การพัฒนาต้นแบบไมโครเมาส์ที่ใช้ในการแข่งขันเพื่อการศึกษา

Development of Competitive Micromouse's prototype For Education

โดย

นาย ชาคริต เศวตบวร

นาย ชาญชัย นิรุตตินานนท์

อาจารย์ที่ปรึกษา

รศ. สมศักดิ์ มิตะถา

อาจารย์ นเรศ มาลัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาต้นแบบไมโครเมาส์ที่ใช้ในการแข่งขันเพื่อการศึกษา

DEVELOPMENT OF COMPETITIVE MICROMOUSE'S PROTOTYPE FOR EDUCATION

ผู้จัดทำ

1. นาย ชาคริต เศวตบวร รหัสประจำตัว 41014102
2. นาย ชาญชัย นิรุตตินานนท์ รหัสประจำตัว 41014103

_____ อาจารย์ที่ปรึกษา

(รศ. สมศักดิ์ มิตะถา)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ นเรศ มาลัย)

การพัฒนาต้นแบบไมโครเมาส์ที่ใช้ในการแข่งขันเพื่อการศึกษา

นาย ชาคริต เศวตบวร
นาย ชาญชัย นิรุตตินานนท์

รศ. สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
อาจารย์ นเรศ มาลัย อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ในปัจจุบันการที่จะศึกษาการใช้ไมโครคอนโทลเลอร์ในการควบคุมอุปกรณ์ต่างๆนั้น สามารถที่จะเริ่มศึกษาจากไมโครเมาส์ เนื่องจากเป็นที่รู้จักอย่างกว้างขวาง มีการกำหนดเป็นรูปแบบที่เป็นมาตรฐานและมีการจัดการแข่งขันอยู่เสมอ แต่ไมโครเมาส์ในปัจจุบันนั้น ส่วนมากเน้นเพื่อการแข่งขันจึงทำให้ยากต่อการเข้าใจและเรียนรู้ต่อบุคคลทั่วไปที่สนใจ

โครงงานไมโครเมาส์นี้ได้จัดทำขึ้นนี้เป็นการพัฒนาจากตัวต้นแบบให้มีประสิทธิภาพที่สูงขึ้น สามารถใช้ในการแข่งขันจริงได้ โดยใช้อุปกรณ์ที่หาได้ง่าย ราคาถูก แต่มีประสิทธิภาพดี ในส่วนของการออกแบบทางด้านซอร์ฟแวร์และฮาร์ดแวร์ จะใช้วิธีการของObject-Oriented เพื่อให้ง่ายต่อการเรียนรู้และเข้าใจถึงวิธีการใช้ซอร์ฟแวร์ ในการควบคุมการทำงานของอุปกรณ์รวมถึงวิธีการทำงานของฮาร์ดแวร์ แต่ละส่วน ในโครงงานนี้จะใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ของ Atmel สำหรับในส่วนของโปรแกรมที่ใช้ควบคุมจะพัฒนาโดยภาษาแอสเซมบลี และ C-51 โดยขึ้นอยู่กับความซับซ้อนของฟังก์ชั่นการทำงานที่เราต้องการ

Development of Competitive Micromouse's Prototype For Education

Chakrit Savedboworn 41014102

Chanchai Nirutinanon 41014103

Assoc. Prof. Somsak Mitatha Advisor

Nareth Malai Advisor

ABSTRACT

Today, the use of microcontrollers can be studied from the micromouses which are well known and have standard defined. Most of the micromouses are aimed for competition, so their design are complex and this make them hard to be studied. .

The main goal of this project is to develop the micromouse which can be used in the real competition but still simple enough to be studied by the others. Another goal of the project is to create the micromouse which its components can be found easily in the market and also at low price. We apply the object-oriented designing methodology to both hardware and software design. In this project, the Atmel's microcontroller MCS-51 is implemented as the main controlling system of the micromouse, and the controlling software is developed base on assembly and C-51 language.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ หากไม่ได้รับความช่วยเหลือ หลายๆฝ่ายด้วยกันบุคคลแรกที่ต้องกล่าวถึงเพราะถ้าไม่มีบุคคลท่านนี้วิทยานิพนธ์ฉบับนี้ก็ไม่อาจสำเสร็จลงได้ก็คือ อาจารย์ สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอบคุณพี่ๆ เพื่อนๆ ที่คอยให้ทำปรึกษา เอ็นเตอร์เทนยามเหงา อาหารยามท้องหิว รวมทั้งเสียงเพลง เสียงกีตาร์ที่คอยขับกล่อม

ขอบคุณน้องๆ ที่คอยถามใถ่ความคืบหน้าของโปรเจค

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

ชาคริต เศวตบวร

ชาญชัย นิรุตตินานนท์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เนื่องจากในปัจจุบันการพัฒนาไมโครเมาส์ได้ก้าวหน้าไปมาก ทั้งทางด้านฮาร์ดแวร์และทางด้านซอร์ฟแวร์ ซึ่งผู้พัฒนาจำนวนมากได้ทำการวิจัยและเพิ่มประสิทธิภาพการทำงานอยู่เสมอโดยเน้นที่การแข่งขัน ทำให้ไมโครเมาส์มีการทำงานมีความซับซ้อนมาก มีการใช้อุปกรณ์ที่มีราคาแพง ทำให้มีผู้ที่เริ่มต้นสนใจไม่สามารถทำความเข้าใจ เกี่ยวกับระบบการทำงานของไมโครเมาส์ เช่นในการศึกษาการใช้ไมโครคอนโทรลเลอร์ในการควบคุมอุปกรณ์ต่างๆ เป็นต้น

เพื่อสนับสนุนกลุ่มบุคคลดังกล่าว จึงได้มีโครงการนี้ขึ้นมา สามารถลดค่าใช้จ่ายในการศึกษา สามารถเข้าใจการทำงานได้ง่าย

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อนำไมโครเมาส์มาทางเลือกหนึ่ง ในการศึกษาการใช้งานไมโครคอนโทรลเลอร์ในลักษณะต่างๆ เช่น การควบคุมอุปกรณ์ การติดต่อกับคอมพิวเตอร์

1.2.2 เพื่อให้ไมโครเมาส์มีความแพร่หลาย เป็นที่รู้จักมากยิ่งขึ้นมากขึ้น

1.2.3 เพื่อเป็นต้นแบบในการสร้างและออกแบบไมโครเมาส์ใช้วิธีของ ออบเจค-โอเรียลเต็ด(Object-Oriented) มาพัฒนาการออกแบบ นำความรู้ที่ได้รับนี้ไปประยุกต์ใช้ในงานต่างๆ ได้

1.2.4 เพื่อเป็นการนำเสนอแนวคิด และความเป็นไปได้ในการไมโครเมาส์ที่มีความสามารถสูง มีราคาไม่แพง สามารถใช้งานและศึกษาได้ง่าย

1.2.5 ใช้งานไมโครคอนโทรลเลอร์ และเขียนโปรแกรมควบคุมด้วย แอสแซมบลี (Assembly) และ C-51

1.2.6 เพื่อให้สามารถออกแบบ และสร้างวงจรทางด้าน ฮาร์ดแวร์ และซอร์ฟแวร์

1.3 ขอบเขตของงานวิจัย

1.3.1 สามารถใช้เป็นสื่อการสอน เพื่อให้เข้าใจในการใช้งานไมโครคอนโทรลเลอร์ในลักษณะต่างๆ เพื่อให้ผู้เข้าใจ สามารถออกแบบ,สร้างวงจรทางด้านฮาร์ดแวร์ และเขียนโปรแกรมควบคุมได้

1.3.2 สามารถนำไปพัฒนาต่อในเชิงพาณิชย์ และใช้ในการแข่งขัน เนื่องจากใช้อุปกรณ์ที่มีราคาไม่แพง แต่มีประสิทธิภาพ สามารถใช้ในการแข่งขันได้

1.3.3 มีฟังก์ชั่นพื้นฐานและฟังก์ชั่นที่ใช้ในการควบคุมอุปกรณ์ต่างๆเพื่อความสะดวกในการเขียนโปรแกรมและเป็นตัวอย่าง

1.3.4 สามารถส่งคำสั่งที่เขียนแล้วไปยังไมโครเมาส์ได้

1.3.5 มีอีดิตเตอร์สำหรับเขียนโปรแกรม คอมไพล์และ ส่งโปรแกรมไปยังไมโครเมาส์

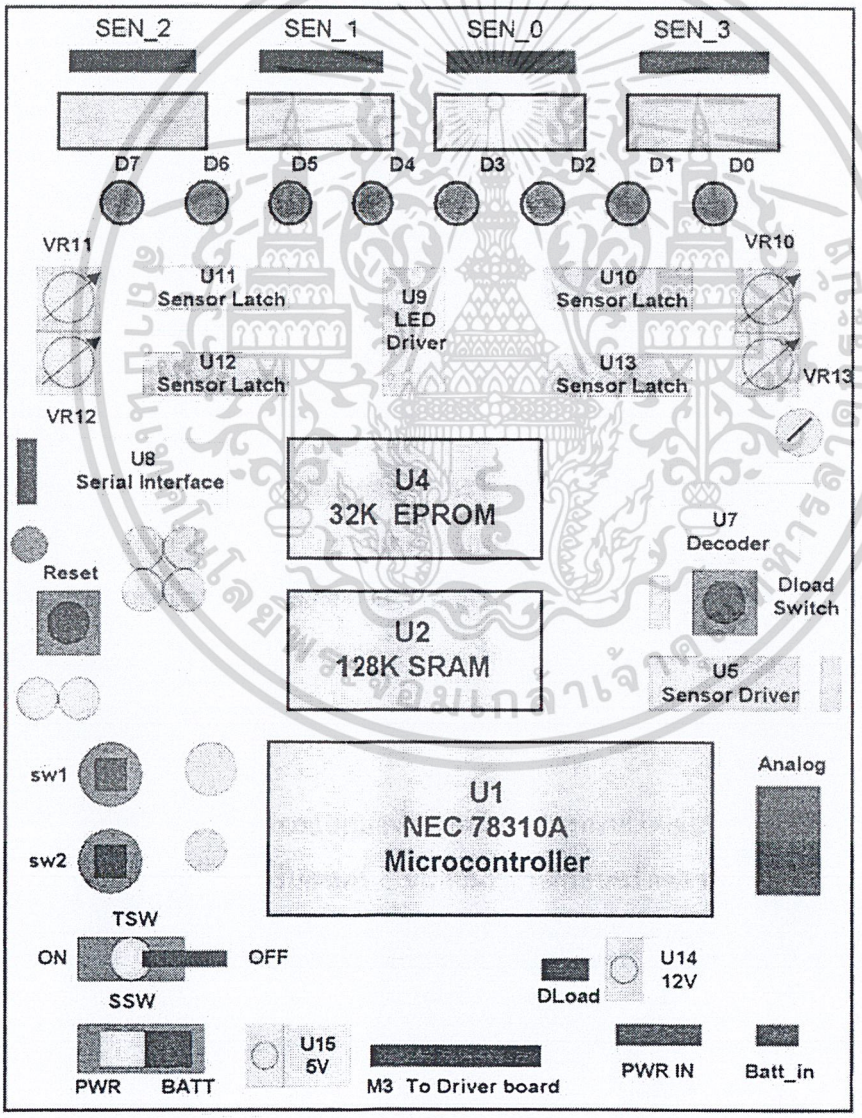
บทที่ 2
ลักษณะไมโครเมาส์ที่มีในท้องตลาด

อธิบายถึงข้อมูลรายละเอียดเกี่ยวกับความสามารถของไมโครเมาส์ที่มีจำหน่ายในท้องตลาด ทางด้านฮาร์ดแวร์และซอฟท์แวร์ และหลักการทำงานของอุปกรณ์ส่วนต่าง ๆ ของโครงงานไมโครเมาส์

ส่วนประกอบต่าง ๆ ออกเป็น 2 ส่วนใหญ่ ๆ คือ ฮาร์ดแวร์และซอร์ฟแวร์

2.1 ฮาร์ดแวร์ของเครื่องต้นแบบ

จากรูปข้างล่างนี้แสดงถึงการวางตำแหน่งของไมโครคอนโทรลเลอร์บอร์ด ซึ่งมีส่วนประกอบต่าง ๆ ด้วยกันดังนี้

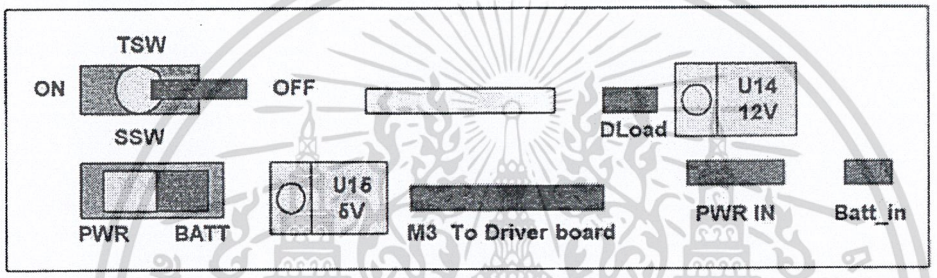


รูปที่ 2.1 ไมโครคอนโทรลเลอร์บอร์ด

2.1.1 ระบบจ่ายพลังงาน

ไมโครเมาส์ต้องการแหล่งจ่ายไฟด้วยกัน 3 ส่วน ซึ่งมีแรงดันที่แตกต่างกัน อันแรกเป็นแรงดันที่ไปเลี้ยงวงจรลอจิกและไมโครคอนโทรลเลอร์ (Vcc) อีกอันหนึ่งเอาไว้สำหรับเลี้ยงตัวส่งของชุดอินฟาเรด ใช้แรงดัน 12 V และส่วนสุดท้ายไว้สำหรับเลี้ยงวงจรขับสเตปปิ้งมอเตอร์ สามารถใช้แรงดันตั้งแต่ 15-24 V แรงดันนี้ได้มาจากแบตเตอรี่ หรือ จากแหล่งจ่ายไฟ DC

แหล่งจ่ายไฟ 5V หรือ Vcc ที่ได้มาจากแหล่งจ่ายภายนอก ซึ่งใช้วงจร สวิตซ์ชิ่ง เรคกูเลเตอร์ (Switching Regulator) DC-DC เป็นตัวแปลงแรงดันให้เหลือ 5V เป็นวงจรที่มีเสถียรภาพมากกว่าวงจร Regulator ทางลิเนียร์(linear) ซึ่งในโครงงานนี้ได้ใช้ไอซีเบอร์ LM2575T5(U15) สามารถดูอุปกรณ์ตามรูปข้างล่างนี้ได้



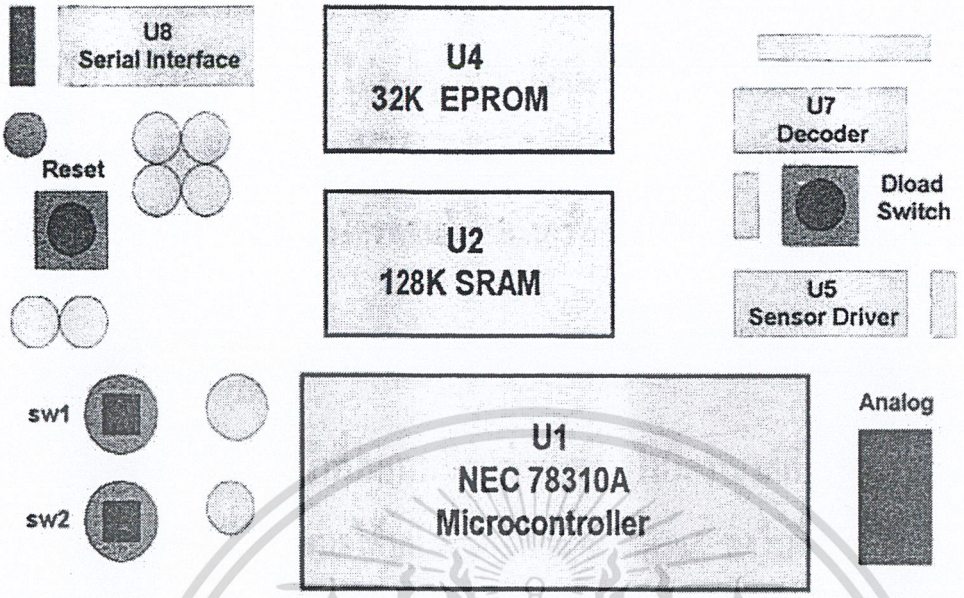
รูปที่ 2.2 ชุดจ่ายไฟของไมโครคอนโทรลเลอร์บอร์ด

แรงดันที่จ่ายให้กับตัวเซนเซอร์ ซึ่งใช้แรงดัน12Vโดยเราใช้ ไอซีเรคกูเลต(IC Regulate) ธรรมดาเบอร์ L7812 (U14) แรงดันเอาท์พุทจะมีค่าเท่ากับ 12V แม้ว่า Regulator แบบ Linear จะมีประสิทธิภาพที่ด้อย แต่ยังอยู่ในเกณฑ์ที่ยอมรับได้ เนื่องจากวงจรในส่วนนี้ใช้กระแสไม่เกิน 100 mA แรงดันนี้จะถูกจ่ายไปยังชุดของเซนเซอร์ทั้ง 4 คือ SEN_0 – SEN_3

แรงดันที่จ่ายให้กับชุดขับสเตปปิ้งมอเตอร์ (Stepping Motor) จะใช้แรงดันจากแหล่งจ่ายไฟที่ป้อนเข้าโดยตรง ในกรณีที่แหล่งจ่ายไฟได้จากแบตเตอรี่ หากแรงดันของแบตเตอรี่ลดลง ก็ไม่เป็นอุปสรรคต่อการทำงานของมอเตอร์ เพราะโดยประสิทธิภาพของมอเตอร์นั้น สามารถทำงานได้แม้ว่าแรงดันจะลดต่ำลงก็ตาม แรงดันส่วนนี้จะป้อนให้กับชุดขับมอเตอร์อีกทางหนึ่งด้วย โดยผ่านคอนเนคเตอร์(Connector) M3

2.1.2 ไมโครคอนโทรลเลอร์บอร์ด

ไมโครคอนโทรลเลอร์ ได้ออกแบบมาเพื่อสำหรับไมโครเมาส์โดยเฉพาะ มีจำนวน I/O 24 บิท สำหรับเซนเซอร์ (Sensor) และสามารถทดสอบหรือตรวจสอบการทำงานของชุดขับสเตปปิ้งมอเตอร์ได้ โดยผ่านพอร์ต (port) สื่อสารอนุกรมได้



รูปที่ 2.3 ส่วนไมโครคอนโทรลเลอร์บอร์ด

- **ไมโครคอนโทรลเลอร์**

ในโครงงานนี้ได้ใช้ไมโครคอนโทรลเลอร์เบอร์ NEC 78K310A (U1) ขนาด 8 บิท เป็นเทคโนโลยี CMOS ความเร็วสูง มี 64 ขา ซึ่งมีคุณสมบัติทางเทคนิคดังนี้

- ทำงานที่ความถี่ 6 MHz ใช้คริสตอล 12 MHz
- มีหน่วยความจำภายใน 128 byte
- มีALU ที่สามารถคำนวณ ได้ทั้ง 8bit และ16 bit
- สามารถคูณและหารข้อมูลได้ทั้ง 8bit และ16 bit
- สามารถอ้างหน่วยความจำได้ 64 Kbyte
- มี I/O 24 bit
- มีวงจร Timer 2 ชุด
- มีวงจร Counter 2 ชุด
- มีวงจร A to D 4 ชุด
- มีพอร์ตสื่อสารแบบอนุกรม (UART)

- **การกำหนดหน่วยความจำ**

หน่วยความจำของระบบที่พัฒนาขึ้นจะใช้ EPROM ขนาด 32 Kbyte (U3) และ RAM ขนาด 128 Kbyte (U4) ซึ่งมี GAL เป็นตัว กำหนดแอดเดส ให้กับหน่วยความจำ

- สวิตช์

บนบอร์ดจะมีสวิตช์รีเซต (Reset) ซึ่งมีหน้าที่รีเซทระบบการทำงานทั้งหมดให้เริ่มต้นใหม่ โดยโปรแกรมจะเริ่มอ่านคำสั่งจากตัว EPROM โดยเวลาที่ใช้ในการรีเซต 10 ไมโครวินาที

สวิตช์ทางด้านขวาจะเป็นสวิตช์ Dload ที่สั่งให้ไมโครคอนโทรลเลอร์มาเอาคำสั่งจากหน่วยความจำมาประมวลผลซึ่งโปรแกรมนี้ได้มาจาก โปรแกรมที่เขียนจากคอมพิวเตอร์แล้วแปลงเป็นภาษาเครื่อง

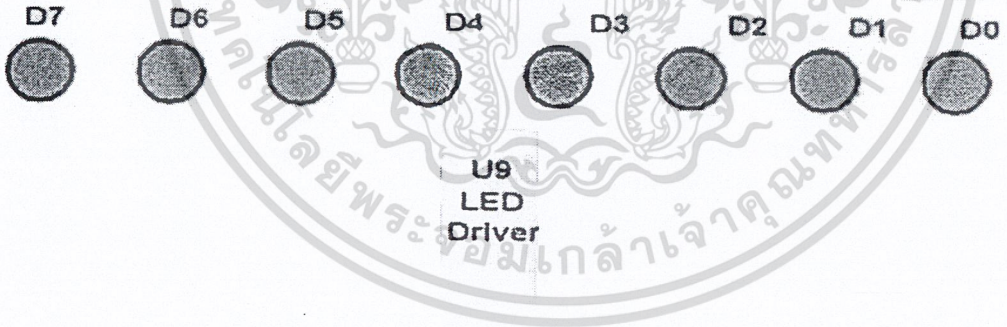
สวิตช์ทางซ้าย 2 ตัวซึ่งออกแบบไว้ใช้งานสำหรับ ผู้ใช้ประกอบด้วย Sw1 และ Sw2 สวิตช์สองตัวนี้เมื่อกดจะไปทำการอินเตอร์รัพท์(interrrupt)ไมโครคอนโทรลเลอร์

- การสื่อสารอนุกรม

พอร์ตสื่อสารอนุกรม ทำหน้าที่ในการเชื่อมต่อระหว่างไมโครเมาส์กับคอมพิวเตอร์ โปรแกรมของไมโครเมาส์ สามารถที่จะพัฒนาได้บน PC และ ดาวน์โหลดลงไมโครเมาส์ผ่านพอร์ตอนุกรม RS232 โดยในโครงงานนี้จะใช้ไอซี MAX232 เพื่อปรับระดับของสัญญาณให้อยู่ในระดับที่สามารถติดต่อกับเครื่องคอมพิวเตอร์ได้

- ส่วนแสดงผล

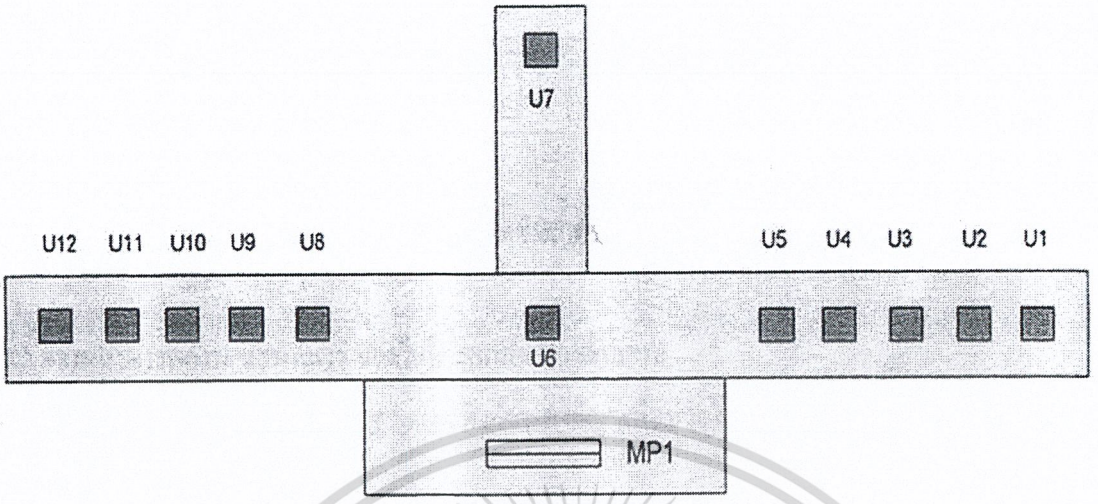
LED ทั้ง 8 ตัว บนบอร์ด จะถูกออกแบบมาให้แสดงสถานะของเซนเซอร์ที่ได้รับ เราสามารถใช้ LED เหล่านี้ในการตรวจสอบการทำงานของเซนเซอร์ วงจร DISPLAY นี้จะมีไอซี U9 เป็นตัวขับ LED



รูปที่ 2.4 ชุดขับ LED

2.1.3 เซนเซอร์

ในชุดของเซนเซอร์จะมีวงจรตัวส่งและตัวรับ ตัวขับตัวส่งนั้นจะใช้ทรานซิสเตอร์แบบ dalington ULN2003 (U5) ซึ่งถูกออกแบบเพียงพอต่อการขับตัวส่งสำหรับตัวส่ง LED ตัวรับนั้นจะประกอบด้วย RC filter และวงจร Latch ในการอ่านข้อมูลมาเก็บไว้ที่วงจร Latch วงจรชุดนี้สามารถใช้ในการตรวจสอบกำแพง



รูปที่ 2.5 ชุดเซนเซอร์

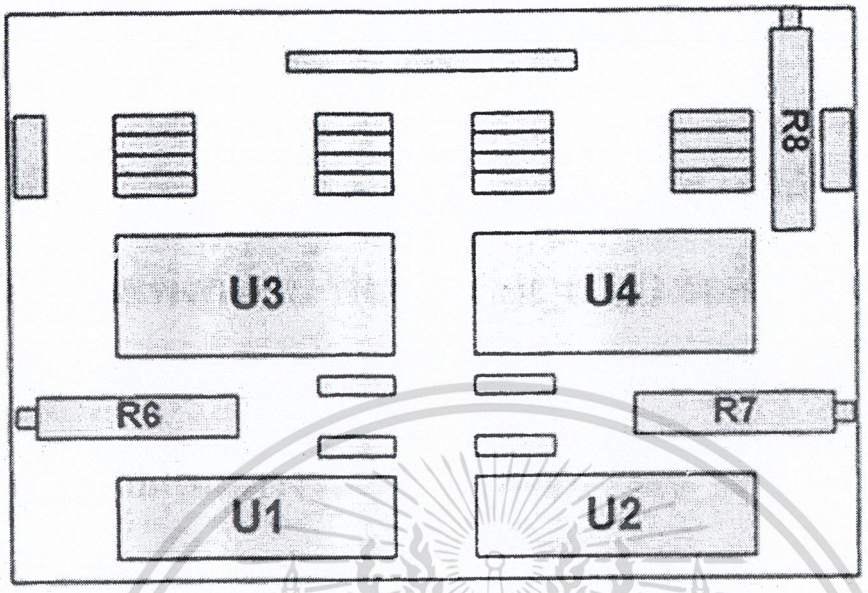
2.1.4 อินพุทเอาท์พุท

คอนเนคเตอร์บนบอร์ด จะเชื่อมต่อกับอุปกรณ์รอบข้างเพื่อรับส่งข้อมูล จากวงจรหนึ่งไปอีกวงจรหนึ่ง ซึ่งมีดังนี้

- BATT_IN เป็นคอนเนคเตอร์นี้จะเป็นตัวที่ใส่แหล่งจ่ายไฟจากแบตเตอรี่ ซึ่งมีแรงดันในช่วง 15 –24 โวลต์
- PWR_IN เป็นคอนเนคเตอร์นี้จะไว้ต่อกับแหล่งจ่ายไฟ ซึ่งสามารถชาร์จแบตเตอรี่ได้ด้วย
- M3 เป็นคอนเนคเตอร์ที่เชื่อมต่อกับชุดขับมอเตอร์
- RS_232 เป็นคอนเนคเตอร์ที่เชื่อมต่อกับพอร์ตอนุกรมของเครื่องคอมพิวเตอร์
- SEN_0 – SEN_3 เป็นคอนเนคเตอร์ที่เชื่อมต่อกับชุดเซนเซอร์
- Dload เป็น jumper ที่ใช้เลือกทำงานจากโปรแกรมที่ดาว์นโหลดมา หรือ จาก EPROM
- ANALOG เป็นคอนเนคเตอร์ที่รับสัญญาณ Anolog เข้ามาเพื่อแปลงเป็นดิจิตอล

