ไมโครเมาส์

MICROMOUSE

โดย

นายสมชาย	อังศุโชติกุล	38013294
นายอดิศักดิ์	โหตระภวานนท์	38013300

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ไมโครเมาส์

MICROMOUSE

โดย

นายสมชาย อังศุโชติกุล 38013294

นายอดิศักดิ์ โหตระภวานนท์ 38013300

อาจารย์ที่ปรึกษา

ผศ. สมศักดิ์ มิตะถา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ไมโครเมาส์

(MICROMOUSE)

ผู้จัดทำ นายสมชาย อังศุโชติกุล 38013294

นายอดิศักดิ์ โหตระภวานนท์ 38013300

.....อาจารย์ที่ปรึกษา

(ผศ. สมศักดิ์ มิตะถา)

ไมโครเมาส์

สมชาย อังศุโชติกุล
อดิศักดิ์ โหตระภวานนท์
ผศ.สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
ปีการศึกษา 2540

บทคัดย่อ

โครงงานไมโครเมาส์ ได้จัดทำขึ้นเพื่อเป็นต้นแบบเพื่อใช้ในการศึกษาทางด้านหุ่นยนต์อัจฉริยะ โดยมีไมโครคอนโทลเลอร์ 8051 เป็นหน่วยประมวลผลกลาง และใช้โปรแกรมภาษา C ในการเขียนโปรแกรมควบคุมฟังก์ชั้นการทำงาน โดยให้ตัวคอมพายเลอร์เป็นตัวแปลงให้เป็นภาษาเครื่อง ส่งคำสั่งที่ได้ไปยังตัวไมโครเมาส์โดนผ่านทางพอร์ทอนุกรมของเครื่องคอมพิวเตอร์ การออกแบบทดสอบการทำงานได้สร้างสนามแบบเขาวงกตขึ้นมาโดยมีหลักการค้นหาเส้นทางของสนามจากจุดเริ่มต้นไปยังจุดเป้าหมายที่สั้นที่สุด โดยวิธีสั่งให้ไมโครเมาส์วิ่งไปยังทุกๆช่องของสนามเพื่อเก็บรูปแบบของกำแพงจนทั่วทั้งสนามจากนั้นก็จะนำลักษณะสนามที่ได้มาวิเคราะห์หาเส้นทางที่สั้นที่สุด จากนั้นไมโคาเมาส์จะวิ่งจากจุดเริ่มต้นไปยังจุดเป้าหมายโดยมีระยะทางที่สั้นที่สุดและใช้เวลาน้อยที่สุด

ผลที่ได้จากโครงงานนี้คือจะทำให้เราทราบถึงวิธีการเขียนโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ 8051 และได้เรียนรู้การทำงานของตัวคอมไพเลอร์ อีกทั้งได้รู้ถึงการทำงานของฟังก์ต่างๆ ของ ซอฟท์แวร์และฮาร์ดแวร์

MICROMOUSE

Somchai Aungsuchotikul

Adisak Hotrapavanon

Assistance Professor Somsak Mitatha **Advisor**

1997

Abstract

The Micro mouse project is built to be prototype that can be used for studying about smart robot. There is microcontroller 8051,used as Central Processing Unit (CPU) and C language programming is added to control the working function. It can be transfromed to machine language and sent to Micro mouse via serial port by using compiler. Work Testing area for Micromouse can be set as maze so the Micro mouse will move from begining point to destination in all the way that possible and fine the shortest way to arrive destination by itself. After that Micro mouse will move along that way and take the shortest time .

From this project we have learnning about microcontroller 8051 and learn how to operate with C language programming in a programming including other function in software and hardware of computer .

กิตติกรรมประกาศ

การทำโครงงานรวมทั้งการทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี โดยได้รับความช่วยเหลือและได้รับคำแนะนำจากบุคคลหลายท่าน ได้แก่

ท่านอาจารย์สมศักดิ์ มิตะถา เป็นอาจารย์ที่ปรึกษาในการวางแผนงานและคอยกระตุ้นให้โครงดำเนินงานไปในแนวทางที่ถูกต้อง

ท่านอาจารย์อภิเนตร อูนากูล พี่ๆเพื่อนในห้อง ESL ที่คอยห่วงใยและให้กำลังใจเสมอมา

เพื่อน ๆ ห้อง 3Pและน้องห้อง 2P 3D ที่คอยถามไถ่ความก้าวหน้าของโครงงาน ทำให้เกิดความกระตือรือร้นมากขึ้น

บริษัท ศิลารีเสริช์ ที่ได้ให้ตัวคอมไพเลอร์ที่ดีเยี่ยมของโครงงานนี้

สุดท้ายขอขอบพระคุณ คุณพ่อคุณแม่และพี่ ๆ ที่ส่งปัจจัยต่างในการเล่าเรียน

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

หัวเรื่อง **หน้า**

สารบัญตาราง

หัวเรื่อง **หน้า**

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

ในปัจจุบันนี้ เทคโนโลยีทางด้านไมโครคอนโทรลเลอร์ ได้พัฒนาไปอย่างรวดเร็ว ซึ่งมีประสิทธิภาพที่แตกต่างกันออกไป รวมทั้งยังขึ้นอยู่กับการนำไปใช้งานเฉพาะทางอีกด้วย ซึ่งการที่ให้ไมโครคอนโทรลเลอร์นำไปใช้งานได้นั้น ต้องมีอุปกรณ์ที่เป็นตัวตรวจสอบและการอุปกรณ์ที่ไปควบคุม นั้นก็คือ อุปกรณ์เซนเซอร์ อุปกรณ์แสดงผล อุปกรณ์ทางด้านขับเคลื่อนมอเตอร์ เป็นต้น ไมโครเมาส์ มีอุปกรณ์ไมโครคอนโทรลเลอร์เพื่อควบคุม ซึ่งเป็นอุปกรณ์ที่ได้สั่งซื้อจากต่างประเทศ มีราคาแพงและเสียเวลา ดังนั้นจึงได้สร้างชุดควบคุมใหม่ ซึ่งใช้อุปกรณ์ที่ใช้ภายในประเทศ ซึ่งมีราคาถูกและหาได้ง่าย

1.1 จุดประสงค์ของโครงงาน

1. เพื่อทดแทนการนำเข้าสินค้าจากต่างประเทศ
2. เพื่อให้สามารถเขียนโปรแกรมภาษาซี สำหรับไมโครคอนโทรลเลอร์ 8051 ได้
3. เพื่อให้สามารถใช้คอมไพลเลอร์ได้ถูกต้อง
4. เพื่อให้สามาถออกแบบและสร้างวงจรที่ใช้ควบคุมได้

1.2 ขอบข่ายของโครงงาน

1. สามารถออกแบบและสร้างวงจรควบคุมการทำงานของไมโครเมาส์โดยใช้ไมโครคอนโทรลเลอร์ 8051 ได้
2. สามารถเขียนโปรแกรมควบคุมด้วยภาษาซีได้
3. สามารถส่งคำสั่งที่เขียนแล้วส่งไปยังไมโครเมาส์โดยผ่านพอร์ทอนุกรมได้
4. มีฟังก์ชั่นพื้นฐานต่างๆเพื่อสะดวกในการเขียนโปรแกรม
5. มีการแสดงรูปแบบของสนามบนหน้าจอคอมพิวเตอร์ เพื่อช่วยในการตรวจสอบความถูกต้องของ ไมโครเมาส์ในการแก็บรูปแบบของสนาม
6. มีการแสดงเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดเป้าหมาย

ประโยชน์ที่ได้จากโครงงานนี้ ทำให้เราเรียนรู้การทำงานของไมโครคอนโทรลเลอร์ และ เข้าใจเทคนิคของการเขียนโปรแกรมในลักษณะที่แตกต่างจากโปรแกรมทั่วไป เรียนรู้การเขียนโปรแกรมภาษาซีและคอมไพเลอร์

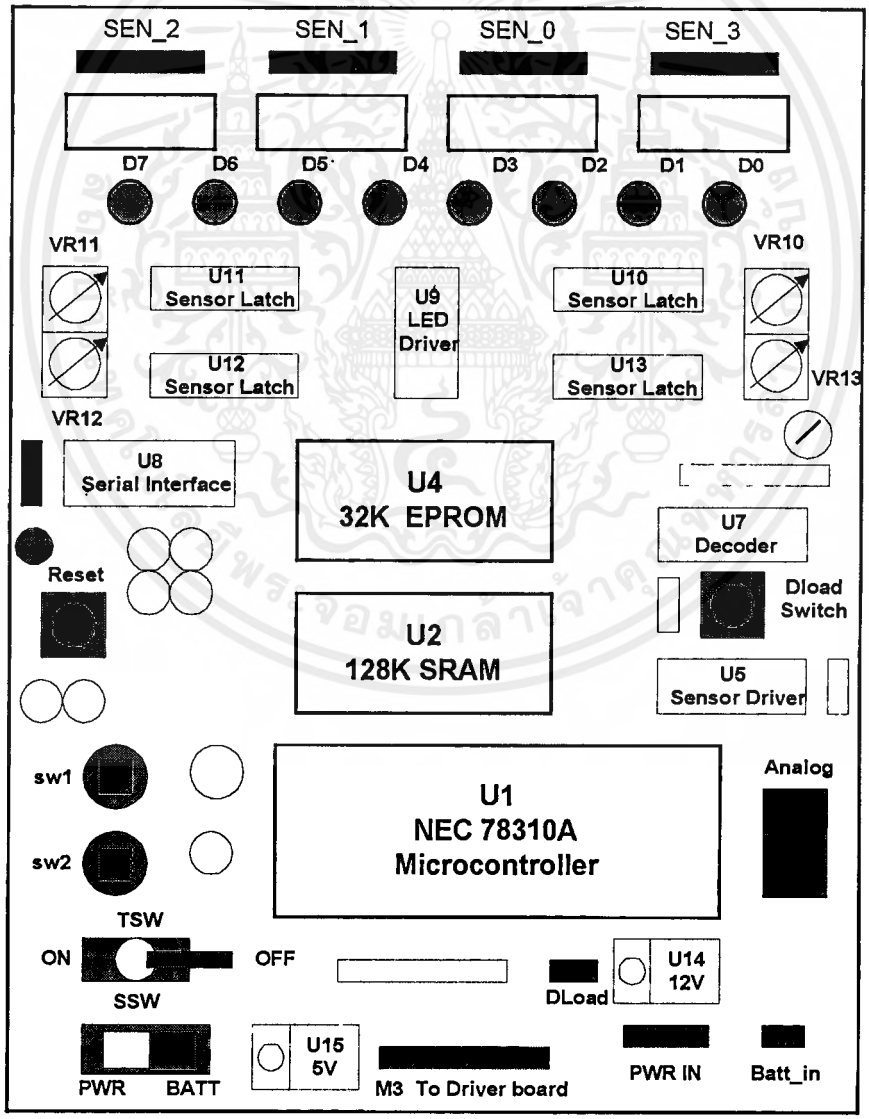
บทที่ 2
ทฤษฎีและหลักการ

ในบทนี้เราจะอธิบายถึงข้อมูลรายละเอียดเกี่ยวกับความสามารถต่างๆของเครื่องต้นแบบทางด้านฮาร์ดแวร์และซอฟท์แวร์ และหลักการทำงานของอุปกรณ์ส่วนต่างๆของโครงงานไมโครเมาส์

เราจะแบ่งส่วนประกอบต่างๆออกเป็น 2 ส่วนใหญ่ๆ คือ ฮาร์ดแวร์และซอฟท์แวร์ รายละเอียดต่างๆของฮาร์ดแวร์มีดังนี้

2.1 ฮาร์ดแวร์ของเครื่องต้นแบบ

จากรูปข้างล่างนี้แสดงถึงการวางตำแหน่งของไมโครคอนโทรลเลอร์บอร์ด ซึ่งมีส่วนประกอบต่าง ๆ ด้วยกันดังนี้

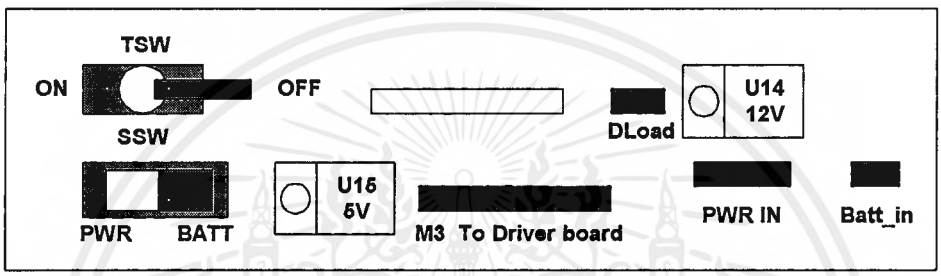


รูปที่ 2.1 ไมโครคอนโทรลเลอร์บอร์ด

2.1.1 System Power

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

แหล่งจ่ายไฟ 5 V หรือ VCC ที่ได้มาจากแหล่งจ่ายภายนอก ซึ่งใช้วงจร Switcing Regulator DC - DC เป็นตัวแปลงแรงดันให้เหลือ 5 V เป็นวงจรที่มีเสถียรภาพมากกว่าวงจร Regulator ทาง linear ซึ่งเราได้ใช้เบอร์ LM2575T5 (U15) ของบริษัท National สามารถดูอุปกรณ์ตามรูปข้างล่างนี้ได้



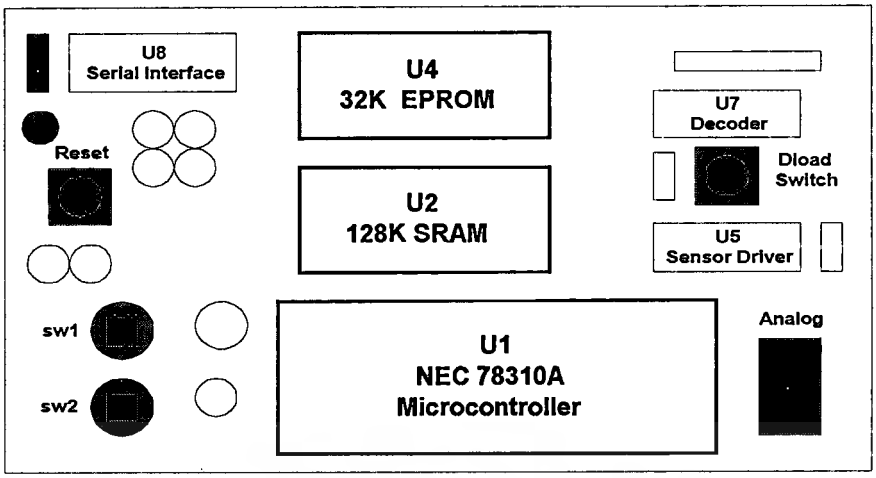
รูปที่ 2.2 ชุดจ่ายไฟของไมโครคอนโทรลเลอร์บอร์ด

แรงดันที่จ่ายให้กับตัวเซนเซอร์ ซึ่งใช้แรงดัน 12V โดยเราไอซี Regulate ธรรมดาเบอร์ L7812 (U14) แรงดันเอาท์พุทจะมีค่าเท่ากับ 12 V แม้ว่า Regulator แบบ Linear จะมีประสิทธิภาพที่ด้อย แต่มันก็ไม่ค่อยมีผลอะไรมากนัก เพราะเราใช้กระแสเอาท์พุทของมันไม่เกิน 100 mA แรงดันนี้จะถูกจ่ายไปยังชุดของเซนเซอร์ทั้ง 4 คือ SEN_0 - SEN_3

แรงดันที่จ่ายให้กับชุดขับสเตปปิ้งมอเตอร์ จะใช้แรงดันจากแหล่งจ่ายไฟที่ป้อนเข้าโดยตรงเลย ในกรณีที่แหล่งจ่ายไฟได้จากแบตตอรี่ หากแรงดันของแบตตอรี่จะลดลง ก็ไม่ต้องกลัวว่ามอเตอร์จะหยุดทำงาน เพราะประสิทธิภาพของมอเตอร์นั้น สามารถทำงานได้แม้ว่าแรงดันจะลดต่ำลงก็ตาม แรงดันส่วนนี้จะป้อนให้กับชุดขับมอเตอร์อีกทางหนึ่งด้วย โดยผ่านคอนเนคเตอร์ M3

2.1.2 Microcomputer

ไมโครคอนโทรลเลอร์ ได้ออกแบบมาเพื่อสำหรับไมโครเมาส์โดยเฉพาะ มีจำนวน I/O 24 bit สำหรับเซนเซอร์และชุดขับสเตปเปอร์มอเตอร์ มันสามารถทดสอบหรือตรวจสอบการทำงาน โดยการผ่านพอร์ทสื่อสารอนุกรมได้



รูปที่ 2.3 ส่วนไมโครคอนโทรลเลอร์บอร์ด

- **Microcontroller**

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

- ทำงานที่ความถี่ 6 Mhz ใช้คริสตอล 12 Mhz
- มีหน่วยความจำภายใน 128 byte
- มี ALU ที่สามารถคำนวนได้ทั้ง 8 bit และ 16 bit
- สามารถคูณและหารข้อมูลได้ทั้ง 8 bit และ 16 bit
- สามารถอ้างหน่วยความจำได้ 64 Kbyte
- มี I/O 24 bit
- มีวงจร Timer 2 ชุด
- มีวงจร Couter 2 ชุด
- มีวงจร A to D 4 ชุด
- มีพอร์ทสื่อสารแบบอนุกรม (UART)

- **Memory Mapping**

หน่วยความจำของเราจะมี EPROM 32 Kbyte (U3) และ RAM 128 Kbyte (U4) ซึ่งมี GAL เป็นตัว decode Address ให้กับหน่วยความจำ

- **Other Sw**

บนบอร์ดจะมีสวิท Reset ถ้าเรากดมันมันจะทำการรีเซตระบบการทำงานทั้งหมดให้เริ่มต้นใหม่ โดยโปรแกรมจะเริ่มอ่านคำสั่งจากตัว EPROM เวลาที่ใช้ในการรีเซต 10 ไมโครวินาที

สวิททางด้านขวาจะเป็นสวิท Dload ที่สั่งให้ไมโครคอนโทรลเลอร์มาเอาคำสั่งจากหน่วยความจำมาประมวลผลซึ่งโปรแกรมนี้ได้มาจาก โปรแกรมที่เราเขียนจากคอมพิวเตอร์แล้วแปลงเป็นภาษาเครื่อง

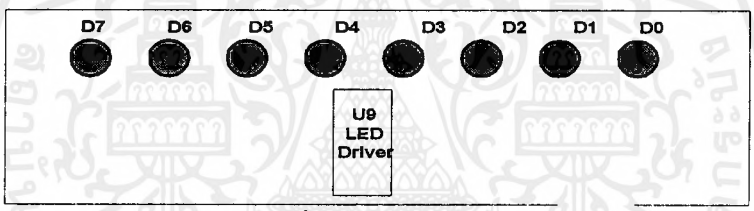
สวิททางซ้าย 2 ตัว ซึ่งออกแบบไว้ใช้งานสำหรับ ผู้ใช้ประกอบด้วย Sw1 และ Sw2 สวิทสองตัวนี้เมื่อกดจะไปทำการอินเตอร์รัพท์ไมโครคอนโทรลเลอร์

- **Serial Communition Interface**

พอร์ทสื่อสารอนุกรม ทำหน้าที่ในการเชื่อมต่อระหว่างไมโครเมาส์กับคอมพิวเตอร์ โปรแกรมของไมโครเมาส์ เราสามารถที่จะพัฒนาได้บน PC และ download ลงไมโครเมาส์ผ่านพอร์ทอนุกรม RS232 โดยเราใช้ไอซี MAX232

