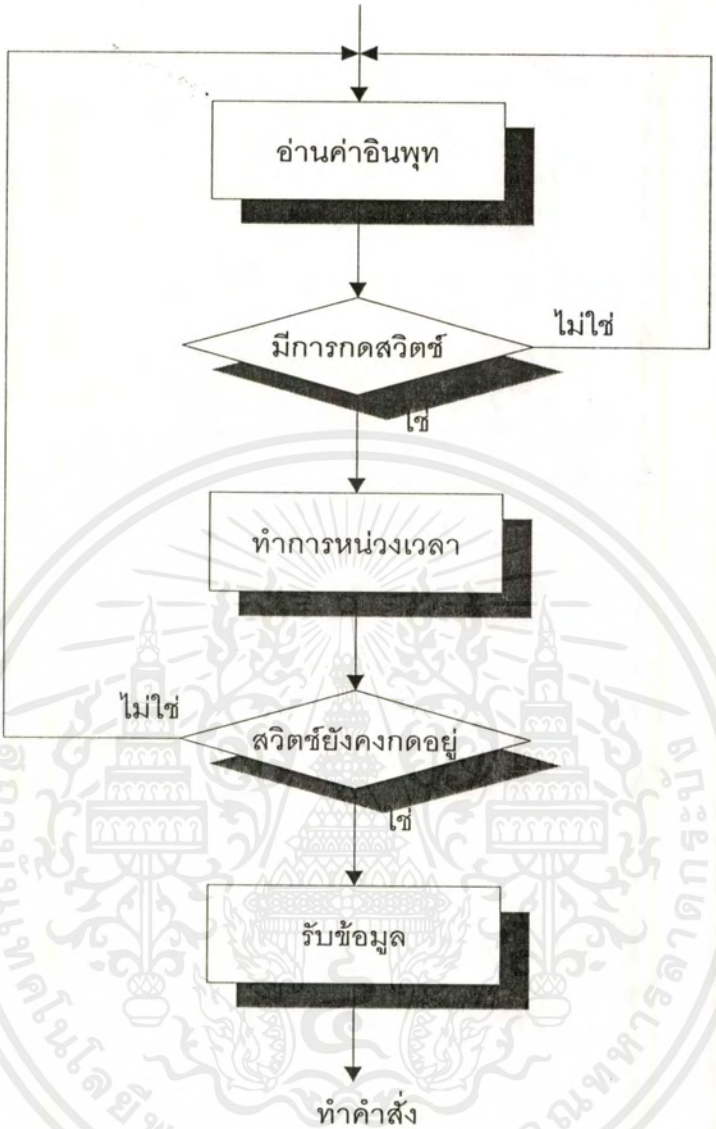
อุปกรณ์ในการรับข้อมูลที่ใช้ คือคีย์บอร์ด โดยคีย์บอร์ดจะทำหน้าที่ในการรับคำสั่งจากผู้ใช้ โดยคำสั่งประกอบด้วย START , MODE , ENTER , FORWARD , REVERSE , DISTANCE , SPEED , ตัวเลข 0-9 โดยพื้นฐานของคีย์บอร์ด คือสวิตช์โดยมีวงจรดังรูปที่ 5.4
ขณะยังไม่กดสวิตช์ เอาท์พุทจะมีค่าเท่ากับ +Vcc และเมื่อกดสวิตช์ เอาท์พุทจะมีค่าเท่ากับ 0 โวลต์ เนื่องจากในการกดสวิตช์ โดยทั่วไปจะมีพัลส์ที่ไม่ต้องการปรากฏขึ้นมา ซึ่งเกิดจากการกระเด้งของหน้าสัมผัส ปรากฏการณ์นี้เรียกว่า การเกิดบาวนซ์(Bounce) ดังรูปที่ 5.5



รูปที่ 5.4 แสดงวงจรของสวิตช์

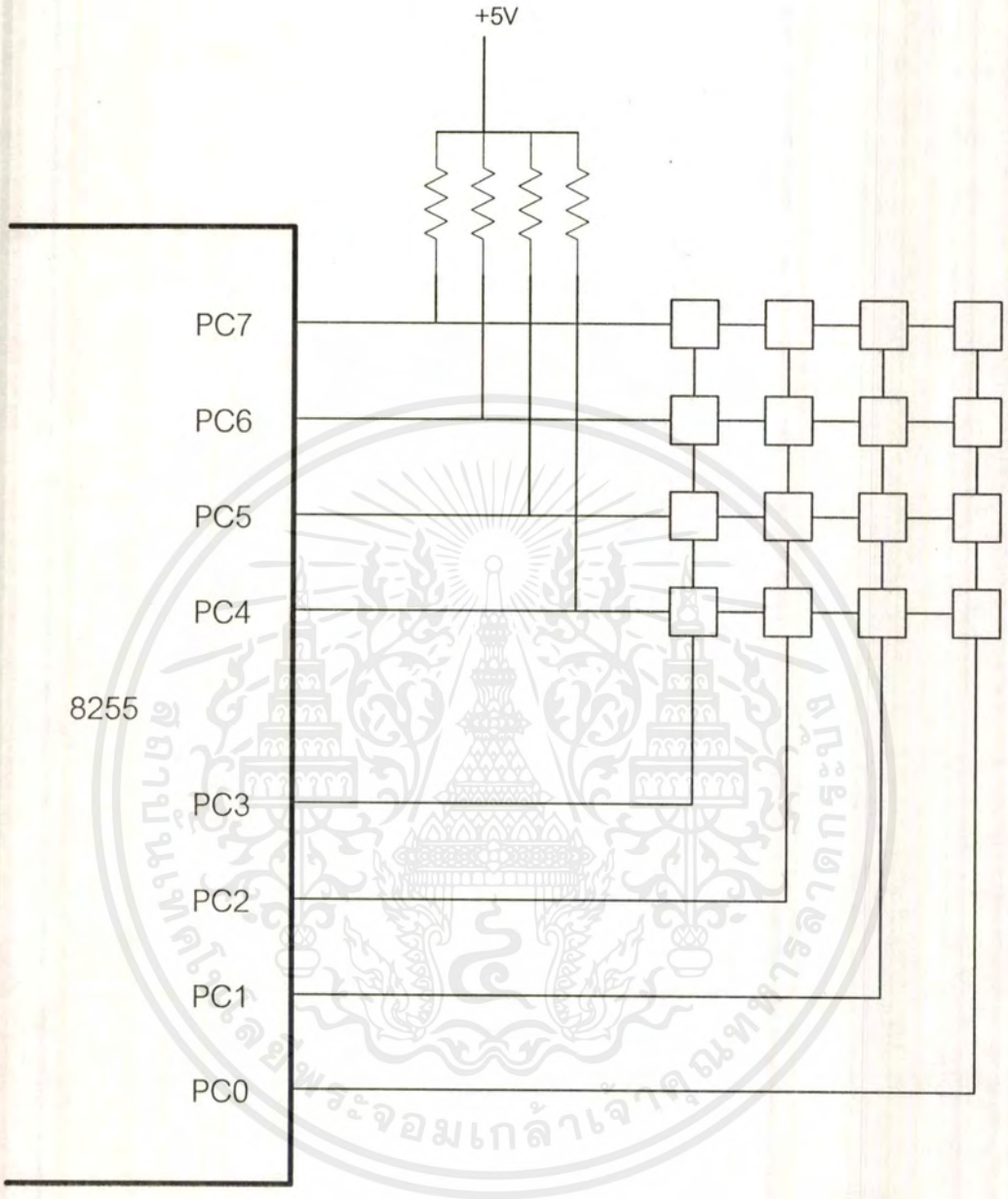
รูปที่5.5 แสดงการเกิดบาวนซ์ขณะกดสวิตช์

การเกิดบาวนซ์จะทำให้การรับข้อมูลผิดพลาดได้ดังนั้น จึงต้องแก้สัญญาณบาวนซ์โดยใช้ซอฟแวร์โปรแกรม จะทำงานโดยการตรวจสอบการกดสวิตช์ครั้งแรกแล้วรอสักระยะเวลาช่วงหนึ่งเพื่อที่จะให้ผ่านช่วงเวลาที่เกิดการบาวนซ์ผ่านไป แล้วทำการอ่านข้อมูลจากสวิตช์อีกครั้งหนึ่งเพื่อจะนำไปเป็นข้อมูล การแก้สัญญาณบาวนซ์นั้นสามารถนำมาเขียนเป็นโฟลว์ชาร์ทได้ดังรูปที่ 5.6



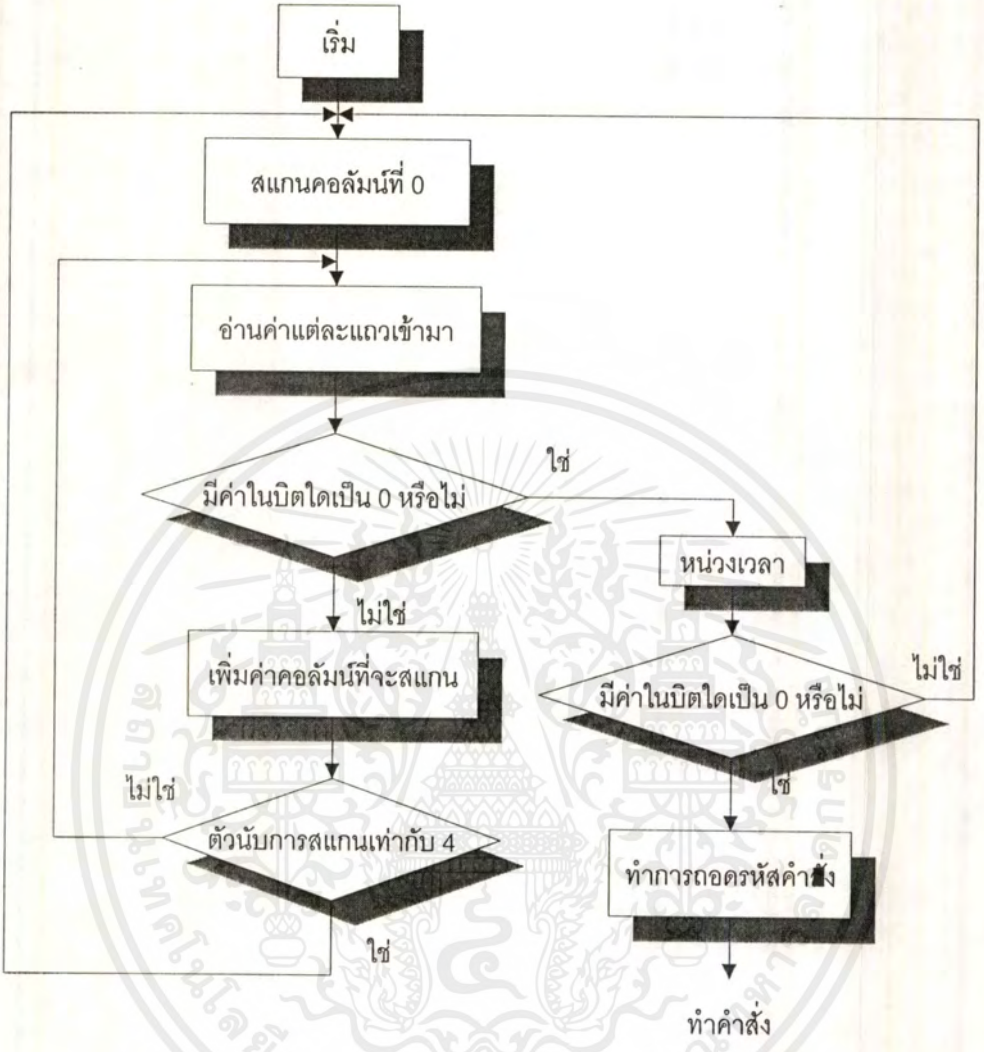
รูปที่ 5.6 แสดงโฟลว์ชาร์ทการแก้สัญญาณบาวนซ์

จะเห็นว่าสวิตช์ 1 ตัว แทน 1 คำสั่ง จะทำให้เกิดการเปลืองสวิทช์จำนวนมากรวมทั้งเปลืองสายที่จะใช้ติดต่อกับบอร์ดคอนโทรล ดังนั้นจึงใช้คีย์บอร์ดสำเร็จรูปซึ่งเป็นคีย์บอร์ดแบบเมทตริกซ์ชนิด 4×4 มีปุ่มกดทั้งหมด 16 ปุ่ม ดังนั้นจะมีบางปุ่มรับ 2 คำสั่งดังรูปที่ 5.7



รูปที่ 5.7 แสดงวงจรคีย์บอร์ด

การตรวจสอบสวิตช์ที่กดจะใช้ซอฟแวร์เข้ามาช่วยในการรับข้อมูลจากคีย์บอร์ดโดยโปรแกรมจะใช้หลักการในการสแกน เพื่อหาว่าสวิตช์ใดที่ถูกกด โดยจากรูปที่5.7 จะเห็นว่าจะใช้ PC0-PC3 ในการส่งค่าออกไป แล้วอ่านค่ากลับมาจาก PC4-PC7 การทำงานของโปรแกรมสแกนมีลักษณะดังรูปที่ 5.8



รูปที่ 5.8 แสดงโฟลว์ชาร์ทโปรแกรมสแกนคีย์บอร์ด

การรับข้อมูลจากปุ่มกด

ในส่วนนี้จะมีการรับค่าจากปุ่มกดอีก 2 ปุ่มคือ ปุ่ม STOP กับ RESET โดยปุ่ม STOP จะทำหน้าที่หยุดการทำงานของเครื่องขณะเครื่องกำลังทำงานอยู่ ส่วนปุ่ม RESET จะทำหน้าที่รีเซตเครื่องเพื่อที่จะป้อนค่าอินพุทใหม่