2.1.5 วงจรขับสเตปปิ้งมอเตอร์

วงจรนี้คือวงจรที่ทำหน้าที่เป็นตัวขับให้สเตปปิ้งมอเตอร์ให้ทำงานได้ โดยจะรับคำสั่งจากตัวไมโครคอนโทรลเลอร์ ซึ่งเราใช้ L298 (U3,4) และ L297(U1,2) เป็นตัวขับมอเตอร์ โดยมี U1 และ U2 เป็นตัวรับลอจิกจากตัวไมโครคอนโทรลเลอร์บอร์ด แล้วส่งข้อมูลไปให้ U3 และ U4 เพื่อไปขับสเตปเปอร์มอเตอร์ ซึ่งมี R6,R7 เป็นตัวจำกัดกระแสที่ไหลในมอเตอร์



รูปที่ 2.6 ชุดขับสเตปเปอร์มอเตอร์

2.2 ซอร์ฟแวร์ของเครื่องต้นแบบ

ด้านซอร์ฟแวร์จะมีฟังก์ชั่นสำคัญๆ เพื่อช่วยในการพัฒนาโปรแกรมไมโครเมาส์ที่เขียนขึ้น โดยทำเป็นไลบลาลี ให้ #Include ไว้ในโปรแกรม โดยแบ่งการทำงานเป็นส่วนต่างๆ

- ฟังก์ชั่นติดต่อกับบอร์ดคอนโทรลเลอร์
- ฟังก์ชั่นเกี่ยวกับบิท
- ฟังก์ชั่นการทำงานเกี่ยวกับการเคลื่อนไหว
- ฟังก์ชั่นการทำงานเกี่ยวกับการรับ/ส่งข้อมูล
- ฟังก์ชั่นเกี่ยวกับการจำลองการเดินทาง

ที่ผ่านมาเป็นการกล่าวถึง ฮาร์ดแวร์ และซอร์ฟแวร์ของเครื่องต้นแบบที่นำมาใช้ศึกษา เพื่อที่จะนำมากำหนดขอบเขตของโครงงาน ศึกษาข้อดีข้อเสียของส่วนของฮาร์ดแวร์ของต้นแบบ

บทที่3
ทฤษฎีและหลักการ
ของอุปกรณ์ไมโครเมาส์

3.1 หลักการทำงานของ 8051

ในระบบของไมโครคอนโทรลเลอร์ 8051 จำเป็นต้องมีหน่วยความจำสำหรับบรรจุคำสั่ง หรือโปรแกรมที่ผู้ใช้พัฒนาขึ้นจัดเก็บไว้ภายในหน่วยความจำ เรียกว่า หน่วยความจำโปรแกรม (Program Memory) โดยอาจจะประกอบอยู่ภายในตัวไอซีของ 8051 เอง หรือเป็นไอซีหน่วยความจำ EPROM หรือ ROM แยกออกต่างหากได้ ในกรณีหลังนี้จำเป็นจะต้องมีการใช้พอร์ตอินพุต/เอาท์พุต ทำหน้าที่เป็นบัสแอดเดรสและบัสข้อมูลเพื่อให้สามารถทำการเชื่อมต่อเข้ากับไอซีหน่วยความจำมาตรฐานทั่วไปได้ ดังนั้นในบทนี้นอกจากจะได้มีการอธิบายเกี่ยวกับเรื่องหน่วยความจำโปรแกรมของ 8051 แล้ว ยังจะได้อธิบายรวมไปถึงพื้นฐานการใช้งานของไอซีหน่วยความจำมาตรฐานแบบต่าง ๆ ด้วย เพื่อให้มีความเข้าใจการเชื่อมต่อสัญญาณเข้ากับขาสัญญาณควบคุมและบัสของ 8051 ที่ชัดเจนขึ้น

หน่วยความจำโปรแกรมของ 8051 เป็นบริเวณหน่วยความจำ สำหรับเก็บข้อมูลและคำสั่งใช้งานต่าง ๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบ ข้อมูลเหล่านี้ก็ยังคงอยู่ไม่สูญหาย โครงสร้างของหน่วยความจำโปรแกรม มีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไอซี หน่วยความจำประเภทต่าง ๆ เช่น หน่วยความจำแบบ ROM (Read Only Memory) หรือ EPROM (Erasable Programmable Read Only Memory) เป็นต้น

จากตารางที่ 3.1 แสดงให้เห็นถึงสัญญาณต่างๆของ 8051 ซึ่งนำมาใช้ในการติดต่อกับหน่วยความจำภายนอก

สัญญาณ	คำจำกัดความ	ขาสัญญาณ	หน้าที่
EA	External Access	31	เลือกประเภทหน่วยความจำภายในหรือภายนอก
ALE	Adress Enable	30	สัญญาณเอาต์พุตสำหรับการแลตซ์ข้อมูลแอดเดรสจากบัส
P 2.0 – P 2.3	Port 2	21 – 28	เป็นข้อมูลแอดเดรสไบต์สูงของหน่วยความจำ
P 0.0 – P 0.7	Port 0	39 – 32	มัลติเพลกซ์สัญญาณของบัสแอดเดรสและบัสข้อมูล
PSEN	Program Store Enable	29	สัญญาณระบุการ Read ให้กับหน่วยความจำ EPROM

ตารางที่ 3.1 สัญญาณของ 8051 ที่ใช้ระหว่างการติดต่อเพื่ออ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

3.1.1 การเพิ่มวงจรขับสัญญาณบัส

การเชื่อมต่อกับหน่วยความจำโปรแกรมดังได้กล่าวมาแล้วนั้น เป็นการกระทำแบบพื้นฐาน โดยเชื่อมโยงสัญญาณโดยตรงระหว่าง EPROM กับ 8051 ซึ่งปกติแล้วจะทำได้จำนวนจำกัดอยู่ค่าหนึ่งตามความสามารถในการขับโหลดของพอร์ต 8051

ลองพิจารณาจากค่าลักษณะสมบัติของ 8051 ค่ากระแสเอาต์พุตเมื่อเป็นลอจิกต่ำ (I_{ol}) และค่ากระแสเอาท์พุตเมื่อเป็นลอจิกสูง(I_{oh}) จะมีค่า 1.6 มิลลิแอมแปร์และ – 80 ไมโครแอมแปร์ตามลำดับ (โดยเครื่องหมายลบแสดงว่าทิศทางของกระแสไหลออกมาจาก 8051) ดังนั้นหากมีการเชื่อมต่อเข้ากับขาสัญญาณเหล่านี้ โดยไม่เหมาะสมกับการขับสัญญาณหรือการโหลดสัญญาณแล้วอาจจะมีผลทำให้การทำงานผิดพลาดได้ ดังนั้นในการใช้งานประยุกต์บางระบบจึงอาจจำเป็นจะต้องมีการใช้วงจรขับสัญญาณเพิ่มเติมขึ้น ซึ่งโดยทั่วไปมักใช้ไอซีประเภทบัฟเฟอร์ (Buffer) เช่น 74LS244 ซึ่งภายในจะมีเกตบัฟเฟอร์ที่เป็นลอจิกแบบสามสถานะอยู่ 8 ตัว เป็นต้น

3.1.2 รีจิสเตอร์หน้าที่พิเศษ

รีจิสเตอร์หน้าที่พิเศษ (SFR) เป็นรีจิสเตอร์สำหรับการควบคุมหน้าที่และการทำงานของอุปกรณ์หรือพอร์ตของ 8051 ทั้งหมด โดยมีตำแหน่งอยู่ในบริเวณแอดเดรส 80H – FFH การใช้งานรีจิสเตอร์หน้าที่พิเศษเหล่านี้สามารถทำได้ทั้งการระบุถึงชื่อของรีจิสเตอร์หรือตำแหน่งแอดเดรสที่เป็นของรีจิสเตอร์นั้นก็ได้

ลักษณะการจัดพื้นที่หน่วยความจำ สำหรับรีจิสเตอร์หน้าที่พิเศษเหล่านี้ โดยมีข้อสังเกตว่ารีจิสเตอร์ที่อยู่ในตำแหน่งแอดเดรสที่เป็นจำนวนทวีคูณของค่า 8 จะสามารถอ้างถึงในระดับบิตได้ด้วย (นั่นคือแอดเดรส 80H, 88H, 90H, A0H, A8H, B0H, B8H, D0H, E0H และ F0H)

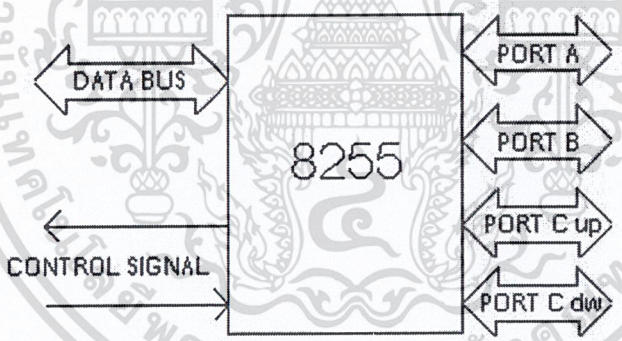
ชื่อรีจิสเตอร์	คำจำกัดความ	ความสามารถการอ้างถึงแบบบิต
ACC	Accumulator	ได้
B	B register	ได้
PSW	Program Status Word	ได้
DPTR	Stack Pointer	ได้
P0	Data Pointer (DPH & DPL)	ได้
P1	Port 0	ได้
P2	Port 1	ได้
P3	Port 2	ได้
IP	Port 3	ได้
IE	Interrupt Priority	ได้
TMOD	Interrupt Enable	ได้
TCON	Timer/counter mode	ไม่ได้

TH0	Timer/counter control	ได้
TH1	Timer/ counter 0 (high)	ไม่ได้
TL0	Timer/ counter 0 (low)	ไม่ได้
TH1	Timer/ counter 1 (high)	ไม่ได้
TL1	Timer/ counter 1 (low)	ไม่ได้
SCON	Seria ! Control	ไม่ได้
SBUF	Serial data buffer	ไม่ได้
PCON	Power control	ไม่ได้

ตารางที่ 3.2 แสดงชื่อรีจิสเตอร์และความสามารถในการอ้างถึงระดับบิต

3.2 ไอซีขยายพอร์ต8255

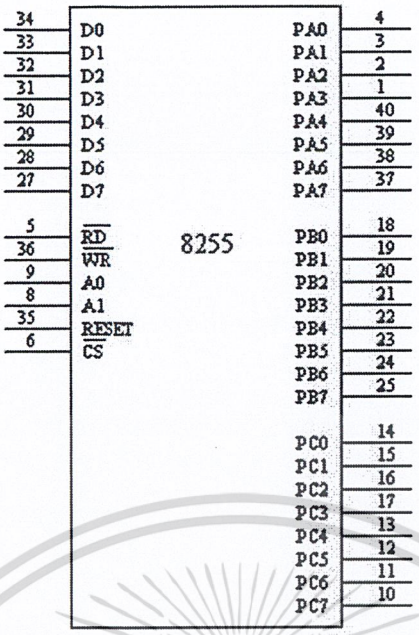
ไอซีที่มี 40 ขา ได้รับการออกแบบมาเพื่อให้สามารถทำงานรวมกับไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์ ซึ่ง 8255 นั้นมีพอร์ตใช้งาน 3 ชุด โดยมีโครงสร้างพื้นฐานแสดงไว้ดังรูปต่อไป



รูปที่ 3.1 ผังโครงสร้างของไอซี 8255

ขาต่าง ๆ ของ 8255

เพื่อให้เข้าใจวิธีการใช้งาน จึงจำเป็นต้องเข้าใจความหมายและตำแหน่งของขาต่าง ๆ เสียก่อน ขาทั้ง 40 ขามีดังต่อไปนี้



รูปที่ 3.2 แผนผังวงจรภายในและการจัดขาของไอซี 8255

D0 – D7 เป็นขาที่ข้อมูลอินพุทเอาท์พุทจะต้องผ่านเข้าออกจากส่วนนี้ D0 – D7 จึงต่อเข้าระบบของไมโครคอนโทรลเลอร์ เพื่อให้ไมโครคอนโทรลเลอร์สามารถอ่านหรือเขียนข้อมูลจากพอร์ตผ่านทางบัสนี้

CS (สัญญาณเลือกชิฟ) ขานี้เป็นขาอินพุทที่รับสัญญาณมาจากภายนอกเพื่อเลือกชิฟ 8255 โดยเมื่อขานี้เป็น “0” จะทำให้ 8255 ต่อเข้ากับระบบบัสของไมโครโปรเซสเซอร์ เพื่อให้ไมโครโปรเซสเซอร์เขียนหรืออ่านข้อมูลจากพอร์ตได้

RD (สัญญาณการอ่าน) เป็นสัญญาณอินพุทที่จะรับเข้ามาจากซีพียูเมื่อสัญญาณที่ขานี้เป็น “0” และสัญญาณขานี้ CS เป็น “0” ด้วยไอซี 8255 จะทำตัวให้ซีพียูอ่านข้อมูลจากบัสในขณะที่เป็นพอร์ตอินพุท

WR เป็นสัญญาณการเขียนจะแอดตีฟเมื่อมีสัญญาณ WR และสัญญาณ CS เป็น “0” สัญญาณนี้มาจากไมโครคอนโทรลเลอร์ เมื่อต้องการเขียนข้อมูลลงบนพอร์ตที่กำหนด

RESET (สัญญาณรีเซต) เป็นสัญญาณที่ส่งจากภายนอกเข้ามาทำการรีเซต 8255 เพื่อเคลียร์สถานะต่าง ๆ ของ 8255 เมื่อ 8255 ได้รับการรีเซต ก็จะกลับเข้าสู่โหมดอินพุทหรือทุกพอร์ตที่เป็นพอร์ตอินพุท

PA0- PA7 เป็นสายสัญญาณที่เป็นพอร์ต 8255 ที่ชื่อพอร์ต A การเลือกพอร์ตจะเลือกโดยสัญญาณแอดเดรส A0 – A7

PB0 –PB7 เป็นสายสัญญาณที่เป็นพอร์ต B ของ 8255 ถูกเลือกโดยสัญญาณแอดเดรส A0 – A1

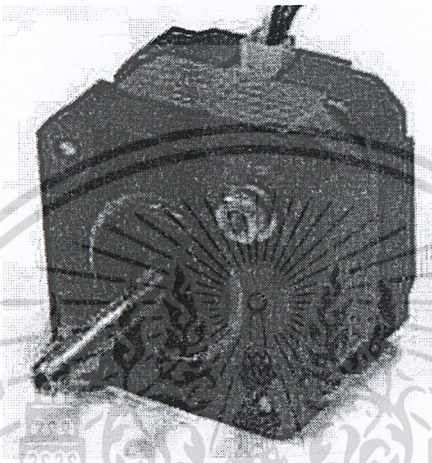
PC0 – PC7 เป็นสายสัญญาณที่เป็นพอร์ต C ของ 8255 การกำหนดพอร์ตจะได้รับการกำหนดโดยการส่งสัญญาณแอดเดรส A0 – A1 พอร์ต C นี้แบ่งเป็น 2 กลุ่ม PC0- PC3 และกลุ่ม PC4 – PC7

การใช้งาน 8255 จะต้องส่งรหัสควบคุม (control code) เข้าไปยังพอร์ตหมายเลข 13H การควบคุมการทำงานของ 8255 โดยใช้สัญญาณควบคุมพอร์ตหมายเลข 13H การควบคุมการทำงานของ 8255 มีหลายโหมด

แต่ละโหมดจะแตกต่างกันออกไป การโปรแกรมให้ 8255 ทำงานจะทำ 3 โหมด คือ โหมด 0 โหมด 1 และโหมด 2

3.3 สเตปปิ้งมอเตอร์

สเตปปิ้งมอเตอร์ทำหน้าที่เปลี่ยนสัญญาณข้อมูลแบบดิจิตอล ไปสู่การเคลื่อนที่ทางกล อย่างได้สัดส่วนกัน โดยที่แกนหรือโรเตอร์ของมอเตอร์ชนิดนี้มักถูกควบคุมให้หมุนเป็นลำดับขั้น



รูป 3.3 ตัวอย่างสเต็ปปิ้งมอเตอร์

ประโยชน์จากการใช้งานสเตปปิ้งมอเตอร์ที่มีการควบคุมโดยใช้สัญญาณดิจิตอลคือ มีความถูกต้องเที่ยงตรงและสามารถเปลี่ยนตำแหน่งของโหลด ได้อย่างรวดเร็ว เนื่องจากแต่ละอินพุทพัลซ์ จะทำให้สเตปปิ้งมอเตอร์เคลื่อนที่ไปหนึ่งสเตป อย่างเที่ยงตรง

เมื่อพิจารณาในแง่ทางกลจะพบว่าสเตปปิ้งมอเตอร์มีความเที่ยงตรงที่ยอมรับได้ ซึ่งในการเปลี่ยนตำแหน่งอย่างง่ายอาจใช้สวิตซ์ ในการควบคุมมอเตอร์ นั่นคือ สร้างวงจรควบคุมมอเตอร์ได้ง่าย ถ้าตองการเพิ่มประสิทธิภาพของส่วนควบคุมให้ดีขึ้น อาจใช้ไอซีที่มีความเร็วสูงเนื่องจากประกอบด้วยมอสเฟสอยู่ภายใน จึงได้กำลังงานสูง ความเร็วสูง และต้นทุนต่ำ ความสะดวกในการใช้งานนี้เองจึงทำให้นำไปสู่การใช้งานอย่างแพร่หลาย

การได้รับประโยชน์จากการใช้สเตปปิ้งมอเตอร์อย่างเต็มที่ นั้นขึ้นอยู่กับการขับอย่างถูกต้องและเหมาะสม ซึ่งภาคขับนี้จะต้องประกอบไปด้วย แหล่งจ่ายไฟฟ้ากระแสตรง อิเล็กทรอนิกส์สวิตซ์ โดยอาจจะเป็นทรานซิสเตอร์หรือมอสเฟส และแหล่งจ่ายสัญญาณข้อมูลแบบดิจิตอล ซึ่งมีลักษณะเป็นพัลซ์ เพื่อใช้ควบคุมทิศทางการหมุนของมอเตอร์

ทิศทางของกระแสไฟฟ้าที่ป้อนเข้าสู่สเตปปิ้งมอเตอร์ ถูกควบคุมโดยการทำงานของอิเล็กทรอนิกส์สวิตซ์ ดังนั้นสเตปปิ้งมอเตอร์จะหมุนไปหนึ่งช่วงในแต่ละอินพุทพัลซ์ที่ป้อนเข้าสู่วงจรอิเล็กทรอนิกส์สวิตซ์ ซึ่งขนาดมุมที่หมุนไปจะขึ้นอยู่กับการออกแบบสเตปปิ้งมอเตอร์ โดยมีค่าตั้งแต่ 1.8 องศาถึง 15 องศา ถ้าส่งสัญญาณไปยังวงจรสวิตซ์ชิ่ง 24 พัลซ์ โดยมีค่าของหนึ่งช่วงการหมุนเท่ากับ 15 องศา สเตปปิ้งมอเตอร์ก็จะ

หมุนไปครบ 1 รอบพอดี เวลาที่ใช้ในการหมุนครบ 1 รอบ ขึ้นอยู่กับอัตราการจ่ายสัญญาณพัลซ์ควบคุมโดยสัญญาณนี้อาจถูกผลิตโดยวงจรกำเนิดสัญญาณ ที่ปรับความถี่ได้หรือจากไอซีควบคุม

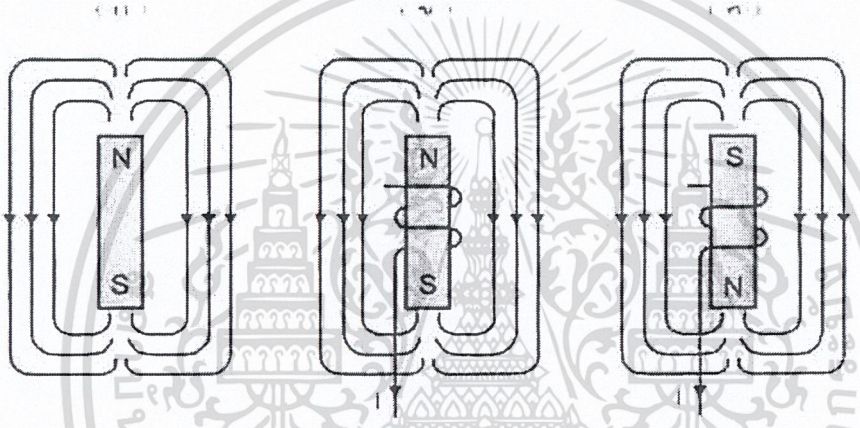
3.3.1 หลักการพื้นฐานของสเตปปิ้งมอเตอร์

ในรูปที่ 1 หลักการพื้นฐานของเส้นแรงแม่เหล็ก

ในรูปที่ 2 (ก) สนามแม่เหล็กที่เกิดจากแม่เหล็กถาวรมีทิศทางพุ่งจากขั้วเหนือไปยังขั้วใต้

ในรูปที่ 2 (ข) สนามแม่เหล็กของแม่เหล็กไฟฟ้าที่เกิดจากกระแส (I)

ในรูปที่ 2 (ค) ขั้วแม่เหล็กกลับทิศทาง เมื่อขดลวดถูกพันกลับทิศทาง และทิศทางการไหลของกระแสไม่เปลี่ยนแปลง



รูปที่ 3.4 สนามแม่เหล็กที่เกิดขึ้นในลักษณะต่าง ๆ

สเตปปิ้งมอเตอร์ แบ่งออกได้ 3 ชนิด คือ

- สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร
- สเตปปิ้งมอเตอร์แบบค่ารีดับแตนซ์แปรค่าได้
- สเตปปิ้งมอเตอร์แบบไฮบริด

ในที่นี้จะกล่าวเฉพาะสเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร

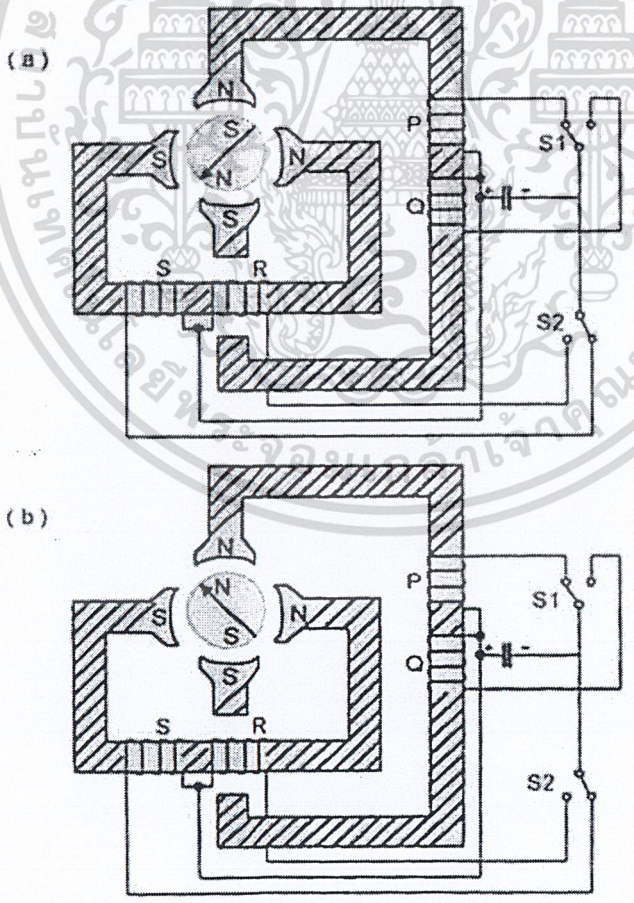
3.3.2 สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร

มุมการหมุน (Step Angle) ของมอเตอร์แบบแม่เหล็กถาวรขึ้นอยู่กับความสัมพันธ์ ระหว่างจำนวนขั้วแม่เหล็กบนส่วนที่อยู่กับที่หรือเรียกว่าสเตเตอร์ และจำนวนขั้วแม่เหล็กบนส่วนเคลื่อนที่หรือที่เรียกว่า โรเตอร์ เนื่องจากโรเตอร์ซึ่งเป็นแม่เหล็กถาวรรูปทรงกระบอกจำนวนขั้วแม่เหล็กจึงถูกจำกัดสูงสุดที่ค่าหนึ่ง การเพิ่มขนาดเส้นผ่านศูนย์กลางของโรเตอร์จะทำให้ได้จำนวนขั้วแม่เหล็กบนโรเตอร์เพิ่มมากขึ้น แต่มีข้อเสียคือเกิดแรงเฉื่อยเพิ่มมากตามไปด้วย ซึ่งจะลดประสิทธิภาพขณะเริ่มต้นการหมุนของมอเตอร์ สเตปปิ้งมอเตอร์แบบนี้มีขนาดความกว้างของมุมหนึ่งช่วงการหมุนมาก แต่สามารถลดลงได้โดยการทำให้มีหลายสแต็ค หรือ

มากกว่าหนึ่งสแต็คขึ้นไป ตามแนวความยาวของมอเตอร์ (สแต็คในที่นี้หมายถึงเฟส ซึ่งประกอบด้วยโรเตอร์ที่เป็นซี่ฟัน และโครงร่างของสเตเตอร์อยู่รอบนอก)

โครงร่างสเตเตอร์ อาจจะประกอบด้วยสองสเตเตอร์ หรือมากกว่าโดยในแต่ละขั้วสเตเตอร์จะมีขดลวดพันอยู่ กระแสที่ไหลผ่านขดลวดจะทำให้เกิดสนามแม่เหล็ก โดยมีขั้วเป็นขั้วเหนือหรือขั้วใต้ขึ้นอยู่กับทิศทางการไหลของกระแสไฟฟ้า กระแสไฟฟ้าที่ไหลในขดสเตเตอร์อย่างต่อเนื่องนี้จะสร้างสนามแม่เหล็กหมุน (Rotating Magnaticfield) ซึ่งมีผลทำให้โรเตอรแม่เหล็กถาวรถูกดึงดูดให้หมุนตาม โดยความเร็วการหมุนขึ้นอยู่กับอัตราการเปลี่ยนทิศทางการไหลของกระแสไฟฟ้าในขดสเตเตอร์ และจำนวนขั้วแม่เหล็กไฟฟ้า

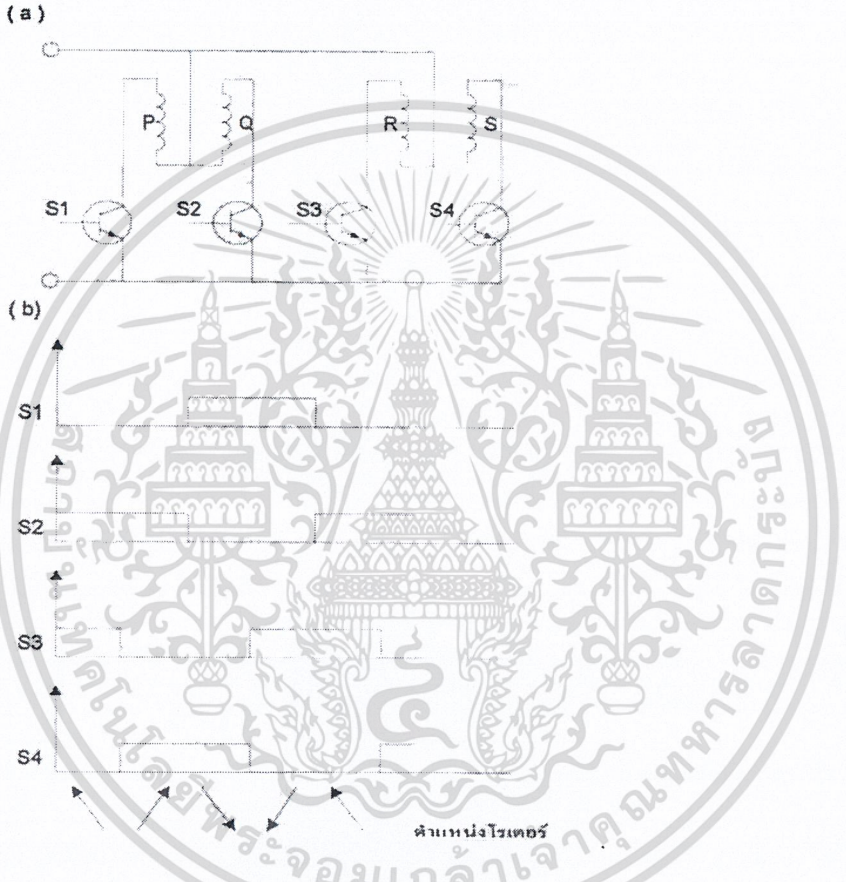
ทิศทางการหมุนของมอเตอร์ สามารถควบคุมได้ โดยการจัดลำดับการขับเฟสของมอเตอร์ การควบคุมทิศทางของกระแสในขดลวดสเตเตอร์อาจทำได้สองวิธี คือ การขับมอเตอร์แบบขั้วเดียว และการขับมอเตอร์แบบสองขั้ว พิจารณาสเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร ซึ่งมี 2 ขั้วที่โรเตอร์ (โดยอันที่จริงอาจมีถึง 24 ขั้ว)