- **Display**

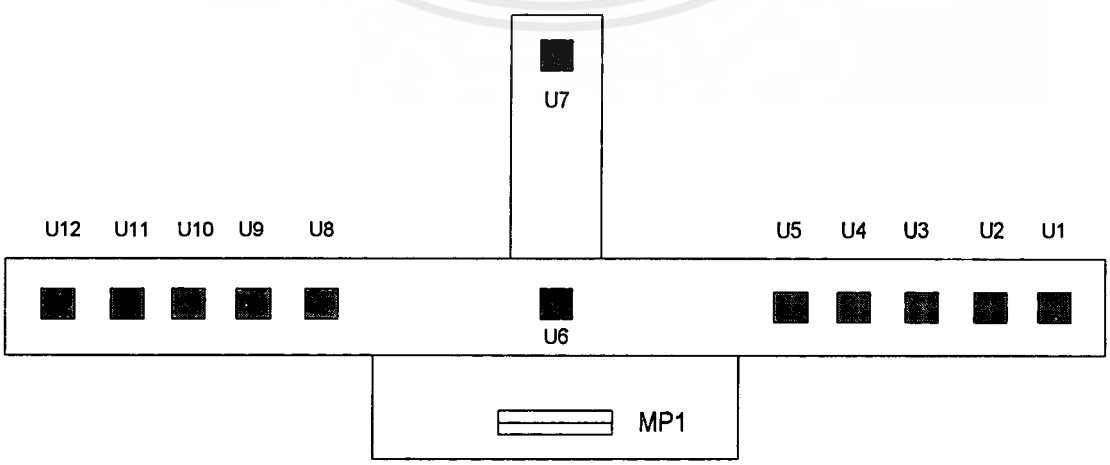
LED ทั้ง 8 บนบอร์ด จะถูกออกแบบมาให้แสดงค่าต่าง ๆ ที่เราต้องการ เราสามารถใช้ LED เหล่านี้ในการตรวจสอบการทำงานของไมโครเมาส์ได้ วงจร DISPLAY นี้จะมีไอซี U9 เป็นตัวขับ LED ทั้ง 8



รูปที่ 2.4 ชุดขับ LED

2.1.3 Sensor Interface

ในชุดของเซนเซอร์จะมีวงจรตัวส่งและตัวรับ ตัวขับตัวส่งนั้นจะทรานซิสเตอร์แบบ darlington โดยใช้เบอร์ ULN2003 (U5) ซึ่งถูกออกแบบเพียงพอต่อการขับตัวส่งสำหรับตัวส่ง LED ตัวรับนั้นจะประกอบด้วย RC filter และวงจร latch ในการอ่านข้อมูลมาเก็บไว้ที่วงจร latch วงจรชุดนี้สามารถใช้ในการตรวจสอบกำแพง



รูปที่ 2.5 ชุดเซนเซอร์

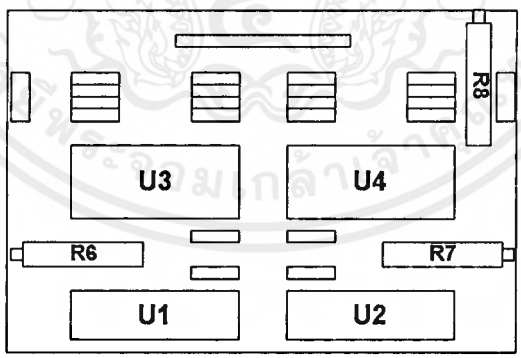
2.1.4 Input and Output Connectors

คอนเนคเตอร์บนบอร์ด จะเชื่อมต่อกับอุปกรณ์รอบข้างเพื่อรับส่งข้อมูล จากวงจรหนึ่งไปอีกวงจรหนึ่ง ซึ่งมีดังนี้

- BATT_IN เป็นคอนเนคเตอร์นี้จะเป็นตัวที่ใส่แหล่งจ่ายไฟจากแบตตอรี่ ซึ่งมีแรงดันในช่วง 15 ถึง 24โวลท์
- PWR_IN เป็นคอนเนคเตอร์นี้จะไว้ต่อกับแหล่งจ่ายไฟ ซึ่งสามารถชาร์จแบตตอรี่ได้ด้วย
- M3 เป็นคอนเนคเตอร์ที่เชื่อมต่อกับชุดขับมอเตอร์
- RS-232 เป็นคอนเนคเตอร์ที่เชื่อมต่อกับพอร์ทอนุกรมของเครื่องคอมพิวเตอร์
- SEN_0 - SEN_3 เป็นคอนเนคเตอร์ที่เชื่อมต่อกับชุดเซนเซอร์
- Dload เป็น jumper ที่ใช้เลือกทำงานจากโปรแกรมที่ดาว์นโหลดมา หรือ จาก EPROM
- ANALOG เป็นคอนเนคเตอร์ที่รับสัญญาณ Analog เข้ามาเพื่อแปลงเป็นดิจิตอล

2.1.5 DRIVER

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



รูปที่ 2.6 ชุดขับสเตปเปอร์มอเตอร์

2.2 ความสามารถด้าน ซอฟท์แวร์ ของไมโครเมาส์เครื่องต้นแบบ

ด้าน ซอฟท์แวร์ จะมีฟังก์ชั่นพื้นฐานและฟังก์ชั่นสำคัญๆเพื่อช่วยให้เราได้ใช้ในการพัฒนาโปรแกรมต่างๆดังนี้

2.2.1 ฟังก์ชั่นติดต่อกับบอร์ดคอลโทรลเลอร์

1.1 ฟังก์ชั่น read_switch

Header : mkit_io.h

Prototype : int **read_switch**(int **sw**)

ฟังก์ชั่นจะส่งค่าที่ไม่เป็นศูนย์ ถ้าปุ่มของสวิทใน MCB ถูกกำหนดโดยตัวแปร **sw** จะเป็นตัวเลือกสวิท ถ้ามันส่งค่ากับเป็น 0 แสดงว่าไม่มีการถูกกด ค่าของ **sw** คือ 1 หรือ 2 ตามโปรแกรมจะแสดงข้อความ " Switch 2 pressed" ถ้าสวิทตัวที่ 2 ถูกกดและแสดงข้อความ "Switch 2 NOT pressed"ถ้าไม่ได้ถูกกด

```
#include <i03x0.h>
#include <mkit_i0.h>
main() {
        while (1) {
                if (read_swtch(2)) printf("\nSwitch 2 pressed ");
                else printf("\nSwitch 2 NOT pressed");
                }
        }
```

1.2 ฟังก์ชั่น trig

Header : mkit_io.h

Prototype : int **trig**(int **sw**)

ฟังก์ชั่นนี้จะส่งค่าที่ไม่ใช้ศูนย์ ถ้าสวิทบนบอร์ด MCU ที่ถูกกำหนดโดยตัวแปร **sw** ถูกกดตั้งแต่การเรียกครั้งสุดท้ายของฟังก์ชั่นนี้ วิธีการใช้ฟังก์ชั่นนี้ ฟังก์ชั่น **init_trig()** จะต้องถูกเรียกก่อนเริ่มเขียนโปรแกรม ค่าของ **sw** มีค่าเป็น 1 หรือ 2

ตัวอย่าง : โปรแกรมนี้จะแสดงความแตกต่างระหว่าง ฟังก์ชั่น **read_switch()** กับ **trig()**

```
#include,io3x0.h>
#include <mkit_io.h>
main () {
        unsigned int x;
        while (1) {
                for (x=0;x=1==;x++) printf("\nTesting %u",x);
                 if(read_switch(2))printf("\nSwitch 2 is detected using  read_switch")
```

```
            if(trig(2))  printf("\nSwitch 2 is detected using trig");
            printf("\nAny  key to continue");
            getch();
           }
    }
```

1.3 ฟังก์ชั่น write_led

Header : mkit_io.h

Prototype : void **write_led**(char **p**)

ฟังก์ชั่นนี้ แสดงค่าของตัวแปร **p** ในรูปแบบของ binary โดยใช้ LED 8 ตัว บนบอร์ท MCU การแสดง บิทของตัวแปร **p** จะเรียงตามลำดับจากทางขวามือ (D1) ไปยังซ้ายมือ (D8) ตัว LED จะสว่างค่าของบิทนั้นมีค่าเป็น 1 และดับเมื่อบิทนั้นเป็น 0 ก่อนใช้ฟังก์ชั่นนี้ต้องเรียกฟังก์ชั่น init_sensor() ก่อนที่เริ่มเขียนโปรแกรม

ตัวอย่าง : โปรแกรมนี้แสดงผลที่ LED เป็น 0x5A และ 0xA5 สลับกันไปเรื่อย ๆ

```
#include <mkit_io.h>
main(0 {
        unsigned int x;
        init_sensor();
        while (1) {
                write_sensor(0x5a);
                for (x=0;x<20000;x++);
                write_sensor(0xa5);
                for (x=0;x<20000,x++);
                }
            }
```

1.4 ฟังก์ชั่น init_trig,init_timer

Header : mkit_io.h

Prototype : void **init_trid**(void),void **init_timer**(void)

ฟังก์ชั่น **init_trig** () จะทำการ setup ให้สอดคล้องกับอินพุทพอร์ท และ อินเตอร์รัพท์ของสวิทบนบอร์ด MCU มัน จะต้องถูกเรียกก่อนเริ่มเขียนโปรแกรม ซึ่งจะเรียกก่อนใช้ฟังก์ชั่น **trig** ()

ฟังก์ชั่น **init_timer ()** จะมีการกระทำดังนี้

- setup ความสอดคล้องระหว่าง timer และ interrupt
- ให้ timer มีค่าเท่ากับ 0
- ให้ timer เริ่มทำงาน

ฟังก์ชั่นนี้ จะต้องถูกเรียกก่อนการเขียนโปรแกรม และก่อนที่เรียกใช้ฟังก์ชั่น **read_timer()**

ตัวอย่าง : ให้ดูในฟังก์ชั่น **trig()** และ **read_timer()**

1.5 ฟังก์ชั่น reset_timer,read_timer

Header : mkit_io.h

Prototype : void **reset_timer** (void) , long unsigned **read_timer** (void)

ฟังก์ชั่น **reset_timer()** จะเคลียร์ค่าของ timer ให้เป็น 0 อย่างไรก็ตาม timer ยังไม่หยุดทำงาน มันจะเก็บค่านั้นแล้วเพิ่มค่าขึ้นไปเรื่อย ๆ การอ่านค่า timer จะใช้ฟังก์ชั่น **read_timer()** ฟังก์ชั่นนี้จะส่งค่าเวลาเป็น microsecond

ตามโปรแกรมนี้จะแสดงเวลาเป็นวินาทีตั้งแต่เริ่มโปรแกรม

```
#include <io3x0.h>
#include <mkit_io.h>
mian() {
        clrscr();
        init_timer();
        while (1) {
                gotoxy(1,1);
                printf(" %6l",read_timer()/1000000L);
                }
        }
```

1.6 ฟังก์ชั่น read_volt

Header : mkit_io.h

Prototype : unsigned int **read_volt**(void)

ฟังก์ชั่นนี้จะส่งค่าของระดับแรงดันของแหล่งจ่ายไฟที่ป้อนให้กับ MCB

ตัวอย่าง : โปรแกรมนี้จะแสดงค่าแรงดันของแหล่งจ่ายไฟ

```
#include <io3x0.h>
```

```
#include <mkit_io.h>
main() {
        clrscr();
        while (1) {
                gotoxy(1,1);
                printf("%w",read_volt());
                }
        }
```

2.2.2 ฟังก์ชั่นเกี่ยวกับบิท

2.1 ฟังก์ชั่น bit, set, clr

Header : bit.h

Protrotype : unsigned char **bit**(int **p**,unsigned char **v**)

void **set**(int **p**, unsigned char **v**)

void **clr**(int **p**, unsigned char **v**)

ฟังก์ชั่น bit() จะส่งค่าที่ไม่เป็นศูนย์ เมื่อบิท **p** ของตัวแปร **v** เป็น 1 และเป็น 0 เมื่อเป็น 0
ฟังก์ชั่น set() เซตบิท **p** ของตัวแปร **v** ฟังก์ชั่น **clr** () เคลียร์ให้เป็น 0 ที่บิท **p** ของตัวแปร **v**
ตัวอย่าง : โปรแกรมนี้จะอ่านค่าเป็นฐาน 16

```
#include <io3x0.h>
#include <bit.h>
main() {
        unsigned int a;
        while (1) {
                printf("\nEnter a Hex : ");
                scanf("%x,&a);
                if (bit(7,(unsigned char) a)) {
                putbin(a);
                printf("\nBit 7 is set");
                clr(7,(unsigned char)a);
                putbin(a);
```

```
            printf("\nBit 7 is now clear")
            }
        }
```

2.2 ฟังก์ชั่น rotr4, rotl4

Header bit.h

Prototype : void rotr4(unsigned char **v**)

void rotl4(unsigned char **v**)

ฟังก์ชั่นนี้จะเลื่อนบิทแบบ nibbles ของตัวแปร **v** ไปทางขวา 1 ครั้ง ถ้าตัวแปร **v** เป็น " abcd efgh " มันจะกลายเป็น " dabc hefg "

ฟังก์ชั่นนี้จะเลื่อนบิทแบบ nibbles ของตัวแปร **v** ไปทางซ้าย 1 ครั้ง ถ้าตัวแปร **v** เป็น " abcd efgh " มันจะกลายเป็น " bcda fghe "

โปรแกรมข้างล่างนี้แสดงตัวแปรโดยกำหนดค่าเริ่มต้นเป็น 0x94 ในรูปแบบไบนารี แต่ละครั้งในการเลื่อน มันจะเลื่อนไปทางขวาโดยใช้ฟังก์ชั่น **rot()**

```
#include <bit.h>
#include <mkit_io.h>
#include ,io3x0.h>
main () {
        unsigned char p=0x94;
        while (1) {
            putbin(p);
            getch();
            rotr4(p);
            }
        }
```

2.3 ฟังก์ชั่น rotr8, rotl8

Header bit.h

Prototype : void **rotr8** (unsigned char **v**)

void **rotl8** (unsigned char **v**)

ฟังก์ชั่น rotr8 () จะทำการเลื่อนบิทของตัวแปร **v** ไปทางขวา ถ้าตัวแปร **v** มีค่าเป็น "abcdefgh" ผลที่ได้ก็จะมีค่าเป็น "habcdefg"

ฟังก์ชั่น **rotl8 ()** จะทำการเลื่อนบิทของตัวแปร **v** ไปทางซ้าย ถ้าตัวแปร **v** มีค่าเป็น "abcdefgh" ผลที่ได้ก็จะมีค่าเป็น "bcdefgha"

ตามโปรแกรม ข้างล่างจะแสดงตัวแปรโดยให้ค่าเริ่มต้นมีค่าเป็น 0x94 ในรูปแบบของเลขฐานสอง ในแต่ละครั้งของคำสั่งมันจะถูกเลื่อนไปทางขวา ซึ่งเราใช้ฟังก์ชั่น **rotr 8()**

```
#include <bit.h>
#include <mkit_io.h>
#include <io3x0.h>
main() {
        unsigned char p = 0x94;
        while (1) {
                putin(p);
                getch(0;
                rotr8(p);
                }
        }
```

2.2.3 ฟังก์ชั่นเกี่ยวกับการเคลื่อนไหว

3.1 ฟังก์ชั่น write_motor

Header : kmit_io.h

Prototype : void **write_mptor** (char **cmd** ,unsigned int **tim**)

ฟังก์ชั่นนี้ใช้ควบคุมการหมุน, ทิศทาง และ ความเร็ว ของสเตปเป้อร์มอเตอร์ ครั้งแรกที่ฟังก์ชั่นได้ทำงาน มอเตอร์จะหมุนตามที่เรากำหนดจนได้เจอคำสั่งต่อไป ทิศทางกำหนดโดยตัวแปร **cmd** รายละเอียดของคำสั่งมีดังต่อไปนี้

FORWARD_FORWARD	ล้อซ้ายหมุนไปข้างหน้า	ล้อขวาหมุนไปข้างหน้า
FORWARD_REVERSE	ล้อซ้ายหมุนไปข้างหน้า	ล้อขวาหมุนถอยหลัง
FORWARD_STOP	ล้อซ้ายหมุนไปข้างหน้า	ล้อขวาหยุดหมุน
REVERSE_FORWARD	ล้อซ้ายหมุนถอยหลัง	ล้อขวาหมุนไปข้างหน้า
REVERSE_REVERSE	ล้อซ้ายหมุนถอยหลัง	ล้อขวาหมุนถอยหลัง
REVERSE_STOP	ล้อซ้ายหมุนถอยหลัง	ล้อขวาหยุดหมุน
STOP_FORWARD	ล้อซ้ายหยุดหมุน	ล้อขวาหมุนไปข้างหน้า
STOP_REVERSE	ล้อซ้ายหยุดหมุน	ล้อขวาหมุนถอยหลัง

STOP_STOP	ล้อซ้ายหยุดหมุน	ล้อขวาหยุดหมุน
DISABLE	ทั้งล้อซ้ายและขวาไม่ทำงาน	

ฟังก์ชั่น **init_motor()** จะต้องถูกเรียกใช้ก่อนที่จะเริ่มใช้ฟังก์ชั่นนี้ โปรแกรมที่อยู่ข้างล่างนี้จะทำให้มอเตอร์หมุนไปข้างหน้า 5 วินาที จากนั้นจะหมุนถอยหลัง 5 วินาที ด้วยความเร็ว 2000 ไมโครวินาทีต่อสเตป ต่อไปมันจะหยุดและสุดท้ายก็จะหยุดทำงาน

```
#include <mkit_io.h>
main() {
        init_motor();
        init_timer();
        write_motor(FORWARD_FORWARD,2000);
        delay(5000000L);
        write_motor(REVERSE_REVERSE,2000);
        delay(5000000L);
        write_motor(STOP_STOP,2000);
        delay(5000000L);
        write_motor(DISABLE,2000);
        }
```

3.2 ฟังก์ชั่น reset_step, read_step

Header : mkit_io

Prototype : void **reset_step**(void), unsigned int **read_step**(void)

เมื่อไรที่มอเตอร์เริ่มหมุน ตัวนับจะเก็บค่าของจำนวนสเตปที่เคลื่อนไป ตัวนับจะถูกนับไปเรื่อย ตัวนับนี้เราสามารถที่จะทำการเคลียร์ให้เป็นศูนย์โดยใช้ฟังก์ชั่น **reset_step()** และจำนวนของสเตปที่นับได้นั้นสามารถอ่านค่ามาได้โดยใช้ฟังก์ชั่น **read_step()**

โปรแกรมข้างล่างนี้ จะทำให้มอเตอร์หมุนไปข้างหน้า 200 สเตป จากนั้นจะถอยหลังหมุนถอยหลัง 200 สเตป ด้วยความเร็ว 2000 ไมโครวินาทีต่อสเตป

```
#include <mkitl_io.h>
main() {
        init_motor();
        while (1) {
```

```
                reset_step();
                motor(FORWARD_FORWARD,2000);
                while (read_step() < 200);
                reset_step();
                motor(REVERSE_REVERSE,2000);
                while (read_step() <200);
                  }
          }
```

3.3 ฟังก์ชั่น delay

Header : mkit_io.h

Prototype : void delay(unsigned long tim)

หน่วงเวลาการทำงานของโปรแกรม

3.4 ฟังก์ชั่น read_sensor

Header : mkit_io.h

Prototype : unsigned char read_sensor(int bnk)

ฟังก์ชั่นส่งสถานะของเซนเซอร์ของ bank ที่กำหนดโดยตัวแปร **bnk** มีอยู่ 4 bank คือ bank 0 -3 ก่อนใช้ฟังก์ชั่นนี้ต้องมีการเรียกใช้ฟังก์ชั่น **init_sensor()** ก่อนจึงจะสามารถใช้ฟังก์ชั่นนี้ได้

โปรแกรมข้างล่างนี้ จะอ่านค่าของเซนเซอร์จากแบงค์ 0 และแสดงผลบนหน้าจอและ LED

```
#include <io3x0.h>
#include <mkit_io.h>
main() {
        unsigned char s;
        clrscr();
        while (1) {
                read_sensor(0);
                gotoxy(1,1);
                putbin(s);
                write_led(s);
                }
        }
```

3.5 ฟังก์ชั่น profile

Header : mkit_io.h

Prototype : unsigned int **profile**(unsigned int ***bf**, unsigned int **vi**,unsigned int **vf**, unsigned int **stp)**

ฟังก์ชั่นนี้จะสร้างความเร่งแบบเชิงเส้นขึ้นมา โดยดูจากจำนวนสเตปที่เริ่มต้นกำหนดด้วยตัวแปร **stp** โดยเราสามารถกำหนดความเร็วเริ่มต้นด้วยตัวแปร **vi** และความเร็วปลายด้วยตัวแปร **vf** ฟังก์ชั่นนี้จะส่งค่าความเร็วสุดท้ายเมื่อทำเสร็จ profile จะถูกสร้าง และ เก็บค่าไว้ในตารางโดยมีตัวแปร **bf** เป็นตัวชี้ มันจะเก็บค่าความเร็วในแต่ละตำแหน่ง ให้มีความเร็วเพิ่มขึ้นเป็นเชิงเส้น เพื่อเป็นค่าความเร็วให้ฟังก์ชั่น **write_motor()**