การแสดงผล

ในการแสดงผลของการกดคีย์บอร์ดนั้นเมื่อทำการสแกนคีย์แล้วตรวจสอบพบว่ามีการกดและทำการถอดรหัสคำสั่งเรียบร้อยแล้วก็จะนำคำสั่งนั้นไปเก็บในแอดเดรสที่ทำหน้าที่เก็บคำสั่ง พร้อมทั้งแสดงคำสั่งที่กดออกมาทางจอ LCD เพื่อให้ผู้ใช้ได้ทราบว่าได้กดคำสั่งอะไร โดยจอ LCD ที่ใช้เป็นแบบ Character 1 แถว 16 ตัวอักษร ลักษณะการทำงานเป็นไปดังโฟลว์ชาร์ทในรูปที่ 5.9

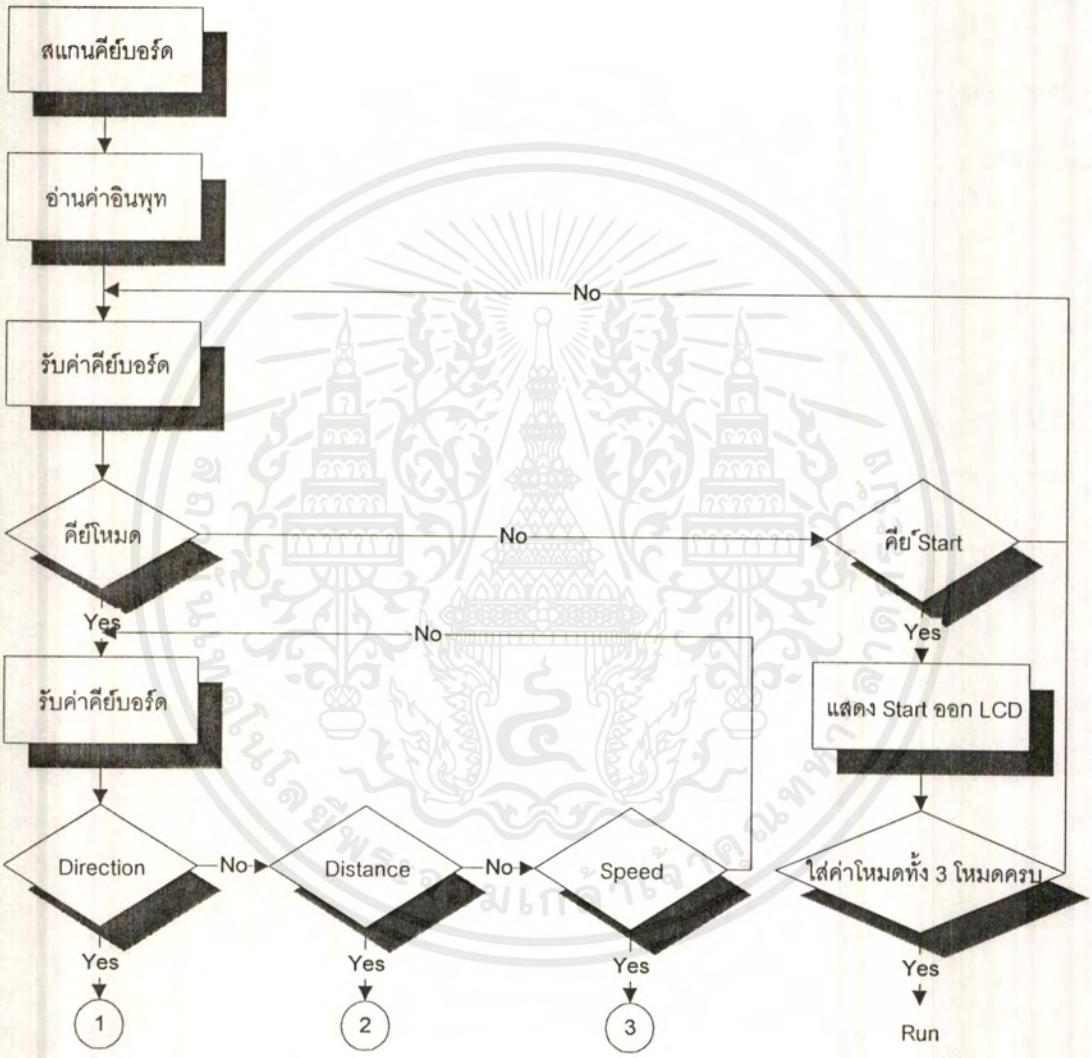
ความเข้าใจพื้นฐาน

1. การเขียนข้อมูลให้กับโมดูล LCD จะแบ่งเป็น 2 ลักษณะคือ คำสั่ง(Instruction) และข้อมูล(Data) โดยจะกำหนดด้วยขาสัญญาณ RS คือถ้า RS=0 จะหมายถึงส่งสัญญาณควบคุม(Instruction)หรืออ่านค่าแฟลก(Flag) สภาพการทำงานของโมดูล LCD และถ้า RS=1จะหมายถึงการเขียนหรืออ่านข้อมูลกับโมดูล LCD

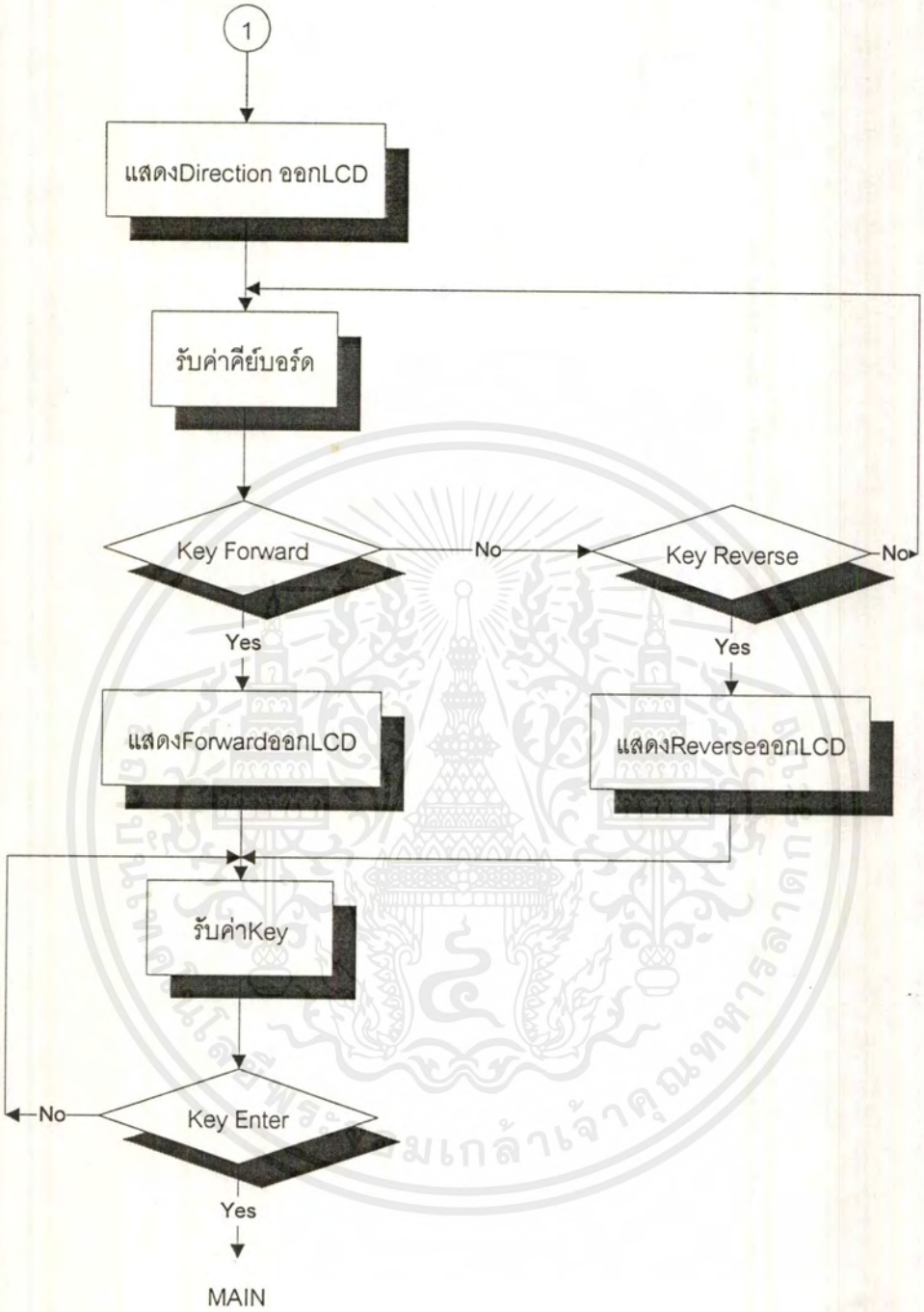
2. หลักการในการเขียนข้อมูลให้ โมดูล LCD นี้ คือเมื่อมีการเขียนข้อมูลไปแล้ว ตัวมันเองจะต้องใช้เวลาในการทำงานชั่วขณะหนึ่ง ซึ่งระบบไมโครสามารถตรวจสอบได้จากบิซซีแฟลก (Busy Flag (BF)) และถ้าเรียบร้อยแล้ว จึงจะสามารถเขียนข้อมูลอันต่อไปได้ ในกรณีที่การต่อวงจรเป็นแบบพอร์ต I/O คือไม่สามารถอ่านข้อมูลย้อนกลับได้ ระบบไมโครก็จะต้องใช้วิธีการหน่วงเวลาแทน

3. การเขียนข้อมูลให้กับโมดูล LCD นี้สามารถทำได้ทั้งแบบ 8 บิต และ 4 บิต โดยกรณี4บิต จะใช้สายสัญญาณข้อมูลเพียง 4 เส้น คือ DB4-DB7 (ใช้สำหรับระบบไมโครแบบ 4 บิต หรือเพื่อการประหยัดสาย) การเขียนข้อมูลจะกระทำเหมือนกับ 8 บิต เพียงแต่ให้เขียน 2 ครั้ง คือ DB4-DB7 ก่อน แล้วตามด้วย DB0-DB3 และจะต้องกำหนดคุณสมบัติ

4. DDRAM (Display Data Ram) คือหน่วยความจำภายในตัวโมดูล LCD ที่เป็นบัฟเฟอร์ของข้อมูล โดยถ้าเขียนรหัส ASCII ใด ๆ ลงไปในหน่วยความจำนี้ ก็จะปรากฏเป็นตัวอักษรที่แผงแสดงทันที



รูปที่ 5.9 แสดงโฟลว์ชาร์ทการแสดงผลของ LCD



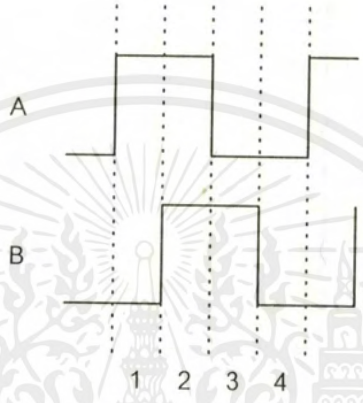
รูปที่ 5.10 แสดงโฟลว์ชาร์ทการแสดงผลLCD (ต่อ)

รูปที่ 5.11 แสดงโฟลว์ชาร์ทการแสดงผลLCD(ต่อ)

รูปที่ 5.12 แสดงโฟลว์ชาร์ทการแสดงผลLCD(ต่อ)

5.2.2 โปรแกรมการเช็คทิศทางและนับค่าพัลส์

โปรแกรมในส่วนนี้นับว่าเป็นส่วนหัวใจของโครงงานเลยทีเดียวเพราะเป็นส่วนแรกที่จะรับค่าอินพุทที่เป็นสัญญาณพัลส์จากเอนโค้ดเดอร์มาทำการตรวจสอบเพื่อประมวลผลโปรแกรมอื่นต่อไป



รูปที่ 5.13 แสดงสัญญาณพัลส์แต่ละเฟสของเอนโค้ดเดอร์

พิจารณาจากรูปที่ 5.13 จะเห็นว่ามีอินพุท 2 บิต ป้อนเข้ามาคือ บิต A และ B เมื่อเรานำค่าอินพุทมาเขียนจะได้ดังตารางที่ 5.2