รูปที่ 3.5 มอเตอร์ 4 เฟสแบบขั้วเดียว

3.3.3 การขับมอเตอร์แบบขั้วเดียว

สำหรับการขับมอเตอร์แบบขั้วเดียวนั้น ขดลวดสเตเตอร์ถูกออกแบบให้มีจุดแบ่งกึ่งกลาง บนขดลวดจุดนี้จะต่อไปยังขั้วใดขั้วหนึ่งของแหล่งจ่ายไฟ ทิศทางกระแสไฟฟ้าที่ไหลผ่านขดลวดจะถูกกำหนดโดยการต่อปลายขดลวดเข้ากับขั้วใดขั้วหนึ่งที่เหลืออยู่ของแหล่งจ่ายไฟ โดยใช้อุปกรณ์ประเภทสวิตช์ชิ่งในการตัดต่อวงจร เช่น ทรานซิสเตอร์ดังรูปที่ 3.6 และการสลับกันทำงานระหว่างขดลวดมีผลทำให้ขั้วแม่เหล็กบนสเตเตอร์เปลี่ยนแปลงดังรูปที่ 3.6



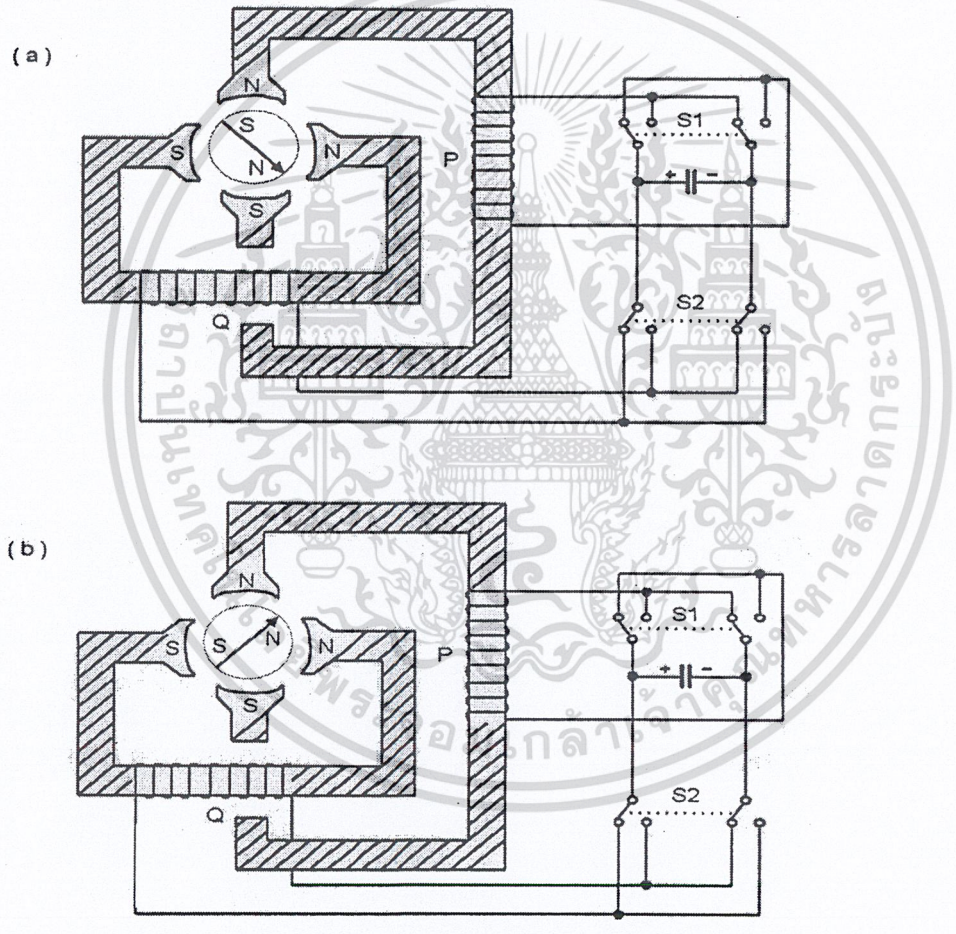
รูปที่ 3.6 การขับแบบขั้วเดียว

รูปที่ 3.5 (A) สเตปปิ้งมอเตอร์ 4 เฟส โดยกระแสไฟฟ้าเข้าเฟส P และ S ดังนั้นจะเกิดขั้วแม่เหล็กบนสเตเตอร์ซึ่งจะดึงดูดให้มอเตอร์หมุนมาอยู่ ณ ตำแหน่งการวางตัวดังรูปที่ 3.5 (A) ต่อมาเมื่อสวิตช์ S_1 เปลี่ยนตำแหน่ง ทำให้กระแสไฟฟ้าไฟลในเฟส Q และ S สนามแม่เหล็กบนสเตเตอร์ จะกลับทิศทาง ทำให้ขั้วแม่เหล็กบนสเตเตอร์กลับกันด้วย โรเตอร์จึงหมุนมาอีกตำแหน่งหนึ่งดังรูปที่ 3.5 (b) แนวลูกศรที่แสดงบนโรเตอร์จะมีทิศทางเดียวกับเส้นแรงแม่เหล็กรวมที่พุ่งออกมาจากสเตเตอร์ จากหลักการทำงานที่กล่าวมา ทำให้สามารถกลับทางหมุนของมอเตอร์ได้โดยการเปลี่ยนแปลงลำดับการทำงานของสวิตช์ S_1 และ S_2

วงจรสวิตซ์ชิ่ง โดยให้สัญญาณพัลซ์ป้อนไปยังขา B ของทรานซิสเตอร์แต่ละตัว จะเห็นว่าที่เวลาใดเวลาหนึ่งจะมีขดลวด 2 เฟส ที่ได้รับกระแสไฟฟ้าในเวลาต่อเนื่องกัน นั่นคือจะเกิดสนามแม่เหล็กหมุนที่สเตเตอร์เหนี่ยวนำให้มอเตอร์หมุนเคลื่อนที่เป็นลำดับได้

3.3.4 การขับมอเตอร์แบบสองขั้ว

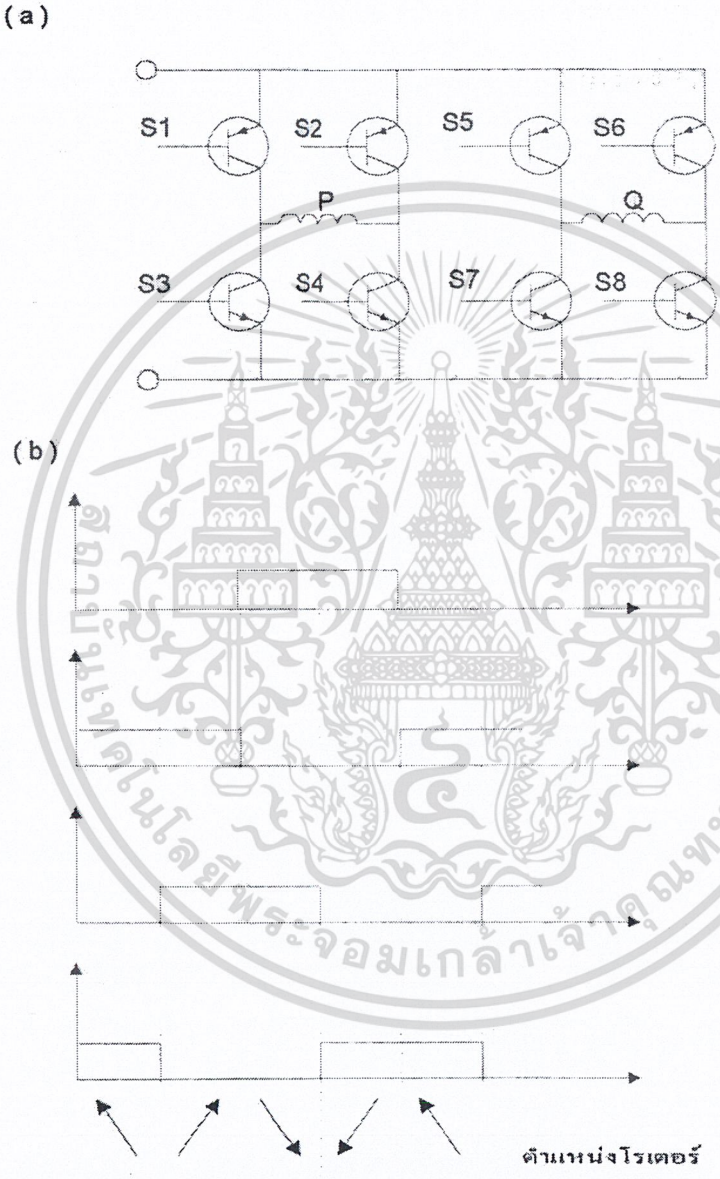
ขดลวดสเตอเตอร์ของมอเตอร์ที่ออกแบบมาสำหรับการขับแบบสองขั้วนั้นจะไม่มีจุดแบ่งกึ่งกลาง หลักการทำงานของมอเตอร์ที่ขับแบบสองขั้วมีลักษณะเช่นเดียวกับการขับมอเตอร์แบบขั้วเดียวดังรูปที่ 3.7 นั่นคือเมื่อกระแสในเฟส P เปลี่ยนทิศทางโดยการสับสวิตซ์ S_1 สนามแม่เหล็กในสเตเตอร์ก็จะเปลี่ยนทิศทางและโรเตอร์จะเคลื่อนที่ไปหนึ่งลำดับ



รูปที่ 3.7 มอเตอร์ 4 เฟสแบบ 2 ขั้ว

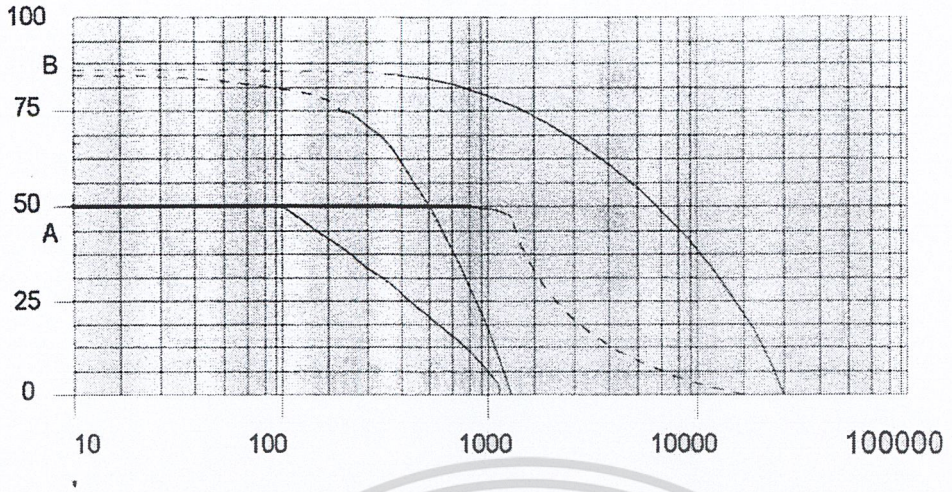
รูปที่ 3.7 (a) วงจรขับสเตปปิ้งมอเตอร์ 4 เฟส แบบสองขั้วทรานซิสเตอร์ในวงจรจะทำงานพร้อมกันเป็นคู่ เช่น S_1 กับ S_4 ทำงาน กระแสจะไหลจากขั้วบวกของแหล่งจ่ายไฟไปเข้าทางปลายขดลวด P หรือ S_6 กับ S_7 ทำงาน กระแสจะกลับทิศทางการไหลคือ ไหลจากขั้วลบของแหล่งจ่ายไฟมาเข้าขดลวดและออกทางปลาย P เป็นต้น

รูปที่ 3.7(b) ลักษณะสัญญาณควบคุมที่ป้อนให้กับวงจรขับสเตปปิ้งมอเตอร์ช่วงเวลาการทำงานของทรานซิสเตอร์ (Switching Time) ในวงจรขับแบบสองขั้ว จะต้องมีความเที่ยงตรงกว่าการขับแบบขั้วเดียว มิฉะนั้นอาจจะทำให้เกิดการลัดวงจรแหล่งจ่ายไฟ เพราะทรานซิสเตอร์ตัวบนและล่าง (เช่น S_1 กับ S_2) นำกระแสพร้อมกัน



รูปที่ 3.8 การขับแบบสองขั้ว (a) วงจรขับสเตปปิ้งมอเตอร์ (b) สัญญาณควบคุม

สัญญาณควบคุมมอเตอร์ที่ขับแบบขั้วเดียวอาจใช้ 2 ขดพันบนบอบบิ้น (Bobbin) ในแต่ละสเตเตอร์ซึ่งเรียกว่าการพันแบบสองแถวสลับกัน (Bifilar Winding) ดังนั้นจึงมี 2 ขดที่ถูกยึดบนบอบบิ้นอันเดียวกัน หลักการทำงานจึงเป็นแบบเดียวกันกับการพันขดลวดเดียวในมอเตอร์แบบสองขั้ว แต่มีขนาดลวดเล็กกว่าและความต้านทานสูงกว่า

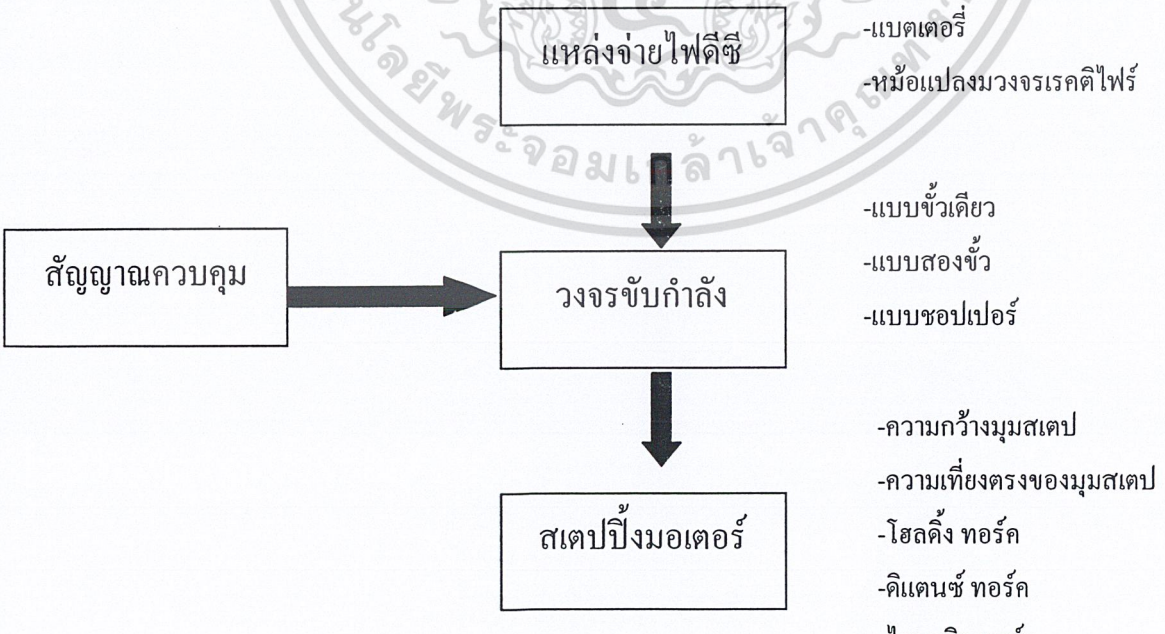


รูปที่ 3.9 โค้งความสัมพันธ์ระหว่างแรงบิดและอัตราการหมุน

ประโยชน์จากการขับมอเตอร์แบบสองขั้วแสดงดังรูปที่ 3.9 กล่าวคือ การขับแบบสองขั้วจะสร้างแรงบิดหรือทอร์ค (Torque) ได้มากกว่าการขับแบบขั้วเดียว ที่อัตราช่วงการหมุนต่ำ ๆ แต่จะมีค่าของแรงบิดใกล้เคียงกัน ที่อัตราช่วงการหมุนสูง ๆ

3.3.5 ระบบสเตปปิ้งมอเตอร์

การเลือกใช้สเตปปิ้งมอเตอร์ เพื่อนำไปใช้ประโยชน์ในงานบางอย่างนั้น จะต้องมีความเข้าใจคุณลักษณะของมอเตอร์และวงจรที่ใช้ขับมอเตอร์ในรูปที่ 3.10 แผนผังของระบบสเตปปิ้งมอเตอร์



รูปที่ 3.10 แผนผังระบบสเตปปิ้งมอเตอร์

ความถูกต้องเที่ยงตรงของมุมสเตปขณะที่ไม่มีโหลด จะถูกระบุสำหรับมอเตอร์แต่ละชนิด เช่น มอเตอร์ที่มีสเตป 7.5 องศา ความผิดพลาด $\pm$ 10 ลิปดา ขณะเคลื่อนที่ไปหนึ่งสเตป เป็นต้น

มุมสเตป	จำนวนสเตปต่อรอบ
0.9 องศา	400
1.8 องศา	200
3.6 องศา	100
3.75 องศา	96
7.5 องศา	48
15.0 องศา	24

ตารางที่ 3.3 แสดงมุมสเตปและจำนวนสเตปต่อรอบ

มอเตอร์ที่มีจำนวนสเตปต่อรอบเท่ากับ 4 จะมีค่าผิดพลาดเป็นศูนย์ เมื่อหมุนครบ 1 รอบ เพราะขณะที่หมุนมา ณ ตำแหน่งเดิมขณะเริ่มต้นขั้วแม่เหล็กและทิศทางของเส้นแรงแม่เหล็ก (Flux) วงเดิมด้วยเหตุนี้การเปลี่ยนตำแหน่งของสเตปปิ้งมอเตอร์ที่ต้องการความถูกต้องสูง ๆ จะต้องแบ่งจำนวนสเตปต่อรอบเป็นจำนวนเท่าของ 4 สเตป เพื่อลดการสะสมของค่าผิดพลาด (Step angle error) ซึ่งเป็นรูปแบบการทำงานแบบสี่สเตป ตัวอย่างของมุมสเตป แสดงดังตารางที่ 3.5

แรงบิด (Torque)

การทำงานของสเตปปิ้งมอเตอร์มีแรงบิดเกี่ยวข้องอยู่ 3 ชนิด คือ

โฮลดิ้งทอร์ค (Holding Torque) คือแรงบิดที่ทำให้สเตปปิ้งมอเตอร์เริ่มหมุนไป สองสเตป ขณะหยุดนิ่ง ถ้าแรงบิดที่ทำให้สเตปปิ้งมอเตอร์มีขนาดมากกว่าโฮลดิ้งทอร์ค จะทำให้มอเตอร์สูญเสียการหมุนแบบสเตป กลายเป็นการหมุนแบบต่อเนื่องโดยปกติแรงบิดขณะทำงานของมอเตอร์จะน้อยกว่าระดับโฮลดิ้ง

ดิเตนซ์ทอร์ค (Detent Torque) ซึ่งเป็นสเตปปิ้งมอเตอร์แบบไฮบริดและแบบแม่เหล็กถาวร จะมีส่วนประกอบของโรเตอร์เป็นแม่เหล็กถาวรซึ่งจะสร้างแรงบิดมาเบรคการหมุนของมอเตอร์อย่างสม่ำเสมอในขณะที่ไม่มีการป้อนกระแสเข้าขดสเตเตอร์ แรงบิดดังกล่าวนี้เรียกว่า ดิเตนซ์ทอร์ค

ไดนามิกทอร์ค (Dinamic or working torque) คือแรงบิดขณะทำงานซึ่งอาจเกิดการเปลี่ยนแปลงได้เนื่องมาจากการปรับเปลี่ยนอัตราเร็วของมอเตอร์ โดยปกติการเปลี่ยนอัตราเร็วของมอเตอร์จะอยู่ในย่าน

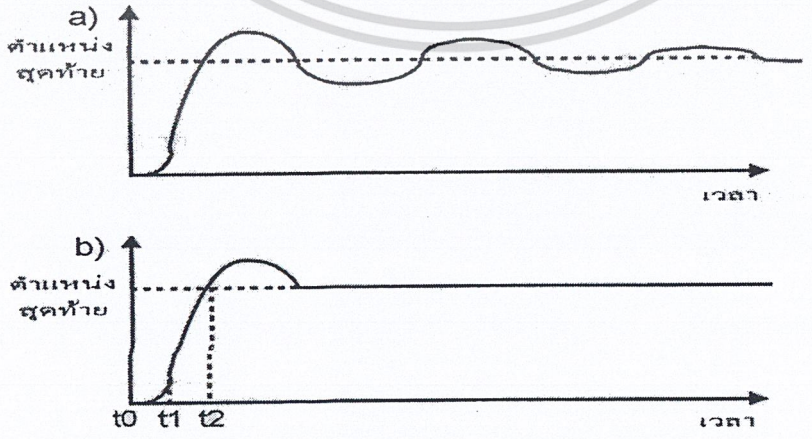
ระหว่าง เส้นโค้งพูลอิน (Pull-in curve) และ เส้นโค้งพูลเอาท์ (Pull-out corve) เพราะถ้าปรับอัตราเร็ว ณ จุดนอกโค้งพูลเอาท์มอเตอร์จะสูญเสียการหมุนแบบเป็นสเตปได้หรือเกิดการหมุนแบบต่อเนื่องนั่นเอง

การแกว่งเข้าสู่สภาวะคงตัว (Over Shoot) ขณะที่มอเตอร์ไปในแต่ละสเตป และหยุด ณ สเตปใด ๆ จะเกิดการแกว่งหรือสั่นของโรเตอร์ เข้าสู่ตำแหน่งสุดท้ายนั้น ๆ และใช้เวลาช่วงหนึ่งในการเข้าสู่สภาวะคงตัว แสดงเปรียบเทียบได้ดังรูปที่ 3.11 (a) ซึ่งเป็นพฤติกรรมปกติ ของระบบที่ใช้สัญญาณพัลซ์โดยทั้งนี้ขึ้นอยู่กับโหลดและกำลังงานที่ได้รับจากภาคขับมอเตอร์ ผลตอบสนองที่เกิดขึ้นนี้สามารถเปลี่ยนแปลงคือลดเวลาในการเข้าสู่สภาวะคงตัวได้โดยการเพิ่มโหลดที่เป็นแรงเสียดทานเข้าไปในระบบ ซึ่งเป็นลักษณะการแก้ไขทางกล เช่นการใช้เครื่องต่อกำลังไปเพลาโดยใช้ความฝืดไม่ใช่เฟือง

วิธีการแก้ไขทางไฟฟ้า ทำได้โดยหน่วงสัญญาณพัลซ์ลูกสุดท้ายในขบวนพัลซ์ทั้งหมด โดยอาจถูกเปลี่ยนเป็น 3 ส่วนด้วยกัน ดังรูปที่ 3.11(b) ซึ่งเป็นพฤติกรรมปกติ ของระบบที่ใช้สัญญาณพัลซ์ โดยทั้งนี้ขึ้นอยู่กับโหลดและกำลังงานที่ได้รับจากภาคขับมอเตอร์ ผลตอบสนองที่เกิดขึ้นนี้สามารถเปลี่ยนแปลงคือลดเวลาในการเข้าสู่สภาวะคงตัวได้โดยการเพิ่มโหลดที่เป็นแรงเสียดทานเข้าไปในระบบซึ่งเป็นลักษณะการแก้ไขทางกล เช่นการใช้เครื่องต่อกำลังไปเพลาโดยใช้ความฝืดไม่ใช่เฟือง

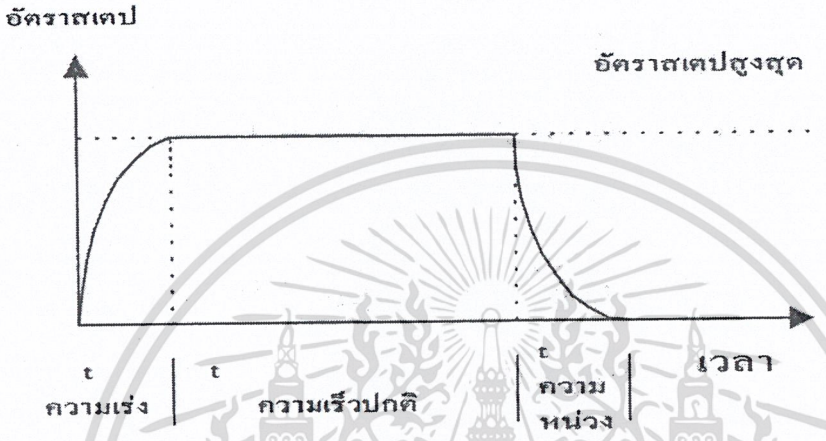
วิธีการแก้ไขทางไฟฟ้า ทำได้โดยหน่วงสัญญาณพัลซ์ลูกสุดท้ายในขบวนพัลซ์ทั้งหมด โดยอาจถูกเปลี่ยนเป็น 3 ส่วนด้วยกัน ดังรูปที่ 3.11 (b) โดยส่วนแรกที่เวลา t จะเป็นการฟอร์เวิร์ดพัลซ์เวลา t จะป้อนเป็นรีเวิร์สพัลซ์เพื่อชะลอการหมุนของโรเตอร์ ส่วนสุดท้ายที่เวลา t จะเป็นฟอร์เวิร์สพัลซ์ ป้อนเพื่อให้โรเตอร์หยุด ณ ตำแหน่งที่ต้องการ ซึ่งวิธีการนี้จะสร้างแรงบิดน้อยกว่าปกติ ดังนั้นจึงลดการสั่นหรือแกว่งของเพลาได้

การทำสเตปที่แตกต่างกัน มีหลายวิธีด้วยกันในการกำหนดจำนวนสเตปที่ใช้ในการเคลื่อนที่จากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่ง เช่น หมุนไป 90 องศา อาจใช้ทั้งหมด 6 สเตป ๆ ละ 15 องศา 12 สเตป สเตปละ 7.5 องศา หรือ 50 สเตป ๆ ละ 1.8 องศา โดยทั่วไปแล้วมอเตอร์ที่มีความกว้างของมุมสเตปน้อยจะลดการแกว่งหรือออสซิเลต และจะมีความแม่นยำดีกว่ามอเตอร์ที่มีความกว้างของมุมสเตปมาก



รูปที่ 3.11 ผลตอบสนองในการเข้าสู่สภาวะคงตัว (a) ผลตอบสนองเมื่อการหน่วง (b) ผลตอบสนองเมื่อมีการหน่วงด้วยไฟฟ้า

อัตราเร่งของมอเตอร์จะถูกควบคุมโดยวงจร VCO (Voltage controlled oscillator) และช่วงเวลาในการเก็บประจุ (RC time Constant) ของคาปาซิเตอร์จะกำหนดอัตราเร่งที่แตกต่างกัน ในรูปที่ 3.12 แสดงอัตราสเตปต่อเวลาซึ่งแบ่งออกเป็น 3 ช่วงคือ ขณะสตาร์ทช่วงแรกมีความเร่งเพิ่มอัตราสเตปอย่างต่อเนื่อง ช่วงที่สองเป็นอัตราสเตปขณะใช้งานสูงสุด และช่วงสุดท้ายเกิดความหน่วง (Deceleration) ลดความเร็วของมอเตอร์ลงมาจนกระทั่งหยุดนิ่ง