ตามโปรแกรมข้างล่างนี้จะสร้าง profile โดยใช้วิ่งที่ความเร็วเริ่มต้นที่ไปถึงความเร็วสุดท้ายและพักที่สเตปที่ 1000

```
#include <mkit_io.h>
main() {
        int x;
        unsigned int p[200]
        profile(p,0,800,200);
        init_motor();
        reset_step();
        while ((x=read_step()) < 200) {   write_motor(FORWARD_FORWARD,p[x]);   }
        while (read_step() < 1000 ){   write_motor(FORWARD_FORWARD,p[199]);   }
        write_motor(STOP_STOP,1000);
        delay(1000000L);
        write_motor(DISABLE,1000);
        }
```

3.6 ฟังก์ชั่น init_motor, init_sensor

Header : mkit_io.h

Prototype : void **init_motor**(void),void **init_sensor**(void)

ฟังก์ชั่นเหล่านี้จะถูกกำหนดค่าเริ่มต้นของ i/o port และการ อินเตอร์รัพท์เพื่อต้องการควบคุมมอเตอร์และเซนเซอร์ ฟังก์ชั่น **init_motor()** จะต้องเรียกใช้ก่อนฟังก์ชั่น **write_motor()** หรือ **read_step()** ในทำนองเดียวกันสำหรับฟังก์ชั่น **init_sensor()** จำเป็นต้องเรียกใช้ก่อนฟังก์ชั่น **read_sensor()** หรือ **write_led()**

3.7 ฟังก์ชั่น write_motor ()

Header ; mkit_io.h

Prototype : unsigned char **write_motor**(void)

ฟังก์ชั่นนี้จะส่งค่าสถานะปัจจุบันของมอเตอร์ที่เริ่มถูกควบคุมค่าที่ส่งมาสามารถนำมาตรวจสอบได้ ซึ่งถูกกำหนดไว้ในฟังก์ชั่น **write_motor()** อยู่แล้ว

โปรแกรมนี้จะทำให้ไมโครเมาส์วิ่งไปทางขวา 5 วินาที ถ้ามอเตอร์ถูกหยุด

```
#include <mkit_io.h>
turn_right() {
        if (read_motor() == STOP_STOP) {
                write_motor(FORWARD_REVERSE,2000);
                delay(5000000L);
                }
        }
```

3.8 ฟังก์ชั่น load_profile

Header : mkit_io.h

Prototype : void **load_profile**(insigned int ***buf**, unsigned int **stp**)

ฟังก์ชั่นนี้จะสร้าง profile โดยทำการย่อความเร่ง ซึ่งถูกสร้างขึ้น โดยโปรแกรมจะเรียกใช้โปรแกรม " profgen.exe " profileที่ถูกขยาย ซึ่งถูกโหลดขึ้นมาเป็นตารางชี้โดยตัวแปร **buf** จำนวนสเต็ป ถูกกำหนดโดยตัวแปร **stp**

การใช้ฟังก์ชั่น profile จะต้องถูกสร้างไว้ก่อนโดยใช้โปรแกรม "profgen.exe" ซึ่งอยู่ในไดเร็คทอรี่ "alpha\bin" การสร้างครั้งแรก ไฟล์จะจะถูกบรรจุ profile จะต้องเป็นส่วนประกอบใน project file ซึ่งถูกรวบรวมเข้าด้วยกันค่าของ **stp** จะต้องมีค่าที่ไม่ใหญ่เกินจำนวนของสเตปที่ประกาศเอาไว้ เมื่อไรที่เริ่มวิ่งโปรแกรมจะทำการสร้าง profile สามารถอ้างไปถึง ตัวสร้าง profile
โปรแกรมข้างล่างนี้จะโหลด profileที่ถูกสร้างขึ้น โดยโปรแกรม profgen.exe และความเร็วสูงขึ้นเป็น 250 สเตป

```
#include <mkit_io.h>
main() {
```

```
unsigned spbuf[300];
unsigned int x;
load_profile(spbuf,250);
init_motor();
reset_step();
while ((x=read_step()) <250) { write_motor(FORWARD_FORWARD,spbuf[x]); }
while (read_step() < 500 ){ write_motor(FORWARD_FORWARD,,spbuf[249]); }
write_motor(DISABLE,2000);
}
```

3.9 ฟังก์ชั่น init_move

Header : mkit_io.h

Protype : void_init_move(unsigned int **atpsq**,unsigned int **stp90**)

ฟังก์ชั่นนี้มีความจำเป็นในการกำหนดค่าเริ่มต้น ซึ่งจะใช้ฟังก์ชั่น **move()** และ **backtrack** ตัวแปร **stpsq** เป็นตัวกำหนดจำนวนสเตปที่ต้องการสำหรับไมโครเมาส์ ในการเคลื่อนที่ไป 1 ช่อง ตัวแปร **stp90** จะกำหนดจำนวนของสเตปที่ต้องการสำหรับไมโครเมาส์ เพื่อการเลี้ยวซ้ายและขวาอยู่ในแนวเก้าสิบองศา

ตัวอย่าง : ให้ดูในตัวอย่างของฟังก์ชั่น **move()**

3.10 ฟังก์ชั่น _move

Header : mkit_io.h

Prototype : **_int_move**(unsigned char ***cmd**,unsigned char ***wall**)

ฟังก์ชั่นนี้จะเคลื่อนไมโครเมาส์สำหรับหนึ่งช่อง ไปยังทิศทางที่กำหนดในบัฟเฟอร์ที่ถูกชี้โดยตัวแปร **cmd** ความสมบูรณ์ของการเคลื่อนที่ ลักษณะกำแพงของช่องนั้นเดินทางจะถูกเก็บไว้ในบัฟเฟอร์ที่ถูกชี้โดย **wall** มันจะส่งค่า 1 ถ้าทำงานถูกต้องและ 0 ถ้าไม่ถูกต้อง

คำสั่ง **cmd** สามารถเป็น FRONT,RIGHT,LEFT,BACK อย่างใดอย่างหนึ่ง ลักษระของกำแพงจะอยู่ในรูปของแบบของบิท ตามรูปแบบดังนี้

Front wall : bit 0 Right wall : bit 1

Left wall : bit 2 Back wall : bit 3

ตัวอย่าง : โปรแกรมนี้จะทำให้ไมโครเมาส์วิ่ง

```
#include <mkit_io.h>
#include <bit.h>
```

```
main() {
        unsigned char cmd,wall;
        cmd = FONT;
        _init_move(424, 190);
        while (!trig(2));
        if (_move(&cmd, &wallinfo)) {  write_motor(DISABLE,1000); }
        else if (!bit(LEFT,wallinfo)) cmd = LEFT;
        else if (!bit(FRONT,wallinfo)) cmd = FRONT;
        else if (!bit(RIGHT,wallinfo))  cmd = RIGHT;
        else cmd = BACK;
        }
```

3.11 ฟังก์ชั่น _backtrack

Header : mkit_io.h

Prototype : int_backtrack(unsigned char *pth)

ฟังก์ชั่น จะเคลื่อนไมโครเมาส์ไปตามเส้นทาง ที่กำหนดซึ่งถูกเก็บไว้โดยมีตัวแปร **pth** เป็นตัวชี้รูปแบบของอะเรย์ซึ่งจะต้องมีรูปแบบเป็นคู่ คือทิศทางกับระยะทางแล้วตามด้วย ENDPATH (oxff) ฟังก์ชั่นนี้จะส่งค่า 1 ถ้าทำงานถูกต้อง

ตัวอย่าง : โปรแกรมนี้จะทำให้ไมโครเมาส์วิ่งไปตามเส้นทางที่ถูกกำหนดเอาไว้แล้ว

```
#include <mkit_io.h>
main() {
        unsigned char movetab[128]
        _init_move(424,190);
        while (!trig(2));
        movetab[0] = FRONT;
        movetab[1] = 4;
        movetab[2] = RIGHT;
        movetab[3] = 5;
        movetab[4] = ENDPATH;
        _backtrack(movetab);
        write_motor(DISABLE,1000);
        }
```

2.2.4 ฟังก์ชั่นเกี่ยวกับการค้นหาเส้นทาง

4.1 ฟังก์ชั่น sloverPath

header : m_slove.h

Prototype : unsigned char **sloverPath**(unsigned char ***path**,unsigmed char ***mptr**,unsigned char **st**,unsigned char **dir**,unsigned char **end**)

mptr เป็นตัวชี้ที่เก็บลักษณะของกำแพง โดยฟังก์ชั่น sloverpath จะสร้างเส้นทางการวิ่งของไมโครเมาส์จากตำแหน่งเริ่มต้น **st** ไปยังตำแหน่งเป้าหมาย **end** ในทิศทางที่กำหนดโดย **dir** เมื่อทำเสร็จมันจะส่งค่าทิศทางและระยะทางของทิศทางนั้นไปจนถึงตำแหน่งเป้าหมาย โดยถ้าฟังก์ชั่นนี้ทำสำเร็จก็จะส่งค่า 0 กลับมา

ตัวอย่างผลที่ได้จากฟังก์ชั่นนี้ : FRONT, 3, RIGHT, 5, LEFT, 7, ENDPATH

ความหมายจากผลก็คือ ไปข้างหน้า 3 ช่อง ไปทางขวา 5 ช่อง ไปทางซ้าย 7 ช่อง จากนั้นจะเป็นโค้ดจบข้อมูลคือ 0xFF

```
#include <m_slover.h>

#include <mkit_io.h>

main() {

        unsigned char pathtab[128];

        unsigned char maze[256];

        int i;

        for (i=0;i<256;i++) maze[i] = 0;

        for (i=0;i<16;i++) addwall(maze, i, WEST, 15);

        for (i=0;i<16;i++) addwall(maze, 0xf=(i*0x10), NORTH, 15);

        for (i=0;i<16;i++) addwall(maze, i*0x10,SOUNTH, 15);

        for (i=0;i<16;i++) addwall(maze, 0xf0+i,EAST, 15);

        addwall(maze, 0, EAST, 15);

        solverPath(pathtab,maze,0,NORTH,0x77);

        for (i=0;pathtab[i] != 0xff;i++)

        printf("Path[%d] = %x", i, pathtab[i] );

}
```

4.2 ฟังก์ชั่น sloverCost

Header : m_slove.h

Prototype : usinged char **sloverCost**(unsigned char ***mptr**, unsigned char **st**, unsigned char **dir**,
unsigned chr **end**)

mptr จะเป็นตัวชี้ข้อมูลของลักษณะกำแพง โดยเริ่มจากช่องแรกที่กำหนดโดย **st** กำหนดทิศทางโดย **dir** และกำหนดตำแหน่งปลายทางโดย **end** ฟังก์ชั่นนี้จะสร้างนำหนักของการหาเส้นทางซึ่งถูกสร้างโดยฟังก์ชั่น **sloverPath()** ถ้าฟังก์ชั่นนี้ทำงานถูกต้องจะส่งค่า 0 กลับมา

```
#include <m_slover.h>
#include <mkit_io.h>
#include <io3x0.h>
mian() {
        unsigned char cost,pathtab[];
        unsigned char maze[256];
        int i;
        for (i=0;i<256;i++) maze[i] = 0;
        for (i=0;i<16;i++) addwall(maze, i, WWALL, 15);
        for (i=0;i<16;i++) addwall(maze, 0xf=(i*0x10), NWALL, 15);
        for (i=0;i<16;i++) addwall(maze, i*0x10,SWALL, 15);
        for (i=0;i<16;i++) addwall(maze, 0xf0+i,EWALL, 15);
        addwall(maze, 0, EWALL, 15);
        solverPath(pathtab,maze,0,NORTH,0x77);
        printf("The Cost = %x", i, pathtab[i] );
        }
```

4.3 ฟังก์ชั่น addWall

Header : m_slover.h

Prototype : void **addWall** (unsigned char ***mptr,** unsigned char **sq** ,unsigned char **dir**, unsigned char **bd**)

ฟังก์ชั่นนี้จะทำการเพิ่มกำแพงตามทิศทางที่กำหนดโดย **dir** ของช่องที่กำหนดโดย **sq** ในอะเรย์ จะเก็บลักษณะของกำแพงที่ชี้โดยตัวแปร **mptr** โดยขอบเขตของขนาดของเขาวงกตกำหนดโดยตัวแปร **bd** สำหรับเขาวงกตขนาด 16 x 16 จะกำหนดให้ค่า **bd** มีค่าเท่ากับ 15 ลักษณะของกำแพงจะถูกใช้โดยฟังก์ชั่น **sloverPath()** และ **sloverCost()**

ตัวอย่าง : ให้ดูจากตัวอย่างในฟังก์ชั่น sloverPath();

4.4 ฟังก์ชั่น delWall

Header : m_slover.h

Prototype : void **delWall**(unsigned char ***mptr,** unsigned char **sq,** unsigned char **dir,**unsigned char **bd**
)

ฟังก์ชั่นนี้จะทำการลบกำแพงตามทิศทางที่กำหนดโดย **dir** ของช่องที่กำหนดโดย **sq** ในอะเรย์ จะเก็บลักษณะของกำแพงที่ชี้โดยตัวแปร **mptr** โดยขอบเขตของขนาดของเขาวงกตกำหนดโดยตัวแปร **bd** สำหรับเขาวงกตขนาด 16 x 16 จะกำหนดให้ค่า **bd** มีค่าเท่ากับ 15 ลักษณะของกำแพงจะถูกใช้โดยฟังก์ชั่น **sloverPath()** และ **sloverCost()**

```
#include <m_slover.h>
#include <mkitc_io.h>
#include <io3x0.h>
main() {
        unsigned char cost, pathtab[128];
        unsigned char maze[256];
        int  i;
       for (i=0;i<256;i++) maze[i] = 0xff;
       delwall(maze, 0, NORTH, 15);
       printf(The maze infor = %x",maze[0]);
       }
```

2.2.5 ฟังก์ชั่นเกี่ยวกับการจำลองเส้นทาง

5.1 ฟังก์ชั่น draw_maze

Header : dmaze.h

Prototype : void **draw_maze** (unsigned char ***mz1**,unsigned char ***mz2**)

ฟังก์ชั่นนี้จะทำการวาดเขาวงกต 2 อันบนหน้าจอ ข้อมูลที่อยู่ในอะเรย์ที่ชี้โดยตัวแปร mz1 และ mz2 ข้อมูลของลักษณะของกำแพงของ mz1 จะถูกวาดไว้ทางด้านซ้ายส่วนของ mz2 จะถูกวาดไว้ทางด้านขวาของหน้าจอ

ตัวอย่าง : ให้ดูตัวอย่างของฟังก์ชั่น **dwall ()**

5.2 ฟังก์ชั่น dwall

Header : dmaze.h

Prototype : void **dwall** (unsigned char **wl**, unsigned char **co**, int **mz**)

ฟังก์ชั่นนี้ จะวาดกำแพงซึ่งลักษณะของกำแพงจะถูกเก็บไว้ในตัวแปร **wl** ที่ตำแหน่งที่ชี้โดยตัวแปร **co** เป็นเขาวงกตของตัวแปร **mz** รูปแบบของกำแพงมีดังนี้

bit 0 แสดงกำแพงทางทิศหนือ

bit 1 แสดงกำแพงทางทิศตะวันออก

bit 2 แสดงกำแพงทางทิศใต้

bit 3 แสดงกำแพงทางทิศตะวันตก

ถ้า **mz** = 1 กำแพงจะถูกวาดบนเขาวงกตทางด้านซ้าย **mz** = 2 กำแพงจะถูกวาดบนเขาวงกตทางด้านขวามือของหน้าจอ โปรแกรมตัวอย่างจะวาดทั้งด้านซ้ายและด้านขวา ข้อมูลของกำแพงทางด้านทิศเหนือและใต้จะเปิด แต่ทางซ้ายและขวาจะปิด ซึ่งถูกวาดที่ตำแหน่ง 0x34 (x = 3 , y = 4)

```
#include <io3x0.h>
#include <dmaze.h>
#include <bit.h>
main() {
    unsigned char omaze[256], cmaze[256],wall;
    int c;
    for (c = 0; c<256;c++) {
                omaze[c] = 0x00;
                cmaze[c] = 0xff;
                    }
    draw_maze(omaze,cmaze);
    set(NORTH,wall);
    set(SOUTH,wall);
```

```
clr(WEST,wall);
dwall(wall,0x34,1);
dwall(wall,0x34,2);
}
```

5.3 ฟังก์ชั่น show

Header : dmaze.h

Prototype : void **show** (unsiged char ***pth**,unsigned char **co**,unsigned char **dir**, unsigned char **md)**

ฟังก์ชั่นนี้จะแสดงเส้นทางที่ถูกกำหนดในอะเรย์ที่ชี้โดยตัวแปร **pth** บนหน้าจอ เส้นทางจะเริ่มจาก ตำแหน่งที่กำหนดโดยตัวแปร **co** กับทิศทางที่กำหนดโดย **dir** มันจะแสดงเส้นทางโดยกำหนดให้ **md** เป็น SHOWN และทำการลบเส้นทางที่แสดงอยู่บนหน้าจอโดยกำหนดให้ **md** เป็น HIDDEN ตามโปรแกรม จะแสดงเส้นทางเป็นเส้นประในเขาวงกตทางด้านซ้ายของหน้าจอ และจะลบมันเมื่อมีการกดคีย์

```
#include <io3x0.h>
#include <dmaze.h>
main () {
        unsigned char omaze[256], cmaze[256],path[128];
        int  c;
        for (c=0;c<256;, c++); {
                        omaze = 0x00;
                        cmaze = oxff;
                      }
      draw_maze(omaze,cmaze);
      path[0] = FRONT;
      path[1] = 7;
      path[2] = RIGHT;
      path[3] = 7;
      path[4] = ENDPATH;
      show(path,0x00,NORTH,SHOWN);
      getch();
```

```
    show(path,0xDD,NORTH,HIDDEN);
}
```

5.4 ฟังก์ชั่น init_smove

Header : dmaze.h

Prototype : void **init_smove**(void)

ฟังก์ชั่นนี้จะถูกกำหนดค่าเริ่มต้น ตำแหน่งของไมโครเมาส์ ที่ตำแหน่ง 0x00 ทางทิศเหนือ สัญลักษณ์ของไมโครเมาส์จะถูกวาดบนเขาวงกตทางด้านซ้ายของหน้าจอ

ตัวอย่าง : แสดงในฟังก์ชั่น smove()

5.5 ฟังก์ชั่น smove

Header : dmaze.h

Prototype : int **smove**(unsigned char **pth**, unsigned char ***wptr**)

ฟังก์ชั่นนี้ทำหน้าที่เคลื่อนไมโครเมาส์จากตำแหน่งปัจจุบัน ไปตามทิศทางซึ่งถูกกำหนดในบัปเฟอร์ของ **path** ค่าของ **path** ต้องมีรูปแบบดังนี้

FRONT ซึ่งถูกกำหนดให้เป็น 0

RIGHT ซึ่งถูกกำหนดให้เป็น 1

BACK ซึ่งถูกกำหนดให้เป็น 2

LEFT ซึ่งถูกกำหนดให้เป็น 3

เมื่อการเคลื่อนที่ของไมโครเมาส์บนหน้าจอถูกต้อง มันจะเก็บลักษณะของกำแพง สำหรับตำแหน่งใหม่ในบัฟเฟอร์จะชี้โดย **mptr** ข้อมูลของเขาวงกตจะถูกเก็บไว้ในอะเรย์ ข้อมูลของกำแพงที่ถูกเก็บไว้จะมีรูปแบบดังนี้

bit 0 เป็นกำแพงด้านหน้า

bit 1 เป็นกำแพงทางด้านขวา

bit 2 เป็นกำแพงทางด้านหลัง

bit 3 เป็นกำแพงทางด้านซ้าย

มันจะส่งค่า 1 ถ้าทำงานถูกต้อง และส่งค่า 0 ถ้าผู้ใช้กดคีย์ ESC โปรแกรมข้างล่างนี้ไมโครเมาส์จะเคลื่อนที่จากตำแน่งปัจจุบันไปยังตำแหน่งถัดไปทางด้านซ้าย โปรแกรมนี้จะเชื่อมโยงกับไฟล์ข้อมูลของเขาวงกต

```
#include <io3x0.h>
#include <dmaze.h>
```

```
#include <bit.h>
main() {
        unsigned char omaze[256], cmaze[256], pathtab[128], wall;
        int  c;
        for (c=0; c<256;c++) {
                        omaze = 0x00;
                        cmaze = 0xff;
                        }
        draw_maze(omaze, cmaze);
        init_smove();
        path[0] = LEFT;
        smove(path,&wall);
        printf("\nwall = %x",wall);
        if (bit(FRONT,wall)) printf("\nthere is a wall in front");
        if (bit(RIGHT,wall)) printf("\nthere is a wall on right");
        if (bit(BACK,wall)) printf("\nthere is a wall at back");
        if (bit(LEFT,wall)) printf("\nthere is a wall on left");
        }
```