A	B
1	0
1	1
0	1
0	0
1	0
1	1
0	1
0	0

ตารางที่ 5.2 แสดงสถานะบิตอินพุทเฟสA เทียบกับเฟสB

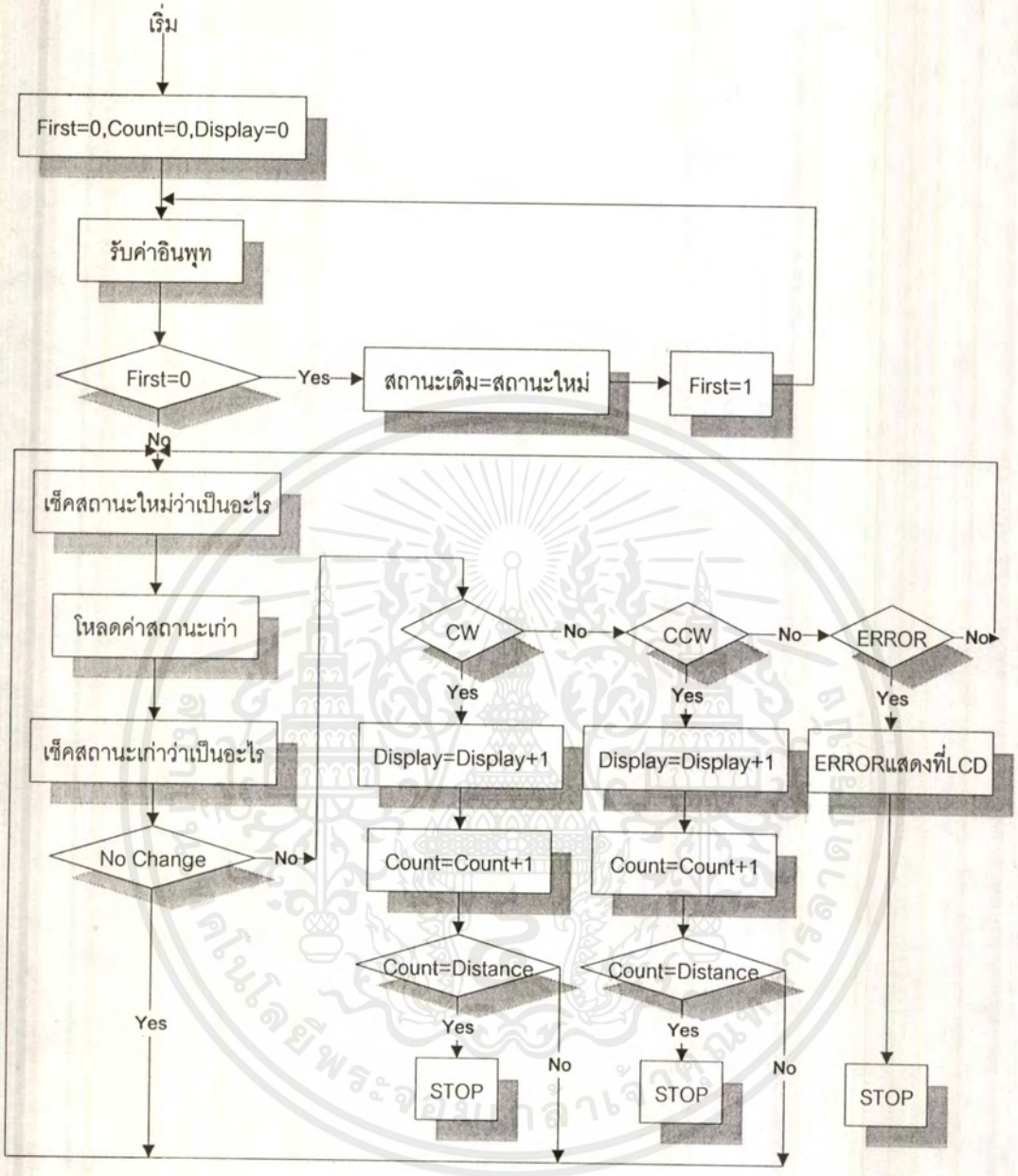
จากตารางที่ 5.2 ถ้าเราพิจารณาที่ค่า ๆ หนึ่ง เช่นที่ 00 ถ้าอินพุทสถานะบิตเฟสA เทียบกับเฟสB ต่อไปเป็น 01 จะเป็นทิศทางตามเข็มนาฬิกา แต่ถ้าอินพุทที่เข้ามาต่อไปเป็น 10 จะเป็นทิศทางทวนเข็มนาฬิกาเป็นต้น(ถ้ามอเตอร์หมุนตามเข็มเฟสA จะนำเฟสB 90 องศา แต่ถ้ามอเตอร์หมุนทวนเข็มนาฬิกาเฟสA จะตามเฟสB 90 องศา)

จากข้อมูลในตารางที่ 5.2 และรูปที่ 5.13 เราสามารถสรุปการตรวจสอบสถานะทิศทางการหมุนจากสถานะบิตเริ่มต้นและสถานะบิตต่อไปของสัญญาณพัลส์จากเอนโค้ดเดอร์ทั้ง 2 เฟสได้ดังตารางที่ 5.3

สถานะเดิม / สถานะใหม่	00	01	11	10
00	no change	ccw	error	cw
01	cw	no change	ccw	error
11	error	cw	no change	ccw
10	ccw	error	cw	no cange

ตารางที่ 5.3 แสดงทิศทางการหมุนต่าง ๆ เมื่อเทียบกับการเปลี่ยนแปลงสถานะพัลส์

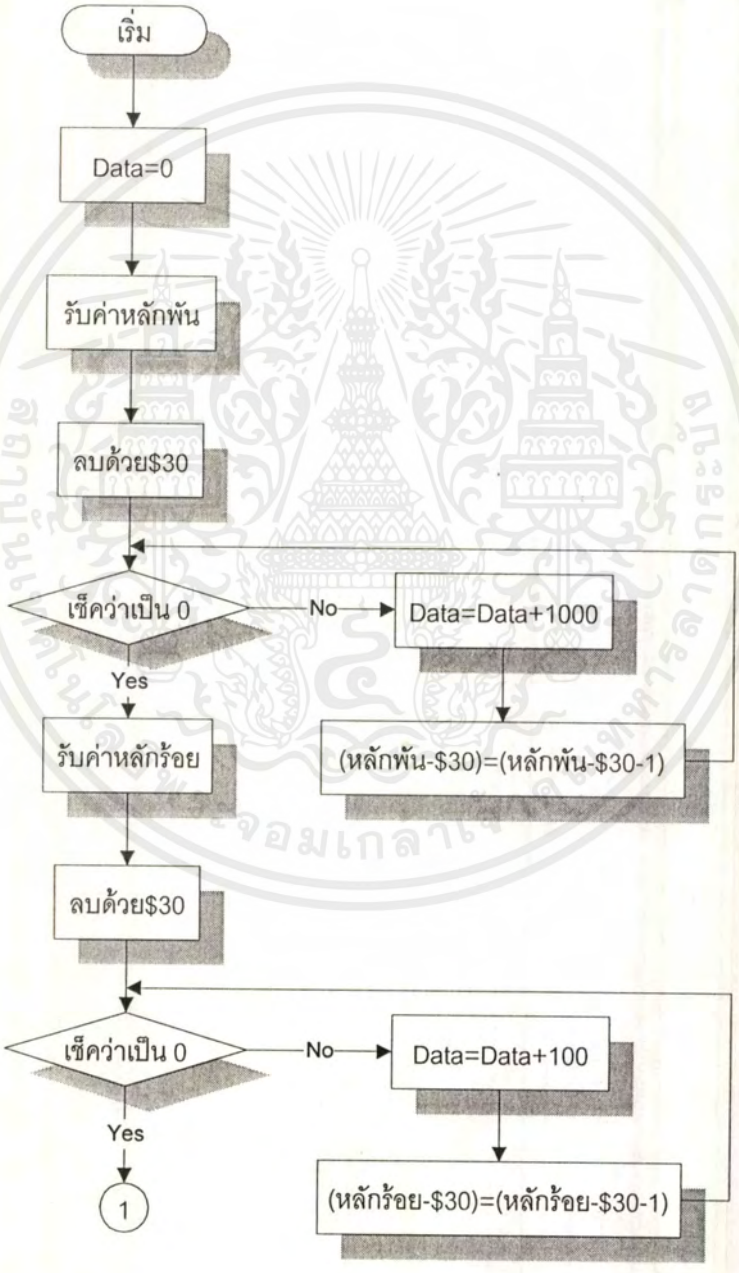
โปรแกรมการนับและตรวจสอบทิศทางแสดงในโฟลว์ชาร์ทในรูปที่ 5.14



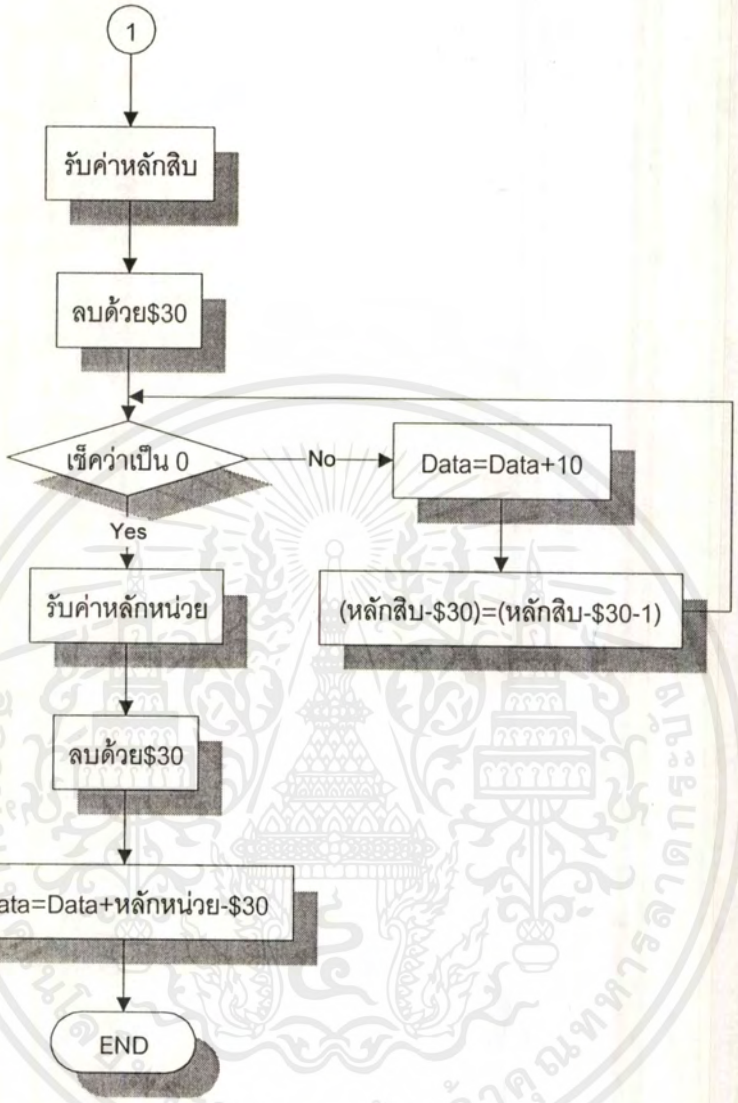
รูปที่ 5.14 แสดงโฟลว์ชาร์ทโปรแกรมการนับสัญญาณพัลส์

5.2.3 การแปลงค่าจากรหัส ASCII เป็นเลขฐาน 16

โปรแกรมในส่วนนี้เป็นส่วนที่รับข้อมูลจากคีย์บอร์ดที่เป็นรหัส ASCII ให้เป็นตัวเลขฐาน 16 ที่ไมโครคอนโทรลเลอร์เข้าใจและสามารถนำไปประมวลผลอื่น ๆ ต่อไปได้ รูปที่ 5.15 แสดงโฟลว์ชาร์ทของโปรแกรม



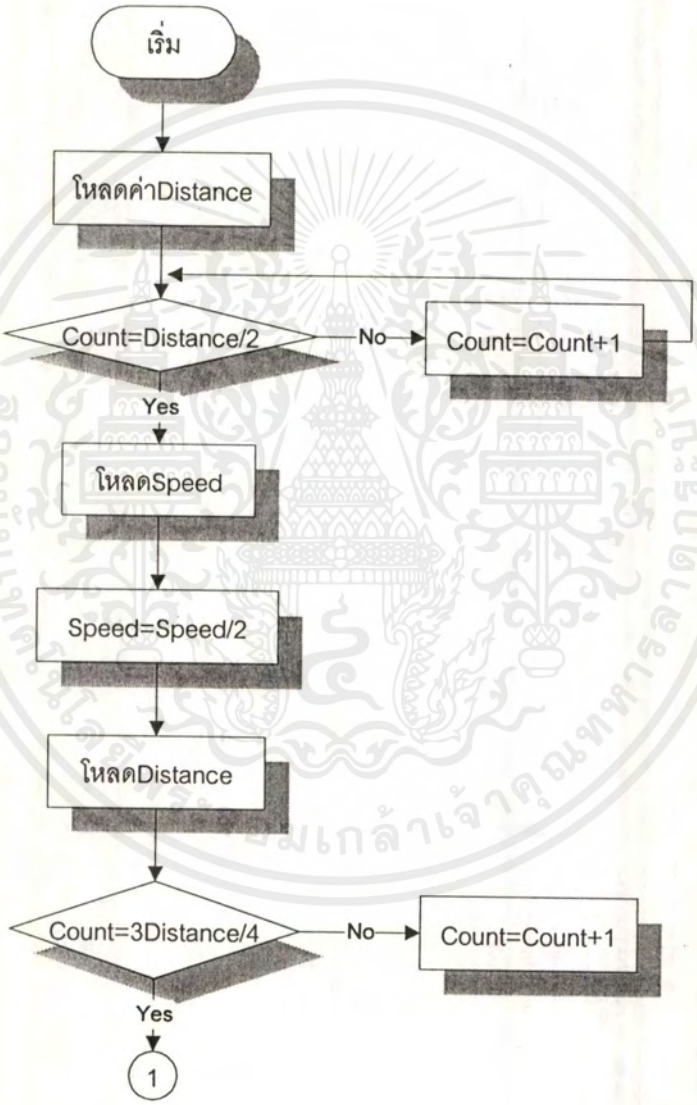
รูปที่ 5.15 แสดงโฟลว์ชาร์ทการแปลงรหัส ASCII เป็นเลขฐาน 16



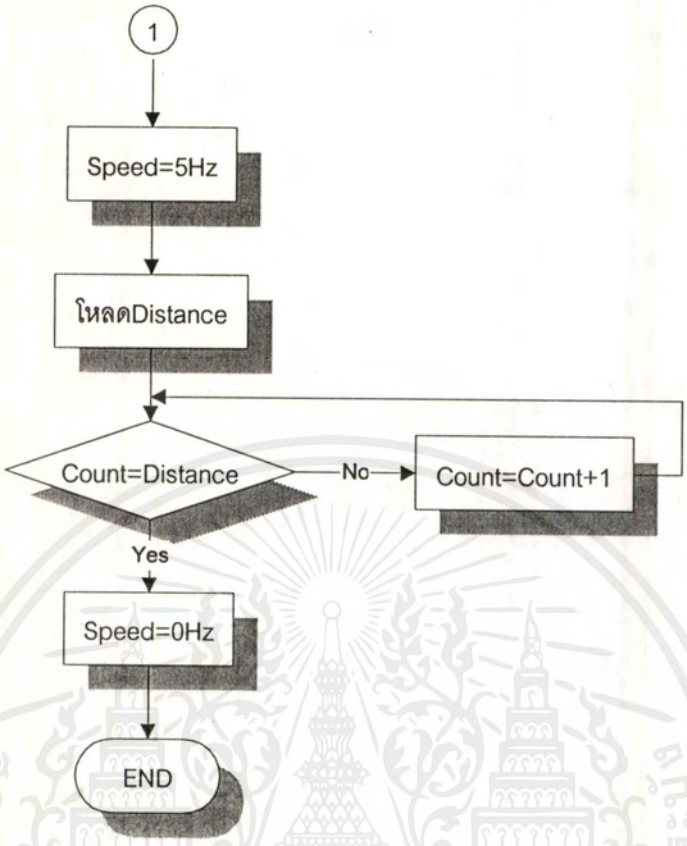
รูปที่ 5.16 แสดงโฟลว์ชาร์ทการแปลงรหัส ASCII เป็นเลขฐาน 16 (ต่อ)