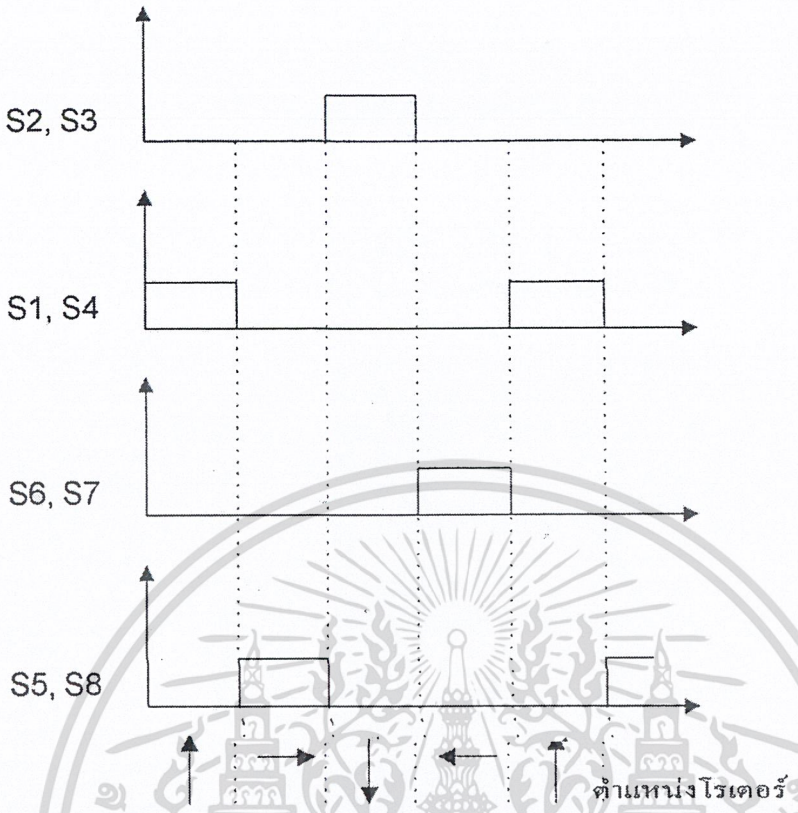
รูปที่ 3.12 โค้งช่วงเวลาการเพิ่มความเร็วและลดความเร็วของมอเตอร์

รีโซแนนซ์ (Resonance) สเตปปิ้งมอเตอร์ขณะทำงานไม่มีโหลด โดยการเปลี่ยนความถี่ใช้งานไปในช่วงหนึ่ง ๆ จะมีความถี่ค่าหนึ่งที่ทำให้เกิดการรีโซแนนซ์ ซึ่งจะเกิดเสียงรบกวน หรือสมารถตรวจสอบได้ด้วยการตรวจจับการสั่นสะเทือน ดังนั้นควรหลีกเลี่ยงการใช้งาน ณ ความถี่เรโซแนนซ์

3.3.6 วิธีขับเฟสสเตปปิ้งมอเตอร์

วิธีที่ใช้ในการขับเฟสสเตปปิ้งมอเตอร์ขึ้นอยู่กับลักษณะการพันของขดลวดและรูปแบบของวงจรขับสเตปปิ้งมอเตอร์

วิธีขับแบบเฟสเดียว (Wave Drive) คือ วิธีการขับที่เฟสใดเฟสหนึ่งของมอเตอร์ได้รับกระแสไฟฟ้าที่เวลาใด ๆ รูปที่ 3.13 ลำดับการส่งสัญญาณควบคุมของสเตปปิ้งมอเตอร์ 4 เฟส ซึ่งมีเฟสเดียวที่ถูกขับ ดังนั้นค่าโฮลดิ้งและไดนามิกทอร์คจะลดลง 30% แต่สามารถชดเชยค่าแรงบิดให้เพิ่มขึ้นได้โดยเพิ่มแรงดันของแหล่งจ่ายไฟ ซึ่งจะทำให้ประสิทธิภาพ (Effeciency) สูง แต่มีความแม่นยำของสเตปน้อย



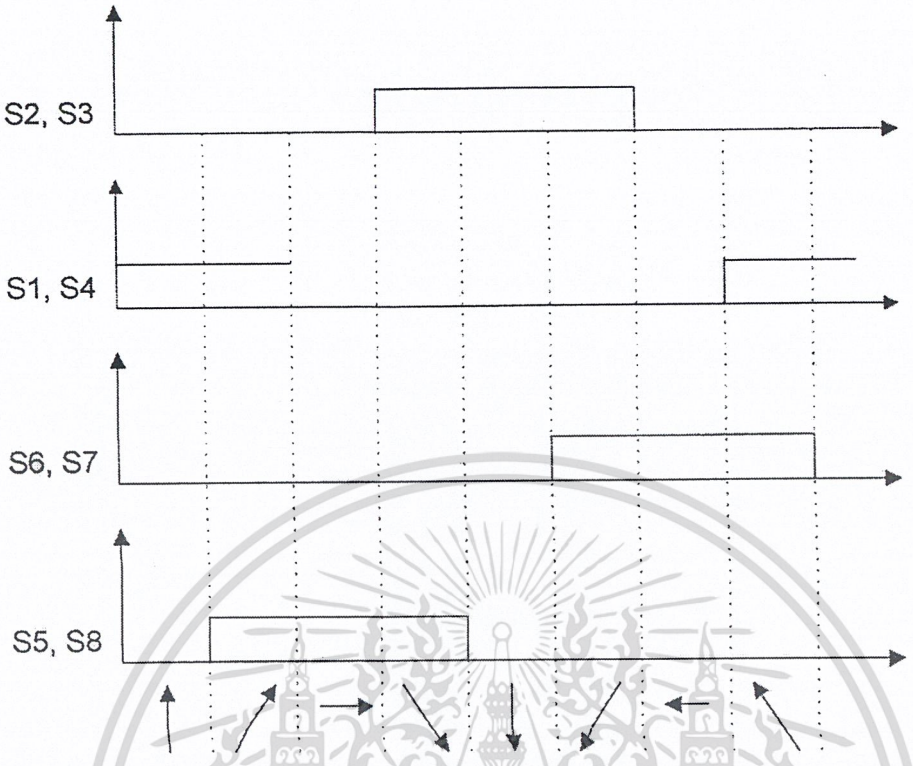
รูปที่ 3.13 สัญญาณควบคุมสเตปปิ้งมอเตอร์ 4 เฟสที่มีวงจรขับสองขั้วแบบเฟสเดียว

วิธีขับแบบหนึ่งและสองเฟส (Half-step mode) คือวิธีการขับแบบครึ่งสเตป เช่น ความกว้าง 1 สเตป เท่ากับ 7.5 องศา แต่การขับวิธีนี้จะหมุนไปครั้งละครึ่งสเตป เท่ากับ 3.75 องศา ผลที่ได้รับขณะใช้งานคือ โฮลดิ้งทอร์คจะมากและน้อยสลับกันไป เพราะถูกขับ 2 เฟส และ 1 เฟส สลับกัน ดังนั้นค่าแรงบิดเฉลี่ยจะสูงกว่าวิธีขับเฟสเดียว แต่จะมีความแม่นยำน้อยลงขณะที่หมุนครบสเตปลำดับการส่งสัญญาณควบคุม

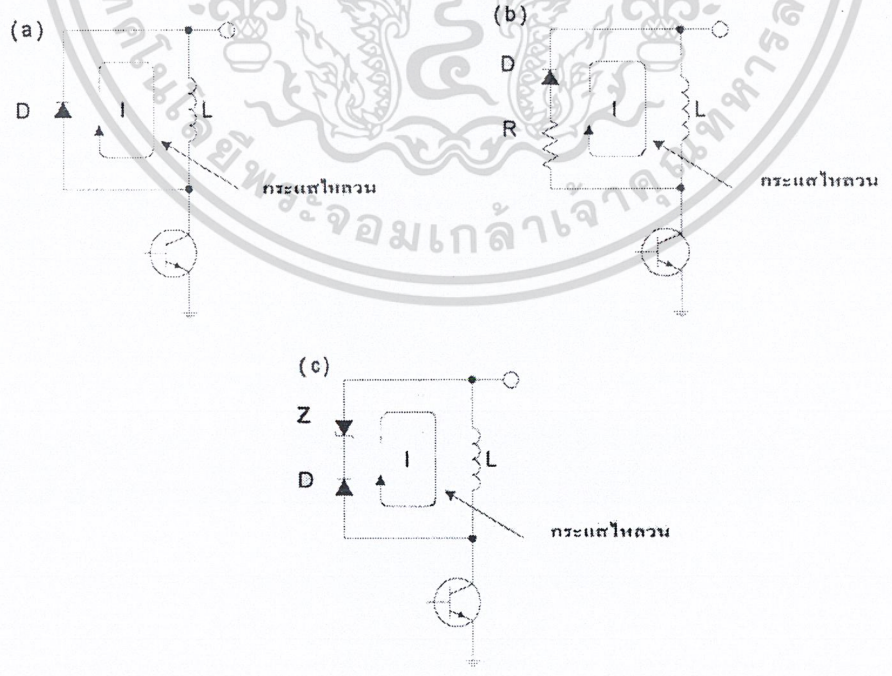
3.3.7 การกำจัดสไปค์ (Spike Suppression)

ขณะที่ทรานซิสเตอร์ในวงจรขับสเตปปิ้งมอเตอร์หยุดนำกระแสจะทำให้เกิดค่าแรงดันสูง ซึ่งเกิดการยุบตัวของสนามแม่เหล็กในขดลวด ตามสมการ V = L di/dt แรงดันที่เกิดขึ้นนี้ อาจทำให้รอยต่อของทรานซิสเตอร์เสียหายได้ ถ้าไม่กำจัดออกไป ปัญหาเหล่านี้แก้โดยใช้วงจร ฟรีวิลลิ่ง (Freewheeling Circuit) เป็นทางผ่านของกระแส จากขดลวดขณะที่ทรานซิสเตอร์หยุดทำงาน ซึ่งมีรูปแบบดังนี้คือ

การกำจัดโดยใช้ไดโอด (Diode Suppression) คือการนำไดโอดมาต่อขนานกันกับขดลวด ดังรูปที่ 3.14 หลังจากที่ทรานซิสเตอร์หยุดนำกระแส จะเกิดกระแสไหลวนอยู่เป็นวงกลมรอบขดลวดและไดโอด กระแสนี้จะลดค่าลงตามเวลา วิธีนี้เป็นวิธีที่ง่ายที่สุดแต่การทำให้กระแสไหลวนหมดต้องใช้เวลานาน



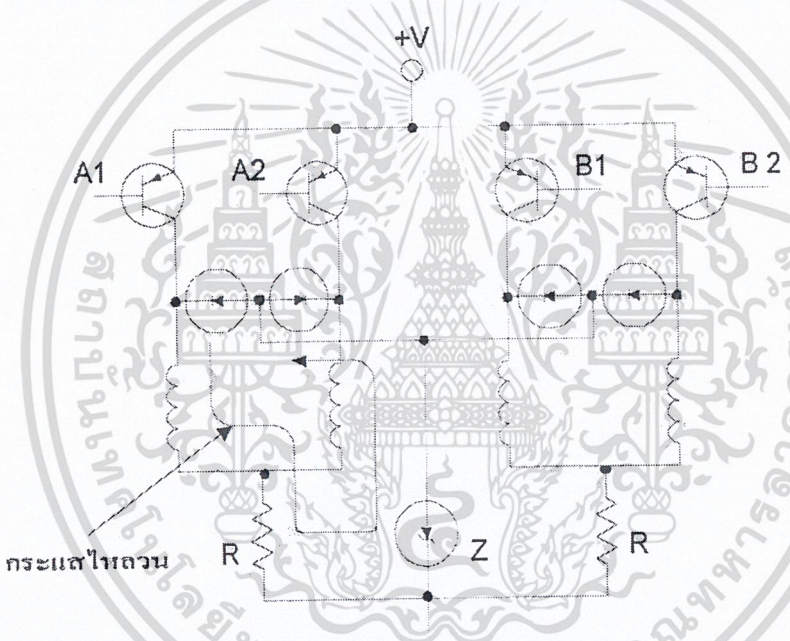
รูปที่ 3.14 สัญญาณควบคุมสเตปปิ้งมอเตอร์ 4 เฟส ที่มีวงจรขับสองขั้วแบบ 1 และ 2 เฟส



***รูปที่ 3.15** วิธีการกำจัดสไปค์ (a) กำจัดด้วยไดโอด (b) กำจัดด้วยไดโอดและความต้านทาน (c) กำจัดด้วยไดโอดและซีเนอร์ไดโอด*

การกำจัดโดยใช้ไดโอดและความต้านทาน (Diode and Resister Suppression) คือการนำค่าความต้านทานมาอนุกรมกับไดโอด ค่าความต้านทานช่วยให้กระแสไหลวนลดค่าลงเร็วขึ้น เพราะมีค่าเวลาลงตัว (Time constant = L/R) น้อยลง

การกำจัดโดยใช้ไดโอดและซีเนอร์ไดโอด (Zener Diode Suppression) การใช้ไดโอดเพียงอย่างเดียวในการกำจัดแรงดันสไปค์มีข้อเสียคือ แรงบิดที่ได้รับจะลดน้อยลงไป นอกเสียจากว่าแรงดันตกคร่อมทรานซิสเตอร์จะมีค่าเพิ่มขึ้นประมาณ 2 เท่า ของแหล่งจ่ายแรงดัน ซึ่งแรงดันที่สูงขึ้นนี้จะทำให้สนามแม่เหล็กและกระแสไหลวนลดค่าอย่างรวดเร็ว ทำให้แรงบิดจึงดีขึ้น ด้วยเหตุนี้จึงใช้ความต้านทานต่ออนุกรมกับไดโอดหรือซีเนอร์ไดโอดต่ออนุกรมกับไดโอดแทน ซึ่งวิธีการใช้ซีเนอร์ไดโอดจะให้ผลดีที่สุด โดยเปรียบเทียบเวลาที่ใช้ในการลดค่ากระแสไหลวน และรูปที่3.16 ตัวอย่างวงจรที่ใช้ซีเนอร์ไดโอด และไดโอดต่ออนุกรมเพื่อลดแรงดันสไปค์ สำหรับความต้านทาน (R) ในวงจรใช้สำหรับสร้างกระแสกระตุ้นขดลวดให้เร็วขึ้น



รูปที่ 3.16 ตัวอย่างวงจรกำจัดแรงดันสไปค์

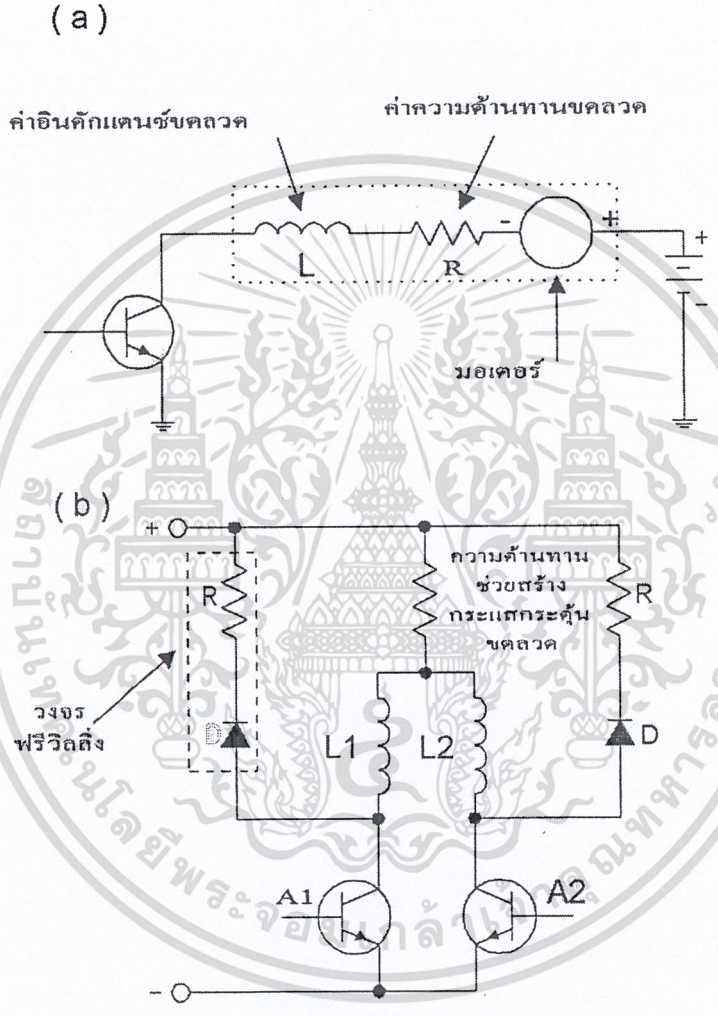
3.3.8 การพิจารณาวงจรขับสเตปปิ้งมอเตอร์

ในสภาวะที่มอเตอร์ทำงานด้วยระดับแรงดันคงที่ค่าหนึ่งแรงบิดที่ได้ จะลดลงถ้ามีการเพิ่มอัตราสเตปเนื่องมาจากการเพิ่มค่าของแรงดันต้านกลับ (Back EMF) และช่วงเวลาขาขึ้น (Rise Time) ของกระแสในขดลวดถูกจำกัดหรือจะมีค่ามีค่ามากวงจรเทียบเคียงของสเตปปิ้งมอเตอร์แสดงดังรูปที่3.17 วิธีแก้ไขให้เกิดการเปลี่ยนแปลงของกระแสในขดลวดเร็วขึ้นทำได้โดยเพิ่มแรงดันของแหล่งจ่ายไฟให้สูงขึ้นเพื่อรักษาระดับกระแสให้คงที่ และขณะทำงานที่อัตราสเตปสูง ๆ หรืออาจคงค่าของแรงดันเอาไว้ด้วย แต่เพิ่มค่าความต้านทานอนุกรมเข้าไปในวงจรดังรูปที่ 3.17 ค่าความต้านทานนี้เรียกว่า ฟอสซิ่ง รีซิสแตนท์ (Forcing Resistance)

ค่าความต้านทานที่ต่อเพิ่มนี้จะลดค่าเวลาคงตัว (t = L/R) ทำให้ใช้งานที่ความเร็วสูงได้ดี ดังนั้นถ้าเพิ่มความต้านทานทีค่า 3 เท่าของความต้านทานขดลวด จะทำให้มีค่าเวลาคงตัวเท่ากับ L/4R และควรเพิ่มขนาด

แรงดัน แหล่งจ่ายไฟเป็น 4 เท่าด้วย เพื่อรักษาระดับกระแสผ่านขดลวดให้คงที่ซึ่งในกรณีนี้อาจมีปัญหาในเรื่องแหล่งจ่ายไฟ หรือเกิดการสูญเสียที่ Forcing resistance ในรูปของความร้อน ดังนั้นที่ความเร็วต่ำ ๆ จึงไม่เหมาะที่จะนำวิธีการนี้มาใช้

ในการออกแบบวงจรขับสเตปปิ้งมอเตอร์ ควรให้มีกำลังงานสูญเสีย จากค่า Forcing Resistance น้อยที่สุด วงจรขับที่จะกล่าวถึงต่อไปนี้จะแก้ปัญหาเหล่านี้ได้

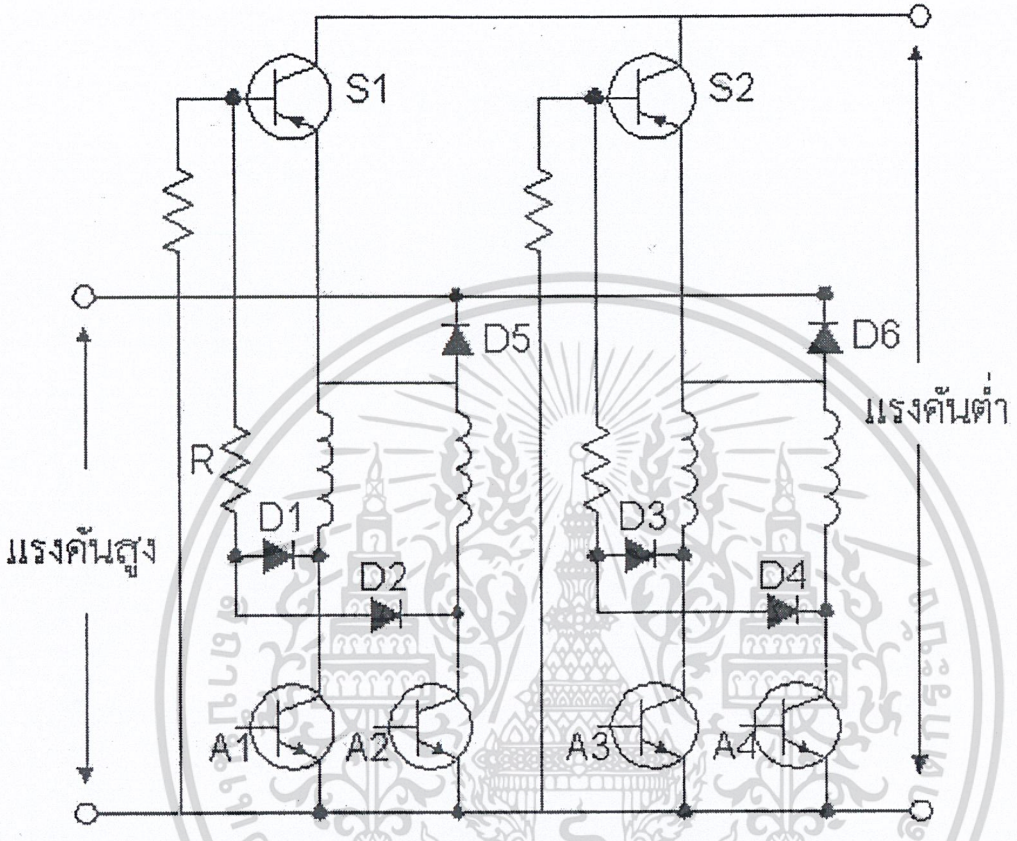


รูปที่ 3.17 (a) แสดงวงจรเทียบเคียงขอลวดของสเตปปิ้งมอเตอร์

(b) แสดงวงจรขับที่มี Forcing resistamce

วงจรขับแบบสองสถานะ (Bi-level drive) วงจรขับแบบสองสถานะ มีแรงดันที่จ่ายให้กับมอเตอร์ 2 ค่า คือค่าสูงและต่ำขณะอัตราสเตปเป็นศูนย์จะรับแรงดันต่ำกว่าขีดจำกัดของมอเตอร์และเมื่ออัตราสเตปเพิ่มจากค่าศูนย์ มอเตอร์จะทำงานที่ระดับแรงดันสูงกว่าขีดจำกัด จากวงจรขับแบบสถานะที่แสดงดังรูป แรงดันค่าต่ำถูกใช้ช่วงอัตราสเตปเป็นศูนย์จนกระทั่งระดับกระแสในขดลวดมีค่าระดับหนึ่งจึงต่อแรงดันค่าสูงเข้าไปแทน

และแรงดันค่าต่ำถูกตัดออกไปโดยการหยุดนำกระแส (Cut off) ของ D5 หรือ D6, D1, D2, D3 และ D4 เป็นฟรีวิลลิ่งไดโอดและ R เป็นฟรีวิลลิ่ง รีซิสแตนซ์



รูปที่ 3.18 วงจรขับแบบสองสถานะ

วงจรขับแบบชอปเปอร์ (Chopper drive) วงจรขับแบบนี้ใช้ระดับแรงดันจากแหล่งจ่ายเพียงค่าเดียวจ่ายให้กับขดลวด โดยกระแสที่จ่ายเข้ามามอเตอร์จะถูกรักษาระดับที่ค่าเฉลี่ยค่าหนึ่งด้วยวิธีการควบคุมระดับกระแสนั่นคือ วงจรออปแอมป์ดีเทคเตอร์จะตรวจสอบค่าของกระแสโดยเปรียบเทียบจากแรงดันที่ตกคร่อม R เมื่อกระแสสูงกว่าระดับเฉลี่ย วงจรออปแอมป์ดีเทคเตอร์จะหยุดขับทรานซิสเตอร์ กระแสที่ไหลผ่านขดลวดจึงลดค่าลง เมื่อลดลงมาต่ำกว่าค่าระดับเฉลี่ย ทรานซิสเตอร์ก็จะถูกขับให้นำกระแสเพื่อให้มีกระแสไหลผ่านขดลวดเพิ่มขึ้น และจะทำงานสลับกันไปเพื่อให้ได้ระดับกระแสเฉลี่ยค่าหนึ่ง

ลักษณะของสัญญาณที่ขับขา B ของทรานซิสเตอร์ A1 หรือ A2 จึงเป็นลักษณะไฟกระแสตรงที่ถูกชอปแรงดัน +V มีค่าประมาณ 5 ถึง 10 เท่า ของระดับแรงดันปกติของมอเตอร์ D1, D2, D3 และ D4 เป็นฟรีวิลลิ่งไดโอด ต่ออนุกรมกับซีเนอร์ไดโอด Z วงจรขับแบบนี้เหมาะสำหรับการใช้งานที่มีการเปลี่ยนแปลงความเร่งอย่างรวดเร็ว

3.4 อุปกรณ์ตรวจจับ (Sensor)

ในวงจรอิเล็กทรอนิกส์ในส่วนนำสัญญาณเข้าที่ทำหน้าที่เป็นส่วนรับรู้ความรู้สึกต่าง ๆ เราเรียกว่า ตัวตรวจจับ (เซนเซอร์) ซึ่งจะทำการเปลี่ยนแปลงความรู้สึกต่าง ๆ ที่ได้รับเป็นสัญญาณทางไฟฟ้าซึ่งอาจจะเป็นแรงดันหรือกระแสก็ได้ และส่งให้กับวงจรอิเล็กทรอนิกส์เพื่อตีความหมาย และเอาผลดังกล่าวไปใช้งานได้ตามต้องการ

ตัวตรวจจับแบบพื้นฐานที่เราคุ้นเคยกันอย่างดี เช่น สวิตซ์กลไก, สวิตซ์แม่เหล็ก, เซลล์รับแสง, โฟโตทรานซิสเตอร์, ออปโต้คัปเปลอร์, ตัวตรวจจับตำแหน่ง, ตัวตรวจจับแรงดัน, ตัวตรวจรับอุณหภูมิ, ตัวตรวจจับเสียง เป็นต้น ตัวตรวจจับต่าง ๆ เหล่านี้ จะทำหน้าที่เปลี่ยนสถานภาพทางฟิสิกส์ให้เป็นสัญญาณทางไฟฟ้า เพื่อนำไปประยุกต์ใช้งานในวงจรอิเล็กทรอนิกส์ให้สามารถทำงานได้ตามต้องการ

3.4.1 โฟโต้ทรานซิสเตอร์ (Photo Transister)

โดยภาวะปกติ สารกึ่งตัวนำจะมีคุณสมบัติที่ไวต่อแสงอยู่แล้ว เมื่อมีการนำเอาสารกึ่งตัวนำมาสร้างเป็นโฟโต้ทรานซิสเตอร์ โปรตอนจากแสงจะทำให้เกิดอิเล็กตรอนอิสระขึ้น เป็นผลทำให้เกิดการไหลของกระแสไฟฟ้าขึ้นได้ ดังนั้นโฟโต้ทรานซิสเตอร์เป็นตัวตรวจจับแสงชนิดหนึ่งซึ่งถูกออกแบบขึ้นมาจากการเกิดประสบการณ์อย่างหนึ่งของสารกึ่งตัวนำ และมีรอยต่อ P-N ระหว่างสารสองชนิดของโฟโตทรานซิสเตอร์ ซึ่งรอยต่อนี้มีขนาดใหญ่กว่ารอยต่อ P-N ของทรานซิสเตอร์โดยทั่วไป ความแตกต่างจากทรานซิสเตอร์ทั่วไปคือที่ตัวถัง (Case) ด้านบนของโฟโต้ทรานซิสเตอร์จะมีช่องสำหรับรับแสงเพื่อส่งไปยังรอยต่อ P-N โดยช่องรับแสงนี้จะมีวัสดุเคลียร์ไมก้า (Clear Mica) หรือควอตเลนซ์ (Quartz Lenz) ติดอยู่บนช่องรับแสงดังกล่าว