5.6 ฟังก์ชั่น marksq, unmarksq

Header : dmaze.h

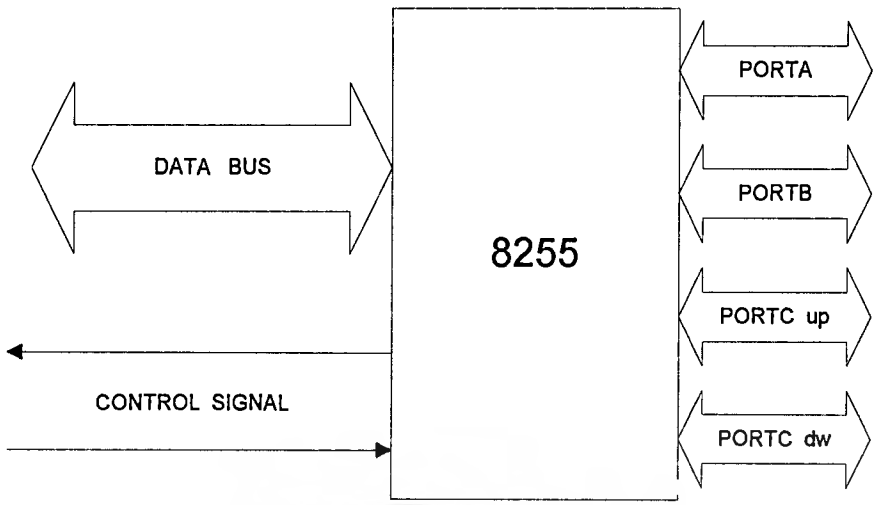
Prototype : void **marksq**(int **mz**, BYTE **co**)

void **unmarksq**(int **mz**,BYTE **co**)

โปรแกรมเหล่านี้จะทำการ mark และ unmark ของช่องนั้น ๆ ในเขาวงกตแสดงบนหน้าจอ ตัวแปร **mz** เป็นเขาวงกตที่ถูกใช้ โดยถ้าเป็น 1 สำหรับเขาวงกตทางด้านซ้าย และถ้าเป็น 2 จะใช้ทางด้านขวา ตัวแปร **co** แทนตำแหน่งที่ถูก mark หรือ unmark

2.3 ไอซี 8255

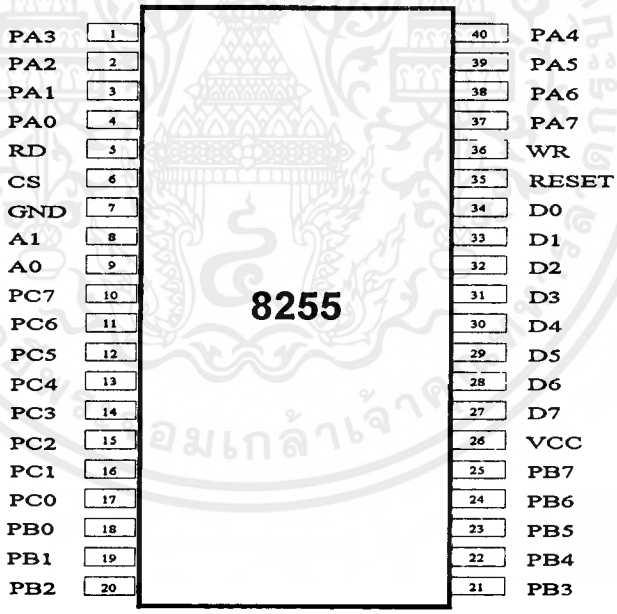
เป็นไอซีที่มี 40 ขา ได้รับการออกแบบมาเพื่อให้สามารถทำงานรวมกับไมโครโปรเซสเซอร์ และ ไมโครคอนโทรลเลอร์ ซึ่ง 8255 นั้นมีพอร์ทใช้งาน 3 ชุด โดยมีโครงสร้างพื้นฐานแสดงไว้ดังรูปที่ 2.1



รูปที่ 2.7 แผนผังโครงสร้างของไอซี 8255

ขาต่าง ๆ ของ 8255

เพื่อให้ เข้าใจวิธีการต่อใช้งาน จึงจำเป็นต้องเข้าใจความหมายและตำแหน่งของขาต่าง ๆ เสียก่อน ขาทั้ง 40 ขามีดังต่อไปนี้



รูปที่ 2.8 แผนผังวงจรภายในและการจัดขาของไอซี 8255

D0 - D7 เป็นขาที่ข้อมูลอินพุทเอาท์พุทจะต้องผ่านเข้าออกจากส่วนนี้ D0 - D7 จึงต่อเข้าระบบของไมโครคอนโทรลเลอร์ เพื่อให้ไมโครคอนโทรลเลอร์สามารถอ่านหรือเขียนข้อมูลจากพอร์ทผ่านทางบัสนี้

CS (สัญญาณเลือกชิฟ) ขานี้เป็นขาอินพุทที่รับสัญญาณมาจากภายนอกเพื่อเลือกชิฟ 8255 โดย เมื่อขานี้เป็น "0" จะทำให้ 8255 ต่อเข้ากับระบบบัสของไมโครโปรเซสเซอร์ เพื่อให้ไมโครโปรเซสเซอร์เขียนหรืออ่านข้อมูลจากพอร์ทได้

RD (สัญญาณการอ่าน) เป็นสัญญาณอินพุทที่จะรับเข้ามาจากซีพียูเมื่อสัญญาณที่ขานี้เป็น "0" และสัญญาณขานี้ CS เป็น "0" ด้วยไอซี 8255 จะทำตัวให้ซีพียูอ่านข้อมูลจากบัสในขณะที่เป็นพอร์ทอินพุท

WR เป็นสัญญาณการเขียนจะแอคตีฟเมื่อมีสัญญาณ WR และสัญญาณ CS เป็น "0" สัญญาณนี้มาจากไมโครคอนโทรลเลอร์ เมื่อต้องการเขียนข้อมูลลงบนพอร์ทที่กำหนด

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

PA0 - PA7 เป็นสายสัญญาณที่เป็นพอร์ทของ 8255 ที่ชื่อพอร์ท A การเลือกพอร์ทจะเลือกโดยสัญญาณแอดเดรส A0 - A1

PB0 - PB7 เป็นสายสัญญาณที่เป็นพอร์ท B ของ 8255 ถูกเลือกโดยสัญญาณแอดเดรส A0 - A1

PC0 - PC7 เป็นสายสัญญาณที่เป็นพอร์ท C ของ 8255 การกำหนดพอร์ทจะได้รับการกำหนดโดยการสัญญาณแอดเดรส A0 - A1 พอร์ท C นี้แบ่งเป็น 2 กลุ่ม PC0 - PC3 และกลุ่ม PC4 - PC7

การใช้งาน 8255 จะต้องส่งรหัสควบคุม (control code) เข้าไปยังพอร์ทหมายเลข 13H การควบคุมการทำงานของ 8255 โดยใช้สัญญาณควบคุมพอร์ทหมายเลข 13H การควบคุมการทำงานของ 8255 มีหลายโหลด แต่ละโหมดจะแตกต่างกันออกไป การโปรแกรมให้ 8255 ทำงานจะทำ 3 โหมดคือ โหมด 0 โหมด 1 โหมด 2

2.4 สเตปปิ้งมอเตอร์

สเตปปิ้งมอเตอร์ทำหน้าที่เปลี่ยนสัญญาณข้อมูลแบบดิจิตอล ไปสู่การเคลื่อนที่ทางกล อย่างได้สัดส่วนกัน โดยที่แกนหรือโรเตอร์ของมอเตอร์ชนิดนี้มักถูกควบคุมให้หมุนเป็นลำดับขั้น

ประโยชน์จากการใช้งานสเตปปิ้งมอเตอร์ที่มีการควบคุมโดยใช้สัญญาณดิจิตอลคือ มีความถูกต้องเที่ยงตรงและสามารถเปลี่ยนตำแหน่งของโหลด ได้อย่างรวดเร็ว เนื่องจากแต่ละอินพุทพัลซ์ จะทำให้สเตปปิ้งมอเตอร์เคลื่อนที่ไปหนึ่งสเตป อย่างเที่ยงตรง

เมื่อพิจารณาในแง่ทางกลจะพบว่าสเตปปิ้งมอเตอร์มีความเที่ยงตรงที่ยอมรับได้ ซึ่งในการเปลี่ยนตำแหน่งอย่างง่ายอาจใช้สวิตช์ ในการควบคุมมอเตอร์ นั่นคือ สร้างวงจรควบคุมมอเตอร์ได้ง่าย ถ้าต้องการเพิ่มประสิทธิภาพของส่วนควบคุม ให้ดีขึ้น อาจใช้ไอซีที่มีความเร็วสูงเนื่องจากประกอบด้วยมอสเฟสอยู่ภายใน จึงได้กำลังงานสูง ความเร็วสูง และต้นทุนต่ำ ความสะดวกในการใช้งานนี้เองจึงทำให้นำไปสู่การใช้งานอย่างแพร่หลาย

การได้รับประโยชน์จากการใช้สเตปปิ้งมอเตอร์อย่างเต็มที่ นั้นขึ้นอยู่กับการขับอย่างถูกต้องเหมาะสม ซึ่งภาคขับนี้จะต้องประกอบไปด้วย แหล่งจ่ายไฟฟ้ากระแสตรง อิเล็กทรอนิกส์สวิตช์ โดยอาจจะเป็นทรานซิสเตอร์หรือมอสเฟส และแหล่งจ่ายสัญญาณข้อมูลแบบดิจิตอล ซึ่งมีลักษณะเป็นพัลซ์ เพื่อใช้ควบคุมทิศทางการหมุนของมอเตอร์

ทิศทางของกระแสไฟฟ้าที่ป้อนเข้าสู่สเตปปิ้งมอเตอร์ ถูกควบคุมโดยการทำงานของอิเล็กทรอนิกส์สวิตช์ ดังนั้นสเตปปิ้งมอเตอร์จะหมุนไปหนึ่งช่วงในแต่ละอินพุทพัลซ์ที่ป้อนเข้าสู่วงจรอิเล็กทรอนิกส์สวิตช์ ซึ่งขนาดมุมที่หมุนไปจะขึ้นอยู่กับการออกแบบ
สเตปปิ้งมอเตอร์ โดยมีค่าตั้งแต่ 1.8° ถึง 15° ถ้าส่งสัญญาณไปยังวงจรสวิตช์ชิ่ง 24 พัลซ์ โดยมีค่าของหนึ่งช่วงการหมุนเท่ากับ 15° สเตปปิ้งมอเตอร์ก็จะหมุนไปครบ 1 รอบพอดี เวลาที่ใช้ในการหมุนครบ 1 รอบ ขึ้นอยู่กับอัตราการจ่ายสัญญาณพัลซ์ควบคุมโดยสัญญาณนี้อาจถูกผลิตโดยวงจรกำเนิดสัญญาณ ที่ปรับความถี่ได้หรือจากไอซีควบคุม

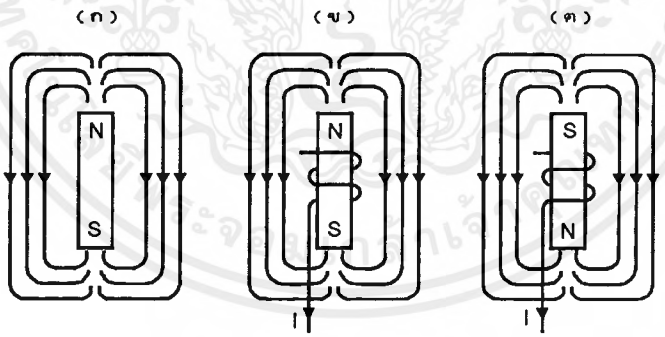
2.4.1 หลักการพื้นฐานของสเตปปิ้งมอเตอร์

ในรูปที่ 2.9 หลักการพื้นฐานของเส้นแรงแม่เหล็ก

ในรูปที่ 2.9 (ก) สนามแม่เหล็กที่เกิดจากแม่เหล็กถาวรมีทิศทางพุ่งจากขั้วเหนือไปยังขั้วใต้

ในรูปที่ 2.9 (ข) สนามแม่เหล็กของแม่เหล็กไฟฟ้าที่เกิดจากกระแส (I)

ในรูปที่ 2.9 (ค) ขั้วแม่เหล็กกลับทิศทาง เมื่อขดลวดถูกพันกลับทิศทาง และทิศทางการไหลของกระแสไม่เปลี่ยนแปลง



รูปที่ 2.9 สนามแม่เหล็กที่เกิดขึ้นในลักษณะต่างๆ

สเตปปิ้งมอเตอร์ แบ่งออกได้ 3 ชนิด คือ

- สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร
- สเตปปิ้งมอเตอร์แบบค่ารีลัคแตนซ์แปรค่าได้
- สเตปปิ้งมอเตอร์แบบไฮบริด

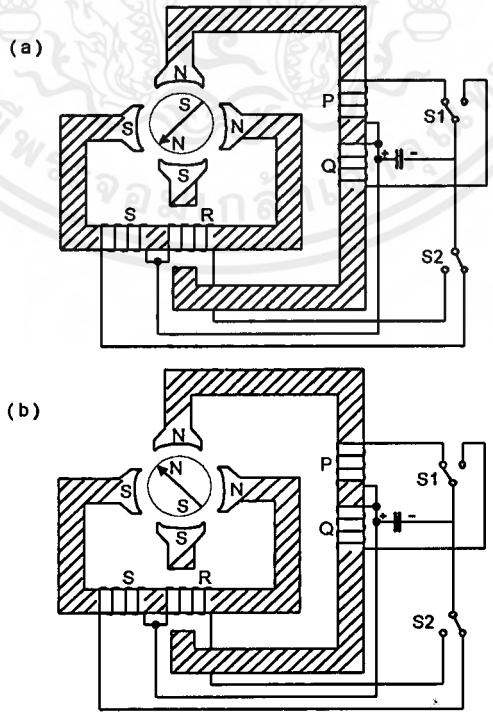
ในที่นี้จะกล่าวเฉพาะสเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร

2.4.2 สเตปปิ้งมอเตอร์แบบแม่เหล็กถาวร

มุมการหมุน (Step Angle) ของมอเตอร์แบบแม่เหล็กถาวรขึ้นอยู่กับความสัมพันธ์ ระหว่างจำนวนขั้วแม่เหล็กบนส่วนที่อยู่กับที่หรือเรียกว่าสเตเตอร์ และจำนวนขั้วแม่เหล็กบนส่วนเคลื่อนที่หรือที่เรียกว่าโรเตอร์ เนื่องจากโรเตอร์ซึ่งเป็นแม่เหล็กถาวรรูปทรงกระบอกจำนวนขั้วแม่เหล็กจึงถูกจำกัดสูงสุดที่ค่าหนึ่ง การเพิ่มขนาดเส้นผ่านศูนย์กลางของโรเตอร์จะทำให้ได้จำนวนขั้วแม่เหล็กบนโรเตอร์เพิ่มมากขึ้น แต่มีข้อเสียคือเกิดแรงเฉื่อยเพิ่มมากตามไปด้วย ซึ่งจะลดประสิทธิภาพขณะเริ่มต้นการหมุนของมอเตอร์ สเตปปิ้งมอเตอร์แบบนี้มีขนาดความกว้างของมุมหนึ่งช่วงการหมุนมาก แต่สามารถลดลงได้โดยการทำให้มีหลายสแต็ค หรือมากกว่าหนึ่งสแต็คขึ้นไป ตามแนวความยาวของมอเตอร์ (สแต็คในที่นี้หมายถึงเฟส ซึ่งประกอบด้วยโรเตอร์ที่เป็นซี่ฟัน และโครงร่างของสเตเตอร์อยู่รอบนอก)

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

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

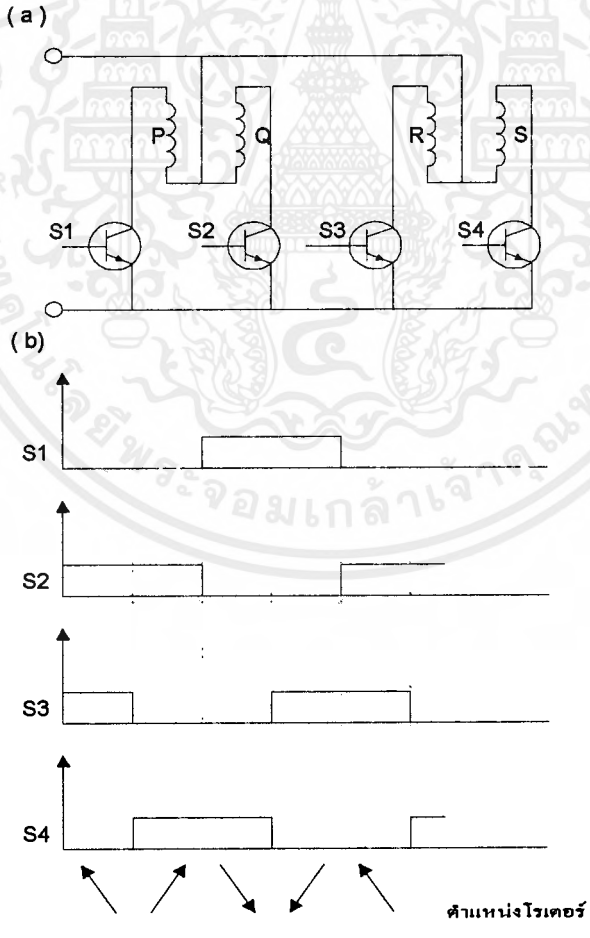


รูปที่ 2.10 มอเตอร์ 4 เฟส แบบขั้วเดียว

2.4.3 การขับมอเตอร์แบบขั้วเดียว

สำหรับการขับมอเตอร์แบบขั้วเดียวนั้น ขดลวดสเตเตอร์ถูกออกแบบให้มีจุดแบ่งกึ่งกลาง บนขดลวด จุดนี้จะต่อไปยังขั้วใดขั้วหนึ่งของแหล่งจ่ายไฟ ทิศทางกระแสไฟฟ้าที่ไหลผ่านขดลวดจะถูกกำหนดโดยการต่อปลายขดลวดเข้ากับขั้วใดขั้วหนึ่งที่เหลืออยู่ของแหล่งจ่ายไฟ โดยใช้อุปกรณ์ประเภทสวิตช์ชิ่งในการตัดต่อวงจร เช่น ทรานซิสเตอร์ดังรูปที่ 2.11 และการสลับกันทำงานระหว่างขดลวดมีผลทำให้ขั้วแม่เหล็กบนสเตเตอร์ เปลี่ยนแปลง ดังรูปที่ 2.10

รูปที่ 2.10 (a) สเตปปิ้งมอเตอร์ 4 เฟส โดยกระแสไฟฟ้าเข้าเฟส P และ S ดังนั้นจะเกิดขั้วแม่เหล็กบนสเตเตอร์ซึ่งจะดึงดูดให้มอเตอร์หมุนมาอยู่ ณ ตำแหน่งการวางตัวดังรูปที่ 2.10 (a) ต่อมาเมื่อสวิตช์ S_1 เปลี่ยนตำแหน่ง ทำให้กระแสไฟฟ้าไหลในเฟส Q และ S สนามแม่เหล็กบนสเตเตอร์จะกลับทิศทาง ทำให้ขั้วแม่เหล็กบนสเตเตอร์กลับกันด้วย โรเตอร์จึงหมุนมาอีกตำแหน่งหนึ่งดังรูปที่ 2.10 (b) แนวลูกศรที่แสดงบนโรเตอร์จะมีทิศทางเดียวกับเส้นแรงแม่เหล็กรวมที่พุ่งออกมาจากสเตเตอร์ จากหลักการทำงานที่กล่าวมา ทำให้สามารถกลับทางหมุนของมอเตอร์ได้โดยการเปลี่ยนลำดับการทำงานของสวิตช์ S_1 และ S_2