5.2.4 การคำนวณค่าเพื่อส่งค่าดิจิตอลมาที่วงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก

โปรแกรมในส่วนนี้ทำต่อจากโปรแกรมนับสัญญาณพัลส์เพื่อคำนวณและส่งค่าดิจิตอลที่เหมาะสมที่จะส่งมายังวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอกเพื่อควบคุมเครื่องVCD อินเวอร์เตอร์ให้จ่ายความถี่สัมพันธ์กับค่าอนาลอกที่ได้รับ แสดงโฟลว์ชาร์ทดังรูปที่ 5.17



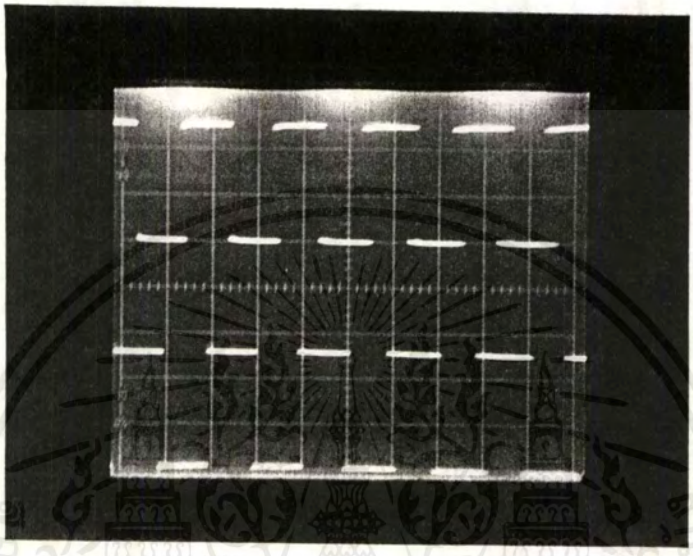
รูปที่ 5.17 แสดงโฟลว์ชาร์ทการคำนวณค่าระยะทางเพื่อปรับลดความเร็วมอเตอร์



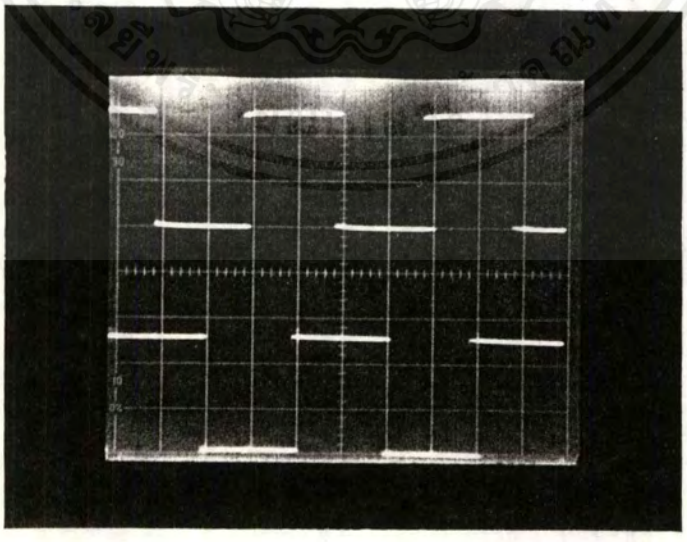
รูปที่ 5.18 แสดงโฟลว์ชาร์ทการคำนวณค่าระยะทางเพื่อปรับความเร็วมอเตอร์(ต่อ)

บทที่ 6

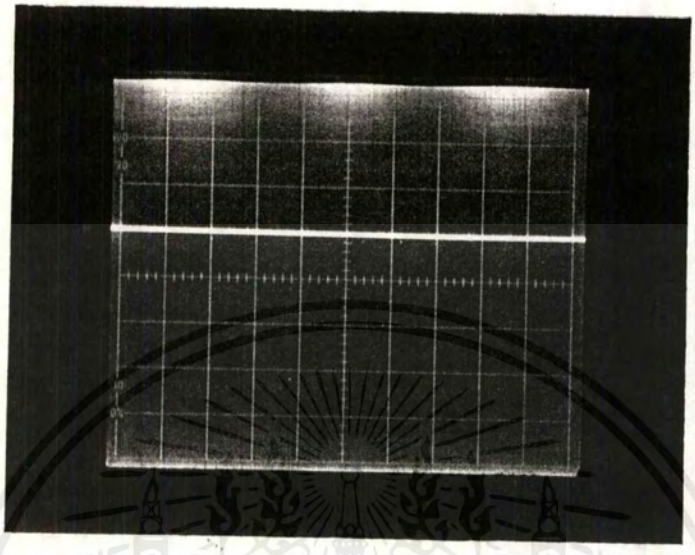
ผลการทดลอง



รูปที่ 6.1 แสดงสัญญาณพัลส์จากเอนโค้ดเดอร์ที่ความถี่ 50 Hz(Volt/div = 2v, Time/div = 20ms)



รูปที่ 6.2 แสดงสัญญาณพัลส์จากเอนโค้ดเดอร์ที่ความถี่ 25 Hz(Volt/div = 2v, Time/div = 20 ms)



รูปที่ 6.3 แสดงสัญญาณอนาลอกขนาด 5 โวลต์ที่ได้จากวงจรD/A

ตารางผลการทดลอง

ครั้งที่	FORWARD			REVERSE		
	ความถี่	ระยะทาง	ค่าผิดพลาด	ความถี่	ระยะทาง	ค่าผิดพลาด
1	2	20	0.3	2	20	0.23
2	2	20	0.15	2	20	0.1
3	2	20	0.14	2	20	0
4	2	20	0.2	2	20	0.28
5	2	20	0.15	2	20	0.04
		เฉลี่ย	0.188%		เฉลี่ย	0.13%
1	5	20	0.08	5	20	0.05
2	5	20	0.14	5	20	0.08
3	5	20	0.18	5	20	0.14
4	5	20	0.2	5	20	0.15
5	5	20	0.13	5	20	0.19
		เฉลี่ย	0.146%		เฉลี่ย	0.122%
1	10	100	0	10	100	0.012
2	10	100	0.03	10	100	0.034
3	10	100	0.03	10	100	0.014
4	10	100	0.04	10	100	0.02
5	10	100	0.01	10	100	0.02
		เฉลี่ย	0.022%		เฉลี่ย	0.02%
1	15	100	0.006	15	100	0.018
2	15	100	0.016	15	100	0.036
3	15	100	0.008	15	100	0.03
4	15	100	0.01	15	100	0.04
5	15	100	0.01	15	100	0.04
		เฉลี่ย	0.01%		เฉลี่ย	0.0328%

ตารางที่ 6.1 ตารางแสดงผลการทดลองวัดระยะที่สั่งและค่าคลาดเคลื่อน

ครั้งที่	FORWARD			REVERSE		
	ความถี่	ระยะทาง	ค่าผิดพลาด	ความถี่	ระยะทาง	ค่าผิดพลาด
1	20	400	0.004	20	400	0.006
2	20	400	0.011	20	400	0.009
3	20	400	0.0095	20	400	0.007
4	20	400	0.0005	20	400	0.0095
5	20	400	0.0095	20	400	0.01
		เฉลี่ย	0.0069%		เฉลี่ย	0.0084%
1	25	400	0.006	25	400	0.004
2	25	400	0.004	25	400	0.0016
3	25	400	0.0105	25	400	0.0095
4	25	400	0.007	25	400	0.002
5	25	400	0.001	25	400	0.014
		เฉลี่ย	0.0057%		เฉลี่ย	0.0091%

ตารางที่ 6.2 ตารางผลการทดลองวัดระยะที่สั่งและค่าคลาดเคลื่อน(ต่อ)

โดยความถี่มีหน่วยเป็น Hz และระยะทางมีหน่วยเป็นจำนวนรอบของมอเตอร์และค่าคลาดเคลื่อนมีหน่วยเป็นเปอร์เซนต์เมื่อเทียบกับระยะทางทั้งหมดที่สั่ง

บทที่ 7
สรุปและวิจารณ์

ในโครงงานนี้มีการสั่งงานโดยผ่านทางคีย์บอร์ดและมีส่วนแสดงผลเป็นแบบจอผลึกเหลว ซึ่งการรับค่าคำสั่งจากคีย์บอร์ดและการแสดงผลเป็นที่น่าพอใจและการสั่งค่าให้มอเตอร์หมุนที่ความเร็วต่าง ๆ (สั่งผ่านวงจรแปลงสัญญาณดิจิตอลเป็นอนาลอกให้เครื่องVCDอินเวอร์เตอร์จ่ายที่ความถี่ต่าง ๆ) นั้นได้ผลเป็นที่น่าพอใจเช่นกัน

แต่ในส่วนของการป้อนสัญญาณจากเอนโค้ดเดอร์กลับมา ยังมีข้อผิดพลาดอยู่เนื่องมาจากมีขีดจำกัดของการประมวลผลของซีพียูของไมโครคอนโทรลเลอร์ 68HC11 ซึ่งเป็นซีพียูขนาด 8 บิต ซึ่งถึงแม้ไมโครคอนโทรลเลอร์ 68HC11 จะเป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต ที่ดีที่สุดตัวหนึ่ง แต่ถ้าจะพัฒนาโครงงานในภายหลังน่าจะใช้ซีพียูที่มีขนาด 16 บิต หรือ 32 บิต ขึ้นไปจะทำให้มีประสิทธิภาพมากกว่า

สัญญาณพัลส์ที่มาจากเอนโค้ดเดอร์ควรจะมีขนาดที่จะสามารถนำมาทำการนับและประมวลผลได้เลยไม่ควรมีการปรับปรุงสัญญาณพัลส์ใด ๆ เพราะเป็นสัญญาณที่มีความถี่มากพอสมควรถ้าทำงานที่ความถี่ไฟมาก โอกาสที่จะเกิดสัญญาณรบกวนหรือสัญญาณแทรกอื่น ๆ อันเนื่องมาจากผ่านวงจรอีกวงจรหนึ่งนั้นสามารถมีได้ เพราะสัญญาณที่ได้จะต้องนำไปเป็นข้อมูลสำคัญในการประมวลผลของโครงงานเลยทีเดียว

โครงงานนี้สามารถนำไปประยุกต์พัฒนาต่อไปได้อีก เกี่ยวกับการควบคุมระยะทางมอเตอร์ ทั้งนี้เพราะการประยุกต์ใช้เอนโค้ดเดอร์นั้นยังมีได้อีกหลายลักษณะ เช่น ใช้ตรวจระยะการเคลื่อนที่ของสายพาน หรือประยุกต์ใช้เป็นพวกเครื่องพันตัวเหนี่ยวนำที่ต้องคำนวณระยะของลวดที่ใช้พันเป็นต้น

ภาคผนวก ก.

หลักการเบื้องต้นในการควบคุมความเร็วมอเตอร์

การควบคุมความเร็วของมอเตอร์เหนี่ยวนำ

วิธีการที่ใช้เป็นส่วนมากในการควบคุมความเร็วของมอเตอร์เหนี่ยวนำมีดังนี้

1. การใส่ค่าความต้านทานเข้าไปในวงจรโรเตอร์
2. การปรับเปลี่ยนค่าจำนวนขั้ว(pole)ของมอเตอร์
3. การปรับเปลี่ยนค่าความถี่ของแหล่งจ่ายไฟ
4. การต่อมอเตอร์เข้ากับเครื่องจักรอื่น ๆ เป็นขั้น ๆ(cascade)

นอกจากวิธีการข้างต้นแล้วยังมีวิธีการพิเศษอื่นอีกหลายวิธี เช่น การควบคุมโดยใช้วิธี saturable reactors, ควบคุมโดยวิธีควบคุมสัญญาณพัลส์ของแหล่งจ่ายไฟ ฯลฯ

ในโครงงานที่ศึกษาโดยใช้แหล่งจ่ายไฟแบบอินเวอร์เตอร์เป็นแหล่งจ่ายไฟให้มอเตอร์ จะเห็นว่าวิธีการควบคุมความเร็วโดยการปรับเปลี่ยนค่าความถี่ของแหล่งจ่ายไฟเป็นวิธีที่สะดวกดังนั้นจะขอกล่าวหลักการเบื้องต้นของการควบคุมความเร็วโดยวิธีปรับเปลี่ยนค่าความถี่ของแหล่งจ่ายไฟ

การควบคุมความเร็วของมอเตอร์เหนี่ยวนำด้วยวิธีการปรับเปลี่ยนค่าความถี่แหล่งจ่ายไฟ

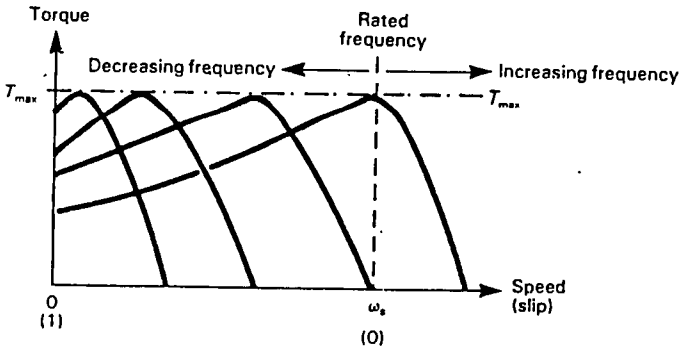
พื้นฐานในการปรับความเร็วโดยวิธีปรับค่าความถี่แหล่งจ่ายไฟนั้นเกี่ยวข้องกับสมการความเร็วสนามแม่เหล็กหมุน(Synchronous speed) $\omega = 2\pi f/P$

เมื่อ ω = ความเร็วสนามแม่เหล็กหมุน (rad/sec)

f = ความถี่แหล่งจ่ายไฟ

P = จำนวนขั้วของมอเตอร์

ในมอเตอร์เหนี่ยวนำสามารถสมมุติได้ว่าผลคูณ $f\phi$ ประมาณเป็นสัดส่วนกับแรงดัน V เพื่อรักษาให้ฟลักซ์แม่เหล็กคงที่ ดังนั้นจึงจำเป็นที่ต้องควบคุมความถี่ให้เป็นไปในอัตราส่วน $V/f = \phi$ = ค่าคงที่ ลักษณะของความเร็ว-แรงบิดของมอเตอร์เหนี่ยวนำควบคุมโดยการปรับความถี่ของแหล่งจ่ายไฟ



รูป ก. แสดงลักษณะแรงบิด-ความเร็วของมอเตอร์เหนี่ยวนำขณะทำให้ V/f คงที่

ภาคผนวก ข.