วงจรสมมูลย์และการทำงาน วงจรสมมูลย์ของโฟโตทรานซิสเตอร์ ซึ่งก็คือการนำทรานซิสเตอร์มาต่อร่วมกับโฟโตไดโอด โดยตัวโฟโต้ไดโอดจะเป็นตัวควบคุมการจัดแรงดันให้ทรานซิสเตอร์ทำงาน เมื่อเกิดแสงมาตกกระทบที่ตัวโฟโต้ไดโอด จะทำให้เกิดแรงดันไปยังขาเบสของทรานซิสเตอร์ก่อให้เกิดกระแสเบสขึ้น ส่งผลให้ทรานซิสเตอร์ทำงานในที่สุด

ตามปกติการคำนวณหาค่าของกระแสอิมิตเตอร์จะใช้ความสัมพันธ์ดังนี้

$$I_e = I_b(h_{fe}+1)$$

แต่ในกรณีของโฟโตทรานซิสเตอร์ และเนื่องจากที่เบสและคอลเลคเตอร์มีโฟโต้ไดโอดต่อคร่อมอยู่ ดังนั้นเมื่อโฟโตทรานซิสเตอร์ทำงานกระแสที่ไหลผ่านตัวโฟโตไดโอดจึงต้องนำมาพิจารณาเป็นกระแสไหลเข้าร่วมกับกระแสเบสด้วยทำให้สมการของกระแสอิมิเตอร์ของโฟโตทรานซิสเตอร์จึงกลายเป็น

$$I_e = (I_p + I_b).(h_{fe}+1)$$

โดยที่ h_{fe} คือ อัตราการขยายกระแสของตัวโฟโตทรานซิสเตอร์

I_p คือ กระแสที่ไหลผ่านตัวโฟโต้ไดโอด

I_b คือ กระแสเบสของโฟโต้ทรานซิสเตอร์

สำหรับเครื่องหมายบวกและลบของ I_b ในสมการเป็นตัวบ่งบอกถึงชนิดของทรานซิสเตอร์ หากเป็นโฟโต้ทรานซิสเตอร์ชนิดเอ็นพีเอ็นค่าของ I_b จะเป็นบวก แต่ถ้าเป็นชนิด P-N-P ค่าของ I_b จะเป็นลบ

สำหรับความต้านทานด้านไฟฟ้ากระแสสลับของส่วนรับของโฟโตทรานซิสเตอร์จะมีค่าเท่ากับ R_{in} x h_{fe} ในภาวะที่โฟโต้ทรานซิสเตอร์ยังไม่ทำงานค่าความต้านทานภายใน (R) ของโฟโต้ทรานซิสเตอร์จะสูงมาก เนื่องจากการที่โฟโต้ไดโอดภายในโฟโต้ไดโอดภายในโฟโต้ทรานซิสเตอร์ถูกไบแอสกลับไว้ ทำให้เกิดค่าความต้านทานสูงมากขึ้น ซึ่งค่าความต้านทานอินพุทนี้เองจะเป็นตัวท่กำหนดความเร็วในการทำงานของตัวโฟโต้ทรานซิสเตอร์ ดังนั้นหากต้องการนำโฟโต้ทรานซิสเตอร์ไปใช้งานที่มีการสวิตซ์ความเร็วสูงต้องพิจารณาถึงพารามิเตอร์ตัวนี้ด้วย สำหรับสัญลักษณ์ของโฟโต้ทรานซิสเตอร์

พารามิเตอร์อีกตัวหนึ่งที่ต้องให้ความสำคัญคือ ค่าของกระแสรั่วไหลที่เกิดขึ้นภายในตัวโฟโต้ทรานซิสเตอร์ในขณะที่ยังไม่ทำงาน นั่นคือกระแสรั่วไหลระหว่างขาของคอลเล็คเตอร์และอิมิตเตอร์ซึ่งจะเกิดขึ้นในขณะที่โฟโต้ทรานซิสเตอร์ยังไม่มีแสงมาตกกระทบให้ตัวมันทำงาน หรือ $I_{ceo(dark)}$ ซึ่งสามารถคำนวณได้จากสมการ

$$I_{ceo(dark)} = h_{fe} \times I_{CBO}$$

โดยที่ I_{CBO} คือ ค่าของกระแสรั่วไหลที่ขาคอลเล็คเตอร์และเบส ซึ่งก็คือกระแสรั่วไหลของตัวโฟโต้ไดโอดนั่นเอง

ปกติในโฟโต้ทรานซิสเตอร์ทั่ว ๆ ไปค่าของกระแสรั่วไหลนี้จะต่ำมาก ๆ อยู่ระหว่าง 4-8 ไมโครแอมป์ที่อุณหภูมิห้อง

3.4.2 อินฟาเรด แอลอีดี (Infrared LED)

อินฟาเรด แอลอีดี ถูกสร้างขึ้นมาเพื่อกำเนิดแสงในย่านอินฟาแรด เมื่ออินฟาแรด แอลอีดีนำกระแสอิเล็คตรอนจะเคลื่อนที่ผ่านสารกึ่งตัวนำชนิดพิเศษ และเกิดพลังงานจากโฟตอน การเกิดพลังงานดังกล่าวเป็นไปในทันที ที่มีกระแสไหลผ่าน

อินฟาเรด แอลอีดี สามารถกำเนิดแสงอินฟาเรดได้ในช่วงสองความยาวคลื่นคือ อินฟาเรดแอลอีดีที่สร้างจากสารแกลเลี่ยมอาเซไนต์ (Gallium Arsenide : GaAs) จะให้ความยาวคลื่นประมาณ 940 นาโนเมตร และอินฟาแรดแอลอีดีที่สามารถสร้างจากสารแกลเลี่ยมอลูมินั่มอาเซไนต์ (Gallium Aluminium Arsenied :GaAlAs) ซึ่งจะกำเนิดแสงอินฟาเรดที่มีความยาวคลื่นประมาณ 880 นาโนเมตร

3.4.3 โมดูลแสดงผลแบบผลึกเหลว (LCD Module)

ในโมดุล LCD ประกอยด้วยส่วนประกอบหลักๆ 3 ส่วนคือ

- ตัวแสดงผล (Display) ภายในจะเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นดดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบน LCD
- ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้จะใช้ชิปควบคุมโดยฉพาะ ชิปที่นิยมคือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก
- ตัวขับ (วงจรขับสเตปปิ้งมอเตอร์) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดง

- ข้อมูลตามที่กำหนด ชิปที่ทำหน้าที่นี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

โมดูล LCD มีขาต่อทั้งสิ้น 14 ขา มีการจัดขาและรายละเอียดดังนี้

Vss (ขา 1) : ต่อกราวด์

Vdd (ขา 2) : ต่อไฟเลี้ยง +5 V

Vo (ขา 3) : เป็นขาอินพุทรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) : เป็นขาอินพุทในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็น ข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา 5) : เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล ถ้าเป็น ”1” จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาอีนาเบิล LCD ให้ทำงาน

D0-D7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิท

บทที่ 4
การสร้างและการออกแบบฮาร์ดแวร์ไมโครเมาส์

ในบทนี้จะอธิบายถึงรายละเอียดการสร้างและการออกแบบของโครงงาน ไมโครเมาส์นี้โดยจะอธิบายด้านฮาร์ดแวร์ว่ามีการใช้อะไรเป็นส่วนประกอบบ้าง

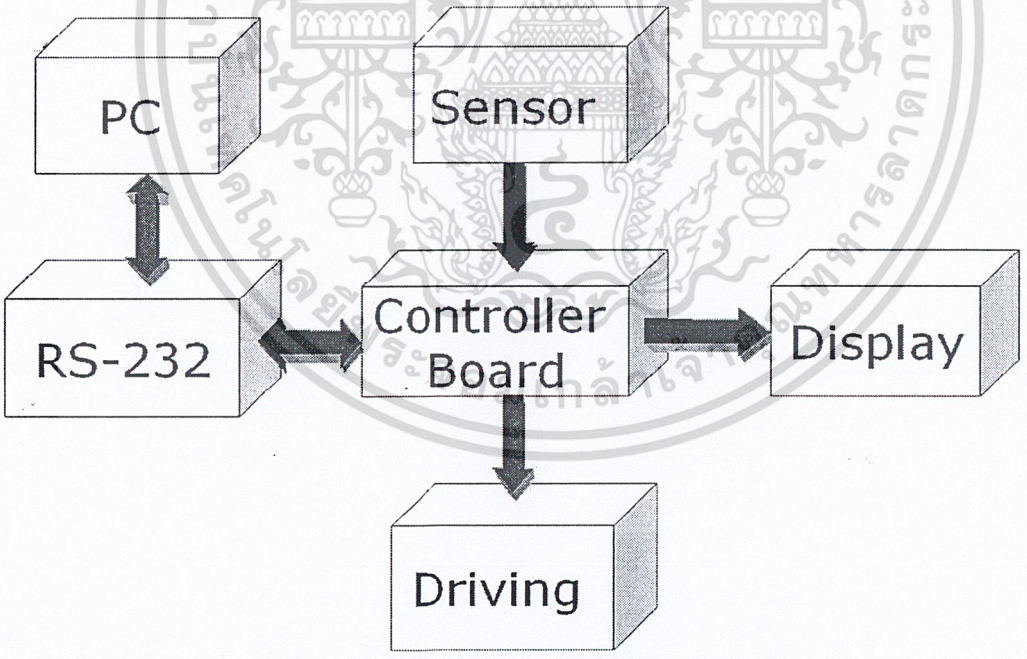
ในส่วนของการสร้างและการออกแบบสามารถแบ่งออกได้เป็นส่วน ๆ ได้ดังนี้

ส่วนประกอบของฮาร์ดแวร์

- ไมโครคอนโทรลเลอร์(MCS-51)
- เซนเซอร์(infrared)
- สเตปปิ้งมอเตอร์
- แสดงผล(LCD)
- อุปกรณ์เชื่อมต่อกับคอมพิวเตอร์(RS-232)

4.1 การออกแบบและการสร้างในส่วนของฮาร์ดแวร์

ในโครงงานไมโครเมาส์ สามารถแบ่งการทำงานออกเป็นส่วน ๆ ดังนี้



รูปที่ 4.1 Block Diagram ของไมโครคอนโทรลเลอร์

4.1.1 คอนโทรลเลอร์บอร์ด

เริ่มจากตัวไมโครคอนโทรลเลอร์ MCS-51จะเชื่อมต่อกับชุดขับมอเตอร์ โดยผ่านทางพอร์ต A ของ 8255 ชุดวงจรเซนเซอร์จะต่อกับชุดวงจรเปรียบเทียบแรงดัน ซึ่งจะทำการเปรียบเทียบแรงดันที่มาจากชุดเซนเซอร์แล้วส่งไปยัง MCS-51 โดยผ่านทางพอร์ต B ของ8255 ส่วนชุดการแสดงผลจะต้องกับพอร์ต C ของ 8255

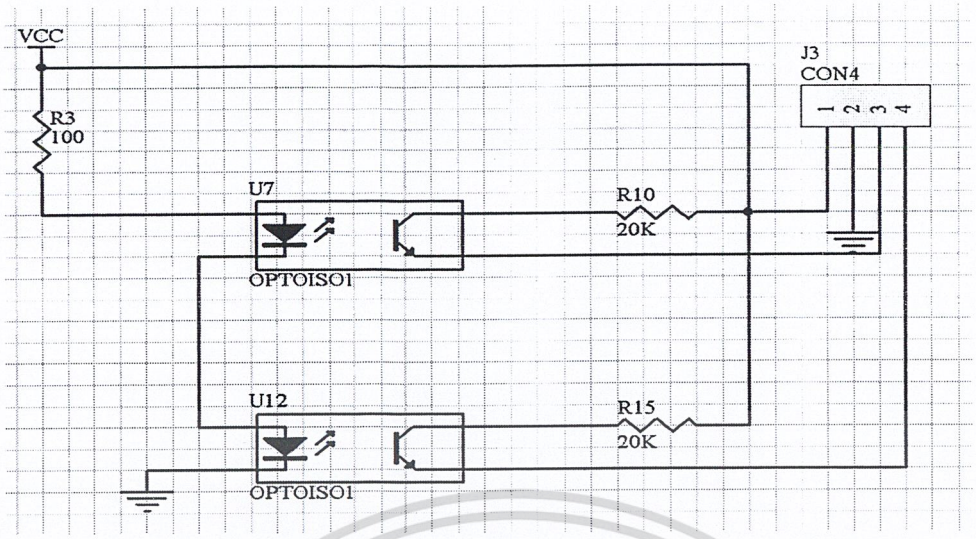
ในส่วนของพอร์ต 2 และ 0 ของMCS-51จะเป็นขาแอดเดส ตัวเชื่อมต่อระหว่างหน่วยความจำและตัว 8255 โดยมีไอซี 74LS373 เป็นตัวมัลติเพล็กระหว่างแอดเดรสต่ำกับ ข้อมูล ส่วนIC 74LS138 เป็นตัวรหัสแอดเดส เพื่อเลือกอุปกรณ์ที่ ไมโครคอนโทรลเลอร์จะติดต่อด้วย หน่วยความจำRAM จะมีขนาด 32 KByte โดยใช้เบอร์ 62256 โดย 8 KByte แรกจะเป็น Flash ROM ซึ่งอยู่ในตัว MCS-51 โดยมี MAX232 จะเป็นตัวเชื่อมต่อระหว่าง MCS-51 กับเครื่องคอมพิวเตอร์ โดยข้อมูลจะถูกส่งและรับที่ขา TXD, RXD ของ MCS-51 ผ่านตัว ชิป MAX232 ไปยังเครื่องคอมพิวเตอร์

ส่วนการทำงานของพอร์ต 1 นั้นจะไว้ใช้สำหรับการรับโปรแกรมจากคอมพิวเตอร์ของ MCS-51 ตระกูล xxSxx ถ้าไม่ใช้เป็นพอร์ต ก็สามารถนำไปใช้ต่อกับอุปกรณ์อื่นได้อิสระ โดยเป็นได้ทั้งอินพุทและเอาท์พุท

สวิตช์เป็นอุปกรณ์อินพุทให้กับวงจร ในการรับคำสั่งจาก User ซึ่งในการต่อจะต้องมี ความต้านทานพลูอัพ ต่ออยู่เพื่อรักษาสถานะหนึ่งให้คงที่ ในกรณีที่ สวิตช์ ไม่มีการกด ตามวงจรเบื้องต้นจะได้ค่าลอจิกเป็น “1” เมื่อกด สวิตช์ จะได้ลอจิก “0” ในโครงงานนี้จะต่ออยู่กับพอร์ต 1 ของไมโครคอนโทรลเลอร์ ปุ่ม select และ enter ของ MCS-51 มีสวิตช์ที่ต่อกับ Interrupt 0 เพื่อเข้าสู่เมนูการทำงานหลัก และ Jumper ที่ Interrupt 1 เพื่อให้ ผู้ใช้ได้ใช้งาน หากต้องการใช้งาน Interrupt เพื่อต่อกับอุปกรณ์อื่น

4.1.2 เซนเซอร์

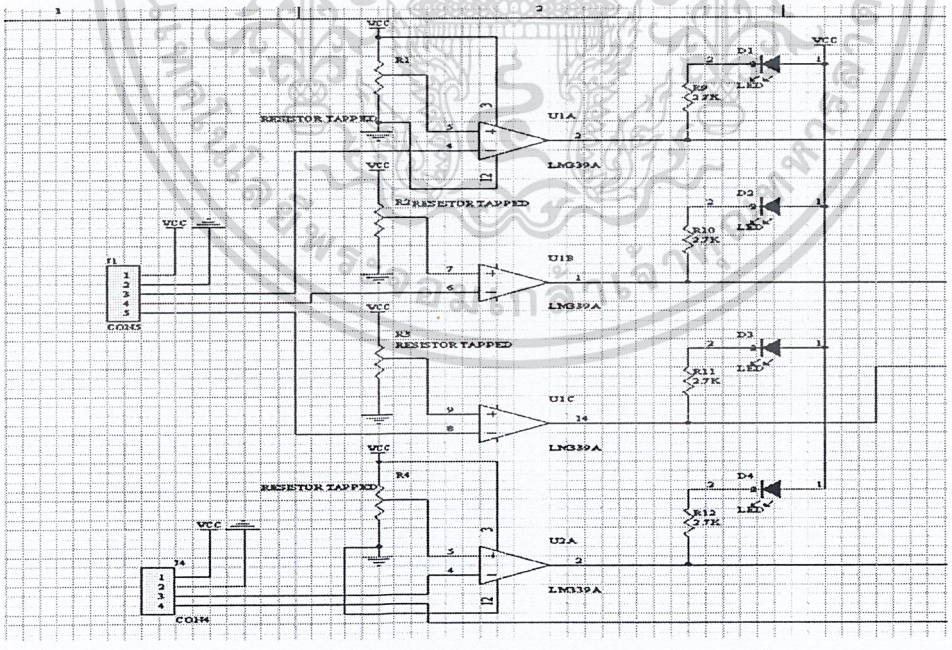
เป็นอุปกรณ์ที่รับข้อมูลจากสิ่งแวดล้อม ซึ่งในโครงงานนี้จะใช้เป็นตัวตรวจสอบกำแพง อุปกรณ์ที่ใช้กับโครงงานนี้คือ โฟโต้ทรานซิสเตอร์ ซึ่งจะเป็นอุปกรณ์ที่หาง่ายและราคาก็ไม่ค่อยแพง หลักการในการทำงานเบื้องต้นนั้นก็คือ ในตัว เซนเซอร์ จะมีตัวรับและตัวส่งภายในตัวเดียวกัน ในกรณีที่เราทำการส่งแสงออกไปจากตัวส่ง เมื่อแสงนั้นถูกส่งออกแล้วเกิดไปกระทบกับวัสดุ ที่สามารถสะท้อนแสงได้ดีเช่น สีขาว แสงนั้นก็จะถูกสะท้อนไปยังตัวรับแสง ทำให้เกิดกระแสไหลในโฟโต้ทรานซิสเตอร์ แต่ถ้าไม่มีแสงมากระทบกับตัวรับกระแสก็จะไม่สามารถไหลผ่านตัวโฟโต้ทรานซิสเตอร์ได้ ซึ่งหลักการทำงานเบื้องต้นของ โฟโต้ทรานซิสเตอร์จึงสามารถที่จะนำมาใช้ในโครงงานนี้ได้



รูปที่ 4.2 วงจร เซนเซอร์

4.1.3 วงจรเปรียบเทียบความต่างศักย์

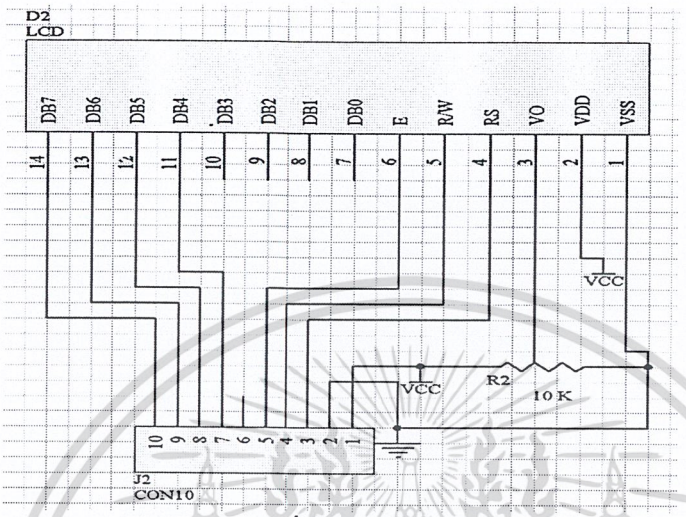
เมื่อวงจรเซนเซอร์ตรวจจับกำแพงก็จะส่งผลการทำงานมายังวงจรเพื่อเปรียบเทียบกับความต่างศักย์อ้างอิง ที่เกิดจาก ความต้านทานปรับค่าได้ เราใช้ ออบ-แอมป์ เบอร์ LM 339 โดยจะต้องต่อความต้านทานพลูอัพไว้ มี LED เพื่ออำนวยความสะดวกในการตรวจสอบผล



รูปที่ 4.3 วงจรเทียบความต่างศักย์

4.1.4 หน่วยแสดงผล

เป็นอุปกรณ์เอาท์พุตที่ทำหน้าที่แสดงผล การต่อ LCD ในโครงงานนี้จะต่อผ่านตัวต้านทานปรับค่าได้ เพื่อปรับความสว่าง แสดงค่าได้ 2 แถว แถวละ 20 ตัวอักษร ใช้การต่อแบบ 4 บิทเพื่อเป็นการประหยัดขาสัญญาณที่นำมาใช้



รูปที่ 4.4 วงจรแสดงผล

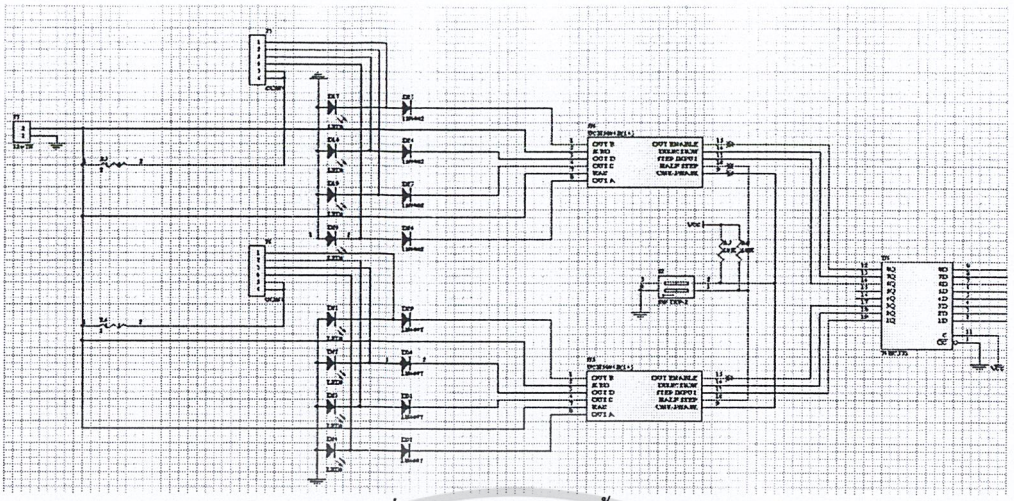
4.1.5 วงจรขับสเตปปิ้งมอเตอร์

วงจรนี้คือวงจรที่ทำหน้าที่เป็นตัวขับให้สเตปปิ้งมอเตอร์ให้ทำงานได้ โดยจะรับคำสั่งจากตัวไมโครคอนโทรลเลอร์ ว่าจะให้ทำงานอย่างไร สิ่งที่ วงจรขับสเตปปิ้งมอเตอร์ ควรจะมีก็คือ วิ่งไปข้างหน้า, วิ่งถอยหลังได้, หยุดอยู่กับที่, หยุดการทำงาน, และที่สำคัญส่วนหนึ่งก็คือวิธีการขับ สเตปปิ้งมอเตอร์ ซึ่งมีอยู่ 3 แบบ เมื่อได้กำหนดลักษณะการทำงานของวงจรแล้ว ก็ทำการหาอุปกรณ์อิเลคทรอนิกส์ ที่สามารถตอบสนองความสามารถที่ต้องการนี้ ในตลาดอุปกรณ์ที่ใช้ในการขับนั้นมีอยู่หลายตัวเช่น L298, L297 และ TEA3718, TCA1560, ULN 2803 ทั้งสี่ตัวนี้สามารถที่จะทำงานในสิ่งที่ต้องการได้ แต่เมื่อพิจารณาถึงองค์ประกอบต่าง ๆ แล้ว UCN5804 เป็นตัวที่เหมาะสมที่สุด

เหตุผลที่เลือก UCN 5804B

- ใช้พลังงานไม่มากนักในการทำงานและเที่ยงตรง เพราะเป็นวงจรสวิตช์ชิ่ง
- มีความเที่ยงตรงเนื่องจากมีโหมดในการเลือกเฟสที่ใช้ในการทำงาน
- มีการตรวจจับอุณหภูมิหากมีอุณหภูมิสูงเกินไป
- ประหยัดสายสัญญาณที่ใช้
- สามารถขับมอเตอร์ได้ถึง 35 V 1.5 A
- สามารถขับสเตปปิ้งมอเตอร์ได้ถึง 3 รูปแบบ

ข้อเสีย คือมีราคาแพงกว่าเมื่อเทียบกับตัวอื่น แต่ถ้าเทียบความสามารถจากที่กล่าวมาแล้วถือว่าคุ้มค่ากับราคา



รูปที่ 4.5 วงจรขับสเตปปิ้งมอเตอร์

4.1.6 หน่วยความจำ

ในโครงงานนี้จะใช้ Flash ROM และ RAM ทำหน้าที่เป็นหน่วยความจำ โดย Flash ROM จะเก็บโปรแกรมในการควบคุมการทำงานหน้าที่ต่าง ๆ ดังนี้

- โปรแกรมรับโปรแกรมจากเครื่องคอมพิวเตอร์
- โปรแกรมส่งข้อมูลสถานะของไมโครเมาส์
- โปรแกรมควบคุมความเร็วของสเตปปิ้งมอเตอร์

ในส่วนของ RAM นี้สำหรับเก็บโปรแกรมที่รับมาจากเครื่องคอมพิวเตอร์ ซึ่งบนเครื่องคอมพิวเตอร์นั้นจะเขียนขึ้นด้วยภาษา C แล้วคอมไพล์เป็น Assembly แล้วแปลงเป็นภาษาเครื่อง(เลขฐาน16) จากนั้นก็ส่งมายังไมโครเมาส์โดยผ่านพอร์ตอนุกรมของเครื่องคอมพิวเตอร์

หน่วยความจำภายใน (89C51)	Address 0000H - 1FFFH
หน่วยความจำภายใน (89C52)	Address 0000H - 3FFFH
หน่วยความจำภายนอก	Address 8000H – EFFFH

ตารางที่ 4.1 แสดงแอดเดสของหน่วยความจำทั้งภายในและภายนอก

4.1.7 พอร์ตอินพุท/เอาท์พุท

ในส่วนของวงจรนี้เราใช้ไอซีเบอร์ 8255 เป็นส่วนเชื่อมต่อกับ เซนเซอร์ และ LCD โดยมีตำแหน่งของ Port ดังนี้

Display	เอาท์พุท	เชื่อมต่ออยู่ที่ Port A	Address F000H
เซนเซอร์	อินพุท/เอาท์พุท	เชื่อมต่ออยู่ที่ Port B	Address F001H
Motor	อินพุท	เชื่อมต่ออยู่ที่ Port C	Address F002H
พอร์ตควบคุม	เอาท์พุท	เชื่อมต่ออยู่ที่ Port Control	Address F003H

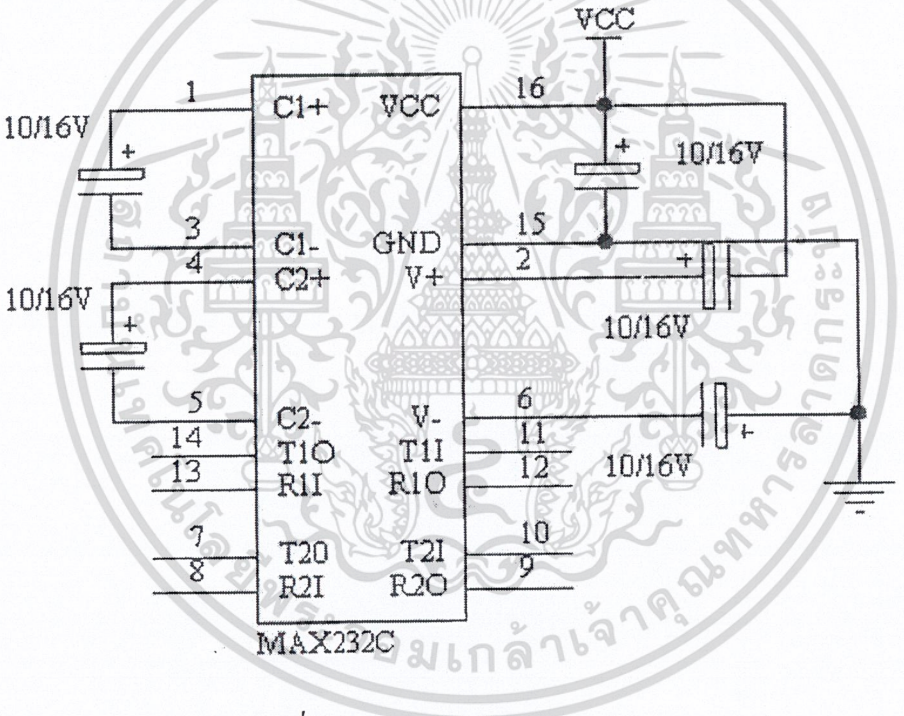
ตารางที่ 4.2 แสดงการเชื่อมต่ออุปกรณ์กับพอร์ต 8255