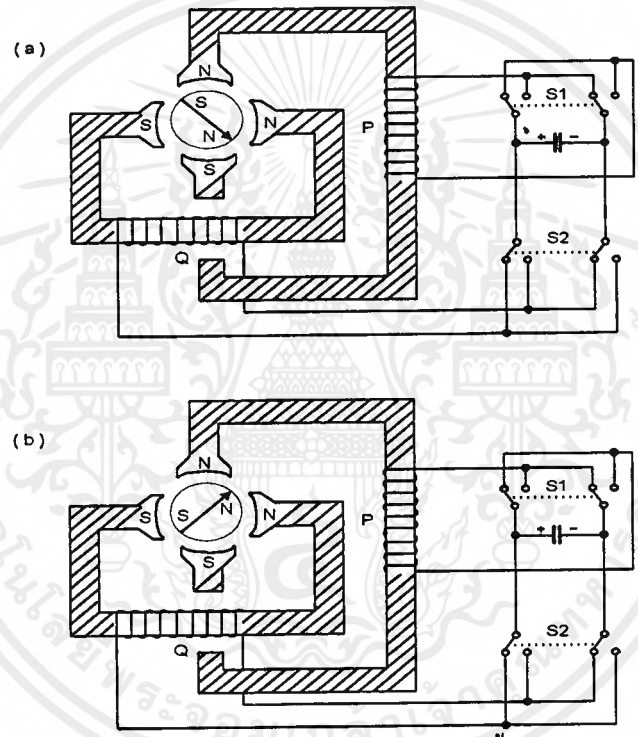


รูปที่ 2.11 การขับแบบขั้วเดียว 2.5 (a) วงจรขับมอเตอร์ 2.5 (b)สัญญาณควบคุม

วงจรสวิตช์ชิ่ง โดยให้สัญญาณพัลซ์ป้อนไปยังขา B ของทรานซิสเตอร์แต่ละตัว จะเห็นว่าที่เวลาใดเวลาหนึ่งจะมีขดลวด 2 เฟส ที่ได้รับกระแสไฟฟ้าในเวลาที่ต่อเนื่องกัน นั่นคือจะเกิดสนามแม่เหล็กหมุนที่สเตเตอร์เหนี่ยวนำให้มอเตอร์หมุนเคลื่อนที่เป็นลำดับได้

2.4.4 การขับมอเตอร์แบบสองขั้ว

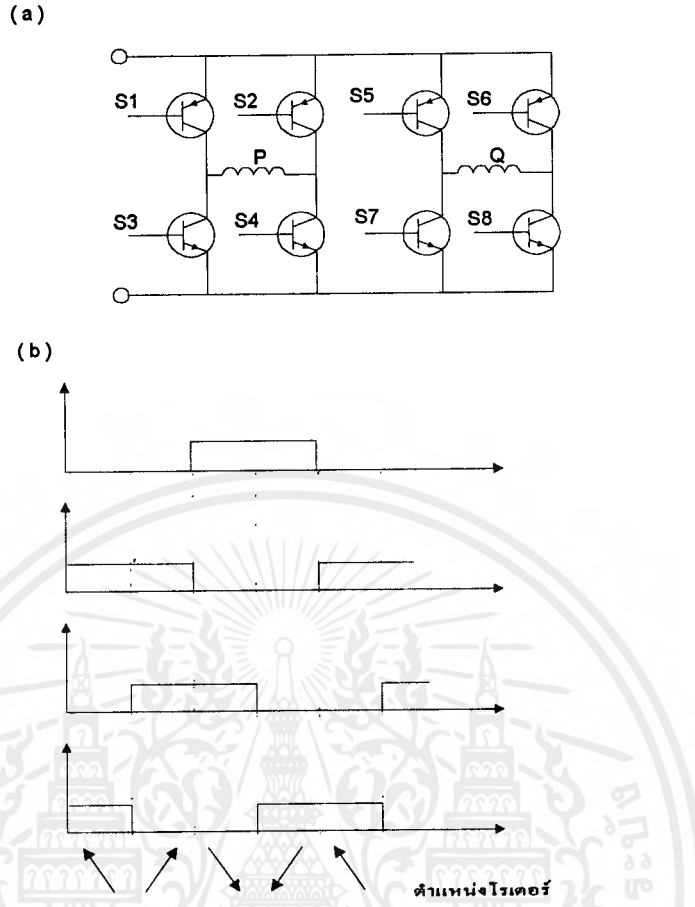
ขดลวดสเตเตอร์ของมอเตอร์ที่ออกแบบมาสำหรับการขับแบบสองขั้วนั้นจะไม่มีจุดแบ่งกึ่งกลาง หลักการทำงานของมอเตอร์ที่ขับแบบสองขั้วมีลักษณะเช่นเดียวกับการขับมอเตอร์แบบขั้วเดียวดังรูปที่ 2.12 นั่นคือเมื่อกระแสในเฟส P เปลี่ยนทิศทางโดยการสับสวิตช์ S_1 สนามแม่เหล็กในสเตเตอร์ก็จะเปลี่ยนทิศทางและโรเตอร์จะเคลื่อนที่ไปหนึ่งลำดับ



รูปที่ 2.12 มอเตอร์ 4 เฟสแบบ 2 ขั้ว

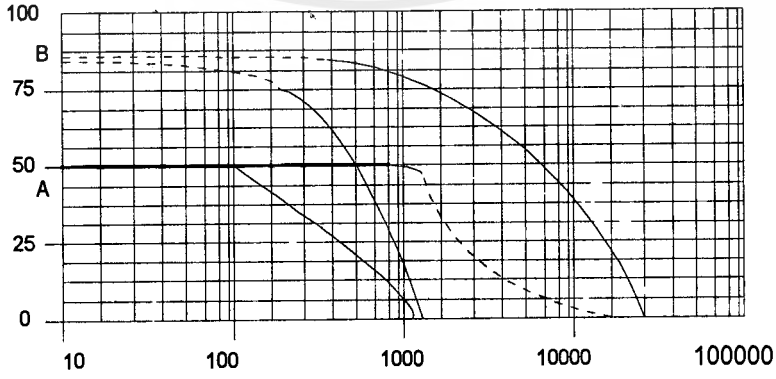
รูปที่ 2.13 (a) วงจรขับมอเตอร์ 4 เฟส แบบสองขั้วทรานซิสเตอร์ในวงจรจะทำงานพร้อมกันเป็นคู่ เช่น S_1 กับ S_4 ทำงาน กระแสจะไหลจากขั้วบวกของแหล่งจ่ายไฟไปเข้าทางปลายขดลวด P หรือ S_6 กับ S_7 ทำงาน กระแสจะกลับทิศทางการไหลคือไหลจากขั้วลบของแหล่งจ่ายไฟมาเข้าขดลวดและออกทางปลาย P เป็นต้น

รูปที่ 2.13 (b) ลักษณะสัญญาณควบคุมที่ป้อนให้กับวงจรขับมอเตอร์ช่วงเวลาการทำงานของทรานซิสเตอร์ (Switching Time) ในวงจรขับแบบสองขั้ว จะต้องมีความเที่ยงตรงกว่าการขับแบบขั้วเดียว มิฉะนั้นอาจจะทำให้เกิดการลัดวงจรแหล่งจ่ายไฟ เพราะทรานซิสเตอร์ตัวบนและล่าง (เช่น S_1 กับ S_2) นำกระแสพร้อมกัน



รูปที่ 2.13 การขับแบบสองขั้ว (a) วงจรขับมอเตอร์ (b) สัญญาณควบคุม

สัญญาณควบคุมมอเตอร์ที่ขับแบบขั้วเดียวในรูปที่ 2.10 อาจใช้ 2 ขดพันบนบอบบิ้น (Bobbin) ในแต่ละสเตเตอร์ซึ่งเรียกว่าการพันแบบสองแถวสลับกัน (Bifilar Winding) ดังนั้นจึงมี 2 ขดที่ถูกยึดบนบอบบิ้นอันเดียวกัน หลักการทำงานจึงเป็นแบบเดียวกับการพันขดลวดขดเดียวในมอเตอร์แบบสองขั้ว แต่มีขนาดลวดเล็กกว่าและความต้านทานสูงกว่า

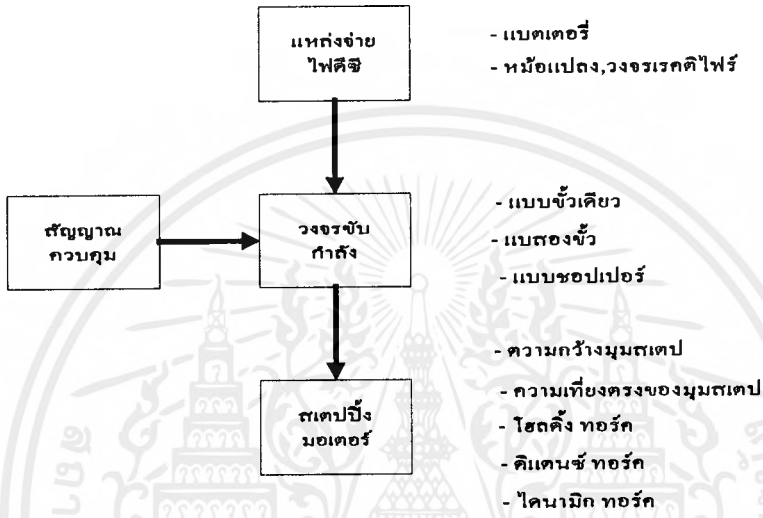


รูปที่ 2.14 โค้งความสัมพันธ์ระหว่างแรงบิดและอัตราการหมุน
(A) มอเตอร์แบบขั้วเดียว (B) มอเตอร์แบบสองขั้ว

ประโยชน์จากการขับมอเตอร์แบบสองขั้วแสดงดังรูปที่ 2.14 กล่าวคือ การขับแบบสองขั้วจะสร้างแรงบิดหรือทอร์ค (Torque) ได้มากกว่าการขับแบบขั้วเดียว ที่อัตราช่วงการหมุนต่ำๆ แต่จะมีค่าของแรงบิดใกล้เคียงกัน ที่อัตราช่วงการหมุนสูงๆ

2.4.5 ระบบสเตปปิ้งมอเตอร์

การเลือกใช้สเตปปิ้งมอเตอร์ เพื่อนำไปใช้ประโยชน์ในงานบางอย่างนั้น จะต้องมีความเข้าใจคุณลักษณะของมอเตอร์และวงจรที่ใช้ขับมอเตอร์ในรูปที่ 2.15 แผนผังของระบบสเตปปิ้งมอเตอร์



รูปที่ 2.15 แผนผังระบบสเตปปิ้งมอเตอร์

ความถูกต้องเที่ยงตรงของมุมสเตปขณะที่ไม่มีโหลด จะถูกระบุสำหรับมอเตอร์แต่ละชนิด เช่น มอเตอร์ที่มีสเตป 7.5° ความผิดพลาด ±10 ลิปดา ขณะเคลื่อนที่ไป หนึ่งสเตป เป็นต้น

มุมสเตป	จำนวนเตปต่อรอบ
0.9°	400
1.8°	200
3.6°	100
3.75°	96
7.5°	48
15.0°	24

ตารางที่ 2.1 ตัวอย่างของมมุมสเตป

มอเตอร์ที่มีจำนวนสเตปต่อรอบเท่ากับ 4 จะมีค่าผิดพลาดเป็นศูนย์ เมื่อหมุนครบ 1 รอบ เพราะขณะที่หมุนมา ณ ตำแหน่งเดิมขณะเริ่มต้นขั้วแม่เหล็กและทิศทางของเส้นแรงแม่เหล็ก (Flux) วงเดิม ด้วย

เหตุนี้การเปลี่ยนตำแหน่งของสเตปปิ้งมอเตอร์ที่ต้องการความถูกต้องสูงๆ จะต้องแบ่งจำนวนสเตปต่อรอบเป็นจำนวนเท่าของ สี่สเตป เพื่อลดการสะสมของค่าผิดพลาด (Step angle error) ซึ่งเป็นรูปแบบการทำงานแบบสี่สเตปตัวอย่างของมุมสเตป แสดงดังตารางที่ 2.1

แรงบิด (Torque)

การทำงานของสเตปปิ้งมอเตอร์มีแรงบิดเกี่ยวข้องอยู่ 3 ชนิดคือ

โฮลดิ้งทอร์ค (Holding Torque) คือแรงบิดที่ทำให้สเตปปิ้งมอเตอร์เริ่มหมุนไป สองสเตป ขณะหยุดนิ่ง ถ้าแรงบิดที่ทำให้สเตปปิ้งมอเตอร์มีขนาดมากกว่าโฮลดิ้งทอร์ค จะทำให้มอเตอร์สูญเสียการหมุนแบบสเตปกลายเป็นการหมุนแบบต่อเนื่องโดยปกติแรงบิดขณะทำงานของมอเตอร์จะน้อยกว่าระดับโฮลดิ้ง

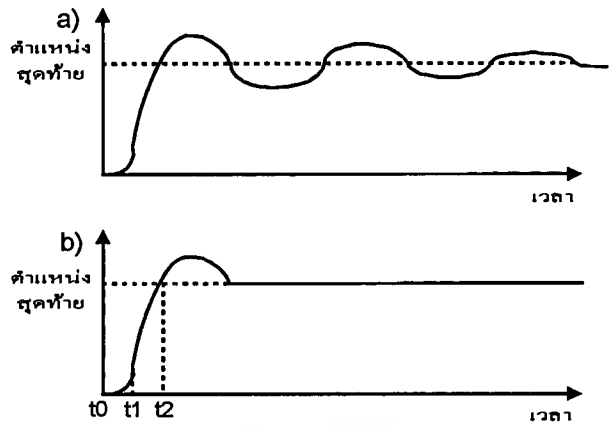
ดิเทนซ์ทอร์ค (Detent Torque) ซึ่งเป็นสเตปปิ้งมอเตอร์แบบไฮบริดและแบบแม่เหล็กถาวร จะมีส่วนประกอบของโรเตอร์เป็นแม่เหล็กถาวรซึ่งจะสร้างแรงบิดมาเบรคการหมุนของมอเตอร์อย่างสม่ำเสมอในขณะที่ไม่มีการป้อนกระแสเข้าขดสเตเตอร์ แรงบิดดังกล่าวนี้เรียกว่า ดิเทนซ์ทอร์ค

ไดนามิกทอร์ค (Dinamic or working torque) คือแรงบิดขณะทำงานซึ่งอาจเกิดการเปลี่ยนแปลงได้ เนื่องมาจากการปรับเปลี่ยนอัตราเร็วของมอเตอร์ โดยปกติการเปลี่ยนอัตราเร็วของมอเตอร์จะอยู่ในย่านระหว่าง เส้นโค้งพูลอิน (Pull-in curve) และ
เส้นโค้งพูลเอ้าท์ (Pull-out corve) เพราะถ้าปรับอัตราเร็ว ณ จุดนอกโค้งพูลเอ้าท์มอเตอร์จะสูญเสียการหมุนแบบเป็นสเตปได้หรือเกิดการหมุนแบบต่อเนื่องนั่นเอง

การแกว่งเข้าสู่สภาวะคงตัว (Over Shoot) ขณะที่มอเตอร์ไปในแต่ละสเตป และหยุด ณ สเตปใดๆ จะเกิดการแกว่งหรือสั่นของโรเตอร์ เข้าสู่ตำแหน่งสุดท้ายนั้นๆ และใช้เวลาช่วงหนึ่งในการเข้าสู่สภาวะคงตัว แสดงเปรียบเทียบได้ดังรูปที่ 2.16 (a) ซึ่งเป็นพฤติกรรมปกติ ของระบบที่ใช้สัญญาณพัลซ์ โดยทั้งนี้ขึ้นอยู่กับโหลดและกำลังงานที่ได้รับจากภาคขับมอเตอร์ ผลตอบสนองที่เกิดขึ้นนี้สามารถเปลี่ยนแปลงคือลดเวลาในการเข้าสู่สภาวะคงตัวได้โดยการเพิ่มโหลดที่เป็นแรงเสียดทานเข้าไปในระบบ ซึ่งเป็นลักษณะการแก้ไขทางกล เช่นการใช้เครื่องต่อกำลังไปเพลาโดยใช้ความฝืดไม่ใช่เฟือง

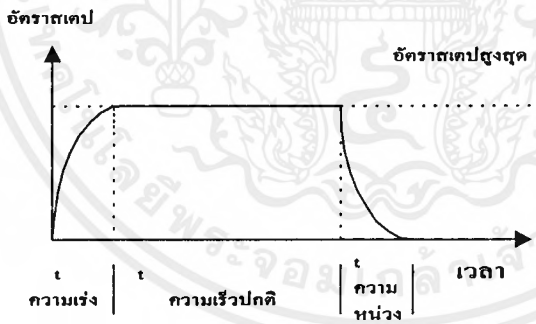
วิธีการแก้ไขทางไฟฟ้า ทำได้โดยหน่วงสัญญาณพัลซ์ลูกสุดท้ายในขบวนพัลซ์ทั้งหมด โดยอาจถูกเปลี่ยนเป็น 3 ส่วนด้วยกัน ดังรูปที่ 2.16 (b) โดยส่วนแรกที่เวลา t_0 จะเป็นการฟอร์เวิร์ดพัลซ์เวลา t_1 จะป้อนเป็นรีเวิร์สพัลซ์เพื่อชะลอการหมุนของโรเตอร์ ส่วนสุดท้ายที่เวลา t_2 จะเป็นฟอร์เวิร์สพัลซ์ ป้อนเพื่อให้โรเตอร์หยุด ณ ตำแหน่งที่ต้องการ ซึ่งวิธีการนี้จะสร้างแรงบิดน้อยกว่าปกติ ดังนั้นจึงลดการสั่นหรือแกว่งของเพลาได้

การสเตปที่แตกต่างกัน มีหลายวิธีด้วยกันในการกำหนดจำนวนสเตปที่ใช้ในการเคลื่อนที่จากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่ง เช่น หมุนไป 90° อาจใช้ทั้งหมด 6 สเตปๆ ละ 15° 12 สเตป ละ 7.5° หรือ 50 สเตปๆ ละ 1.8° โดยทั่วไปแล้วมอเตอร์ที่มีความกว้างของมุมสเตปน้อยจะลดการแกว่งหรือออสซิเลต และจะความมีความแม่นยำดีกว่ามอเตอร์ที่มีความกว้างของมุมสเตปมาก



รูปที่ 2.16 ผลตอบสนองในการเข้าสู่สภาวะคงตัว (a) ผลตอบสนองเมื่อการหน่วงน้อย (b) ผลตอบสนองเมื่อมีการหน่วงทางไฟฟ้า

อัตราเร่งของมอเตอร์จะถูกควบคุมโดยวงจร VCO (Voltage controlled oscillator) และช่วงเวลาในการเก็บประจุ (RC time Constant) ของคาปาซิเตอร์จะกำหนดอัตราเร่งที่แตกต่างกัน ในรูปที่ 2.10 แสดงอัตราสเตปต่อเวลาซึ่งแบ่งออกเป็น 3 ช่วงคือ ขณะสตาร์ทช่วงแรกมีความเร่งเพิ่มอัตราสเตปอย่างต่อเนื่อง ช่วงที่สองเป็นอัตราสเตปขณะใช้งานสูงสุด และช่วงสุดท้ายเกิดความหน่วง (Deceleration) ลดความเร็วของมอเตอร์ลงมาจนกระทั่งหยุดนิ่ง