รายละเอียดของอุปกรณ์ต่าง ๆ

SIKO Incremental Encoder IGV40...41

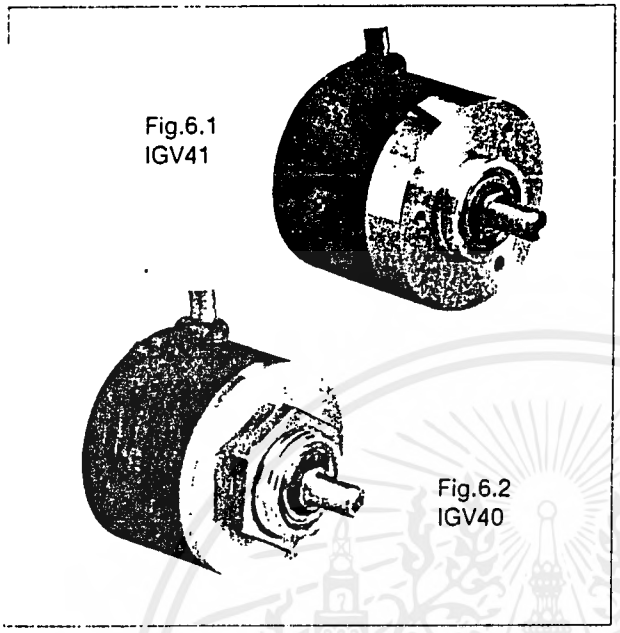


Fig.6.1
IGV41

Fig.6.2
IGV40

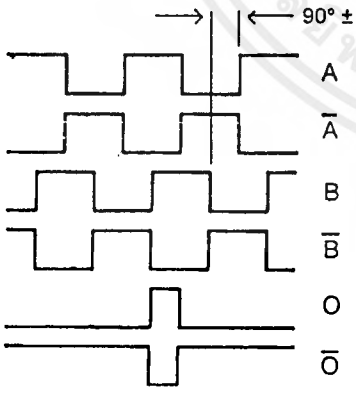
Description

Tough, compact design. Applications throughout industry. Pulse counting uses GaAl diodes and photo-transistors. Output via screened 5 core and 8 core flying leads. These output signal types are available:

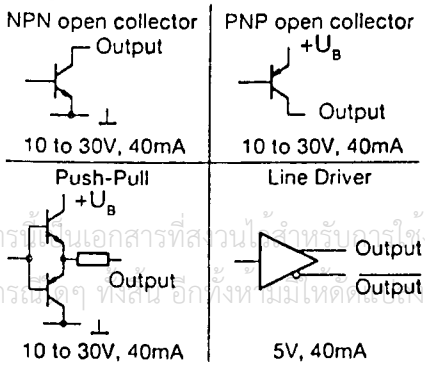
AXX, ABX, ABO, AĀXX, AĀBB̄XX, AĀBB̄OŌ

Output signals

Clockwise rotation see fig. 7.1 and 7.2



Output circuits



Electrical data

Impulses per revolution

8, 10, 20, 25, 30, 40, 50, 60, 90, 100, 125, 127, 150, 200 **250,** 300, **360, 400, 500,** 625, **1000,** 1024, 1200, 1500, 2500.

For others please enquire, preferred values in bold type.

Supply voltage	+10V to 30V		+5V ±5%
Outputs	NPN, PNP open coll.	Rush-Pull	Line driver
Max. output current per channel	40mA	40mA	40mA

Output frequency: 50 kHz max.
Current consumed by encoder: 30 mA max.
Protection: against polarity inversion (except U_B=+5V version)
Life of opto-electronics: >100000 h

Materials and components

Flange: anodised aluminium
Housing: epoxy resin
Shaft: non-magnetic stainless steel
Light source: Ga Al-diode

Mechanical data

Dimensions: see fig. 7.1 and 7.2
Shaft: 4, 6, 8 mm diameter
Shaft load: (axial & radial) 20 N max.
Max. speed: 3000 r/min (short term up to 6000 r/min)
Torque at 20°C: ≤ 0.2 Ncm (typical)
Moment of inertia: ~ 20 gmcm²
Bearing life: >10⁹ revolutions
Weight: ~ 0.1 kg

Environmental factors

Shock: 50 g / 11 ms
Vibration: 10 g / 10 to 2000 Hz
Operational temperature: 0 to +60°C
Storage temperature: -20 to +80°C (98% relative humidity without condensation)
Protection: IP54

Options

- Working temperature -20 to + 70°C
- Protection IP65 (IGV41 only)
- Push-Pull output circuit with short circuit protection
- Output frequency 100 kHz

Mounting dimensions

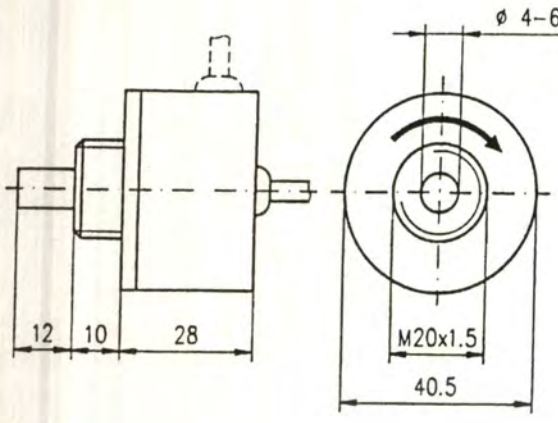


Fig.7.1
IGV40

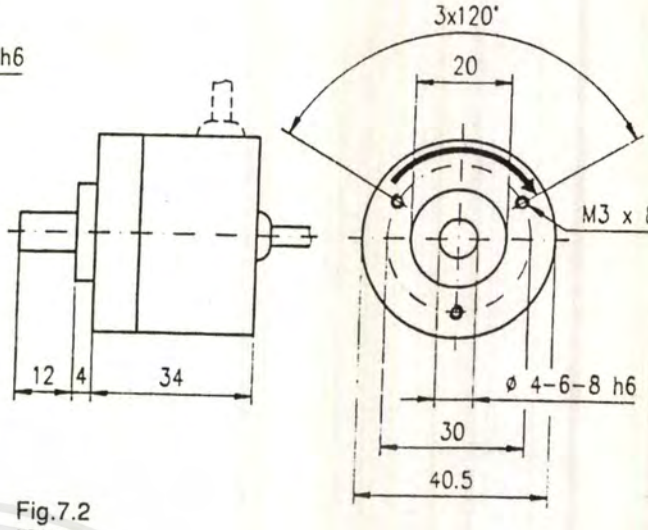


Fig.7.2
IGV41

Electrical connection

Normal output
5 core flying lead (1 m)

black	⊥
red	U_B
brown	A
blue	B
white	O

With inverted output
8 core flying lead (1 m)

yellow	A	grey	$\overline{O}$
blue	$\overline{A}$	red*	U_B
green	B	black	⊥
orange	$\overline{B}$		
white	O		

Order code

Use addnl. codes only if required

I G V X X - X - X X X X - X - X - X - X - X + X - X - X - X - X - X

Encoder type
IGV40
IGV41

Output circuit
N = NPN (O. C.)
P = PNP (O. C.)
Y = Push-Pull
L = Line driver

Impulses per rev.
See electrical data page 6

Output signal
M = AXX
B = ABX
Z = ABO

Shaft
4 = 4 mm
6 = 6 mm
8 = 8 mm

L = cable length reqd. meter
e.g. L4 = 4 m
L7 = 7 m

T = working temperature
– 20° to + 70° C

V = output frequency 100 kHz

P = protection to IP65 (IGV41 only)

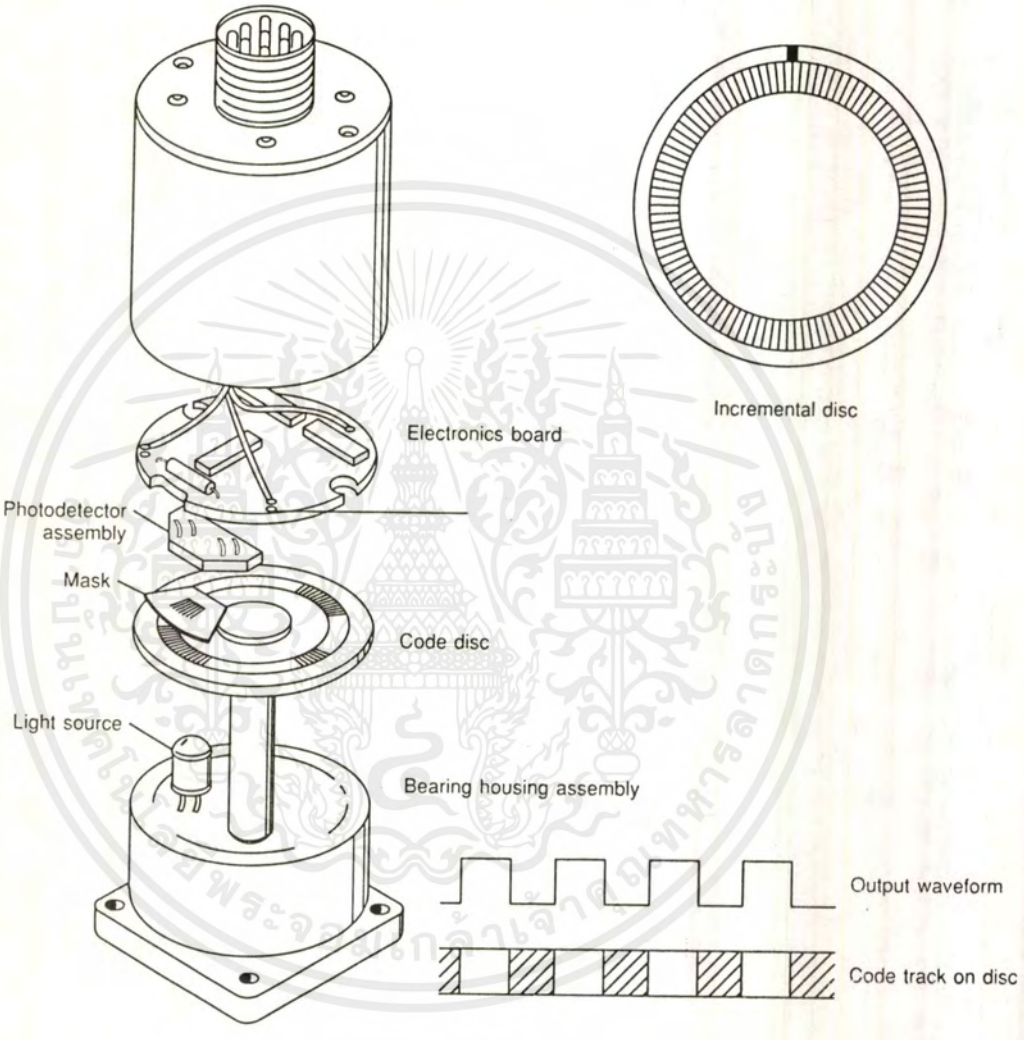
R = radial cable outlet

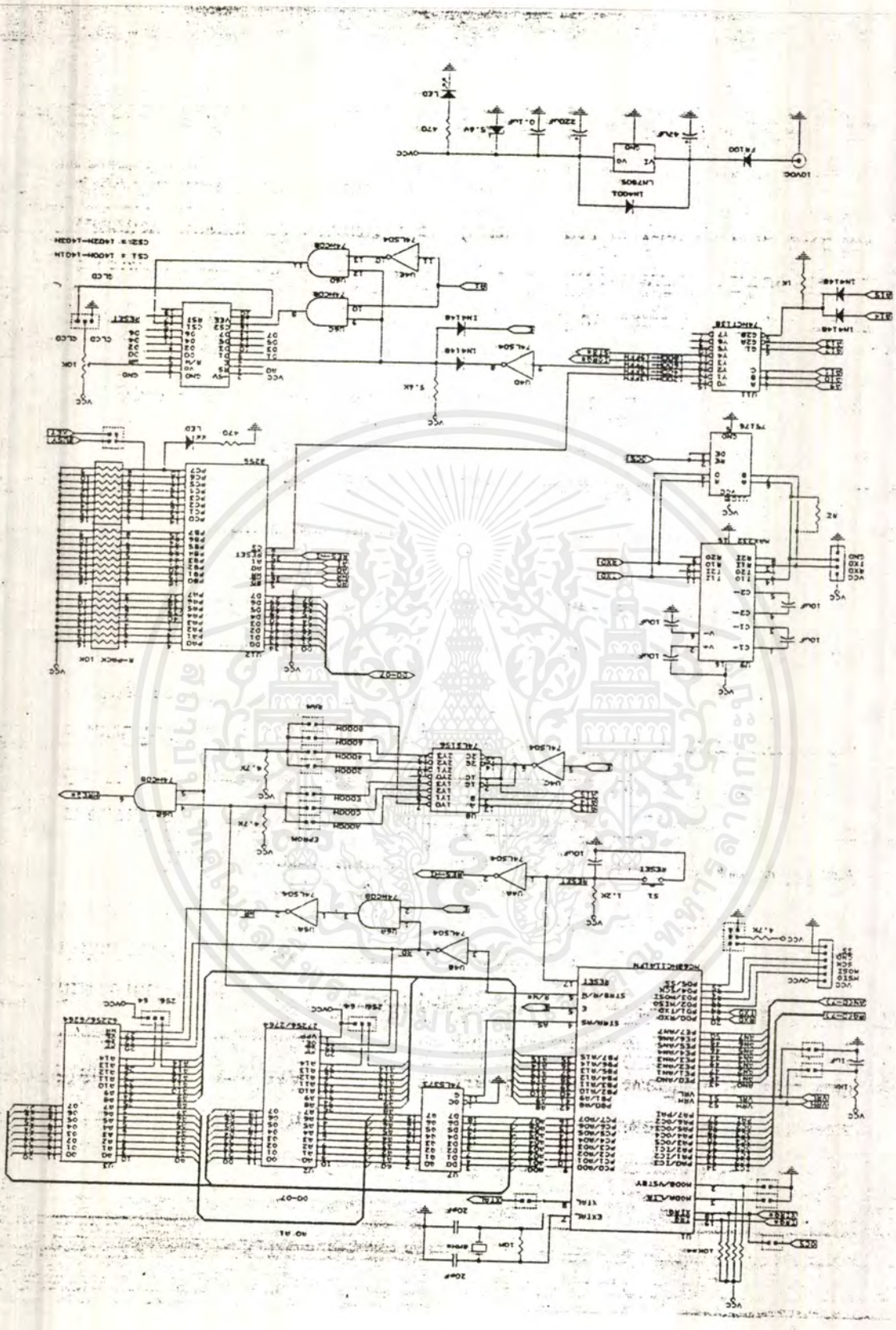
Z = Push-Pull output w. short circuit protection max. 15 mA