4.1.8 พอร์ตอนุกรม

ในการเชื่อมต่อวงจรที่ทำงานในระดับแรงดันแบบทีทีแอล (TTL) เข้ากับพอร์ต RS-232 ของเครื่องคอมพิวเตอร์ ซึ่งมีระดับแรงดัน –15 ถึง +15 นั้น จะต้องมีวงจรพิเศษเพื่อทำการเปลี่ยนแปลงระดับแรงดันให้เหมาะสมซึ่งในโครงงานนี้ได้เลือกใช้ IC เบอร์ MAX232 ซึ่งเป็น IC ที่ใช้อุปกรณ์ประกอบจากภายนอกน้อย คือใช้ C ชนิดอิเล็คโทรไลท์เพียง 5 ตัวเท่านั้น ที่สำคัญคือมีราคาถูกเมื่อเทียบกับ DS275

หลักการทำงาน

จากวงจรจะใช้ C ที่ต่อระหว่างขา 1 กับ 3 , ระหว่างขา 4 กับ 5 , ขา 2 และ ขา 6 เป็นตัวกำหนดระดับแรงดันในการเชื่อมต่อ โดยขา R11 และ R21 จะเป็นขาที่รับระดับแรงดัน –15 ถึง +15 และแปลงออกเป็นแรงดัน 0V, +5V ออกที่ขา R10 และ R20 ส่วนขา T11 และ T21 ก็จะรับแรงดันที่เป็น 0 V ,+5V แปลงเป็นระดับแรงดัน –10 ถึง +10 ออกที่ขา T10 และ T20



รูปที่ 4.6 วงจรรับส่งข้อมูลอนุกรม

4.1.9 วงจรกำหนดแอดเดส

โดยใช้ไอซี 74LS138 เป็นตัวถอดรหัสตำแหน่งของหน่วยความจำและอุปกรณ์ต่างๆ โดยเอาขาแอดเดรสของไมโครคอนโทรลเลอร์ 51 ไปติดต่ออุปกรณ์ดังนี้

- หน่วยความจำ 62256
- 8255 Port I/O

บทที่ 5
การสร้างและการออกแบบซอร์ฟแวร์ของไมโครเมาส์

ในบทนี้จะอธิบายถึงรายละเอียดการสร้างและการออกแบบของโครงงาน ไมโครเมาส์นี้โดยจะอธิบายด้านซอร์ฟแวร์ว่ามีการใช้อะไรเป็นส่วนประกอบบ้าง

โปรแกรมอีดิตเตอร์ เขียนด้วย Visual Basic V.6 ซึ่งเป็นเครื่องมือในการเขียนโปรแกรมบนระบบปฏิบัติการ Windows ที่ใช้งานง่าย โดยการสร้างโปรแกรมใน visual Basic V.6 จะเป็นการเลือกเครื่องมือต่างๆมาออกแบบหน้าจอของโปรแกรมที่จะทำการสร้าง ซึ่งการเขียนโปรแกรมลักษณะนี้เรียกว่า Visual Programming ซึ่งมีข้อดีคือไม่ต้องเขียนคำสั่งมากนัก สามารถสร้างโปรแกรมได้อย่างรวดเร็ว และสามารถสร้างอินเตอเฟสที่สวยงาม

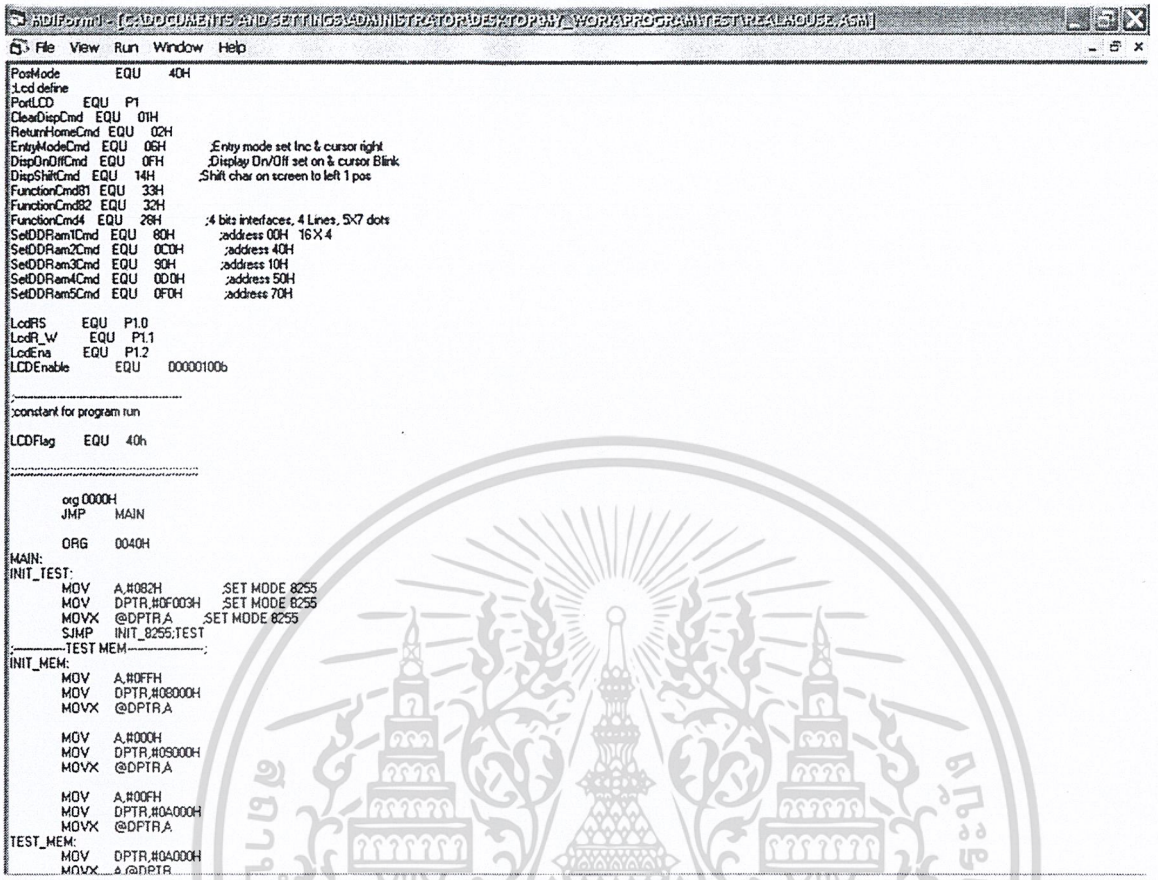
โปรแกรมอีดิตเตอร์ ใช้ในการเขียนโปรแกรมควบคุมฮาร์ดแวร์ เพื่อความสะดวกในการใช้งานไมโครเมาส์ พร้อมทั้งตัวอย่างฟังก์ชั่นการทำงาน การติดต่อฮาร์ดแวร์และการคำนวณ ทั้ง แอสเซมบลีและภาษา C เพื่อนำไปใช้ศึกษาทำความเข้าใจการเขียนพัฒนาโปรแกรมสำหรับไมโครเมาส์ และเขียนโปรแกรมควบคุมอุปกรณ์อื่นๆ ต่อไป

ส่วนประกอบของซอฟท์แวร์

- โปรแกรมอีดิตเตอร์
- การมอนิเตอร์ฮาร์ดแวร์
- การส่งโปรแกรมจากคอมพิวเตอร์ไปยังไมโครเมาส์
- ฟังก์ชั่นพื้นฐานของไมโครเมาส์

5.1 การใช้งานอีดิตเตอร์

ในการเขียนโปรแกรมควบคุมไมโครเมาส์สามารถเขียนได้ 2 ภาษาคือ C 51 และ แอสแซมบี (Assembly) การใช้ภาษาที่ใช้ภาษา C โดยคอมพายไปเป็น โค้ดของไมโครคอนโทรเลอร์ตระกูล 8051 เพราะการเขียนด้วยภาษา C จะง่ายในการทำความเข้าใจและพัฒนา ใช้ในฟังก์ชั่นที่มีการทำงานซับซ้อน ส่วนการใช้ภาษา แอสแซมบี้ จะใช้ในฟังก์ชั่นที่มีการทำงานไม่ซับซ้อน และมีการคำนวณไม่มาก เน้นที่การติดต่อและควบคุมตัวอุปกรณ์ฮาร์ดแวร์ เราได้เลือกใช้ซอพท์แวร์ในการคอมไพล์ คือ C51 ของ Franklin โดยคอมไพล์จากภาษา C ไปเป็นโค้ดของไมโครโปรเซสเซอร์ตระกูล 8051

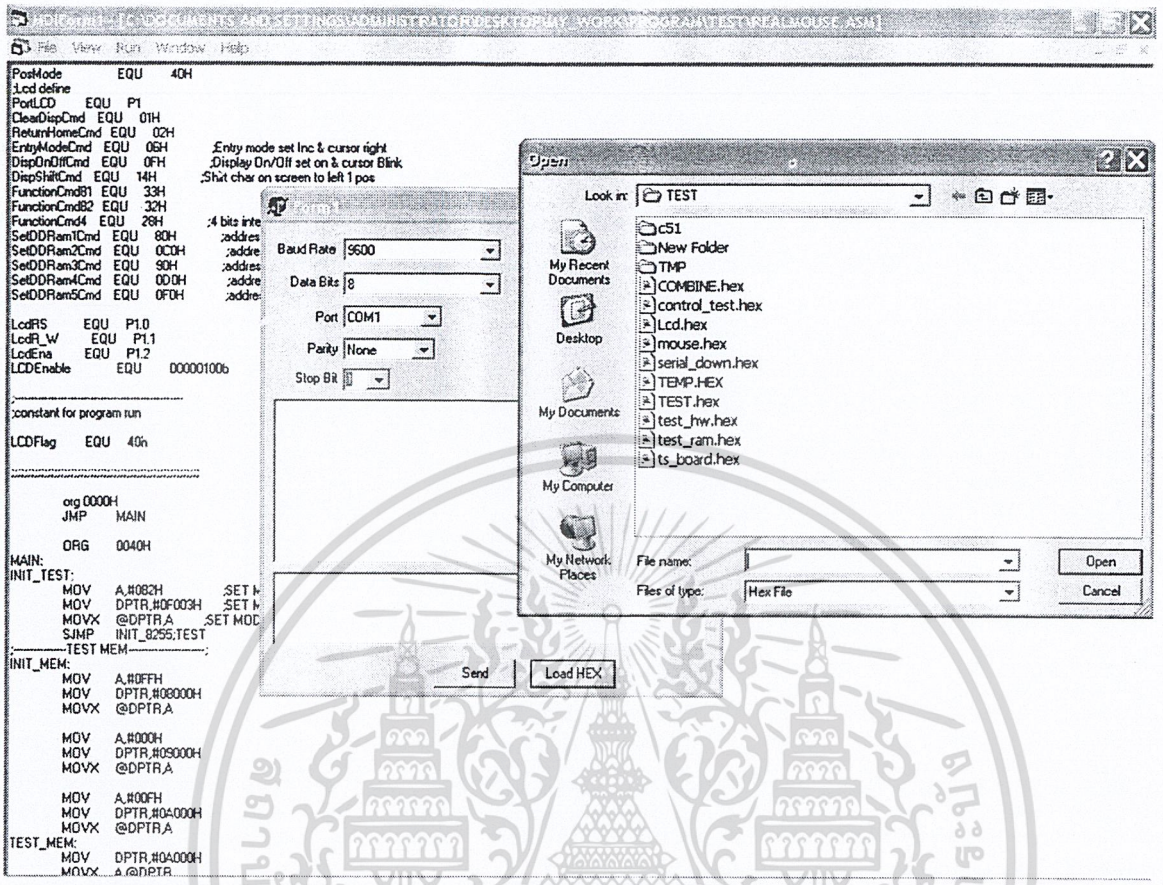


```
ADIEdit1 - [C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\DESKTOP\MY_WORK\PROGRAM\TEST\REALHOUSE.ASM]
File  View  Run  Window  Help
PosMode          EQU     40H
;Lcd define
PortLCD     EQU   P1
ClearDispCmd  EQU   01H
ReturnHomeCmd  EQU   02H
EntryModeCmd  EQU   06H          ;Entry mode set Inc & cursor right
DispOnOffCmd  EQU   0FH          ;Display On/Off set on & cursor Blink
DispShiftCmd  EQU   14H          ;Shift char on screen to left 1 pos
FunctionCmd81  EQU   33H
FunctionCmd82  EQU   32H
FunctionCmd4  EQU   28H          ;4 bits interfaces, 4 Lines, 5X7 dots
SetDDRam1Cmd  EQU   80H          ;address 00H  16X4
SetDDRam2Cmd  EQU   0C0H         ;address 40H
SetDDRam3Cmd  EQU   90H          ;address 10H
SetDDRam4Cmd  EQU   0D0H         ;address 50H
SetDDRam5Cmd  EQU   0F0H         ;address 70H

LcdRS       EQU   P1.0
LcdR_W      EQU   P1.1
LcdEna      EQU   P1.2
LCDEnable         EQU     00000100b

;-------------------------------------------
;constant for program run

LCDFlag     EQU   40h

;-------------------------------------------

        org 0000H
        JMP     MAIN

        ORG     0040H
MAIN:
INIT_TEST:
        MOV     A,#082H          ;SET MODE 8255
        MOV     DPTR,#0F003H     ;SET MODE 8255
        MOVX    @DPTR,A          ;SET MODE 8255
        SJMP    INIT_8255;TEST
;----------TEST MEM----------;
INIT_MEM:
        MOV     A,#0FFH
        MOV     DPTR,#08000H
        MOVX    @DPTR,A

        MOV     A,#000H
        MOV     DPTR,#09000H
        MOVX    @DPTR,A

        MOV     A,#00FH
        MOV     DPTR,#0A000H
        MOVX    @DPTR,A
TEST_MEM:
        MOV     DPTR,#0A000H
        MOVX    A,@DPTR
```

รูปที่ 5.1 ลักษณะของโปรแกรมอีดิตเตอร์

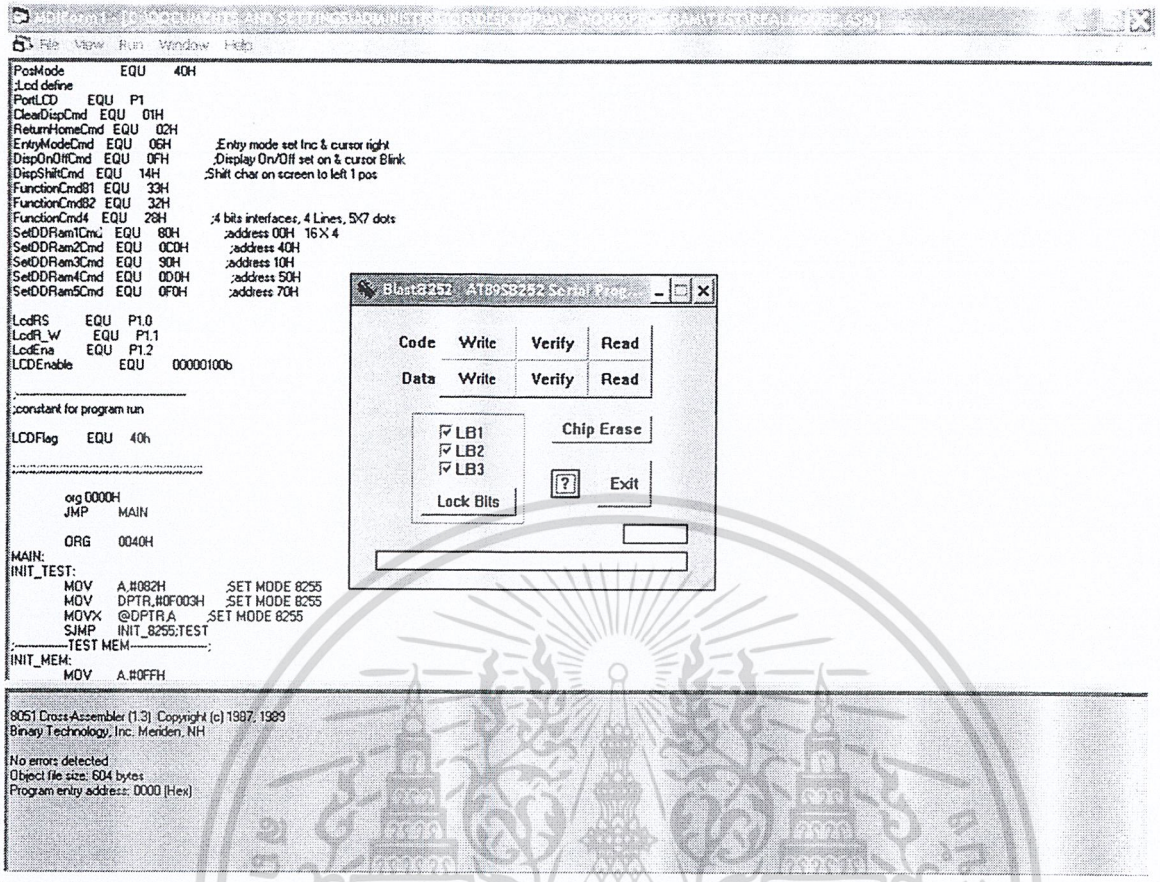
5.2 การส่งโปรแกรมจากคอมพิวเตอร์ไปยังไมโครเมาส์

เมื่อได้คอมไพล์ไฟล์ที่เป็น แอสเซมบี หรือ C51 แล้วถ้าถูกต้องก็จะได้ ไฟล์นามสกุล HEX เมื่อจะทำการส่งโปรแกรมจากคอมพิวเตอร์ไปยังไมโครเมาส์ ต้องทำการเซตไมโครเมาส์ให้เตรียมพร้อมในการรับโปรแกรมก่อน โดยเลือกสถานะโดยดูจาก LCD เมื่อไมโครเมาส์พร้อมรับข้อมูลก็เรียกโปรแกรมส่วนที่ส่งข้อมูลไปยังไมโครเมาส์แล้วเลือกไฟล์ HEX ที่ต้องการ โปรแกรมจะทำการส่งโปรแกรมที่อยู่ในไฟล์ ซึ่งอยู่ในรูปของ Intel Hex From



รูปที่ 5.2 ของโปรแกรมขณะทำการส่งโปรแกรมไปยังไมโครเมาส์ผ่านพอร์ตอนุกรม

จากรูปที่ 5.2 เป็นการส่งโปรแกรมจากคอมพิวเตอร์ไปยังหน่วยความจำโปรแกรมภายนอก ผ่านทางพอร์ตอนุกรม และจะต้องมีโปรแกรม ดาวโหลดมอนิเตอร์ (Download Monitor) ในตัว MCS-51 โดยโปรแกรมดาวโหลดมอนิเตอร์ ให้ผู้ใช้ได้เลือกการทำงาน ว่าต้องการจะเริ่มการทำงานของไมโครเมาส์, มอนิเตอร์โปรแกรม หรือ รอรับโปรแกรมจากคอมพิวเตอร์ ถ้าปิดไมโครเมาส์โปรแกรมก็จะหายไป

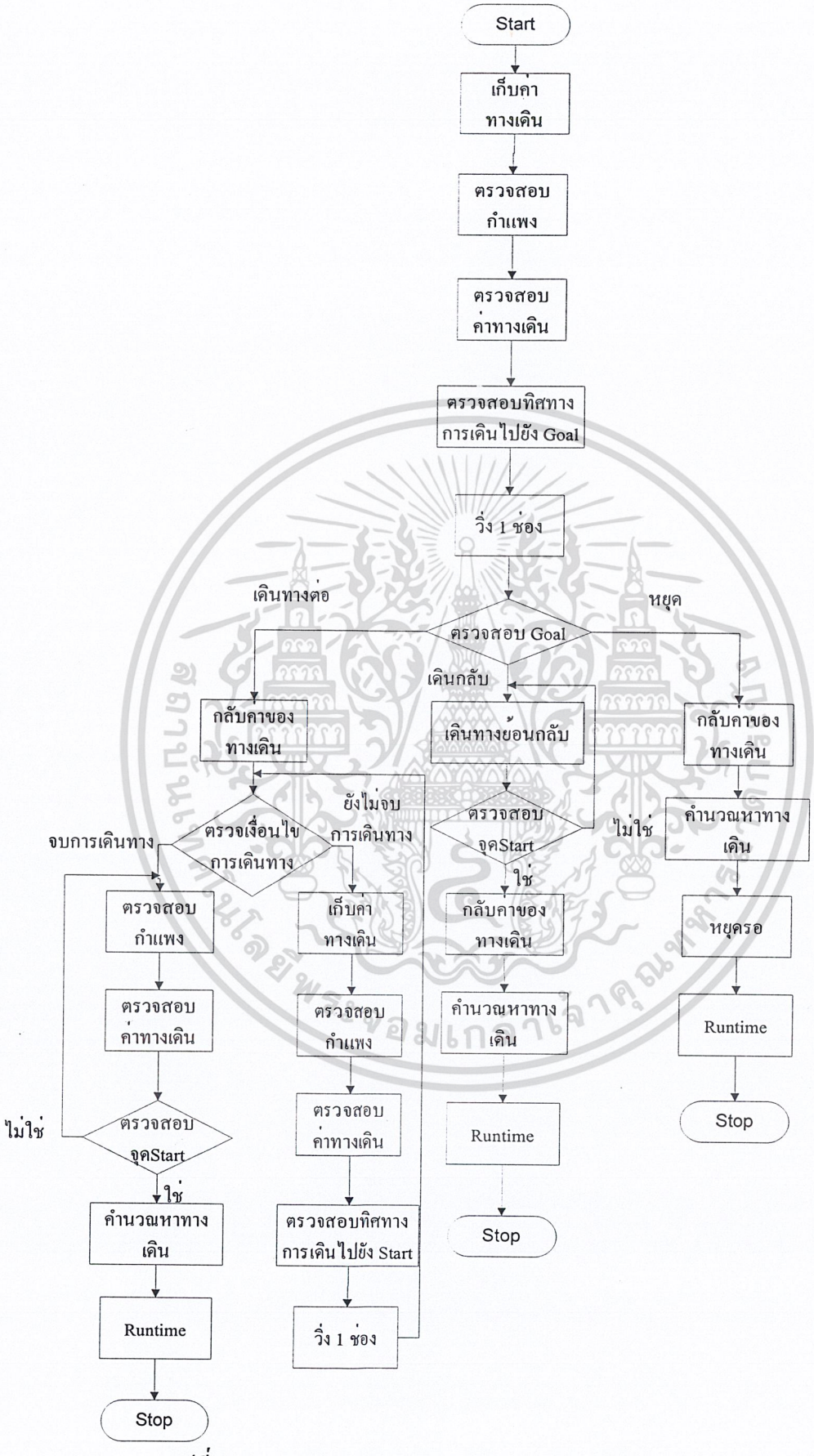


รูปที่ 5.3 ของโปรแกรมขณะทำการส่งโปรแกรมไปยังไมโครเมาส์ด้วย ISP

จากรูปที่ 5.3 เป็นการส่งโปรแกรมจากคอมพิวเตอร์ไปยังหน่วยความจำโปรแกรมภายใน ผ่านทางพอร์ตขนาน ซึ่งวิธีการนี้ไม่ต้องมีโปรแกรม ดาวโหลดมอนิเตอร์ (Download Monitor) ในตัว MCS-51 โดยจะส่งโปรแกรมไปยังหน่วยความจำโปรแกรมภายในของ MCS-51 เลย แต่ต้องใช้กับ MCS-51 รุ่น xxSxx เท่านั้น เช่น AT89S8252 เป็นต้น

5.3 การมอนิเตอร์ฮาร์ดแวร์

สามารถสั่งควบคุมการทำงานของฮาร์ดแวร์ผ่านทาง พอร์ตอนุกรม โดยไม่จำเป็นต้องต่ออุปกรณ์ครบชุด โดยต้องต่อ บอร์ดคอนโทรลเลอร์ กับอุปกรณ์ที่ต้องการจะทำการมอนิเตอร์ และเขียนคำสั่งง่ายๆ เพื่อทดสอบดูว่าอุปกรณ์ทำงานได้ถูกต้องหรือไม่ โปรแกรมจะทำการส่งการทำงานไปยัง บอร์ดคอนโทรลเลอร์ หลังจากนั้น บอร์ดคอนโทรลเลอร์จะรับคำสั่งและทำการควบคุมอุปกรณ์นั้น ซึ่งเป็นการทดสอบอุปกรณ์ว่าทำงานถูกต้องหรือไม่เท่านั้น



รูปที่ 5.4 Block Diagram ตัวอย่างการทำงานของไมโครเมาส์

บทที่ 6
ผลการทดลองและสรุป

6.1 การทำงานของไมโครเมาส์

เมื่อเปิดสวิตช์การทำงานของไมโครเมาส์หรือกดรีเซตตัว MCS-51 จะขึ้นเมนูการทำงานมา 3 อย่างมาให้เลือกคือ

6.1.1 เริ่มต้นการทำงาน

ถ้าผู้ใช้เลือก ไมโครเมาส์จะกระโดดไปทำงานยัง address ของหน่วยความจำโปรแกรมที่อยู่ใน RAM แอดเดส 08000H เป็นการเริ่มต้นการทำงานของไมโครเมาส์ตามโปรแกรมที่รับมาจากคอมพิวเตอร์ ซึ่งมีขนาดของโปรแกรมทั้งหมดไม่เกิน 32KByte ตามขนาดของ 62256 ไมโครเมาส์จะทำงานก็ต่อเมื่อได้มีการรับโปรแกรมจากเครื่องคอมพิวเตอร์มาใส่แล้วเท่านั้น ถ้ายังไม่มีการรับโปรแกรมใดๆมาไมโครเมาส์ก็จะยังไม่ทำงาน

ขณะที่ไมโครเมาส์ทำงานอยู่หากต้องการเข้าเมนูหลัก ผู้ใช้จะต้องทำการกดสวิตช์อินเตอร์รัพท์ ไมโครเมาส์จากนั้นก็จะเข้าเมนูหลักต่อไป

ถ้ามีการปิดสวิตช์ไมโครเมาส์ โปรแกรมที่รับมานั้นจะหายไปจะต้องทำการรับข้อมูลใหม่จากคอมพิวเตอร์

6.1.2 รับโปรแกรมจากคอมพิวเตอร์

เมื่อผู้ใช้ต้องการจะรับโปรแกรมซึ่งเป็นการรับโปรแกรมจากคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม ไมโครเมาส์จทำการหยุดเพื่อรอรับโปรแกรมจากคอมพิวเตอร์ จากนั้นผู้ใช้จะต้องทำการรันโปรแกรม เลือก HEX ไฟล์ที่จะทำการส่งจากคอมพิวเตอร์จากนั้นก็คอมพิวเตอร์ก็จะทำการส่งโปรแกรมมายังไมโครเมาส์ โดยดาวโหลดมอนิเตอร์ในไมโครเมาส์จะทำการรับโปรแกรมไปเก็บยังแอดเดส 08000H จนจบโปรแกรมจากนั้นก็จะกลับไปยังเมนูหลักเพื่อรอรับการทำงานต่อไป

หากระหว่างการทำงานต้องการกลับไปยังเมนูหลัก เพียงกด สวิตช์อินเตอร์รัพท์ ไมโครเมาส์ยกเลิกการทำงานและกลับมาทำงานยังเมนูหลัก