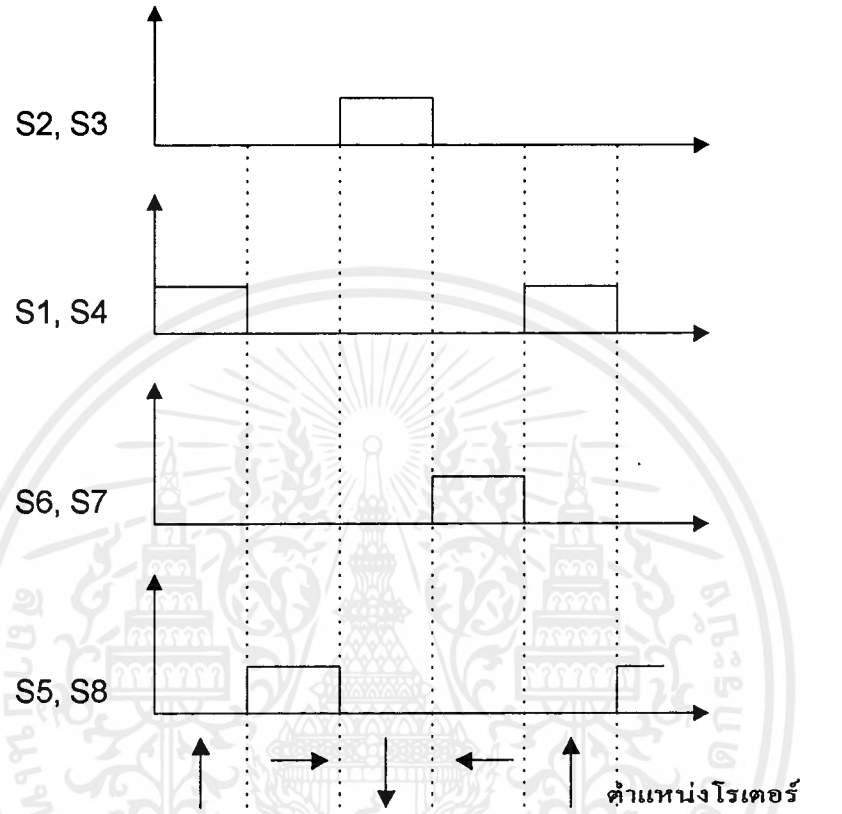
รูปที่ 2.17 โค้งช่วงเวลาการเพิ่มความเร็วและลดความเร็วของมอเตอร์

รีโซแนนซ์ (Resonance) สเตปปิ้งมอเตอร์ขณะทำงานไม่มีโหลด โดยการเปลี่ยนความถี่ใช้งานไปในช่วงหนึ่งๆ จะมีความถี่ค่าหนึ่งที่ทำให้เกิดการรีโซแนนซ์ ซึ่งจะเกิดเสียงรบกวน หรือสามารถตรวจสอบได้ด้วยการตรวจจับการสั่นสะเทือน ดังนั้นควรหลีกเลี่ยงการใช้งาน ณ ความถี่เรโซแนนซ์

2.4.6 วิธีขับเฟสสเตปปิ้งมอเตอร์

วิธีที่ใช้ในการขับเฟสสเตปปิ้งมอเตอร์ขึ้นอยู่กับลักษณะการพันของขดลวดและรูปแบบของวงจรขับมอเตอร์

วิธีขับแบบเฟสเดียว (Wave Drive) คือ วิธีการขับที่เฟสใดเฟสหนึ่งของมอเตอร์ได้รับกระแสไฟฟ้าที่เวลาใดๆ รูปที่2.18 ลำดับการส่งสัญญาณควบคุมของสเตปปิ้งมอเตอร์ 4 เฟส ซึ่งมีเฟสเดียวที่ถูกขับ ดังนั้นค่าโฮลดิ้งและไดนามิกทอร์คจะลดลง 30% แต่สามารถชดเชยค่าแรงบิดให้เพิ่มขึ้นได้โดยเพิ่มแรงดันของแหล่งจ่ายไฟ ซึ่งจะทำให้ประสิทธิภาพ (Efficiency) สูง แต่มีความแม่นยำของสเตปน้อย



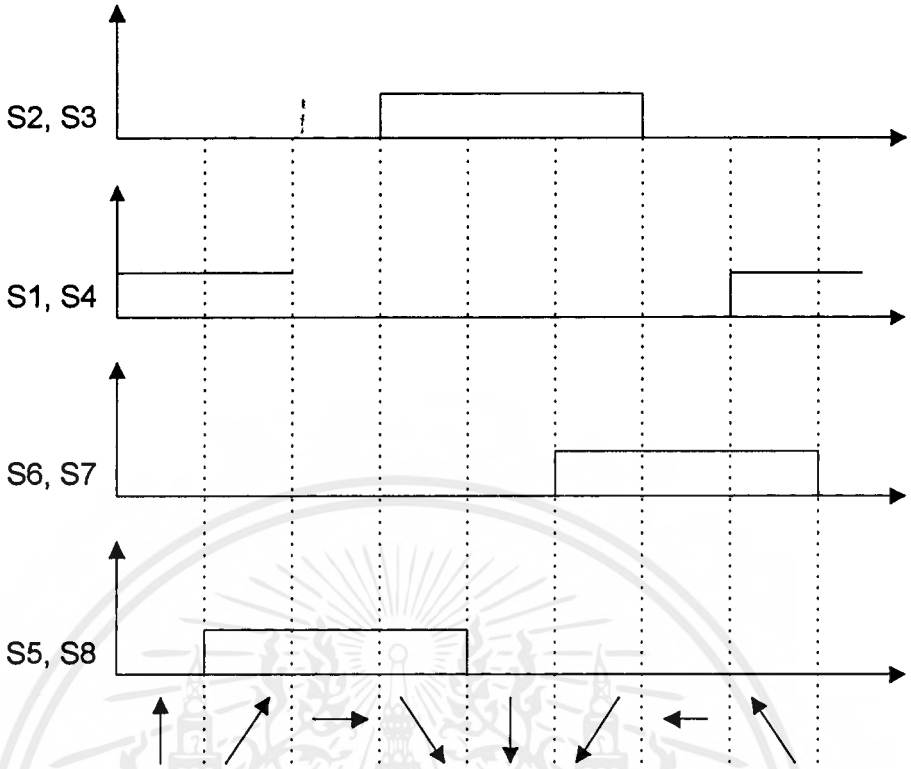
รูปที่ 2.18 สัญญาณควบคุมสเตปปิ้งมอเตอร์ 4 เฟสที่มีวงจรขับสองขั้วแบบเฟสดียว

วิธีขับแบบหนึ่งและสองเฟส (Half-step mode) คือวิธีการขับแบบครึ่งสเตป เช่น ความกว้าง 1 สเตป เท่ากับ 7.5^{o} แต่การขับวิธีนี้จะหมุนไปครั้งละครึ่งสเตป เท่ากับ 3.75^{o} ผลที่ได้รับขณะใช้งานคือโฮลดิ้งทอร์คจะมากและน้อยสลับกันไป เพราะถูกขับ 2 เฟส และ 1 เฟส สลับกัน ดังนั้นค่าแรงบิดเฉลี่ยจะสูงกว่าวิธีขับเฟสเดียว แต่จะมีความแม่นยำน้อยลงขณะที่หมุนครบสเตปลำดับการส่งสัญญาณควบคุม ดังรูปที่ 2.18

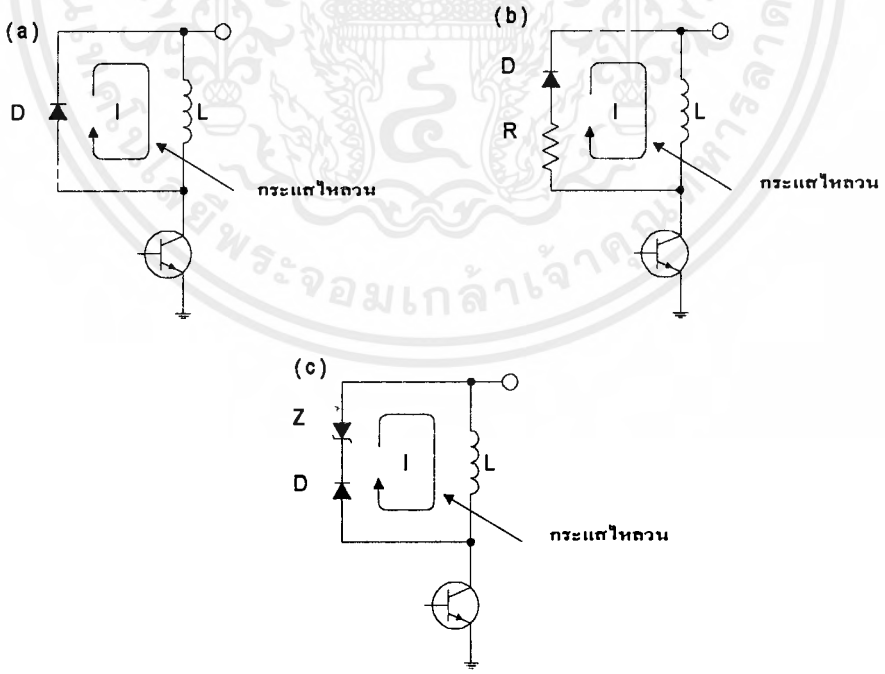
2.4.7 การกำจัดสไปค์ (Spike Suppression)

ขณะที่ทรานซิสเตอร์ในวงจรขับมอเตอร์หยุดนำกระแสจะทำให้เกิดค่าแรงดันสูง ซึ่งเกิดการยุบตัวของสนามแม่เหล็กในขดลวด ตามสมการ $V = L\ di/dt$ แรงดันที่เกิดขึ้นนี้ อาจทำให้รอยต่อของทรานซิสเตอร์เสียหายได้ ถ้าไม่กำจัดออกไป ปัญหาเหล่านี้แก้โดยใช้วงจร ฟรีวิลลิ่ง (Reewheeling Circuit) เป็นทางผ่านของกระแส จากขดลวดขณะที่ทรานซิสเตอร์หยุดทำงาน ซึ่งมีรูปแบบดังนี้คือ

การกำจัดโดยใช้ไดโอด (Diode Suppression) คือการนำไดโอดมาต่อขนานกันกับขดลวด ดังรูปที่ 2.20 (a) หลังจากที่ทรานซิสเตอร์หยุดนำกระแส จะเกิดกระแสไหลวนอยู่เป็นวงกลมรอบขดลวดและ



รูปที่ 2.19 สัญญาณควบคุมสเตปปิ้งมอเตอร์ 4 เฟส ที่มีวงจรขับสองขั้วแบบ 1 และ 2 เฟส

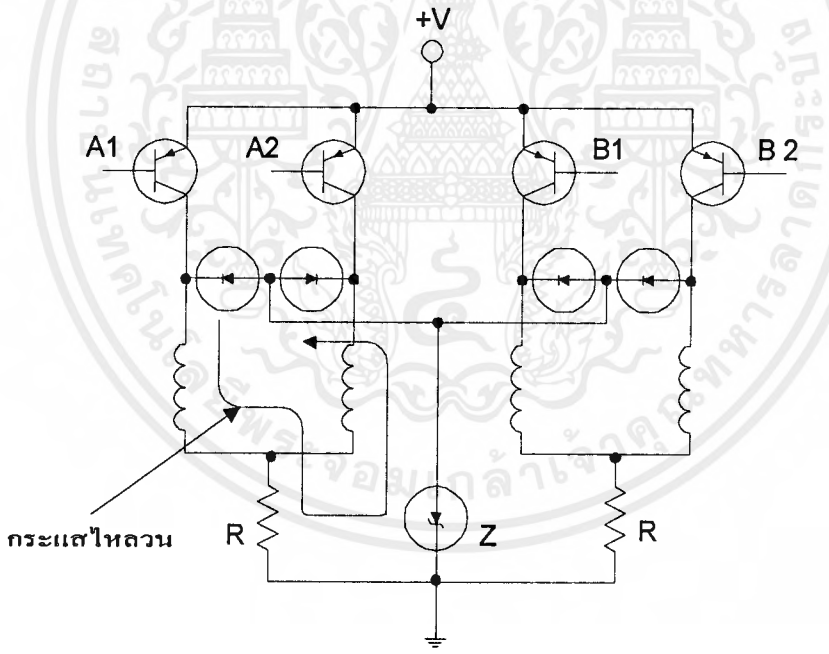


รูปที่ 2.20 วิธีการกำจัดสไปค์ (a) กำจัดด้วยไดโอด (b) กำจัดด้วยไดโอดและความต้านทาน (c) กำจัดด้วยไดโอดและซีเนอร์ไดโอด

ไดโอด กระแสนี้จะลดค่าลงตามเวลา วิธีนี้เป็นวิธีที่ง่ายที่สุดแต่การทำให้กระแสไหลวนหมดต้องใช้เวลานาน

การกำจัดโดยใช้ไดโอดและความต้านทาน (Diode and Resister Suppression) คือการนำค่าความต้านทานมาอนุกรมกับไดโอด ดังรูป 2.20 (b) ค่าความต้านทานช่วยให้กระแสไหลวนลดค่าลงเร็วขึ้น เพราะมีค่าเวลาคงตัว (Time constant = L/R) น้อยลง

การกำจัดโดยใช้ไดโอดและซีเนอร์ไดโอด (Zener Diode Suppression) การใช้ไดโอดเพียงอย่างเดียวในการกำจัดแรงดันสไปค์มีข้อเสียคือ แรงบิดที่ได้รับจะลดน้อยลงไป นอกเสียจากว่าแรงดันตกคร่อมทรานซิสเตอร์จะมีค่าเพิ่มขึ้นประมาณ 2 เท่า ของแหล่งจ่ายแรงดัน ซึ่งแรงดันที่สูงขึ้นนี้จะทำให้สนามแม่เหล็กและกระแสไหลวนลดค่าลงอย่างรวดเร็ว ทำให้แรงบิดจึงดีขึ้น ด้วยเหตุนี้จึงใช้ความต้านทานต่ออนุกรมกับไดโอดหรือซีเนอร์ไดโอดต่ออนุกรมกับไดโอดแทน ดังรูปที่ 2.20 (c) ซึ่งวิธีการใช้ซีเนอร์ไดโอดจะให้ผลดีที่สุด โดยเปรียบเทียบเวลาที่ใช้ในการลดค่ากระแสไหลวน และรูปที่ 2.21 ตัวอย่างวงจรที่ใช้ซีเนอร์ไดโอด และไดโอดต่ออนุกรมเพื่อลดแรงดันสไปค์ สำหรับความต้านทาน R ในวงจรใช้สำหรับสร้างกระแสกระตุ้นขดลวดให้เร็วขึ้น



รูปที่ 2.21 ตัวอย่างวงจรกำจัดแรงดันสไปค์

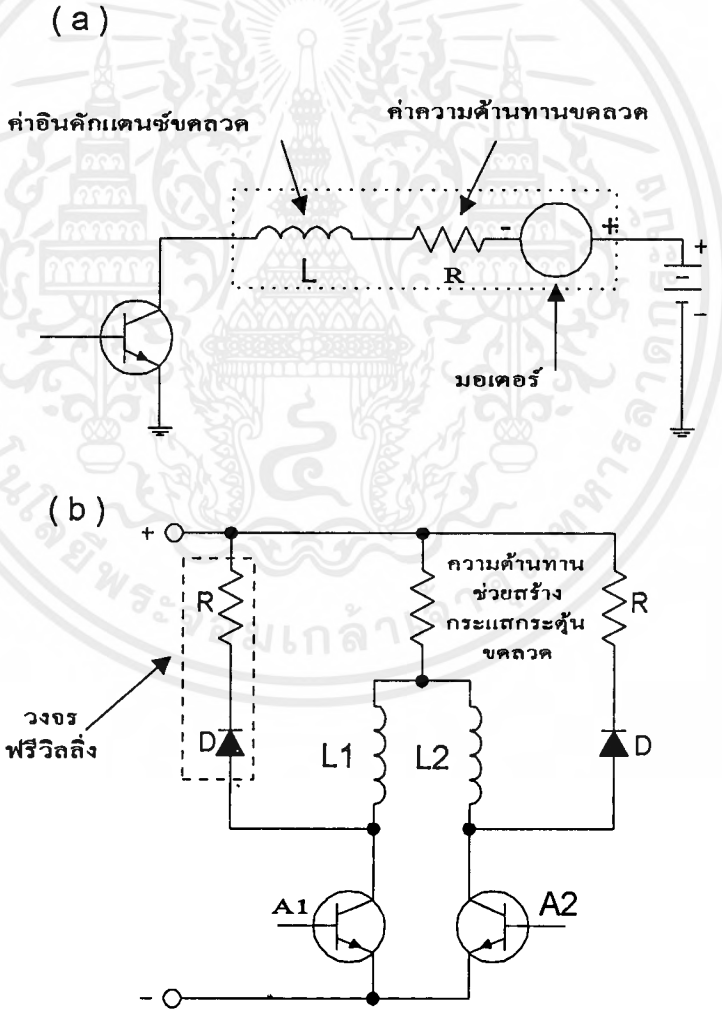
2.4.8 การพิจารณาวงจรขับมอเตอร์

ในสภาวะที่มอเตอร์ทำงานด้วยระดับแรงดันคงที่ค่าหนึ่งแรงบิดที่ได้ จะลดลงถ้ามีการเพิ่มอัตราสเตปเนื่องมาจากการเพิ่มค่าของแรงดันต้านกลับ (Back EMF) และช่วงเวลาขาขึ้น (Rise Time) ของกระแสในขดลวดถูกจำกัดหรือจะมีค่ามีค่ามากวงจรเทียบเคียงของ

สเตปปิ้งมอเตอร์แสดงดังรูปที่ 2.22 (a) วิธีแก้ไขให้เกิดการเปลี่ยนแปลงของกระแสในขดลวดเร็วขึ้นทำได้โดยเพิ่มแรงดันของแหล่งจ่ายไฟให้สูงขึ้นเพื่อรักษาระดับกระแสให้คงที่ และขณะทำงานที่อัตราสเตป สูงๆ หรืออาจคงค่าของแรงดันเอาไว้ด้วย แต่เพิ่มค่าความต้านทานอนุกรมเข้าไปในวงจรดังรูปที่2.22 (b) ค่าความต้านทานนี้เรียกว่า Forcing Resistance

ค่าความต้านทานที่ต่อเพิ่มนี้จะลดค่าเวลาคงตัว (t = L/R)ทำให้ใช้งานที่ความเร็วสูงได้ดี ดังนั้น ถ้าเพิ่มความต้านทานมีค่า 3 เท่าของความต้านทานขดลวด จะทำให้มีค่าเวลาคงตัวเท่ากับ L/4R และควรเพิ่มขนาดแรงดัน แหล่งจ่ายไฟเป็น 4 เท่าด้วย เพื่อรักษาระดับกระแสผ่านขดลวดให้คงที่ซึ่งในกรณีนี้อาจมีปัญหาในเรื่องแหล่งจ่ายไฟ หรือเกิดการสูญเสียที่ Forcing resistance ในรูปของความร้อน ดังนั้นที่ความเร็วต่ำๆ จึงไม่เหมาะที่จะนำวิธีการนี้มาใช้

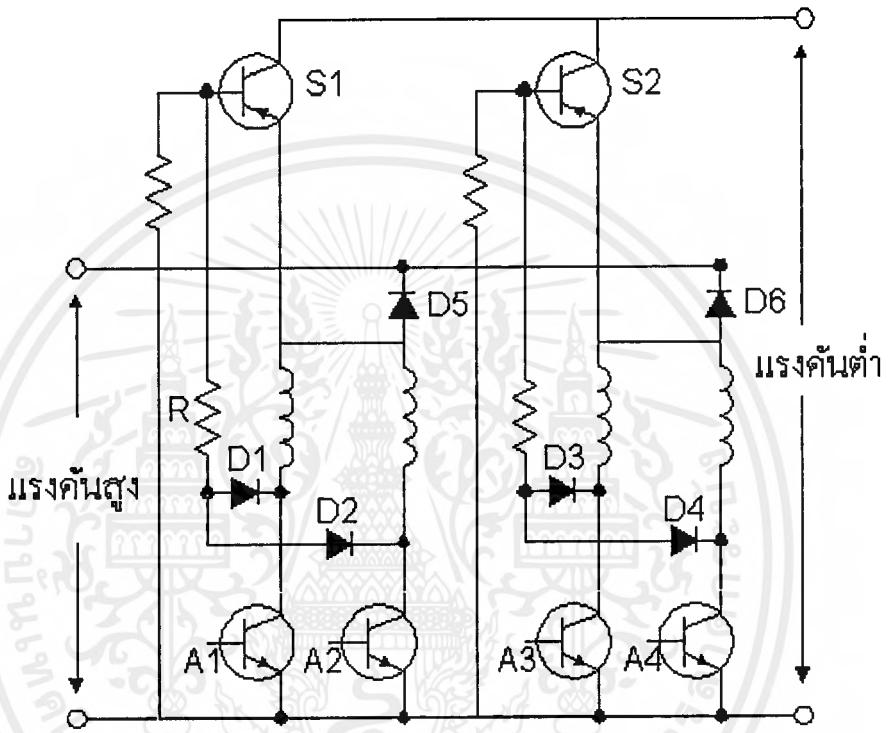
ในการออกแบวงจรขับมอเตอร์ ควรให้มีกำลังงานสูญเสีย จากค่า Forcing Resistance น้อยที่สุด วงจรขับที่จะกล่าวถึงต่อไปนี้จะแก้ปัญหาเหล่านี้ได้