Output options
N = standard output
C = w. inverted output

Connection method
F = 5 core cable
U = 8 core cable

Operating voltage
1 = + 5V ± 5% (L output)
2 = + 10 to 30V (N, P and Y outputs)





วงจรสมบูรณ์ของบอร์ด CP-68HC11

DMC161C

• Display Format(16character ×1line) • Display Fonts(5×8dots) • Driving Method(1/8D)

ABSOLUTE MAXIMUM RATINGS

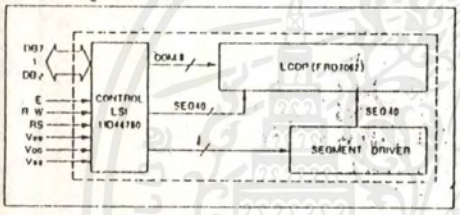
Item	Symbol	Test Condition	Standard Value min.	typ.	max.	Unit
Power Supply Voltage for Logic	$V_{DD}-V_{SS}$	———	0	-	7	V
Power Supply Voltage for LCD Drive	$V_{DD}-V_{EE}$	———	0	—	13.5	V
Input Voltage	V_I	———	V_{SS}	—	V_{DD}	V
Operating Temperature	T_A	———	0	—	+50	℃
Storage Temperature	T_{stg}	- -	−20	—	+70	℃

ELECTRICAL CHARACTERISTICS

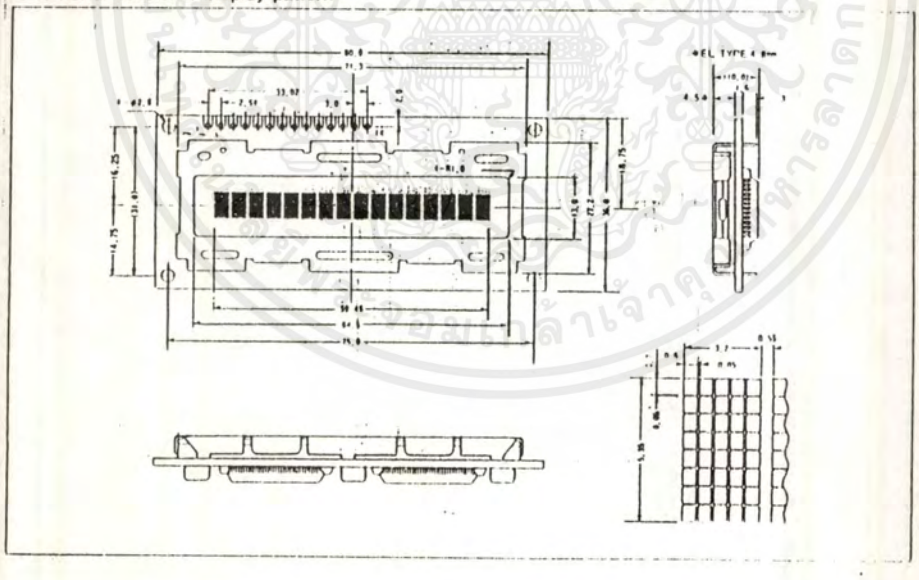
Item	Symbol	Test Condition	Standard Value min.	typ.	max.	Unit
Input "High" Voltage	V_{IH}		2.2		V_{DD}	V
Input "Low" Voltage	V_{IL}		-0.3		0.6	V
Output "High" Voltage	V_{OH}	$-I_{OH}$=0.205mA	2.4			V
Output "Low" Voltage	V_{OL}	I_{OL}=1.2mA			0.4	V
Power Supply Current	I_{DD}	V_{DD}=5.0V		0.5	2.0	mA

* V_{DD}=5.0V±5%, T_A=25℃

Block diagram



External dimensions / Display pattern



รายละเอียดของ LCD

ตารางกำสั่ง HD44780

Instructions

Instruction	Code										Description	Execution Time (max) (when fcp or fosc is 250 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter.	1.64 ms
Return Home	0	0	0	0	0	0	0	0	1	*	Sets DD RAM address 0 in address counter. Also returns display being shifted to original position. DD RAM contents remain unchanged.	1.64 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display. These operations are performed during data write and read.	40μs
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of entire display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor and shifts display without changing DD RAM contents.	40μs
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display lines (L) and character font (F).	40μs
Set CG RAM Address	0	0	0	1	ACG						Sets CG RAM address. CG RAM data is sent and received after this setting.	40μs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. DD RAM data is sent and received after this setting.	40μs
Read Busy Flag & Address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5×10 dots, F = 0: 5×7 dots BF = 1: Internally operating BF = 0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM ACG: CG RAM address ADD: DD RAM Address: Corresponds to cursor address AC: Address counter used for both DD and CG RAM address.	Execution time changes when frequency changes Example: When fcp or fosc is 270 kHz: $40\mu s \times \frac{250}{270} = 37\mu s$

■ Character Codes and Character Pattern

Higher 4 bit / Lower 4 bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
××××0000	CG RAM (1)												
××××0001	(2)												
××××0010	(3)												
××××0011	(4)												
××××0100	(5)												
××××0101	(6)												
××××0110	(7)												
××××0111	(8)												
××××1000	(1)												
××××1001	(2)												
××××1010	(3)												
××××1011	(4)												
××××1100	(5)												
××××1101	(6)												
××××1110	(7)												
××××1111	(8)												

ภาคผนวก ค.

NAME PLATE ของมอเตอร์ที่ใช้ในโครงงาน

THREE PHASE INDUCTION MOTOR

1 HP 4 POLE TYPE SF-J

HERTZ	50			
VOLT	220	380	FRAME	80
AMP	3.5	2.0	RATING	CONT
RPM	1410		INS CLASS	E
JIS C	4004		AMB TEMP	40 C
JP	44		WEIGHT	15 kg
JC	4		BEARING	
			620422	
SERIAL	693		620322	

ภาคผนวก ง.

รายละเอียดของเครื่องอินเวอร์เตอร์ที่ใช้ในโครงงาน

ในโครงงานเราได้ใช้เครื่องอินเวอร์เตอร์ VCD Series ที่ผลิตโดยบริษัท Mikipulley ที่มีลักษณะดังนี้

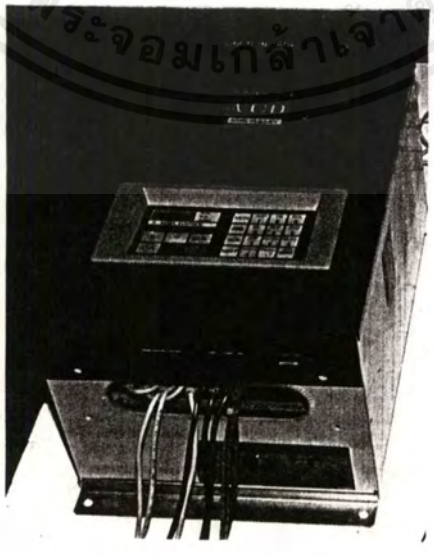
VCD - 55 - H2

Input voltage : 3-phase AC 200/220/230V 50/60Hz
H1 : 3-phase AC 400/440/460V 50/60Hz
H2 : 3-phase AC 380V 50Hz

Capacity 04.....0.4kw 110.....11kw
07.....0.75kw
15.....1.5kw
22.....
37.....
55.....
75.....7.5kw 750.....75kw

Service code

คำว่า VCD เป็นคำแทนคำว่า Service code ซึ่งมีหลายแบบ ในโครงงานเราใช้ VCD-55-H2 คือมี Capacity ขนาด 5.5kw และ Input voltage 3-phase AC 380V 50Hz, Output AC 3-phase 380V 50Hz 13A



รูป ข. เครื่องVCDอินเวอร์เตอร์

ภาคผนวก จ.

คู่มือการใช้คีย์บอร์ดในโครงงาน

คู่มือการใช้คีย์บอร์ดในโครงงาน

1. กด Mode เพื่อเลือกฟังก์ชันทิศการหมุน(Direction) ระยะทาง(Distance) ความเร็ว(Speed) ในขั้นแรกกด Mode แล้วกดปุ่ม1เพื่อเลือกทิศทางโดยเมื่อกดปุ่ม1แล้วที่หน้าจอ LCD จะปรากฏคำว่า Direction ให้เลือกทิศทางหมุนที่ต้องการโดยถ้าต้องการทิศทาง forward ให้กดปุ่ม 1 อีกครั้งหนึ่งแต่ถ้าต้องการทิศทาง reverse ให้กดปุ่ม 2 แล้วEnter

2. เมื่อทำตามข้อ 1 เสร็จแล้วให้กด Mode อีกครั้งและกดปุ่ม 2 เพื่อเลือกค่าระยะทาง โดยเมื่อกดปุ่ม 2 แล้วที่หน้าจอ LCD จะปรากฏคำว่า Distance ให้ใส่ค่าตัวเลข 4 ตัวเช่น ต้องการ20รอบให้ใส่ 0020 เป็นต้น แล้วกดEnter

3. เมื่อทำตามข้อ 2 เสร็จแล้วให้กด Mode อีกครั้งและกดปุ่ม 3 เพื่อเลือกค่าความเร็ว โดยค่าความเร็วที่จะใส่ค่านี้เป็นค่าความถี่ไฟ (เพราะความเร็วมอเตอร์นั้นสัมพันธ์กับค่าความถี่ไฟตามสมการ $N_s = 120f/p$) เมื่อกดปุ่ม 3 แล้วที่หน้าจอ LCD จะปรากฏคำว่า Speed ให้ใส่ค่าตัวเลข 2 ตัวเช่น ต้องการความถี่ไฟ 5Hz ให้ใส่ค่า 05 เป็นต้น แล้วกดEnter

4. เมื่อทำตามขั้นตอนทั้ง 3 ขั้นแล้ว ก็พร้อมที่จะเริ่ม RUN มอเตอร์ โดยกดปุ่ม Start มอเตอร์ก็จะหมุนทันที

* ในขณะกด Mode จะปรากฏคำว่า FEEDING MEASUREMENT แทรกเสมอ*

ภาคผนวก ช.