6.1.3 การมอนิเตอร์ฮาร์ดแวร์

เมื่อผู้ใช้เลือกการทำงานในข้อนี้ เป็นการตรวจสอบการทำงานของอุปกรณ์ทางด้านฮาร์ดแวร์เป็นส่วนๆ ไม่จำเป็นต้องต่ออุปกรณ์ทั้งหมด เพียงมีคอนโทรลเลอร์บอร์ดและอุปกรณ์ที่ต้องการทดสอบเท่านั้น ไมโครเมาส์จะทำการหยุดรอเพื่อรับคำสั่งจากคอมพิวเตอร์ เมื่อไมโครเมาส์ได้รับคำสั่งแล้วจะทำการจากสั่งไปยังอุปกรณ์นั้นๆให้ทำงานตามที่สั่ง

คำสั่งที่ใช้จะเป็นเพียงคำสั่งง่ายๆ ใช้เพียงเพื่อตรวจสอบว่าฮาร์ดแวร์มีการทำงานที่ผิดพลาดหรือไม่หรือเกิดเหตุขัดข้องอันใด หากต้องการจะกลับเมนูหลักก็เพียงกดสวิตช์อินเตอร์รัพท์เท่านั้น

6.1.4 การกลับไปทำงานต่อ

ผู้ใช้จะใช้คำสั่งนี้ก็ต่อเมื่อผู้ใช้มีการทำงานใดๆที่ค้างอยู่แล้วต้องการกลับไปทำงานนั้นๆต่อ ไมโครเมาส์จะทำการกระโดดไปทำงานยังการทำงานเดิมก่อนจะเข้าเมนูหลัก โดยผู้ใช้จะต้องมีการทำงานอยู่แล้วเท่านั้นจึงจะเกิดผลการทำงาน

การกลับไปทำงานตามเดิมจะย้อนกลับได้เพียงการทำงานก่อนเข้าเมนูหลักครั้งสุดท้ายเท่านั้นถ้ามากกว่านั้นอาจเกิดข้อผิดพลาดในการทำงานได้

6.2 การทดสอบวงจรและอุปกรณ์

ขณะการทดลองจะต้องทำการทดสอบวงจรของอุปกรณ์ว่าทำงานได้หรือไม่ ถ้านำมาประกอบเข้าด้วยกันจะทำงานได้เรือไม่ ซึ่งจะต้องมีการเขียนโปรแกรมทดสอบเป็นขั้นตอน โดยจะใช้ภาษาแอสเซมบลีในการทดสอบ

6.2.1 การทดสอบบอร์ดคอนโทรลเลอร์

การทดสอบบอร์ดไมโครคอนโทรลเลอร์ จะทำการทดสอบโดยแบ่งเป็นการทดสอบย่อย 3 การทดลอง

- การทดสอบไมโครคอนโทรลเลอร์

จากโปรแกรมในภาคผนวก ก. จะเป็นการสั่งให้ไมโครคอนโทรลเลอร์ทำการ ส่งค่า 1 และ 0 สลับกันออกมา โดยจะ หน่วงเวลาในการสลับเป็นเวลา 1 วินาที (ส่วนของการหน่วงเวลาดูที่ท้ายบท) ซึ่งเราจะตรวจสอบโดยนำลอจิกโพรบมาตรวจสอบ หรือต่อ LED มาตรวจสอบจะเห็นสัญญาณออกมาเป็น LO และ HI สลับกันที่ทุกพอร์ตของไมโครคอนโทรลเลอร์ ลองทดสอบสวิตช์รีเซตด้วย

ถ้าการแสดงผลไม่เป็นตามนี้เกิดจากไมโครคอนโทรลเลอร์ไม่ทำงาน อาจเกิดจากไมโครคอนโทรลเลอร์เสีย หรือวงจรมีปัญหาขึ้นได้

ปัญหาและการแก้ไข

ไม่เกิดปัญหา ในการทดลองขั้นตอนนี้

- การทดสอบการทำงานของหน่วยความจำโปรแกรมภายนอก

ในการทดสอบนี้จะทดสอบหน่วยความจำว่ารับค่าและอ่านค่าระหว่างหน่วยความจำกับไมโครคอนโทรลเลอร์ได้หรือไม่ การทดสอบจะทดสอบโดยให้เก็บค่าและอ่านค่าเป็นหน่วยความจำแบบข้อมูล

โปรแกรมข้างต้นจะทำการใส่ข้อมูล 0AAH เข้าไปยังหน่วยความจำตั้งแต่แอดเดส 08000H-0EFFFH เมื่อใส่ครบจะตรวจสอบข้อมูลที่ใส่ลงไป โดยดึงข้อมูลมาจากหน่วยความจำที่ใส่เข้าไปมาแสดงที่พอร์ต P1 ทีละแอดเดส และดูว่าข้อมูลนั้นเท่ากับ 0AAH หรือไม่ ถ้าเท่าก็จะตรวจสอบแอดเดสอื่นจนจบ ถ้าไม่มีการผิดพลาดจะแสดงข้อมูล 0FFH ออกมาที่พอร์ต P1 ถ้าไม่เท่าจะแสดงค่า 0F0H ออกมาทาง พอร์ต P1 เพื่อง่ายในการตรวจสอบ

ถ้าการแสดงผลไม่เป็นตามนี้เกิดจาก อาจเกิดจากไมโครคอนโทรลเลอร์หรือ 62256 เสียหรือไม่ทำงาน หรือวงจรมีปัญหาขึ้นได้

ปัญหาและการแก้ไข

ไม่เกิดปัญหา ในการทดลองขั้นตอนนี้

- การทดสอบการทำงานของ 8255

ในการทดสอบนี้จะทดสอบ8255 ว่ารับติดต่อกับหน่วยไมโครคอนโทรลเลอร์ได้ได้หรือไม่ การทดสอบจะทดสอบโดยให้ส่งค่าค่าไปแสดงยัง 8255

โปรแกรมในภาคผนวก ก. จะเป็นการสั่งให้ไมโครคอนโทรลเลอร์ทำการ ส่งค่า 1 และ 0 สลับกันออกมา โดยจะ หน่วงเวลาในการสลับเป็นเวลา 1 วินาที ออกมาจากพอร์ต ของ 8255 ทั้ง 3 พอร์ต ซึ่งเราจะตรวจสอบโดยนำลอจิกโพรบมาตรวจสอบ หรือต่อ LED มาตรวจสอบจะเห็นสัญญาณออกมาเป็น LO และ HI สลับกันที่ทุกพอร์ตของ8255 ลองทดสอบสวิตช์ รีเซตด้วย

ถ้าการแสดงผลไม่เป็นตามนี้เกิดจาก อาจเกิดจากไมโครคอนโทรลเลอร์หรือ 8255 เสียหรือไม่ทำงาน หรือวงจรมีปัญหาขึ้นได้

ปัญหาและการแก้ไข

ไม่เกิดปัญหา ในการทดลองขั้นตอนนี้

6.2.2 การทดสอบบอร์ดขับสเตปปิ้งมอเตอร์

ในการทดสอบนี้จะทดสอบว่าบอร์ดขับสเตปปิ้งมอเตอร์รับค่าเพื่อควบคุมมอเตอร์ได้ตามที่ต้องการหรือไม่

โปรแกรมในภาคผนวก ก. การควบคุมมอเตอร์ในโปรแกรมนี้จะเขียนโดยใช้ไทม์เมอร์ในโหมด 2 ออโต้รีโหลด ของMCS-51 ในการกำเนิดความถี่ที่จะส่งให้บอร์ดขับสเตปปิ้งมอเตอร์ ในการควบคุมความถี่เพื่อกำหนดความเร็ว โดยสามารถกำหนดที่ TL1,TH1,M_L_LIMIT และ M_R_LIMIT โดยที่ M_L_ LIMIT และ M_R_LIMIT จะเป็นจำนวนเท่าความเร็วของ TL1

ถ้าการทดลองถูกต้อง คือมอเตอร์จะหมุนสวนทางกันโดยมีความเร็วเท่ากัน เราสามารถปรับการขับเฟสได้โดยใช้ ดิพสวิตช์ปรับ

ปัญหาและการแก้ไข

ในการทดลองเกิดความผิดพลาดคือ ในช่วงแรกมอเตอร์หมุนได้อย่างถูกต้องเป็นระยะเวลาประมาณ 2 นาที หลังจากนั้นจะมอเตอร์จะหมุนแล้วหยุดสลับกันไป เมื่อตรวจสอบดูพบว่า 7805 ซึ่งเป็นเรคกูเรเตอร์สามารถจ่ายกระแสได้ 1 A เกิดการทำงานผิดพลาดเพราะว่ามีความร้อนเกิดขึ้น เนื่องจากไม่สามารถจ่ายกระแสได้พอ ต้องปิดสวิตช์ไมโครเมาส์ซักพักเพื่อให้เรคกูเรเตอร์เย็นลงจึงทำงานต่อไปได้อย่างถูกต้อง โดยมอเตอร์ 2 ตัว แต่ละตัวกินกระแส 0.8 A

วิธีการแก้ปัญหาโดย

- เปลี่ยนเรคกูเรเตอร์ที่จ่ายกระแสได้มากกว่านี้
- เพิ่มอุปกรณ์ทำความเย็นให้เรคกูเรเตอร์

6.2.3 การทดสอบบอร์ดเซนเซอร์และบอร์ดเปรียบเทียบความต่างศักย์

โปรแกรมในภาคผนวก ก. ในการทดสอบนี้จะทดสอบเซนเซอร์และบอร์ดคอนโทรลว่าส่งข้อมูลได้หรือไม่ได้หรือไม่ การทดสอบจะทดสอบโดยให้ส่งค่าค่าไปแสดงยัง 8255

ปัญหาและการแก้ไข

ผลที่ได้จากการทดลองคือ ค่าที่ส่งมาจากพอร์ต B ของ 8255 จะเป็นไปตามค่าที่ได้จากเซนเซอร์ที่ส่งผ่านบอร์ดเปรียบเทียบเข้ามายังพอร์ต A

ไม่เกิดปัญหา ในการทดลองขั้นตอนนี้

6.2.4 การทดสอบบอร์ดLCD

เป็นการส่งข้อมูลไปแสดงยัง LCD

เป็นการฟังก์ชั่นการทำงานของ LCD ที่ใช้แค่ 4 บิท ทั้งหมด ผลการทกลองที่ถูกต้องคือ LCD จะพิมพ์คำว่า “Micromouse for Education. ” ออกมาที่ LCD

ปัญหาและการแก้ไข

ไม่เกิดปัญหา ในการทดลองขั้นตอนนี้

6.3 สรุป

เปรียบเทียบกับไมโครเมาส์ที่มีวางขาย โดยแสดงเฉพาะ การทำงานที่แตกต่าง

- ไมโครคอนโทรลเลอร์

Microcontroller NEC 78310A

ออกแบบมาสำหรับไมโครเมาส์โดยเฉพาะมีความสามารถดังนี้

- ทำงานที่ความถี่ 6 MHz ใช้ Crystal 12 MHz
- RAM 128 Byte
- ALU 8/16 Bit
- I/O Line 24 Bit
- 16 Bit Timer/Counter 2 ชุด
- Analog to Digital 4 ชุด
- พอร์ตสื่อสารอนุกรม (UART)

ข้อเสีย

ไม่มี ROM ภายในจึงต้องต่อ EPROM ภายนอกทำให้เปลืองเนื้อที่และไม่สะดวก

มีราคาแพง

ข้อดี

มีการทำงานที่เร็วถึง 6 Mhz ที่ Crystal 12 Mhz

และมี Analog to Digital 4 ชุด

	AT89C51	**AT89S8252**
Operating Frequency	1 MHz	1 MHz (cıystal 12Mhz)
ROM	4K	8K Byte
RAM	128 Byte	256 Byte
I/O Line	2	32
16 Bit Timer/Counter	2	3
Interrupt Source	6	6
SPI	-	มี
2 K byte EEPROM	-	มี

ข้อดี

หาซื้อ ได้ง่าย

สามารถใช้ได้ทั้งภาษา C และ แอสเซมบี

มีราคาถูก

- วงจรขับสเตปปิ้งมอเตอร์

L297,L298

ข้อดี

ประหยัดสายสัญญาณที่ใช้

ข้อเสีย

ต้องใช้ชิป 2 ตัวต่อ 1 มอตอร์ ทำให้เกิดความไม่สะดวก

UCN5804B

ข้อดี

ไม่ค่อยสิ้นเปลืองกำลังงาน

มีความเที่ยงตรงเนื่องจากมีโหมดในการเลือกเฟสที่ใช้ในการทำงาน

มีการตรวจจับอุณหภูมิ

ประหยัดสายสัญญาณที่ใช้

ข้อเสีย

อุปกรณ์มีราคาแพง (375 บาท)

6.4 แนวทางแก้ไข

- ค่าใช้จ่ายในการทำโปรโตไทพ์ มีราคาต่อหน่วยสูง เพราะต้องซื้ออุปกรณ์ทีไม่มาก
- วงจร อุปกรณ์ทางด้าน ฮาร์ดแวร์ ไม่มีความเสถียร ซึ่งเกิดจาก สัญญาณรบกวนของกระแสไฟ หรือ สัญญาณรบกวน ที่เกิดจากการหมุนของมอเตอร์ทำให้สเตป ของมอเตอร์ผิดพลาด
- ลักษณะของสนามเช่น สนามไม่เรียบ การมีทางเลี้ยวมากๆ ลักษณะของแสง
- การโหลดโปรแกรมผ่านทางพอร์ตอนุกรมไม่มีความถูกต้องพอทำให้เกิดการผิดพลาด ในการกระโดด ไปทำงาน

6.5 ทางในการพัฒนาต่อไป

- เพิ่มชุดคำสั่งในการทำงานทางซอร์ฟแวร์และควบคุมฮาร์ดแวร์ให้ทำงานได้หลากหลายมากขึ้น
- เพิ่มอุปกรณ์ทางด้านฮาร์ดแวร์ที่สามารถนำมาเปลี่ยน เพื่อสามารถควบคุมอุปกรณ์ได้หลากหลาย
- สามารถที่จะลดสัญญาณรบกวน ที่เกิดขึ้น ทำให้วงจรมีความเสถียร มีการทำงานที่เที่ยงตรงมากขึ้น

ภาคผนวก ก.

- โปรแกรมทดสอบไมโครคอนโทรลเลอร์

```
ORG 0000H
        JMP  INIT_51
        ORG 0040H
;----------------TEST 51-------------------------;
INIT_51:
        MOV    P0,A
        MOV    P1,A
        MOV    P2,A
        MOV    P3,A
TEST_51:
        XRL    A,#0FFH
        MOV    P0,A
        MOV    P1,A
        MOV    P2,A
        MOV    P3,A
        ACALL DELAY_1s
SJMP    TEST_51
```

- โปรแกรมทดสอบหน่วยความจำภายนอก

```
        ORG 0000H
MAIN:
;----------------TEST MEM----------------------;
        MOV    DPTR,#08000H
        MOV    A,#0AAH
        MOV    R0,000H
        MOV    P1,000H
INIT_MEM:
        MOVX  @DPTR,A
        INC    DPTR
        MOV    R0,DPL
        CJNE   R0,#000H, INIT_MEM
```

```
        MOV    R0,DPH
        CJNE   R0,#0F0H, INIT_MEM
        MOV    DPTR,#08000H
        MOV    R0,000H
CHECK_MEM:
        MOVX   A,@DPTR
        MOV    P1,A
        CJNE   A,#0AAH, ERROR
NEXT_MEM:
        ACALL  DELAY_1s
        INC    DPTR
        MOV    R0,DPL
        CJNE   R0,#000H,CHECK_MEM
        MOV    R0,DPH
        CJNE   R0,#0F0H, CHECK_MEM
        MOV    P1,0FFH
        JMP    $
ERROR:
        MOV    P1,0F0H
        JMP    $
```

- โปรแกรมทดสอบการทำงานของ 8255

```
ORG 0000H
MAIN:
        MOV    A,#080H          ;SET MODE 8255
        MOV    DPTR,#0F003H  ;SET MODE 8255  ;PORT A,B,C =OUTPUT PORT
        MOVX   @DPTR,A       ;SET MODE 8255
INIT_8255:
        MOV    A,#0FFH
        MOV    DPTR,#0F000H ;PORT A

        MOVX   @DPTR,A

        MOV    DPTR,#0F001H ;PORT B
```

```
        MOVX  @DPTR,A

        MOV   DPTR,#0F002H ;PORT C
        MOVX  @DPTR,A

TEST_8255:
        XRL   A,#0FFH
        MOV   DPTR,#0F000H ;PORT A
        MOVX  @DPTR,A

        MOV   DPTR,#0F001H ;PORT B
        MOVX  @DPTR,A

        MOV   DPTR,#0F002H ;PORT C
        MOVX  @DPTR,A

        ACALL DELAY_1s
        SJMP  TEST_8255
```

- โปรแกรมทดสอบสเตปปิ้งมอเตอร์

```
M_R_FLAG    EQU   20H    ;BIT ADDRESS
M_L_FLAG    EQU   22H    ;BIT ADDRESS

M_L_COUNT   EQU   30H
M_R_COUNT   EQU   32H

M_L_LIMIT   EQU   34H
M_R_LIMIT   EQU   36H
      ORG   0000H
      JMP   start

      ORG   000BH
      JMP   MOTOR
```

```
        ORG     0040H
START: ACALL DELAY_1s
        MOV     P1,#000H
        MOV     DPTR,#0F003H  ;INIT    8255     A OUTPUT
        MOV     A,#082H       ;B INPUT
        MOVX    @DPTR,A       ;C OUTPUT

        MOV     DPTR,#0F000H  ; OUT A 0F0H;
        MOV     A,#0F0H
        MOVX    @DPTR,A

        MOV     DPTR,#0F002H  ; OUT A 0F0H;
        MOV     A,#00FH
        MOVX    @DPTR,A

        MOV     TMOD,#00000010B      ;SET TIMER 0 MODE 2 AUTO RELOAD
        SETB    EA
        SETB    ET0
        CLR     ES      ;DISABLE SERIAL INTERUPT

        MOV     TL1,#0FEH       ;COUNTER
        MOV     TH1,#0FEH       ;RELOAD

        MOV     M_R_FLAG,#000H
        MOV     M_L_FLAG,#000H

        MOV     M_L_COUNT,#000H
        MOV     M_R_COUNT,#000H

        MOV     M_L_LIMIT,#020
        MOV     M_R_LIMIT,#020
        SETB    TR0     ;INIT    TIMER
        SJMP    $
;---------------MOTOR----------------
```

```
MOTOR:
        PUSH   ACC
        PUSH   PSW

CHK_M_L:
        INC    M_L_COUNT
        MOV    A,M_L_COUNT
        CJNE   A,M_L_LIMIT,CHK_M_R
        CPL    P1.0
        MOV    M_L_COUNT,#000H
CHK_M_R:
        INC    M_R_COUNT
        MOV    A,M_R_COUNT
        CJNE   A,M_R_LIMIT,END_T0
        CPL    P1.5
        MOV    M_R_COUNT,#000H
END_T0:
        POP    PSW
        POP    ACC
        RETI
```

- ทดสอบบอร์ดเซนเซอร์และบอร์ดเปรียบเทียบความต่างศักย์

```
ORG 0000H
        JMP    MAIN
        ORG 0040H
MAIN:
        MOV    A,#082H          ;SET MODE 8255
        MOV    DPTR,#0F003H  ;SET MODE 8255
        MOVX   @DPTR,A          ;SET MODE 8255
        SJMP   T_8255;TEST
;----------TEST 8255----------;
T_8255:
        MOV    DPTR,#0F001H ;PORT B
        MOVX   A,@DPTR
```

```
        MOV     DPTR,#0F002H ;PORT C
        MOVX  @DPTR,A


                SJMP    T_8255
```

- โปรแกรมทดสอบ LCD

```
PosMode                 EQU     40H
PortLCD         EQU     P1
ClearDispCmd    EQU     01H
ReturnHomeCmd   EQU     02H
EntryModeCmd    EQU     06H             ;Entry mode set Inc & cursor right
DispOnOffCmd    EQU     0FH             ;Display On/Off set on & cursor Blink
DispShiftCmd    EQU     14H             ;Shift char on screen to left 1 pos
FunctionCmd81   EQU     33H
FunctionCmd82   EQU     32H
FunctionCmd4    EQU     28H             ;4 bits interfaces, 4 Lines, 5X7 dots
SetDDRam1Cmd    EQU     80H             ;address 00H   16 X 4
SetDDRam2Cmd    EQU     0C0H            ;address 40H
SetDDRam3Cmd    EQU     90H             ;address 10H
SetDDRam4Cmd    EQU     0D0H            ;address 50H
SetDDRam5Cmd    EQU     0F0H            ;address 70H

LcdRS           EQU     P1.0
LcdR_W          EQU     P1.1
LcdEna          EQU     P1.2
LCDEnable       EQU     00000100b
;-------------------------------------------------------
;constant for program run
LCDFlag         EQU     40h
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ORG     0000h
        jmp     Main
        ORG     0040h
Main:           call    Init                    ;init and
```

```
                call    ScrnTitle
                jmp     $
;-----------------------------------------------
Init:       mov   A,#00h
                orl     a,#LCDEnable
                mov   PortLCD,A
          call  InitLCD
                call    ClearLCD
                ret
;===============================================================
; LCD module
;-------------------------------------
;Show Screen Title on screen
ScrnTitle:
          call  ClearLcd
          call  CursorFront
          mov   dptr,#ScrnTitle1
          call  ShowScreen
          call  CursorHide
          ret
;****************** Routine For LCD ******************
;-------------------------------------
;print string on screen all position
; DPTR = address of data to show on screen
ShowScreen:   push  dph
          push  dpl
          push  acc
          push  0

          mov   a,#0
          movc  a,@a+dptr     ;get counter
          inc   a
          mov   r0,a
```

```
        call   ClearLcd
        call   LcdString      ;write first line
        mov    a,dpl          ;add dptr with 17, Next string
        add    a,r0
        mov    dpl,a
        mov    a,dph
        addc   a,#0
        mov    dph,a

        mov    a,#SetDDRam2Cmd
        call   CursorMove
        call   LcdString      ;write second line

        pop    0
        pop    acc
        pop    dpl
        pop    dph
        ret
;---------------------------------------
; Extend delay for clear lcd and return home command
; delay for > 1.64 ms (40us done on lcdcommand)
ExtendDelay:   push   0
        push   1
        mov    r0,#0D0H       ;0A0
ExtendDelay1:  mov    r1,#80H
        djnz   r1,$
        djnz   r0,ExtendDelay1
        pop    1
        pop    0
        ret
;---------------------------------------
;Show key on Lcd
;A = Key
; not destroy acc!!!!
```

```
ShowKey:      push   acc
              push   0
              push   dph
              push   dpl
              rl     a            ;multiply by 2 for locate data
              mov    r0,a
              inc    r0
              mov    dptr,#TestStr
ShowKeyLoop:  inc    dptr
              djnz   r0,ShowKeyLoop  ;calculate address
              call   LcdString
              pop    dpl
              pop    dph
              pop    0
              pop    acc
              ret
;--------------------------------------
;Show key on Lcd but 0a is 11
;A = Key
; not destroy acc!!!!
ShowKey_:     push   acc
              push   0
              push   dph
              push   dpl
              cjne   a,#0Ah,ShowNormalKey
              mov    dptr,#TestStr1
              call   LcdString
              mov    dptr,#TestStr0
              call   LcdString
              jmp    EndShowKey_
ShowNorMalKey: rl    a            ;multiply by 2 for locate data
              mov    r0,a
              inc    r0
              mov    dptr,#TestStr
```

```
ShowKeyLoop_:   inc    dptr
          djnz   r0,ShowKeyLoop_ ;calculate address
          call   LcdString
EndShowKey_:   pop    dpl
          pop    dph
          pop    0
          pop    acc
          ret
;----------------------------------------
;Show data on Lcd 00-FF
;A = data
ShowData:      push   0
          mov    r0,a
          anl    a,#0F0H
          swap   a
          call   ShowKey
          mov    a,r0
          anl    a,#0FH
          call   ShowKey
          pop    0
          ret
;----------------------------------------
;Show num on Lcd 0 - 255
;A = number
ShowNum:       push   0
          push   acc
          mov    b,#10
          div    ab
          mov    r0,b
          mov    b,#10
          div    ab
          jz     LessThan100
          call   ShowKey
          mov    a,b
```

```
        call   ShowKey
        jmp    LessThan10
LessThan100:  mov    a,b
        jz     LessThan10
        call   ShowKey
LessThan10:   mov    a,r0
        call   ShowKey
        pop    acc
        pop    0
        ret
;----------------------------------------
;Clear LCD Display
ClearLcd:
        push   acc
        mov    a,#ClearDispCmd
        call   LcdCommand
        call   ExtendDelay
        pop    acc
        ret
;----------------------------------------
;Move Cursor to any position
; A = Position of cursor
CursorMove:   push   acc
        orl    a,#80h
        call   LcdCommand
        pop    acc
        ret
;----------------------------------------
;Move Cursor to first address
CursorFront:  push   acc
        mov    a,#SetDDRam1Cmd
        call   LcdCommand
        pop    acc
        ret
```

```
;----------------------------------------
;Move Cursore to nineth address
CursorBack:    push   acc
               mov    a,#SetDDRam3Cmd
               call   LcdCommand
               pop    acc
               ret
;----------------------------------------
;Move Cursor out of screen
CursorHide:    push   acc
               mov    a,#SetDDRam5Cmd
               call   LcdCommand
               pop    acc
               ret
;----------------------------------------
;Move Cursor to first line
CursorLine1:   push   acc
               mov    a,#SetDDRam1Cmd
               call   LcdCommand
               pop    acc
               ret
;----------------------------------------
;Move Cursor to line 2
CursorLine2:   push   acc
               mov    a,#SetDDRam2Cmd
               call   LcdCommand
               pop    acc
               ret
;----------------------------------------
;Move Cursor to line 3
CursorLine3:   push   acc
               mov    a,#SetDDRam3Cmd
               call   LcdCommand
               pop    acc
```

```
        ret
;----------------------------------------
;Move Cursor to line 4
CursorLine4:  push  acc
        mov   a,#SetDDRam4Cmd
        call  LcdCommand
        pop   acc
        ret
;----------------------------------------
;Write one byte to LCD at current position
;A = Data in
LcdByte:    call  LcdData
        ret
;----------------------------------------
;Write String <= 8 bytes to LCD start at current position
;first byte is length --> '06','Hello!'
;DPTR = Address of data
LcdString:    push  acc
        push  0
        push  1

        mov   r1,#0
        mov   a,r1
        movc  a,@a+dptr     ;get first byte as counter
        mov   r0,a          ;save counter
LcdStr1:    inc   r1          ;next data
        mov   a,r1
        movc  a,@a+dptr
        call  Lcddata
        djnz  r0,LcdStr1

        pop   1
        pop   0
        pop   acc
```

```
        ret
;---------------------------------------
InitLcd:    push  acc
        call  LcdPowerOnReset
        mov   a,LCDFlag
        cjne  a,#78h,NotBeenInit
        jmp   HasBeenInit
NotBeenInit:  call  Interface4Bits
        mov   LCDFlag,#78h
HasBeenInit:
        mov   a,#EntryModeCmd
        call  LcdCommand
            mov   a,#DispOnOffCmd
        call  LcdCommand
        call  ClearLcd
        pop   acc
        ret
;****  Send command to interface 4 bit **********
Interface4Bits: push  acc
        mov   a,#FunctionCmd4
        anl   a,#0F0h       ;send high nibble only;
       orl      a,#LCDEnable
        mov   PortLCD,a     ;send command on high nibble only
        clr   LcdR_W        ;write to LCD
        clr   LcdRS         ;write to command register
        call  EnablePulse
        call  LcdDelay      ;Delay > 4.1ms
        pop   acc
        ret
;****  Send command to LCD ***
LcdCommand:   push  acc
        push  0
        ;4 bits connection, send high nible first
        mov   r0,a          ;save command
```

```
        ;send high nibble
        mov    a,r0
        anl    a,#0F0h        ;send high nibble first
               orl        a,#LCDEnable
        mov    PortLCD,a      ;send command on high nibble first
        clr    LcdR_W         ;write to LCD
        clr    LcdRS          ;write to command register
        call   EnablePulse
;;      call   LcdDelay       ;Delay > 4.1ms

        ;send low nibble
        mov    a,r0
        anl    a,#0Fh         ;send low nibble
        swap   a              ;bring data to high nibble position
               orl        a,#LCDEnable
        mov    PortLCD,a      ;send command on low nibble
        clr    LcdR_W         ;write to LCD
        clr    LcdRS          ;write to command register
        call   EnablePulse
        call   LcdDelay       ;Delay > 4.1ms

        pop    0
        pop    acc
        ret
;**** Send data to LCD ***
LcdData:    push   acc
        push   0
        ;4 bits connection, send high nible first
        mov    r0,a           ;save data

        ;send high nibble
        mov    a,r0
        anl    a,#0F0h        ;send high nibble first
```

```
                orl     a,#LCDEnable
        mov     PortLCD,a       ;send data on high nibble first
        clr     LcdR_W          ;write to LCD
        setb    LcdRS           ;write to data register
        call    EnablePulse
;;      call    LcdDelay        ;Delay > 4.1ms

        ;send low nibble
        mov     a,r0
        anl     a,#0Fh          ;send low nibble
        swap    a               ;bring data to high nibble position
        orl     a,#LCDEnable
        mov     PortLCD,a       ;send data on low nibble
        clr     LcdR_W          ;write to LCD
        setb    LcdRS           ;write to data register
        call    EnablePulse
        call    LcdDelay        ;Delay > 4.1ms

        pop     0
        pop     acc
        ret
;***** Delay for LCD Power On Reset > 15 ms ******
LcdPowerOnReset:
        push    0
        push    1
        mov     r0,#0
LcdPOR1:    mov     r1,#30h     ;20h
LcdPOR2:    nop
        nop
        djnz    r1,LcdPOR2
        djnz    r0,LcdPOR1
        pop     1
        pop     0
        ret
```

```
;**** Generate enable pluse **** active low
EnablePulse:   push   0
                 nop
          clr    LcdEna        ;Fall enable signal
                 nop
          setb   LcdEna        ;Riase enable signal
          pop    0
          ret
;----------------------------------------
;***  Delay for each command to LCD > 40 us ****
LcdDelay:      push   0
          push   1
          mov    r0,#60        ;Dealy > 40 us ;;40
          djnz   r0,$
          pop    1
          pop    0
          ret
;----------------------------------------
DelayTemp:     push   0
          push   1
          push   2
          mov    r0,#7         ;5
DelayTempLoop2: mov    r1,#0
DelayTempLoop1: mov    r2,#0
          djnz   r2,$
          djnz   r1,DelayTempLoop1
          djnz   r0,DelayTempLoop2
          pop    2
          pop    1
          pop    0
          ret
;----------------------------------------
;Delay a sec
DelayASec:     push   0
```

```
        push    1
        push    2
        mov     r0,#00h
LoopASec2:  mov     r1,#00h
LoopASec1:  mov     r2,#03h
        djnz    r2,$
        djnz    r1,LoopASec1
        djnz    r0,LoopASec2
        pop     2
        pop     1
        pop     0
        ret
;=========================================
;Message defind
ScrnTitle1:   DB    16,' Micromouse for '
ScrnTitle2:   DB    16,'  Education   '
TestStr:      DB    01
TestStr0:     DB    01,'0'
TestStr1:     DB    01,'1'
TestStr2:     DB    01,'2'
TestStr3:     DB    01,'3'
TestStr4:     DB    01,'4'
TestStr5:     DB    01,'5'
TestStr6:     DB    01,'6'
TestStr7:     DB    01,'7'
TestStr8:     DB    01,'8'
TestStr9:  DB    01,'9'
TestStrA:  DB    01,'A'   ;10
TestStrB:  DB    01,'B'   ;11
TestStrC:  DB    01,'C'   ;12
TestStrD:  DB    01,'D'   ;13
TestStrE:  DB    01,'E'   ;14
TestStrF:  DB    01,'F'   ;15
TestStrG:  DB    01,'G'   ;16
```