รูปที่ 2.22 (a) แสดงวงจรเทียบเคียงขดลวดของสเตปปิ้งมอเตอร์

(b) แสดงวงจรขับที่มี Forcing resistance

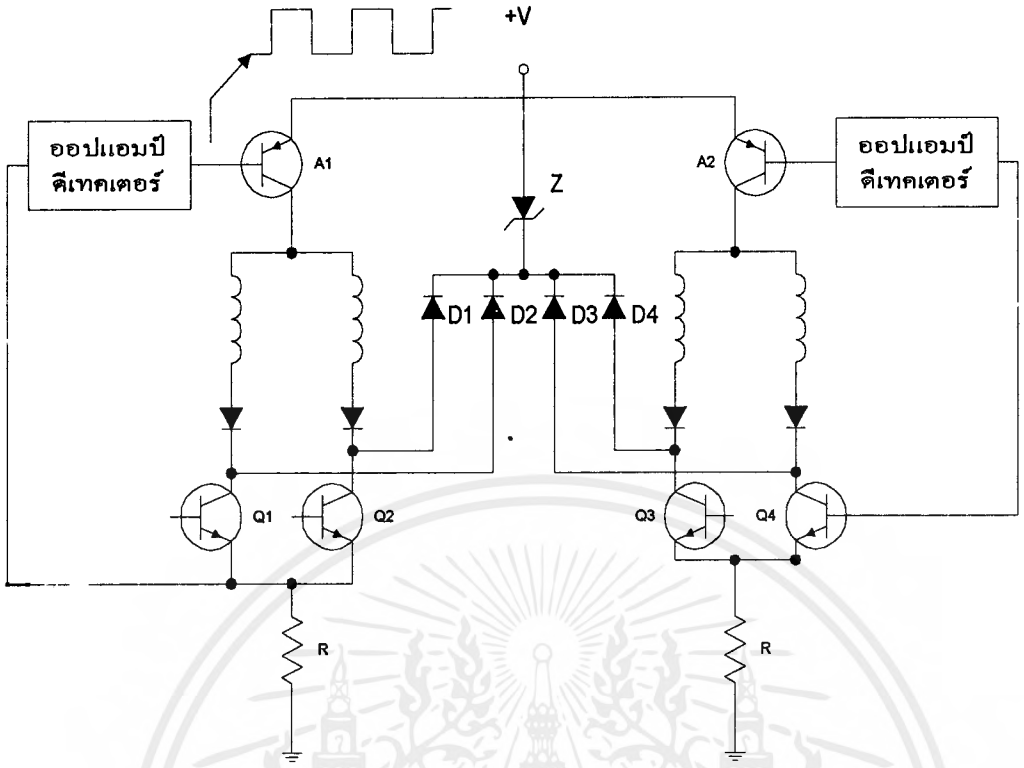
วงจรขับแบบสองสถานะ (Bi-level drive) วงจรขับแบบสองสถานะ มีแรงดันที่จ่ายให้กับมอเตอร์ 2 ค่า คือค่าสูงและต่ำขณะอัตราสเตปเป็นศูนย์จะรับแรงดันต่ำกว่าขีดจำกัดของมอเตอร์และเมื่ออัตราสเตปเพิ่มจากค่าศูนย์ มอเตอร์จะทำงานที่ระดับแรงดันสูงกว่าขีดจำกัด จากวงจรขับแบบสถานะที่แสดงดังรูปที่ 2.22 แรงดันค่าต่ำถูกใช้ช่วงอัตราสเตปเป็นศูนย์จนกระทั่งระดับกระแสในขดลวดมีค่าระดับหนึ่งจึงต่อแรงดันค่าสูงเข้าไปแทนและแรงดันค่าต่ำถูกตัดออกไปโดยการหยุดนำกระแส (Cut off) ของ D5 หรือ D6, D1, D2, D3, D4 เป็น ฟรีวิลลิ่งไดโอดและ R เป็นฟรีวิลลิ่ง รีซิสแตนซ์



รูปที่ 2.23 วงจรขับแบบสองสถานะ

วงจรขับแบบชอปเปอร์ (Chopper drive) วงจรขับแบบนี้ใช้ระดับแรงดันจากแหล่งจ่ายเพียงค่าเดียว จ่ายให้กับขดลวด ดังในรูปที่ 2.24 โดยกระแสที่จ่ายเข้ามอเตอร์จะถูกรักษาระดับที่ค่าเฉลี่ยค่าหนึ่งด้วยวิธีการควบคุมระดับกระแสนั่นคือ วงจรออปแอมป์ดีเทคเตอร์จะตรวจสอบค่าของกระแสโดยเปรียบเทียบจากแรงดันที่ตกคร่อม R เมื่อกระแสสูงกว่าระดับเฉลี่ย วงจรออปแอมป์ดีเทคเตอร์จะหยุดขับทรานซิสเตอร์ กระแสที่ไหลผ่านขอลวดจึงลดค่าลง เมื่อลดลงมาต่ำกว่าค่าระเฉลี่ย ทรานซิสเตอร์ก็จะถูกขับให้นำกระแสเพื่อให้มีกระแสไหลผ่านขดลวดเพิ่มขึ้น และจะทำงานสลับกันไปเพื่อให้ได้ระดับกระแสเฉลี่ยค่าหนึ่ง

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



รูปที่ 2.24 วงจรขับแบบชอปเปอร์

2.5 ไมโครคอนโทรลเลอร์ MCS-51

2.5.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

- ต้องการแหล่งจ่ายไฟ +5V. ชุดเดียว

- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์ สำหรับเบอร์ 8051 และ 8031, 8032 ไม่มีหน่วยความจำชุดนี้ ส่วน 8052 มีหน่วยความจำถึง 8 กิโลไบต์
มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์สำรับ 8052 มีถึง 256 ไบต์

- หน่วยความจำสำหรับโปรแกรมและข้อมูล (Program Memory และ Data Memory แยกจากกันอย่างละ 64 กิโลไบต์)

- คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 µS เมื่อทำงานที่ความถี่ 12 MHz

- มี Timer/Counter ขนาด 16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด

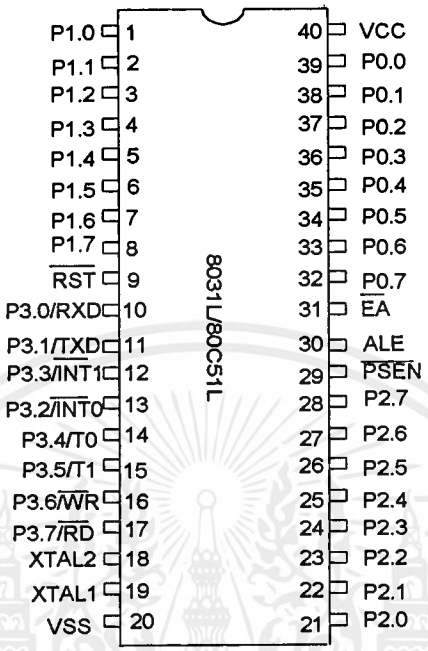
- รับอินเตอร์รับได้ 6 แหล่ง 5 เวคเตอร์

- มีพอร์ทรับและส่งข้อมูลแบบอนุกรม (UART) 2 พอร์ท ทั้งรับและส่งในเวลาเดียวกัน (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมด

- มีคำสั่งในการทำ AND, OR หรือ COMPLEMENT ได้ทั้งแบบ 8 บิต และ 1 บิต

2.5.2 โครงสร้างของ MCS-51

MCS-51 ใช้เทคโนโลยีในการผลิตเป็นแบบ NMOS และ CMOS เบอร์ 8032 และ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวก สำหรับผู้เขียนโปรแกรมที่จะเขียนโปรแกรมด้วยภาษาเบสิก



รูปที่ 2.25 การจัดวางขาของ 8051

2.6 อุปกรณ์ตรวจจับ (Sensor)

ในวงจรอิเล็กทรอนิกส์ในส่วนนำสัญญานเข้าที่ทำหน้าที่เป็นส่วนรับรู้ความรู้สึกต่างๆ เราเรียกว่า ตัวตรวจจับ (Sensor) ซึ่งจะทำการเปลี่ยนแปลงความรู้สึกต่างๆ ที่ได้รับเป็นสัญญาณทางไฟฟ้าซึ่งอาจจะเป็นแรงดันหรือกระแสก็ได้ และส่งให้กับวงจรอิเล็กทรอนิกส์เพื่อตีความหมาย และเอาผลดังกล่าวไปใช้งานได้ตามต้องการ

ตัวตรวจจับแบบพื้นฐานที่เราคุ้นเคยกันอย่างดี เช่น สวิตช์กลไก, สวิตช์แม่เหล็ก, เซลล์รับแสง, โฟโตทรานซิสเตอร์, ออปโต้คัปเปลอร์, ตัวตรวจจับตำแหน่ง, ตัวตรวจจับแรงดัน,ตัวตรวจจับอุณหภูมิ, ตัวตรวจจับเสียง เป็นต้น ตัวตรวจจับต่างๆ เหล่านี้ จะทำหน้าที่เปลี่ยนสถานภาพทางฟิสิกส์ให้เป็นสัญญาณทางไฟฟ้า เพื่อนำไปประยุกต์ใช้งานในวงจรอิเล็กทรอนิกส์ให้สามารถทำงานได้ตามต้องการ

2.6.1 โฟโต้ทรานซิสเตอร์ (Photo Transister)

โดยภาวะปกติ สารกึ่งตัวนำจะมีคุณสมบัติที่ไวต่อแสงอยู่แล้ว เมื่อมีการนำเอาสารกึ่งตัวนำมาสร้างเป็นโฟโต้ทรานซิสเตอร์ โปรตอนจากแสงจะทำให้เกิดอิเล็กตรอนอิสระขึ้น เป็นผลทำให้เกิดการไหลของกระแสไฟฟ้าขึ้นได้ ดังนั้นโฟโต้ทรานซิสเตอร์เป็นตัวตรวจจับแสงชนิดหนึ่งซึ่งถูกออกแบบขึ้นมาจากการเกิดประสบการณ์อย่างหนึ่งของสารกึ่งตัวนำ และมีรอยต่อ P-N ระหว่างสารสองชนิดของ

โฟโตทรานซิสเตอร์ ซึ่งรอยต่อนี้มีขนาดใหญ่กว่ารอยต่อ P-N ของทรานซิสเตอร์โดยทั่วไป ความแตกต่างจากทรานซิสเตอร์ทั่วไปคือที่ตัวถัง (Case) ด้านบนของโฟโต้ทรานซิสเตอร์จะมีช่องสำหรับรับแสงเพื่อส่งไปยังรอยต่อ P-N โดยช่องรับแสงนี้จะมีวัสดุเคลียร์ไมก้า (Clear Mica) หรือควอตซ์เลนซ์ (Quartz Lenz) ติดอยู่บนช่องรับแสงดังกล่าว

วงจรสมมูลย์และการทำงาน วงจรสมมูลย์ของโฟโต้ทรานซิสเตอร์ดังในรูปที่ 2.27 ซึ่งก็คือการนำทรานซิสเตอร์มาต่อร่วมกับโฟโต้ไดโอด โดยตัวโฟโต้ไดโอดจะเป็นตัวควบคุมการจัดแรงดันให้ทรานซิสเตอร์ทำงาน เมื่อเกิดแสงมาตกกระทบที่ตัวโฟโต้ไดโอด จะทำให้เกิดแรงดันไปยังขาเบสของทรานซิสเตอร์ก่อให้เกิดกระแสเบสขึ้น ส่งผลให้ทรานซิสเตอร์ทำงานในที่สุด

ตามปกติการคำนวณหาค่าของกระแสอิมิตเตอร์จะใช้ความสัมพันธ์ดังนี้

$$I_E = I_B . (h_{FE} + 1) \qquad (2.4)$$

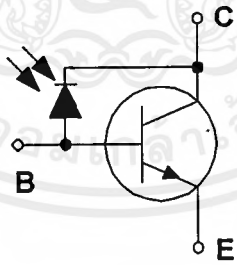
แต่ในกรณีของโฟโต้ทรานซิสเตอร์และเนื่องจากที่เบสและคอลเลคเตอร์มีโฟโต้ไดโอดต่อคร่อมอยู่ดังนั้นเมื่อโฟโต้ทรานซิสเตอร์ทำงานกระแสที่ไหลผ่านตัวโฟโต้ไดโอดจึงต้องนำมาพิจารณาเป็นกระแสไหลเข้าร่วมกับกระแสเบสด้วยทำให้สมการของกระแสอิมิตเตอร์ของโฟโต้ทรานซิสเตอร์จึงกลายเป็น

$$I_E = (I_P + I_B) . (h_{FE} + 1) \qquad (2.5)$$

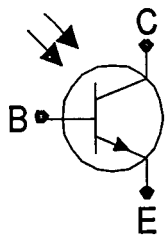
โดยที่ h_{FE} คือ อัตราการขยายกระแสของตัวโฟโต้ทรานซิสเตอร์

I_P คือ กระแสที่ไหลผ่านตัวโฟโต้ไดโอด

I_B คือ กระแสเบสของโฟโต้ทรานซิสเตอร์



รูปที่ 2.26 วงจรสมมูลย์ของโฟโต้ทรานซิสเตอร์



รูปที่ 2.27 สัญลักษณ์ของโฟโต้ทรานซิสเตอร์

สำหรับเครื่องหมายบวกและลบของ I_B ในสมการเป็นตัวบ่งบอกถึงชนิดของทรานซิสเตอร์ หากเป็นโฟโต้ทรานซิสเตอร์ชนิดเอ็นพีเอ็นค่าของ I_B จะเป็นบวก แต่ถ้าเป็นชนิดพีเอ็นพีค่าของ I_B จะเป็นลบ

สำหรับความต้านทานด้านไฟฟ้ากระแสสลับของส่วนรับของโฟโต้ทรานซิสเตอร์จะมีค่าเท่ากับ $R_{in} \times h_{FE}$ ในภาวะที่โฟโต้ทรานซิสเตอร์ยังไม่ทำงานค่าความต้านทานภายใน (R_{in}) ของโฟโต้ทรานซิสเตอร์จะสูงมาก เนื่องจากการที่โฟโต้ไดโอดภายในโฟโต้ทรานซิสเตอร์ถูกไบแอสกลับไว้ ทำให้เกิดค่าความต้านทานสูงมากขึ้น ซึ่งค่าความต้านทานอินพุทนี้เองจะเป็นตัวที่กำหนดความเร็วในการทำงานของตัวโฟโต้ทรานซิสเตอร์ ดังนั้นหากต้องการนำโฟโต้ทรานซิสเตอร์ไปใช้ในงานที่มีการสวิตช์ความเร็วสูงต้องพิจารณาถึงพารามิเตอร์ตัวนี้ด้วย สำหรับสัญลักษณ์ของโฟโต้ทรานซิสเตอร์ดังในรูปที่ 2.26

พารามิเตอร์อีกตัวหนึ่งที่ต้องให้ความสำคัญคือ ค่าของกระแสรั่วไหลที่เกิดขึ้นภายในตัวโฟโต้ทรานซิสเตอร์ในขณะที่ยังไม่ทำงาน นั่นคือกระแสรั่วไหลระหว่างขาของคอลเล็กเตอร์และอิมิตเตอร์ซึ่งจะเกิดขึ้นในขณะที่โฟโต้ทรานซิสเตอร์ยังไม่มีแสงมาตกกระทบให้ตัวมันทำงาน หรือ $I_{CEO(dark)}$ ซึ่งสามารถคำนวณได้จากสมการ

$$I_{CEO(dark)} = h_{FE} \times I_{CBO} \qquad (2.6)$$

โดยที่ I_{CBO} คือ ค่าของกระแสรั่วไหลที่ขาคอลเล็กเตอร์และเบส ซึ่งก็คือกระแสรั่วไหลของตัวโฟโต้ไดโอดนั่นเอง

ปกติในโฟโต้ทรานซิสเตอร์ทั่วๆไปค่าของกระแสรั่วไหลนี้จะต่ำมากๆ อยู่ระหว่าง 4-8 ไมโครแอมป์ที่อุณหภูมิห้อง

2.6.2 อินฟาเรด แอลอีดี (Infrared LED)

อินฟาเรด แอลอีดี ถูกสร้างขึ้นมาเพื่อกำเนิดแสงในย่านอินฟาแรด เมื่ออินฟาเรด แอลอีดีนำกระแส อิเล็กตรอนจะเคลื่อนที่ผ่านสารกึ่งตัวนำชนิดพิเศษ และเกิดพลังงานจากโฟตอน การเกิดพลังงานดังกล่าวเป็นไปในทันที ที่มีกระแสไหลผ่าน

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

บทที่ 3
การสร้างและการออกแบบ

ในบทนี้เราจะอธิบายถึงรายละเอียดการสร้างและการออกแบบของโครงงาน ไมโครเมาส์นี้ โดยจะอธิบายด้านฮาร์ดแวร์ว่ามีการใช้อะไรเป็นส่วนประกอบบ้างและบอกถึงขั้นตอนวิธีการใช้งานด้านซอฟท์แวร์รวมทั้งฟังก์ชั่นพื้นฐานและฟังก์ชั่นการค้นหาเส้นทางเพื่อช่วยในการพัฒนาโปรแกรม

ในส่วนของการสร้างและการออกแบบเราสามารถแบ่งออกได้เป็นส่วน ๆ ดังนี้

ส่วนประกอบของฮาร์ดแวร์

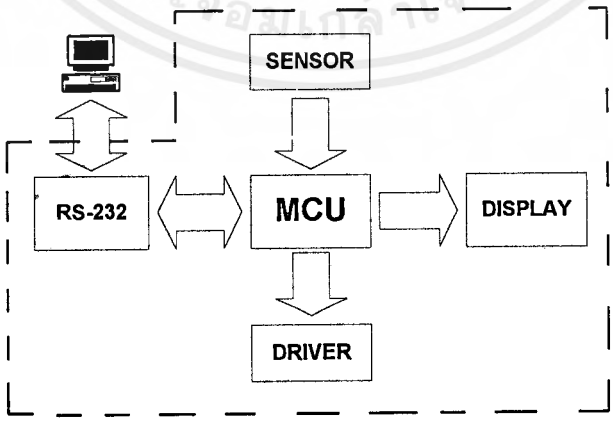
- ไมโครคอนโทรลเลอร์
- เซนเซอร์
- สเต็ปปิ้งมอเตอร์
- แสดงผล
- อุปกรณ์เชื่อมต่อกับอุปกรณ์ภายนอก

ส่วนประกอบของซอฟท์แวร์

- โปรแกรมคอมไพเลอร์
- ฟังก์ชั่นพื้นฐานของไมโครเมาส์
- โปรแกรมแสดงผลทางหน้าจอคอมพิวเตอร์

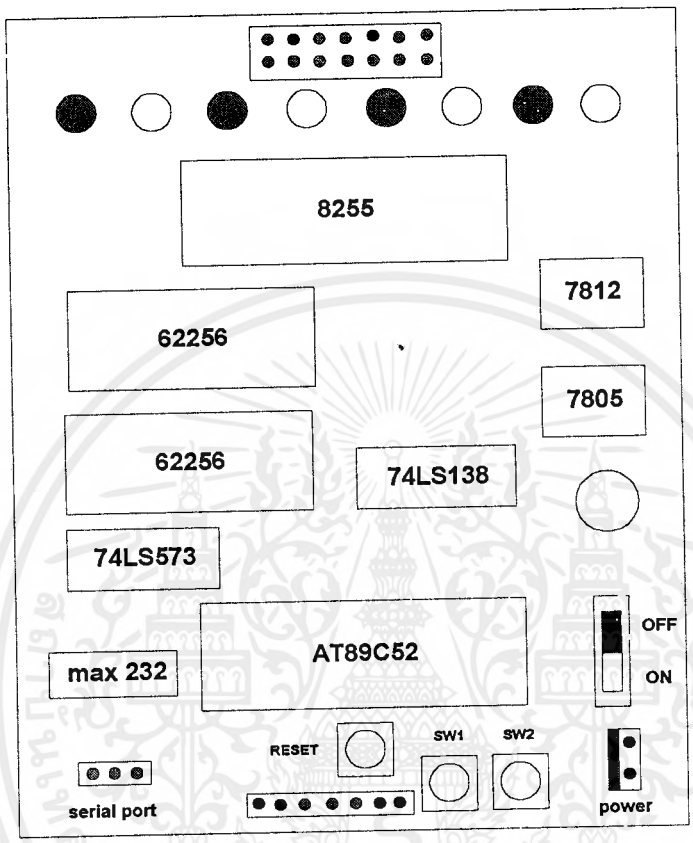
3.1 การออกแบบและการสร้างในส่วนของฮาร์ดแวร์