ส่วนโปรแกรม

```
RELAY           EQU     $1000           *CONTROL DIRECTION
DDAA            EQU     $1200           *DIGITIAL TO ANALOG
ENC_NEW         EQU     $1201           *INPUT FROM ENCODER
PORTC           EQU     $1202           *KEYBOARD
CNTRL_P         EQU     $1203           *CONTROL 8255
SETLCD          EQU     $1400
DATALCD         EQU     $1401
LINE_2          EQU     $40
EOT             EQU     $04             *END OF TRANSMITION

                ORG     $2200
KEYDATA         RMB     1               *DATA KEYBOARD
K_NUMBER        RMB     1               *DATA KEYBOARD ASCII
STATUS          EQU     1
K_RAM           RMB     2               *NUMBER LCD
D_SPEED         RMB     3               *SPEED LCD
D_SPEED1        RMB     1
D_SPEED2        RMB     1
D_DISTANCE      RMB     3               *DISTANCE LCD
D_DISTANCE1     RMB     1
D_DISTANCE2     RMB     1
D_DISTANCE3     RMB     1
D_DISTANCE4     RMB     1
SPEEDS          RMB     1               * DATA SPEED 16H
SPEEDS2         RMB     1
DISTANCES       RMB     2               *DATA DISTANCE 16H
DISTANCES2      RMB     2
DISTANCES5      RMB     2
DISTANCES22     RMB     2
ENC_OLD         RMB     1
TEMPP           RMB     1
TEMP1           RMB     2
COUNT           RMB     2
FIRST           RMB     1               *CHECK FIRST INPUT
SECOND          RMB     1               *CHECK NEW INPUT
XXXX            RMB     2



                ORG     $2300
INTRPT          LDS     #$3FFF
                TPA
                ANDA    #$AF
                TAP
                LDAA    #$7E
                STAA    $20F1
                STAA    $20EE
                LDX     #RESETT
                STX     $20F2
                STX     $20EF
                JSR     INITIAL
                JSR     INITKYBD        *INITIAL KEYBOARD INTERFACE
                JSR     INIT_LCD
                JSR     CLEAR_L

**********
*MAIN
**********
**********
MAIN            JSR     TITLE
MAIN1           JSR     GETKEY          *WAIT FOR KEY TO BE PRESSED
                JSR     CHECKKEY
                JMP     MAIN1
```

```
**********
INITIAL         CLRA
                CLRB
                STAA    D_DISTANCE1
                STAA    D_DISTANCE2
                STAA    D_DISTANCE3
                STAA    D_DISTANCE4
                STAA    D_SPEED1
                STAA    D_SPEED2
                STAA    STATUS
                STAA    COUNT
                STAA    COUNT+1
                STAA    FIRST
                RTS

**********
TITLE           JSR     CLEAR_L
                LDX     #DATA1          *WORD 'FEED'
                JSR     OSTRGO_L
                JSR     LINE2
                LDX     #DATA2          *WORD 'ING'
                JSR     OSTRGO_L
                JSR     DELAY
                JSR     CLEAR_L
                LDX     #DATA3          *WORD 'MEASUR'
                JSR     OSTRGO_L
                JSR     LINE2
                LDX     #DATA4          *WORD 'EMENT'
                JSR     OSTRGO_L
                RTS
**********
CHECKKEY        LDAA    KEYDATA
                CMPA    #$EE
                BEQ     MODESS
                LDAA    KEYDATA
                CMPA    #$DE
                BEQ     STARTSS
                LDAA    KEYDATA
                CMPA    #$7E
                BEQ     ENTERSS
                LDAA    KEYDATA
                CMPA    #$7B
                BEQ     KEY0
                LDAA    KEYDATA
                CMPA    #$E7
                BEQ     KEY1
                LDAA    KEYDATA
                CMPA    #$EB
                BEQ     KEY2
                LDAA    KEYDATA
                CMPA    #$ED
                BEQ     KEY3
                LDAA    KEYDATA
                CMPA    #$D7
                BEQ     KEY4
                LDAA    KEYDATA
```

```
                CMPA    #$DB
                BEQ     KEY5
                LDAA    KEYDATA
                CMPA    #$DD
                BEQ     KEY6
                LDAA    KEYDATA
                CMPA    #$B7
                BEQ     KEY7
                LDAA    KEYDATA
                CMPA    #$BB
                BEQ     KEY8
                LDAA    KEYDATA
                CMPA    #$BD
                BEQ     KEY9
                JMP     MAIN1


MODESS          JMP     MODE
STARTSS         JMP     START
ENTERSS         JMP     ENTER
KEY0            JMP     K_0
KEY1            JMP     K_1
KEY2            JMP     K_2
KEY3            JMP     K_3
KEY4            JMP     K_4
KEY5            JMP     K_5
KEY6            JMP     K_6
KEY7            JMP     K_7
KEY8            JMP     K_8
KEY9            JMP     K_9

**********
START           JSR     CLEAR_L
                LDX     #M_START
                JSR     OSTRGO_L
                JMP     MAIN_I
**********
ENTER           JSR     CLEAR_L
                LDX     #M_ENTER
                JSR     OSTRGO_L
                JSR     DELAY
                JMP     MAIN

**********
K_0             PSHA
                PSHB
                LDAA    #$30
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$30
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_1             PSHA
                PSHB
                LDAA    #$31
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$31
                STAA    K_NUMBER
```

```
                PULB
                PULA
                RTS

K_2             PSHA
                PSHB
                LDAA    #$32
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$32
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_3             PSHA
                PSHB
                LDAA    #$33
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$33
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_4             PSHA
                PSHB
                LDAA    #$34
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$34
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_5             PSHA
                PSHB
                LDAA    #$35
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$35
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_6             PSHA
                PSHB
                LDAA    #$36
                STAA    K_RAM
```

```
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$36
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_7             PSHA
                PSHB
                LDAA    #$37
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$37
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_8             PSHA
                PSHB
                LDAA    #$38
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$38
                STAA    K_NUMBER
                PULB
                PULA
                RTS

K_9             PSHA
                PSHB
                LDAA    #$39
                STAA    K_RAM
                LDAA    #EOT
                STAA    K_RAM+1
                LDAA    #$39
                STAA    K_NUMBER
                PULB
                PULA
                RTS

**********
*MODE    *
**********
**********
MODE            JSR     CLEAR_L
                LDX     #M_MODE
                JSR     OSTRGO_L
MODE1           JSR     GETKEY
                LDAA    KEYDATA
                CMPA    #$E7
                BEQ     DIRECTSS
                LDAA    KEYDATA
                CMPA    #$EB
                BEQ     DISTANCESS
                LDAA    KEYDATA
                CMPA    #$ED
                BEQ     SPEEDSS
                JMP     MODE1

DIRECTSS        JMP     DIRECTION
```

```
DISTANCESS      JMP     DISTANCE
SPEEDSS         JMP     SPEED

**********
DIRECTION       BSET    STATUS $04
                JSR     CLEAR_L
                LDX     #M_DIREC
                JSR     OSTRGO_L
                JSR     LINE2
                LDX     #M_DIREC1
                JSR     OSTRGO_L
DIREC1          JSR     GETKEY
                LDAA    KEYDATA
                CMPA    #$E7
                BEQ     FW
                LDAA    KEYDATA
                CMPA    #$EB
                BEQ     RW
                JSR     DIREC1

DIREC_E         JSR     CLEAR_L
                LDX     #M_ENTER
                JSR     OSTRGO_L
                JSR     DELAY
                JMP     MAIN

FW              PSHA
                BSET    STATUS $02
                JSR     CLEAR_L
                LDX     #M_FW
                JSR     OSTRGO_L
                PULA
FW1             JSR     GETKEY
                LDAA    KEYDATA
                CMPA    #$7E
                BEQ     DIREC_E
                JMP     FW1


RW              PSHA
                BSET    STATUS $01
                JSR     CLEAR_L
                LDX     #M_RW
                JSR     OSTRGO_L
                PULA
RW1             JSR     GETKEY
                LDAA    KEYDATA
                CMPA    #$7E
                BEQ     DIREC_E
                JMP     RW1

**********
*DISTANCE*
**********
DISTANCE        BSET    STATUS $08
                LDX     #$00
```

```
                STX     DISTANCES
                JSR     CLEAR_L
                LDX     #M_DISTANCE
                JSR     OSTRGO_L
DISTANCE1       JSR     GETKEY
                JSR     CHECKKEY
                JSR     CLEAR_L
                JSR     THOUSAND
                JSR     GETKEY
                JSR     CHECKKEY
                JSR     HUNDRED
                JSR     GETKEY
                JSR     CHECKKEY
                JSR     TEN
                JSR     GETKEY
                JSR     CHECKKEY
                JSR     ONE
DISTANCE2       JSR     GETKEY
                LDAA    KEYDATA
                CMPA    #$7E
                BEQ     DIS_E
                JMP     DISTANCE2

DIS_E           JSR     CLEAR_L
                LDX     #M_ENTER
                JSR     OSTRGO_L
                JSR     DELAY

**********
*CONVERT ASCII DISTANCE
*TO 16H DISTANCE
**********

THOUS           LDAA    D_DISTANCE1
                SUBA    #48
                CMPA    #0
                BEQ     HUNDS
                PSHX
                LDX     DISTANCES
                LDAB    #200
                ABX
                ABX
                ABX
                ABX
                ABX
                STX     DISTANCES
                PULX
                PSHA
                LDAA    D_DISTANCE1
                DECA
                STAA    D_DISTANCE1
                PULA
                JMP     THOUS
HUNDS           LDAA    D_DISTANCE2
                SUBA    #48
                CMPA    #0
                BEQ     TENS
                PSHX
                LDX     DISTANCES
                LDAB    #100
                ABX
                STX     DISTANCES
                PULX
                PSHA
```

```
                    LDAA    D_DISTANCE2
                    DECA
                    STAA    D_DISTANCE2
                    PULA
                    JMP     HUNDS
TENS                LDAA    D_DISTANCE3
                    SUBA    #48
                    CMPA    #0
                    BEQ     ONES
                    PSHX
                    LDX     DISTANCES
                    LDAB    #10
                    ABX
                    STX     DISTANCES
                    PULX
                    PSHA
                    LDAA    D_DISTANCE3
                    DECA
                    STAA    D_DISTANCE3
                    PULA
                    JMP     TENS
ONES                PSHX
                    PSHA
                    PSHB
                    LDAA    D_DISTANCE4
                    SUBA    #48
                    STAA    D_DISTANCE4
                    LDX     DISTANCES
                    LDAB    D_DISTANCE4
                    ABX
                    STX     DISTANCES
                    LDD     DISTANCES
                    LSRD
                    STD     DISTANCES2
                    LDD     DISTANCES
                    SUBD    #$0032
                    STD     DISTANCES5
                    LDD     DISTANCES
                    SUBD    #$0014
                    STD     DISTANCES22
                    PULB
                    PULA
                    PULX
                    JMP     MAIN

**********
*OUT DISTANCE TO LCD
**********
THOUSAND            PSHA
                    LDAA    K_NUMBER
                    STAA    D_DISTANCE1
                    LDX     #K_RAM
                    JSR     OSTRGO_L
                    PULA
                    RTS
```

```
HUNDRED         PSHA
                LDAA    K_NUMBER
                STAA    D_DISTANCE2
                LDX     #K_RAM
                JSR     OSTRGO_L
                PULA
                RTS

TEN             PSHA
                LDAA    K_NUMBER
                STAA    D_DISTANCE3
                LDX     #K_RAM
                JSR     OSTRGO_L
                PULA
                RTS

ONE             PSHA
                LDAA    K_NUMBER
                STAA    D_DISTANCE4
                LDX     #K_RAM
                JSR     OSTRGO_L
                PULA
                RTS

**********
* SPEED  *
**********
SPEED           BSET    STATUS $10
                LDX     #$00
                STX     SPEEDS
                JSR     CLEAR_L
                LDX     #M_SPEED
                JSR     OSTRGO_L
SPEED1          JSR     GETKEY
                JSR     CHECKKEY
                JSR     CLEAR_L
                JSR     HZ_1
                JSR     GETKEY
                JSR     CHECKKEY
                JSR     HZ_2
SPEED2          JSR     GETKEY
                LDAA    KEYDATA
                CMPA    #$7E
                BEQ     DIR_E
                JMP     SPEED2

DIR_E           PSHX
                JSR     CLEAR_L
                LDX     #M_ENTER
                JSR     OSTRGO_L
                JSR     DELAY
                PULX

**********
*CONVERT SPEED ASCII
*TO SPEED 16H
**********
HZ_1S           LDAA    D_SPEED1
                SUBA    #$30
                CMPA    #$00
                BEQ     HZ_2S
                PSHA
                LDAA    SPEEDS
                LDAB    #$0A
```

```
                ABA
                STAA    SPEEDS
                LDAA    D_SPEED1
                DECA
                STAA    D_SPEED1
                PULA
                JMP     HZ_1S
HZ_2S           PSHA
                PSHB
                LDAA    D_SPEED2
                SUBA    #$30
                STAA    D_SPEED2
                LDAA    SPEEDS
                LDAB    D_SPEED2
                ABA
                STAA    SPEEDS
                PSHA
                PSHB
                PSHX
                LDAA    SPEEDS
                LDAB    #$FF
                MUL
                LDX     #$32
                IDIV
                XGDX
                STAB    SPEEDS
                LDAA    SPEEDS
                LSRA
                STAA    SPEEDS2
                PULX
                PULB
                PULA
                PULB
                PULA
                JMP     MAIN
*********
*DATA SPEED TO LCD
*********
HZ_1            PSHA
                LDAA    K_NUMBER
                STAA    D_SPEED1
                LDX     #K_RAM
                JSR     OSTRGO_L
                PULA
                RTS

HZ_2            PSHA
                LDAA    K_NUMBER
                STAA    D_SPEED2
                LDX     #K_RAM
                JSR     OSTRGO_L
                PULA
                RTS

**********
*START FEEDING MEASUREMENT
```

```
**********
MAIN_I      LDAA    STATUS
            ANDA    #%00011100
            CMPA    #%00011100
            BEQ     RUN
            JSR     MAIN

RUN         JSR     CLEAR_L
            LDX     #M_RUNNING
            JSR     OSTRGO_L
            LDAA    SPEEDS
            STAA    DDAA
            JSR     FFRR
RUN_II      LDX     #1000
            STX     XXXX
RUN_I       LDAA    ENC_NEW
            ANDA    #%00000011
            CMPA    #%00000001
            BEQ     STATE1
            LDAA    #$00
            STAA    SECOND
            JMP     RUN_I

FFRR        LDAA    STATUS
            ANDA    #%00000011
            CMPA    #%00000001
            BEQ     RRRR
            PSHA
            LDAA    #$10
            STAA    RELAY
            PULA
            RTS
RRRR        PSHA
            LDAA    #$20
            STAA    RELAY
            PULA
            RTS

STATE1      LDAA    SECOND
            ANDA    #%00000001
            CMPA    #%00000001
            BEQ     RUN_I
            LDAA    #$01
            STAA    SECOND
            LDX     COUNT
            CPX     DISTANCES2
            BEQ     DOWN2
            LDX     XXXX
            DEX
            STX     XXXX
            LDX     XXXX
            CPX     #$0000
            BEQ     STATE2
            JMP     RUN_I
STATE2      PSHX
            LDX     COUNT
            INX
            STX     COUNT
            PULX
            JMP     RUN_II

DOWN2       LDAA    SPEEDS2         *NEXT STEP SPEED
            STAA    DDAA
DOWN2I      LDX     #1000
```

```
                STX     XXXX
DOWN2II         LDAA    ENC_NEW
                ANDA    #%00000011
                CMPA    #%00000001
                BEQ     STATE3
                LDAA    #$00
                STAA    SECOND
                JMP     DOWN2II

STATE3          LDAA    SECOND
                ANDA    #%00000001
                CMPA    #%00000001
                BEQ     DOWN2II
                LDAA    #$01
                STAA    SECOND
                LDX     COUNT
                CPX     DISTANCES5
                BEQ     DOWN5
                LDX     XXXX
                DEX
                STX     XXXX
                LDX     XXXX
                CPX     #$0000
                BEQ     STATE4
                JMP     DOWN2II
STATE4          PSHX
                LDX     COUNT
                INX
                STX     COUNT
                PULX
                JMP     DOWN2I

DOWN5           LDAA    #$19            *NEXT STEP SPEED
                STAA    DDAA
DOWN5I          LDX     #1000
                STX     XXXX
DOWN5II         LDAA    ENC_NEW
                ANDA    #%00000011
                CMPA    #%00000001
                BEQ     STATE5
                LDAA    #$00
                STAA    SECOND
                JMP     DOWN5II
STATE5          LDAA    SECOND
                ANDA    #%00000001
                CMPA    #%00000001
                BEQ     DOWN5II
                LDAA    #$01
                STAA    SECOND
                LDX     COUNT
                CPX     DISTANCES22
                BEQ     DOWNE1
                LDX     XXXX
                DEX
                STX     XXXX
                LDX     XXXX
```

```
                CPX     #$0000
                BEQ     STATE6
                JMP     DOWN5II
STATE6          PSHX
                LDX     COUNT
                INX
                STX     COUNT
                PULX
                JMP     DOWN5I

**********
FIRSTT          BSET    FIRST $01       *LAST STEP SPEED
                LDAA    ENC_NEW
                ANDA    #%00000011
                STAA    ENC_OLD
                JMP     DOWNE2

DOWNE1          LDAA    #$0A
                STAA    DDAA
                LDAA    FIRST
                CMPA    #$00
                BEQ     FIRSTT
DOWNE2          LDX     #3990           *INERTIA MOTOR
                STX     XXXX
DOWNE3          LDAA    ENC_NEW
                ANDA    #%00000011
                STAA    TEMPP
                CMPA    #$00
                BEQ     BRANCH1
                LDAA    TEMPP
                CMPA    #$01
                BEQ     BRANCH2
                LDAA    TEMPP
                CMPA    #$02
                BEQ     BRANCH3
                LDAA    TEMPP
                CMPA    #$03
                BEQ     BRANCH4
                JMP     DOWNE3

BRANCH1         LDAA    ENC_OLD
                CMPA    #$00
                BEQ     NOCHANGE
                CMPA    #$01
                BEQ     CCW
                CMPA    #$02
                BEQ     CW
                CMPA    #$03
                BEQ     ERROR
                JMP     BRANCH1
BRANCH2         LDAA    ENC_OLD
                CMPA    #$00
                BEQ     CW
                CMPA    #$01
                BEQ     NOCHANGE
                CMPA    #$02
                BEQ     ERROR
                CMPA    #$03
                BEQ     CCW
                JMP     BRANCH2
NOCHANGE        JMP     NOCHANGE1
CCW             JMP     CCW1
CW              JMP     CW1
```

```
ERROR           JMP     STOPP

BRANCH3         LDAA    ENC_OLD
                CMPA    #$00
                BEQ     CCW
                CMPA    #$01
                BEQ     ERROR
                CMPA    #$02
                BEQ     NOCHANGE
                CMPA    #$03
                BEQ     CW
                JMP     BRANCH3
BRANCH4         LDAA    ENC_OLD
                CMPA    #$00
                BEQ     ERROR
                CMPA    #$01
                BEQ     CW
                CMPA    #$02
                BEQ     CCW
                CMPA    #$03
                BEQ     NOCHANGE
                JMP     BRANCH4

NOCHANGE1       LDAA    TEMPP
                STAA    ENC_OLD
                JMP     DOWNE3
CW1             LDAA    TEMPP
                STAA    ENC_OLD
                LDX     XXXX
                DEX
                STX     XXXX
                CPX     #$0000
                BEQ     MINUSS
                JMP     DOWNE3
CCW1            LDAA    TEMPP
                STAA    ENC_OLD
                LDX     XXXX
                DEX
                STX     XXXX
                CPX     #$0000
                BEQ     PLUSS
                JMP     DOWNE3

PLUSS           LDX     COUNT
                INX
                STX     COUNT
                CPX     DISTANCES
                BEQ     STOPP
                JMP     DOWNE2

MINUSS          LDX     COUNT
                INX
                STX     COUNT
                CPX     DISTANCES
                BEQ     STOPP
                JMP     DOWNE2
```

```
STOPP       PSHA
            LDAA    #$00
            STAA    DDAA
            STAA    RELAY
            PULA
            JSR     CLEAR_L
            LDX     #M_STOP
            JSR     OSTRGO_L

STOPPP      BRA     STOPPP

RESETT      JSR     CLEAR_L
            LDX     #M_RESET
            JSR     OSTRGO_L
            JSR     DELAY
            JMP     INTRPT


**********
*KEYBOARD*
**********
**********
INITKYBD    PSHA
            LDAA    #%10001010      *PORTA O/P,PORTB I/P
            STAA    CNTRL_P         *PORTC 4 BIT LOW O/P
            PULA                    *PORTC 4 BIT HIGH I/P
            RTS

**********
GETKEY      PSHB
GETKEY1     JSR     IDKEY           *RETURN KEY CODE IN ACCA
            CMPA    #0
            BEQ     GETKEY1         *IF KEY=0 THEN REPEAT SCAN
            JSR     DEBOUNCE
            PSHA
            JSR     IDKEY           *RETURN NEW KEY CODE IN ACCA
            PULB
            CBA
            BNE     GETKEY1         *ELSE RETURN
            PULB
            RTS

***********
BREAKKEY    PSHA
BREAK1      JSR     IDKEY           *KEY CODE OF ZERO
            CMPA    #0              *MEANS BREAK OCCURRED
            BNE     BREAK1
            JSR     DEBOUNCE
            JSR     IDKEY           *CHECK FOR BREAK AGAIN
            CMPA    #0
            BNE     BREAK1
            PULA
            RTS

***********
IDKEY       PSHY                    *PRESERVE REGISTERS
            PSHB
            LDY     #KYTAB          *POINT TO TABLE
            CLRA                    *KEY=0
IDKEY1      INCA
            INY
            LDAB    0,Y             *DRIVE PORTC OUTPUT
            STAB    PORTC
```

```
                    STAB    KEYDATA
                    PSHA
                    LDAA    PORTC
                    CBA
                    PULA
                    BEQ     IDKEY2
                    CMPA    #16
                    BLO     IDKEY1
                    CLRA
IDKEY2              PULB
                    PULY
                    RTS



**********
DEBOUNCE            PSHX
                    LDX     #$0D06          *INIT LOOP COUNTER
DEBOUNCE1           DEX
                    BNE     DEBOUNCE1
                    PULX
                    RTS

**********
KYTAB               FCB     $FF             *KEY CODE OF ZERO, DON'T CAR
                    FCB     $EE,$DE,$BE,$7E
                    FCB     $ED,$DD,$BD,$7D
                    FCB     $EB,$DB,$BB,$7B
                    FCB     $E7,$D7,$B7,$77


**********
*INITLCD
**********
INIT_LCD            LDAA    #$10            * clear display
                    STAA    SETLCD
                    JSR     DELAY_L
                    LDAA    #%00111000      * set function 8 bit,5x7,2 l
                    STAA    SETLCD          * DL=1, N=1, F=0
                    JSR     DELAY_L
                    LDAA    #%00001100      * set display on, cursor off
                    STAA    SETLCD          * D=1, C=0, B=0
                    JSR     DELAY_L
                    LDAA    #%00000110      *set mode,DDram address incr
                    STAA    SETLCD          *I/D=1,S=0
                    JSR     DELAY_L
                    RTS

**********
*CLEAR DISPLAY LCD
**********
CLEAR_L             JSR     DELAY_L
                    JSR     READY_L
                    PSHA
```

```
                LDAA    #$01
                STAA    SETLCD
                PULA
                RTS


**********
*CURSOR LOCATION
*A=address
**********
CURSOR_L        JSR     DELAY_L
                JSR     READY_L
                PSHA
                ADDA    #$80
                STAA    SETLCD
                PULA
                RTS

**********
LINE2           LDAA    #LINE_2
                JSR     CURSOR_L
                RTS


**********
*cursor shift right 1 position
**********
SHIFTR_L        JSR     READY_L
                PSHA
                LDAA    #$14
                STAA    SETLCD
                PULA
                RTS

**********
*cursor shift left 1 position
**********
SHIFTL_L        JSR     READY_L
                PSHA
                LDAA    #$10
                STAA    SETLCD
                PULA
                RTS

**********
*write byte of lcd
*A=data (ASCII)
**********
OUT_L           JSR     READY_L
                STAA    DATALCD
                RTS

*********
*write data HEX to ASCII
*A=data (HEX)
*********
OUTH_L          PSHA
                PSHA
                JSR     OLHLF_L
                PULA
                JSR     ORHLF_L
                PULA
                RTS
```

```
**********
*OLHLF_L,ORHLF_L
*convert A from binary to ASCII and output
**********
OLHLF_L         LSRA
                LSRA
                LSRA
                LSRA
ORHLF_L         ANDA    #$0F
                ADDA    #$30
                CMPA    #$39
                BLE     OUTA_L
                ADDA    #$07
OUTA_L          JSR     OUT_L
                RTS

**********
*convert the byte at X to two
*ASCII characters and output. return X pointing
*to next byte
**********
O1BYT_L         PSHA
                LDAA    0,X
                PSHA
                BSR     OLHLF_L
                PULA
                BSR     ORHLF_L
                PULA
                INX
                RTS

**********
*output 1or 2 bytes
*at X followed by a space. returns X pointing to
*next byte
**********
O2BSP_L         JSR     O1BYT_L
O1BSP_L         JSR     O1BYT_L
OSPAC_L         PSHA
                LDAA    #$20
                JSR     OUT_L
                PULA
                RTS

**********
*output string of ASCII bytes
*starting at X unit end of text
**********
OSTRGO_L        PSHA
OSTRG1_L        LDAA    0,X             *list of fix word
                CMPA    #EOT            *increase x result in displa
                BEQ     OSTRG3_L        *STRING
                JSR     OUT_L
                INX
                BRA     OSTRG1_L
OSTRG3_L        PULA
```

```
                    RTS

 **********
 *output string of ASCII bytes
 **********
 OSTR_L             PSHA
 OSTR1_L            LDAA    0,X
                    CMPA    #$0D
                    BEQ     OSTR3_L
                    JSR     OUT_L
                    INX
                    BRA     OSTR1_L
 OSTR3_L            INX
                    INX
                    PULA
                    RTS

 **********
 *read busy flag
 **********
 READY_L            PSHA
 READY1_L           LDAA    SETLCD
                    ANDA    #$80
                    BNE     READY1_L
                    PULA
                    RTS

 **********
 *delay
 **********
 DELAY_L            PSHX
                    LDX     #$1000
 DELAY1_L           DEX
                    BNE     DELAY1_L
                    PULX
                    RTS

 **********
 DELAY              PSHA
                    PSHX
                    LDAA    #$4
 DELAY1             LDX     #$FFFF
 DELAY2             DEX
                    BNE     DELAY2
                    DECA
                    BNE     DELAY1
                    PULX
                    PULA
                    RTS


 ***********
 DATA1              FCC     '    FEED'
                    FCB     EOT

 DATA2              FCC     'ING     '
                    FCB     EOT

 DATA3              FCC     '  MEASUR'
                    FCB     EOT

 DATA4              FCC     'EMENT   '
                    FCB     EOT
```

```
M_MODE          FCC     'MODE'
                FCB     EOT

M_DIREC         FCC     'DIRECTIO'
                FCB     EOT

M_DIREC1        FCC     'N'
                FCB     EOT

M_DISTANCE      FCC     'DISTANCE'
                FCB     EOT

M_SPEED         FCC     'SPEED'
                FCB     EOT

M_FW            FCC     'FORWARD'
                FCB     EOT

M_RW            FCC     'REVERSE'
                FCB     EOT

M_START         FCC     'START'
                FCB     EOT

M_STOP          FCC     'STOP'
                FCB     EOT

M_ENTER         FCC     'ENTER'
                FCB     EOT

M_ERROR         FCC     'ERROR'
                FCB     EOT

M_RUNNING       FCC     'RUNNING'
                FCB     EOT


M_RESET         FCC     'RESET'
                FCB     EOT


M_MODE          FCC     'MODE'
                FCB     EOT

M_DIREC         FCC     'DIRECTIO'
                FCB     EOT

M_DIREC1        FCC     'N'
                FCB     EOT

M_DISTANCE      FCC     'DISTANCE'
                FCB     EOT

M_SPEED         FCC     'SPEED'
                FCB     EOT

M_FW            FCC     'FORWARD'
                FCB     EOT

M_RW            FCC     'REVERSE'
                FCB     EOT

M_START         FCC     'START'
                FCB     EOT
```

```
M_STOP          FCC     'STOP'
                FCB     EOT

M_ENTER         FCC     'ENTER'
                FCB     EOT

M_ERROR         FCC     'ERROR'
                FCB     EOT

M_RUNNING       FCC     'RUNNING'
                FCB     EOT

M_RESET         FCC     'RESET'
                FCB     EOT
```