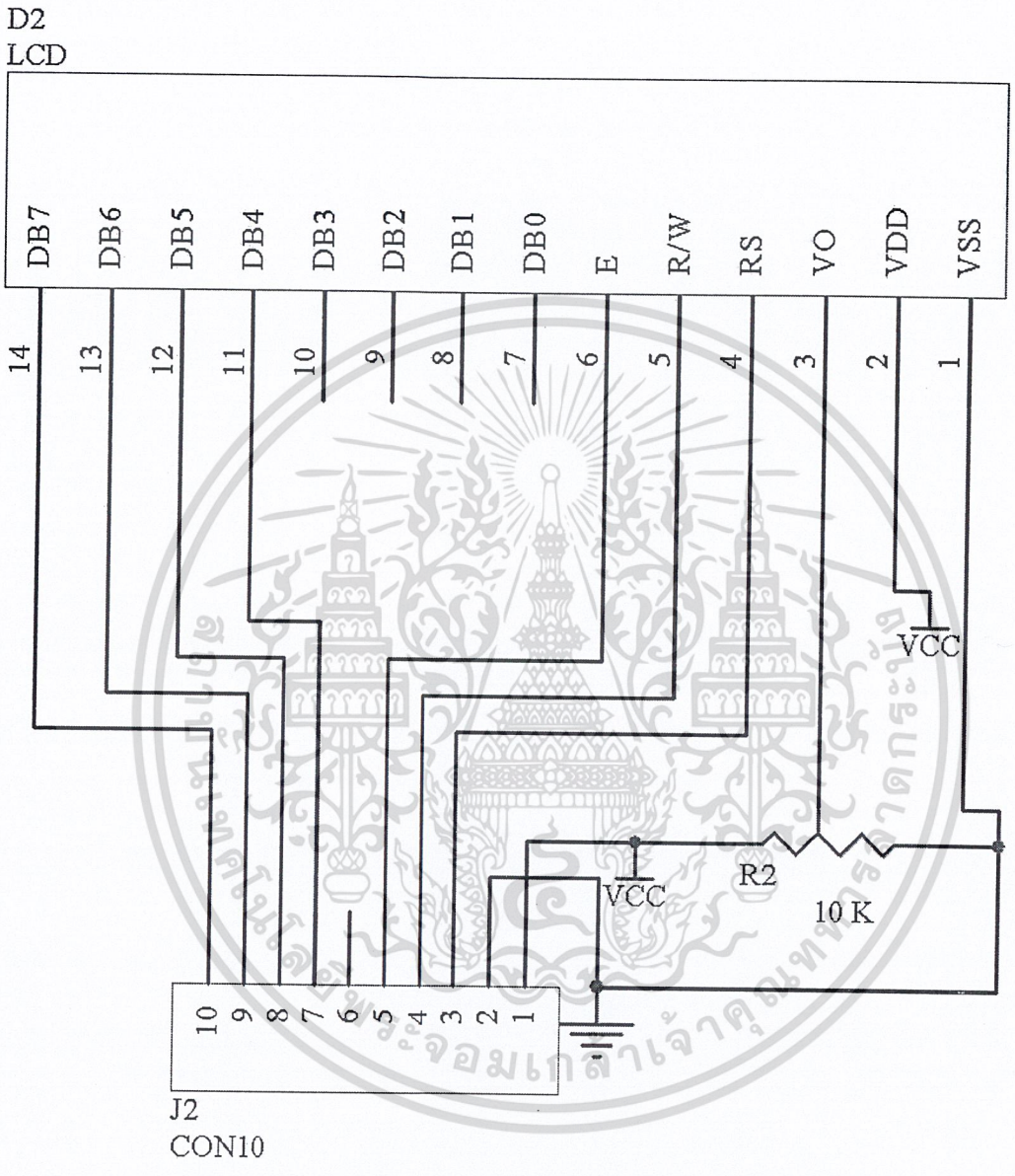
```
TestStrH: DB    01,'H'   ;17
TestStrI:  DB    01,'I'   ;18
TestStrJ:  DB    01,'J'   ;19
TestStrK:  DB    01,'K'   ;20
TestStrL:  DB    01,'L'   ;21
TestStrM:  DB    01,'M'   ;22
TestStrN:  DB    01,'N'   ;23
TestStrO:  DB    01,'O'   ;24
TestStrP:  DB    01,'P'   ;25
TestStrQ:  DB    01,'Q'   ;26
TestStrR:  DB    01,'R'   ;27
TestStrS:  DB    01,'S'   ;28
TestStrT:  DB    01,'T'   ;29
TestStrU:  DB    01,'U'   ;30
TestStrV:  DB    01,'V'   ;31
TestStrW:  DB    01,'W'   ;32
TestStrX:  DB    01,'X'   ;33
TestStrY:  DB    01,'Y'   ;34
TestStrZ:        DB    01,'Z'   ;35
TestStrMinus:    DB    01,'-'
TestStrComma:  DB    01,','
        END
```

- ส่วนการดีเลย์

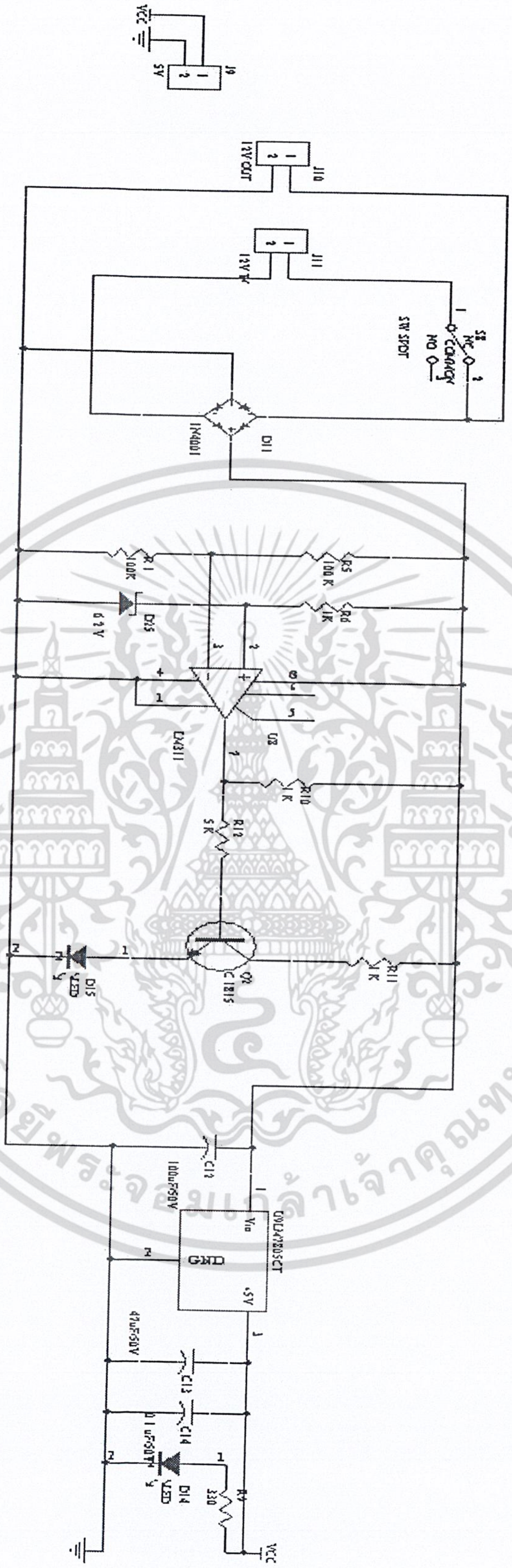
```
;---------DELAY-------------------------------;
DELAY:
        MOV   R7,#002
DELAY_1:
        MOV   R6,#0E6H
DELAY_2:
        NOP
        NOP
        DJNZ   R6,DELAY_2
        DJNZ   R7,DELAY_1
```

```
        RET

DELAY_10ms:
        MOV    R7,#010
DELAY_10ms_1:
        MOV    R6,#0E6H
DELAY_10ms_2:
        NOP
        NOP
        DJNZ   R6,DELAY_10ms_2
        DJNZ   R7,DELAY_10ms_1
        RET

DELAY_100ms:
        MOV    R7,#100
DELAY_100ms_1:
        MOV    R6,#0E6H
DELAY_100ms_2:
        NOP
        NOP
        DJNZ   R6,DELAY_100ms_2
        DJNZ   R7,DELAY_100ms_1
        RET
DELAY_1s:
        MOV    R5,#100
DELAY_1s_1:
        ACALL DELAY_10ms
        DJNZ   R5,DELAY_1s_1
        RET
        END
```

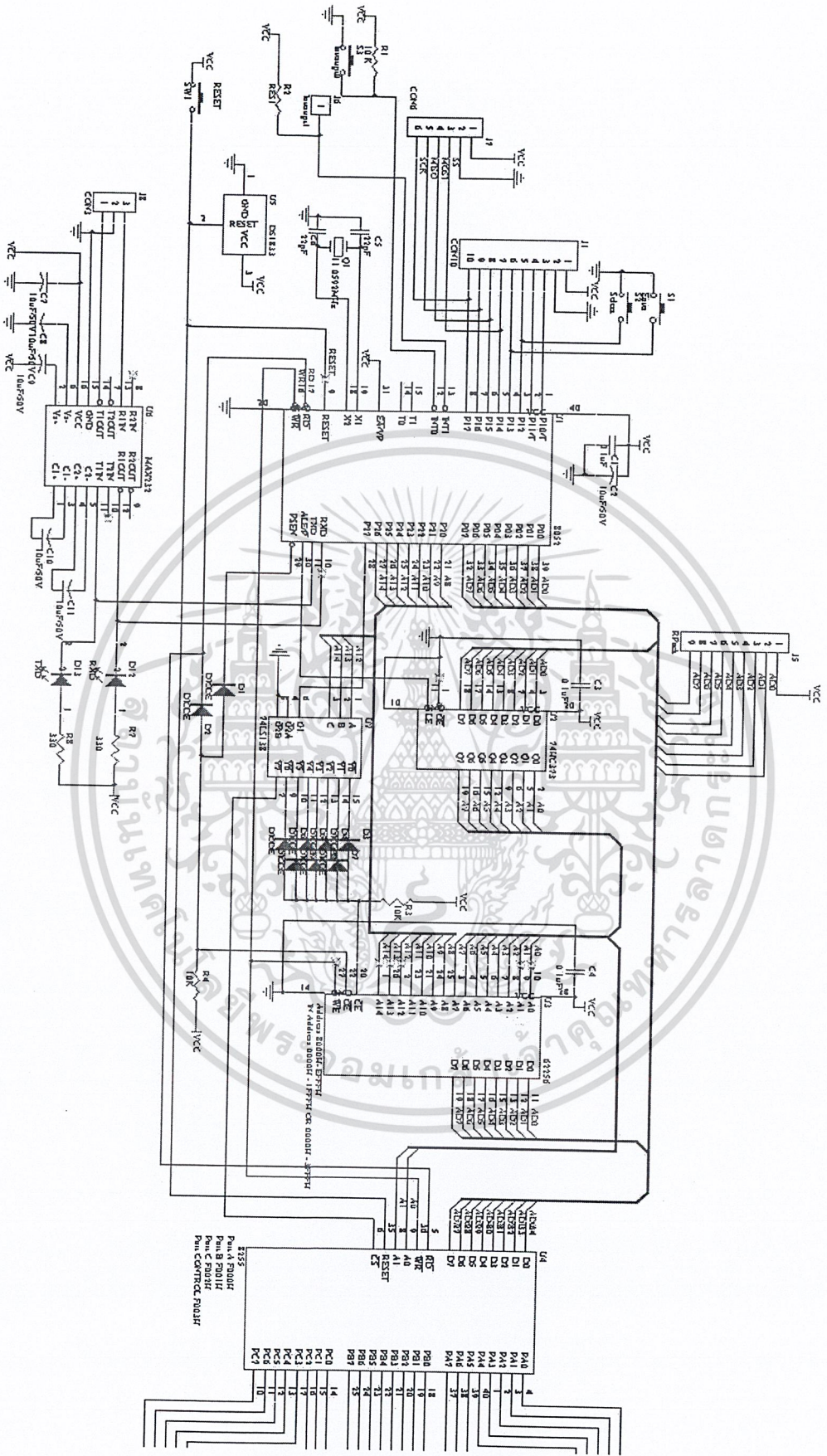
ภาคผนวก ข.



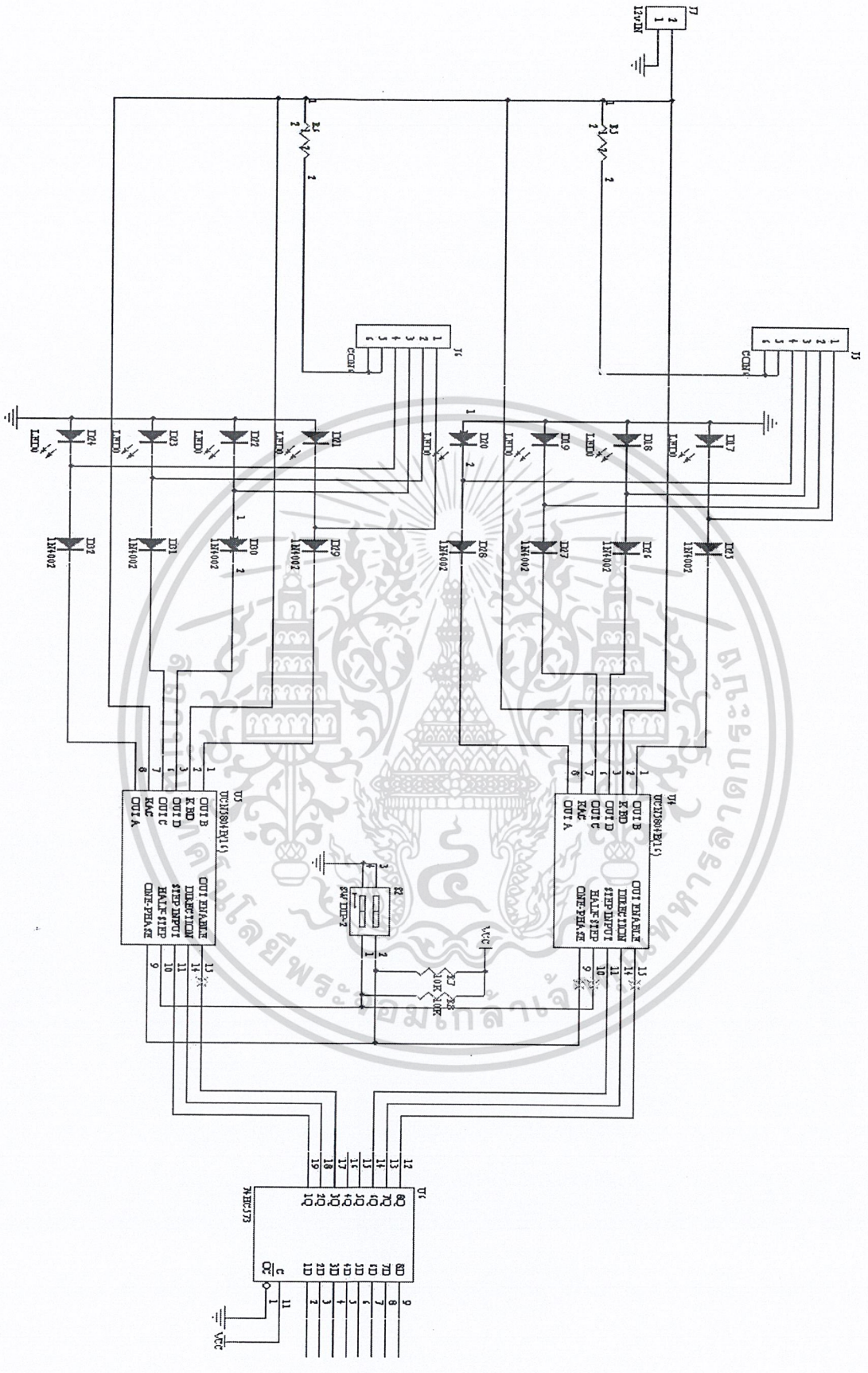
วงจรLCD



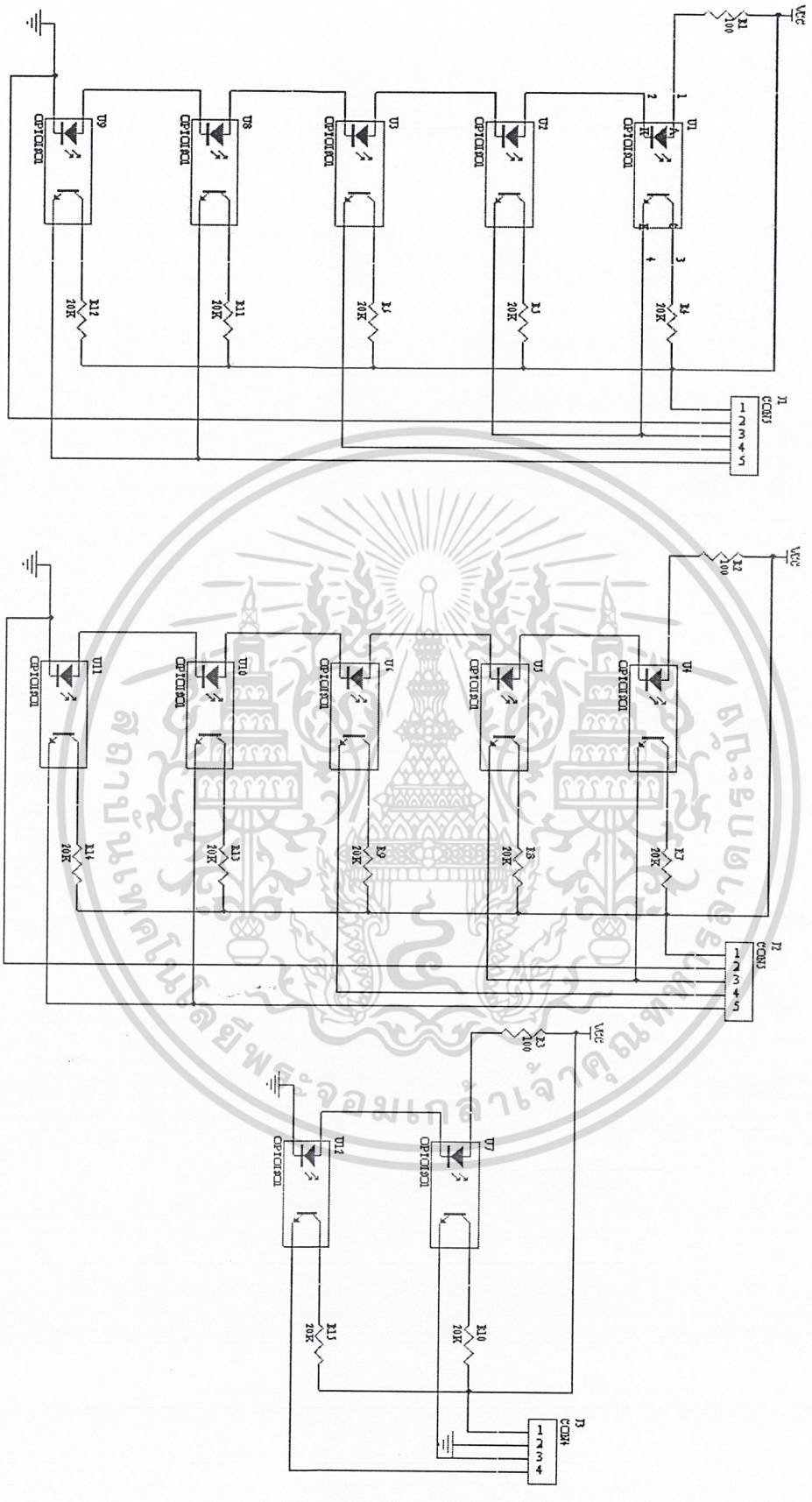
วงจรจ่ายไฟ



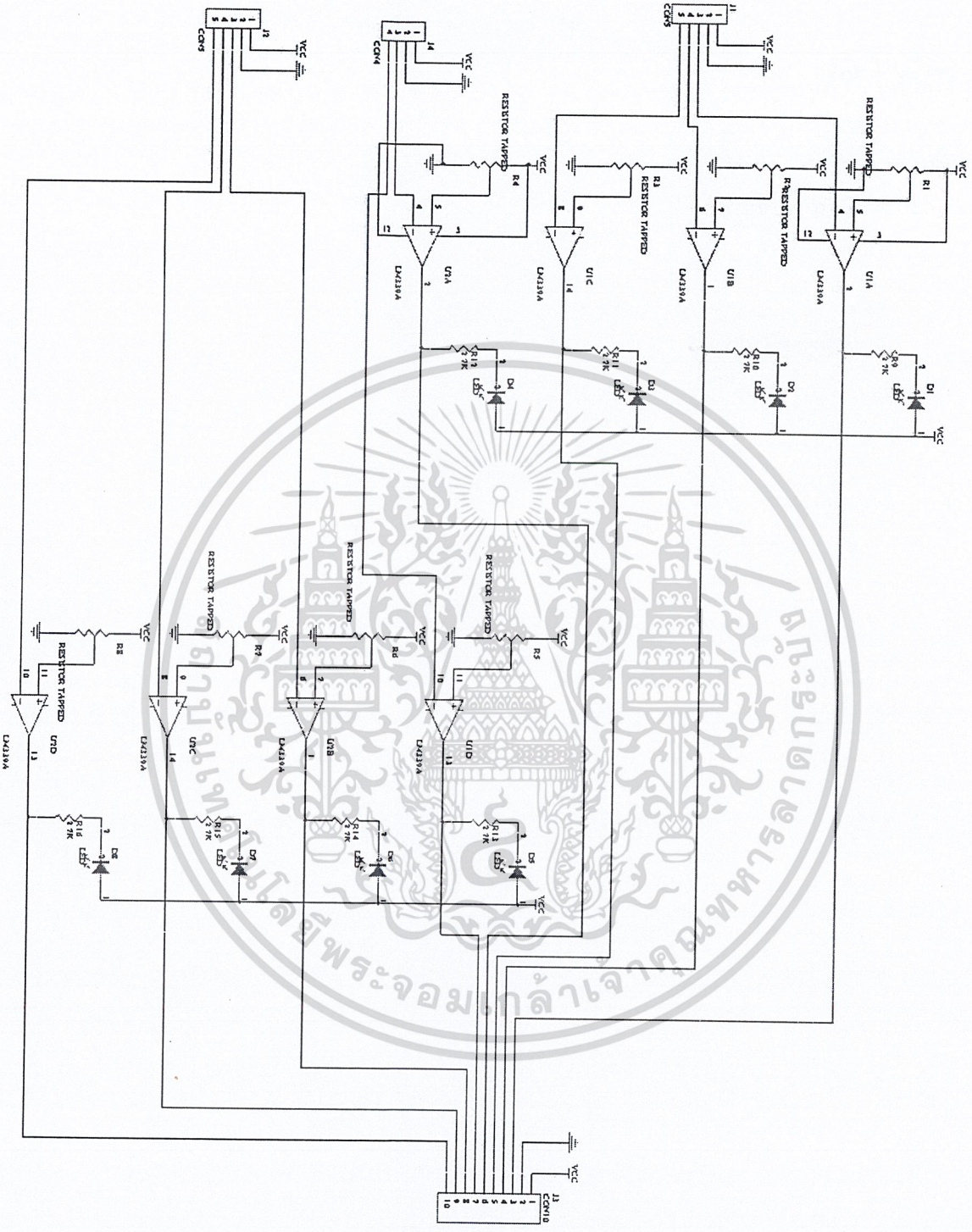
วงจรบอร์ดไมโครคอนทรลเลอร์



วงจรขับสเตปปิ้งมอเตอร์



วงจรเซนเซอร์



วงจรเทียบความต่างศักย์

บรรณานุกรม

[1] สุนทร วิทูสุรพจน์ : “การใช้งานไมโครคอนโทรเลอร์ ตระกูล8051”

[2] ชัยวัฒน์ ลิ้มพรจิตรวิไล, วรพจน์ กรแก้ววัฒนกุล : ”เรียนรู้และปฏิบัติการไมโครคอนโทรเลอร์ MCS-51”

[3] นาย สมชาย อังศุโชติกุล 38013294,นายอดิศักดิ์ โหตระภาวนนท์ 380113300 : ”ปริญญานิพนธ์ไมโครเมาส์” สาขาวิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2540

[4] อ. บัณฑิต จามรภูติ : ”คู่มือการใช้งาน Protel 99”

[5] อ. วรพล ลีลาเกรียรติสกุล , อ. มารุต ตามศรี , พัฒนพล สายกลิ่น : C51 LAB ของ SILA