กิตติกรรมประกาศ

รายงานฉบับนี้ได้สำเร็จลงได้ด้วยความช่วยเหลือจากบุคคลหลายท่าน

ขอขอบคุณอาจารย์ที่ปรึกษา

อาจารย์พิชิต ลำยอง ที่ให้คำปรึกษาโครงงาน

ขอขอบคุณรุ่นพี่ พี่ชนนวรรธ ห้าวหาญ (พี่นก)
และขอขอบคุณเพื่อน ๆปี4 และรุ่นน้องทุกคนตลอดจนเจ้าหน้าที่ในภาควิศวกรรมไฟฟ้าที่ได้ให้ความช่วยเหลือด้านต่าง ๆ

เอกสารอ้างอิง

[1] Yasuhiko Dote, "Servo Motor and Motion Control using Digital Signal Processor", McGraw Hill.

[2] M Chilikin, "Electric Drive", Translated from russian by Jacob Feinberg, Mir Publisher Moscows

[3] Peter Spasov, "Microcontroller Technology THE 68HC11", Prentice Hall

[4] Mikipulley, "High Performance Inverter VCD Series Instruction Manual"

[5] ชัยวัฒน์ ลิ้มพรจิตรวิไล, "เรียนรู้และใช้งานไมโครคอนโทรลเลอร์ 68HC11, ซีเอ็ด 2538