ในโครงงานไมโครเมาส์ สามารถแบ่งการทำงานออกเป็นส่วน ๆ ดังนี้



รูปที่ 3.1 Block Diagram ของ Micro Mouse

3.1.1 Display เป็นอุปกรณ์เอาท์พุทที่ทำหน้าที่แสดงผลของ ไมโครคอนโทรลเลอร์ ในการต่อLED เราจะต่อผ่านรีซีสเตอร์ค่า 330 ohm ซึ่งเป็นค่าที่เหมาะสมที่จะให้แสงสว่างเพียงพอ ตามปกติแล้ว LED จะกินกระแสประมาณ 10 mA ซึ่งโครงงานนี้ต่อกับพอร์ท A ของ 8255 ซึ่งมีทั้งหมด 8 ตัว

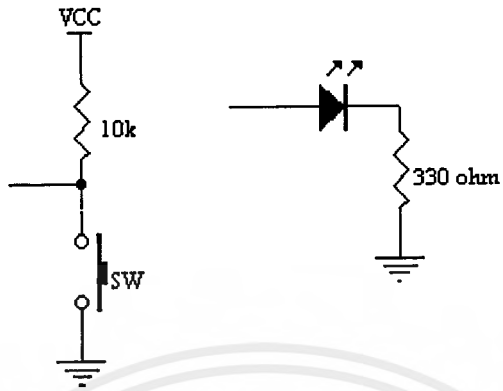


รูปที่ 3.2 แสดงตำแหน่งการวางอุปกรณ์

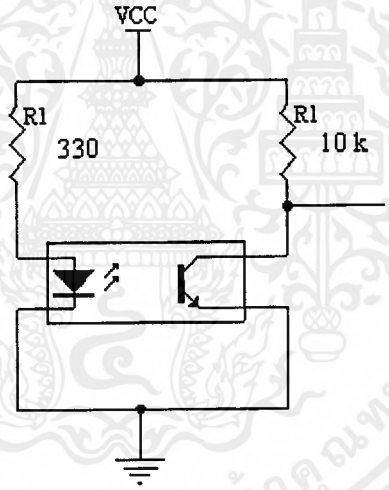
3.12 Switch จะเป็นอุปกรณ์อินพุทให้กับวงจร ในการรับคำสั่งจาก User ซึ่งในการต่อจะต้องมี Resister Pull up ต่ออยู่เพื่อรักษาสถานะใดสถานะหนึ่งให้คงที่ ในกรณีที่ Switch ไม่มีการกด ตามวงจรเบื้องต้นจะได้ค่าลอจิกเป็น "1" เมื่อกด switch จะได้ลอจิก "0" ในโครงงานนี้จะต่ออยู่กับพอร์ท 3 ของไมโครคอนโทรลเลอร์

3.1.3 Sensor เป็นอุปกรณ์ที่รับข้อมูลจาก สิ่งแวดล้อมต่าง ๆ ว่าเป็นยังไงบ้าง ซึ่งในโครงงานนี้จะใช้เป็นตัวตรวจสอบกำแพง อุปกรณ์ที่ใช้ง่ายในการนำมาใช้กับโครงงานนี้คือ Photo Infrared ซึ่งจะเป็นอุปกรณ์ที่หาง่ายและราคาก็ไม่ค่อยแพง หลักการในการทำงานเบื้องต้นนั้นก็คือ ในตัว Sensor จะมีตัวรับและตัวส่งภายในตัวเดียวกัน ในกรณีที่เราทำการส่งแสงออกไปจากตัวส่ง เมื่อแสงนั้นถูกส่งออกแล้วเกิดไปกระทบกับวัสดุ ที่สามารถสะท้อนแสงได้ดีเช่น สีขาว แสงนั้นก็จะถูกสะท้อนไปยังตัวรับแสง Infrared ทำให้มีกระแสไหลในตัวมัน แต่ถ้าไม่มีแสงมากระทบกับตัวรับ กระแสก็จะไม่สามารถ

ไหลผ่านตัวมันได้ ซึ่งหลักการทำงานเบื้องต้นของ photo Infrared จึงสามารถที่จะนำมาใช้ในโครงงานนี้ได้



รูปที่ 3.3 วงจรสวิทช์ และ วงจรDisplay

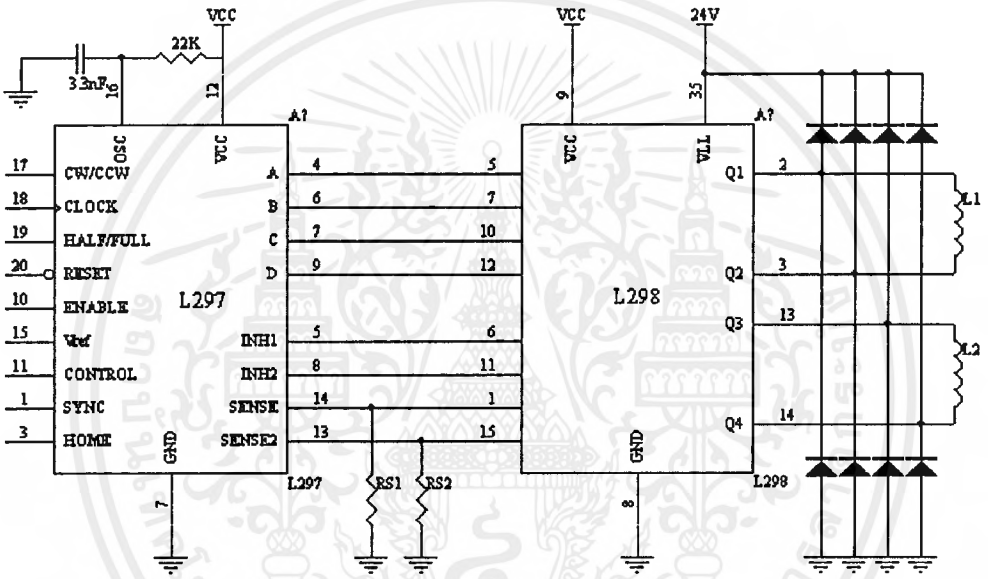


รูปที่ 3.4 วงจร Sensor

3.1.4 Driver วงจรนี้คือวงจรที่ทำหน้าที่เป็นตัวขับให้สเต็ปปิ้งมอเตอร์ให้ทำงานได้ โดยจะรับคำสั่งจากตัวไมโครคอนโทรลเลอร์ ว่าจะให้ทำงานอย่างไร สิ่งที่วงจร Driver ควรจะมีก็คือ วิ่งไปข้างหน้า , วิ่งถอยหลังได้ , หยุดอยู่กับที่ , หยุดการทำงาน , และที่สำคัญส่วนหนึ่งก็คือวิธีการขับ สเต็ปปิ้งมอเตอร์ ซึ่งมีอยู่ 3 แบบ รายละเอียดนี้ได้ถูกบันทึกไว้ในบทที่ 2 แล้ว เมื่อเรากำหนดการทำงานของมันแล้ว เราก็ทำการหาอุปกรณ์อิเลคทรอนิกส์ ที่สามารถตอบสนองความสามารถเรานี้ ในตลาดอุปกรณ์ที่ใช้ในการขับนั้นมีอยู่หลายตัวเช่น L298 , L297 และ TEA3718 , TCA1560 ทั้งสามตัวนี้สามารถที่จะทำงานในสิ่งที่ต้องการได้ แต่เมื่อพิจารณาถึงองค์ประกอบต่าง ๆ แล้ว L298 และ L297 เป็นตัวที่เหมาะสมที่สุด ต่อไปเราก็ต้องมาทำออกแบบวงจร Driver ในขั้นตอนแรก ต้องออกแบบวงจร

ขับสเต็ปปิ้งมอเตอร์ โดยดูวงจรจากวารสารต่าง ๆ หรือ ปริญญานิพนธ์ ก็ได้ ซึ่งก็ได้วงจรที่ใช้ไอซีเบอร์ L297 L298 มาควบคุมการทำงานของมอเตอร์ ตัว L298 จะเป็นตัวขับมอเตอร์ และ ตัว L297 เป็นตัวควบคุม L298 ให้ขับมอเตอร์ตามสัญญาน อินพุทของ L297 ซึ่งมีสัญญาณอินพุท 5 ขาดังนี้

1. CW/CCW เป็นขาที่ใช้ควบคุมการหมุนของมอเตอร์ให้หมุนซ้ายหรือขวา
2. Clock เป็นขาที่ใช้ควบคุมความเร็วถ้าป้อนความถี่มากก็จะวิ่งเร็ว
3. Reset เป็นขาที่บังคับให้มอเตอร์หยุดหมุนแต่มอเตอร์ยังทำงานอยู่
4. Half/Full เป็นขาควบคุมรูปแบบการขับของมอเตอร์ ว่าจะให้ขับแบบ Half หรือ Full Step
5. On/Off เป็นขาที่ควบคุมให้มอเตอร์ทำงานหรือไม่ทำงาน



รูปที่ 3.5 วงจรขับสเต็ปปิ้งมอเตอร์

3.1.5 Memory ซึ่งในโครงงานนี้เราจะแบ่งหน่วยความจำออกเป็น 2 ส่วน คือ

Flash ROM	ขนาด 8 กิโลไบท์	แอดเดรส 0000H – 1FFFH
RAM	ขนาด 52 กิโลไบท์	แอดเดรส 2000H – EFFFH

ตารางที่ 3.1 แสดงการแบ่งหน่วยความจำ

ในส่วนของ Flash ROM นั้นเราจะเก็บโปรแกรมในการควบคุมหน้าที่ต่าง ๆ ดังนี้

- โปรแกรมรับโปรแกรมจากเครื่องคอมพิวเตอร์
- โปรแกรมส่งข้อมูลสถานะของไมโครเมาส์
- โปรแกรมควบคุมความเร็วของสเต็ปปิ้งมอเตอร์

ในส่วนของ RAM นี้สำหรับเก็บโปรแกรมที่รับมาจากเครื่องคอมพิวเตอร์ ซึ่งบนเครื่องคอมพิวเตอร์นั้นจะเขียนขึ้นด้วยภาษา C แล้วคอมไพล์เป็น Assembly แล้วแปลงเป็นภาษาเครื่อง จากนั้นก็ส่งมายังไมโครเมาส์โดยผ่านพอร์ทอนุกรมของเครื่องคอมพิวเตอร์

3.1.6 Port I/O ในส่วนของวงจรนี้เราใช้ 8255 เป็นเชื่อมต่อกับ Sensor และ Display โดยมีตำแหน่งของ Port ดังนี้

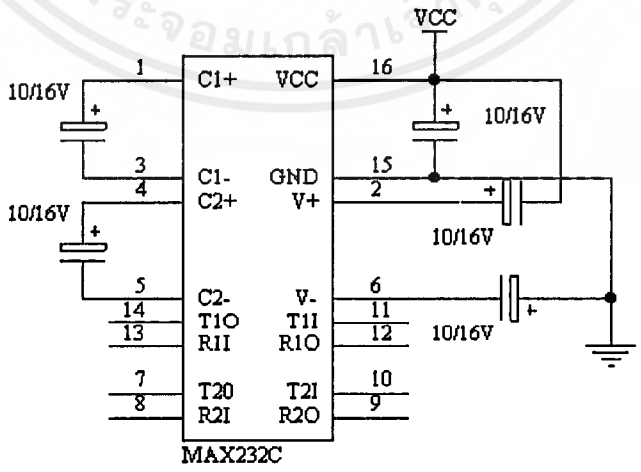
Display เป็น	เอาท์พุท	เชื่อมต่ออยู่ที่	Port A	แอดเดรส	F000H
ใช้งานทั่วไป	อินพุท /เอาท์พุท	เชื่อมต่ออยู่ที่	Port B	แอดเดรส	F001H
Sensor เป็น	อินพุท	เชื่อมต่ออยู่ที่	Port C	แอดเดรส	F002H
พอร์ทควบคุม	เอาท์พุท	เชื่อมต่ออยู่ที่	Port Control	แอดเดรส	F003H

ตารางที่ 3.2 แสดงการใช้งาน Port ของ 8255

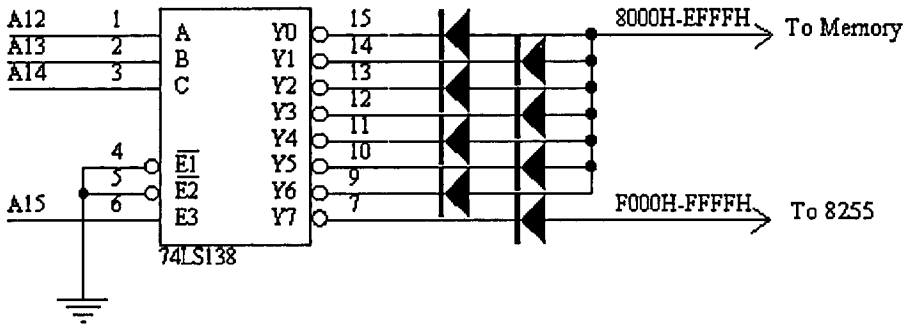
3.1.7 RS-232 Interface ในการเชื่อมต่อวงจรที่ทำงานในระดับแรงดันแบบทีทีแอล (TTL) เข้ากับพอร์ท RS232 ของเครื่องคอมพิวเตอร์ ซึ่งมีระดับแรงดัน –15 ถึง +15 นั้น จะต้องมีวงจรพิเศษเพื่อทำการแปลงระดับแรงดันให้เหมาะสมซึ่งในที่นี้เราได้เลือกใช้ IC เบอร์ MAX 232 ซึ่งเป็น IC ที่ใช้อุปกรณ์ประกอบจากภายนอกน้อย คือใช้ C ชนิดอีเล็คโทรไลท์เพียง 5 ตัวเท่านั้น

หลักการทำงาน

จากวงจรจะใช้ C ที่ต่อระหว่างขา 1 กับ 3 , ระหว่างขา 4 กับ 5 , ขา 2 และ ขา 6 เป็นตัวกำหนดระดับแรงดันในการเชื่อมต่อ โดยขา R1I และ R2I จะเป็นขาที่รับระดับแรงดัน –15 ถึง +15 และแปลงออกเป็นแรงดัน 0V , +5V ออกที่ขา R1O และ R2O ส่วนขา T1I และ T2I ก็จะรับแรงดันที่เป็น 0V , +5V แปลงเป็นระดับแรงดัน –10 ถึง +10 ออกที่ขา T1O และ T2O



รูปที่ 3.6 วงจรรับส่งข้อมูลอนุกรม



รูปที่ 3.7 วงจร DECODER

3.1.8 วงจร Decoder โดยใช้ไอซี 74LS138 เป็นตัวถอดรหัสตำแหน่งของหน่วยความจำ โดยเอาขาแอดเดรสของไมโครคอนโทรลเลอร์ 51 จากตารางที่ 3.3 เราก็จะได้ตำแหน่งดังนี้

แอดเดรส 2000H - 7FFFH	ติดต่อกับ หน่วยความจำตัวที่ 1
แอดเดรส 8000H - EFFFH	ติดต่อกับ หน่วยความจำตัวที่ 2
แอดเดรส F000H - FFFFH	ติดต่อกับ 8255

ตารางที่ 3.3 แสดงการแบ่งเพื่อใช้งานต่างๆ

3.1.9 หลักการทำงานของฮาร์ดแวร์โดยรวม

เริ่มจากตัวไมโครคอนโทรลเลอร์ ซึ่งเราใช้เบอร์ 89C52 จะเชื่อต่อกับชุดขับมอเตอร์ โดยผ่านทางพอร์ท 1 โดยมีรายละของแต่ละขาดังนี้

P1.2	เป็นขาที่กำหนดทิศทางของสเต็ปปิ้งมอเตอร์ตัวที่ 1
P1.3	เป็นขาที่ส่งสัญญาณนาฬิกาออกไป เพื่อกำหนดความเร็วของมอเตอร์ตัวที่ 1
P1.4	เป็นขาที่กำหนดให้สเต็ปปิ้งทั้งสองทำงาน
P1.5	เป็นขาที่กำหนดทิศทางของสเต็ปปิ้งมอเตอร์ตัวที่ 2
P1.6	เป็นขาที่ส่งสัญญาณนาฬิกาออกไป เพื่อกำหนดความเร็วของมอเตอร์ตัวที่ 2

ตารางที่ 3.4 การควบคุมมอเตอร์ผ่านพอร์ท1

ในส่วนของพอร์ท 2 และ 0 จะเป็นตัวเชื่อมต่อกับหน่วยความจำทั้งสองตัวกับตัว 8255 โดยมีไอซี 74LS373 เป็นตัวมัลติเพล็กระหว่างแอดเดรสต่ำกับดาต้า ส่วนไอซี 74LS138 เป็นตัวดีโค้ดพอร์ทเพื่อเลือกชิฟที่ ไมโครคอนโทรลเลอร์จะติดต่อด้วย หน่วยความจำทั้งสองแต่ละตัวจะมีขนาด 32 Kbyte โดยใช้เบอร์ 62256 ซึ่งรวมกันแล้วจะมีขนาด 64 Kbyte แต่ในโครงงานนี้ใช้เพียง 52 Kbyte เท่านั้น โดย 8 Kbyte จะเป็น Flase ROM ซึ่งอยู่ในตัว 89C52 ส่วนอีก 4 Kbyte อยู่ในส่วนของ 8255 ไอซี MAX232

จะเป็นตัวเชื่อมต่อระหว่าง 89C52 กับ เครื่องคอมพิวเตอร์ โดยข้อมูลจะถูกส่งและรับที่ขา TXD,RXD ของ 89C52 ผ่าน ตัวไอซี MAX232 ไปยังเครื่องคอมพิวเตอร์

เมื่อเริ่มเปิดเครื่องตัว 89C52 จะคอยรับข้อมูลที่เป็นโปรแกรมจากคอมพิวเตอร์ ผ่าน MAX232 จากนั้น 89C52 จะนำข้อมูลที่เป็นโปรแกรมลงไปในหน่วยความจำตัวที่ 1 ที่แอดเดรส 2000H ซึ่งโปรแกรมที่ส่งมาต้องมีขนาดไม่เกิน 52 Kbyte เมื่อตัว 89C52 รับข้อมูลจนหมดแล้วตัว 89C52 จะไป Run โปรแกรมที่แอดเดรส 2000H ทันที ในกรณีที่เรารีเซตเครื่องใหม่ ตัวไมโครเมาส์จะรอรับข้อมูลใหม่จากคอมพิวเตอร์ และจะทำการตรวจสอบสวิทตัวที่ 1 ไปพร้อม ๆ กัน โดยถ้ามีการกดที่สวิทตัวที่ 1 89C52 ก็จะกระโดดไปทำงานที่แอดเดรส 2000H ทันที เป็นทำงานโปรแกรมเก่าใหม่อีกครั้ง

3.2 การออกแบบในส่วนของซอฟท์แวร์

3.2.1 การเลือกใช้คอมพายเลอร์ เราได้เลือกใช้ซอพท์แวร์ในการคอมพาย คือ C51 ของ Franklin โดยคอมไพล์จากภาษา C ไปเป็นโค้ดของไมโครโปรเซสเซอร์ตระกูล 8051เพราะการเขียนด้วยภาษา C จะทำให้ง่ายในการทำความเข้าใจและง่ายที่จะนำไปพัฒนาต่อๆไป

ขั้นตอนการใช้งานคอมไพเลอร์ ก่อนใช้งานจะต้องกำหนด PATH ก่อน ดังนี้

PATH=C:\C51;C:\C51\BIN;

SET C51TMP=

SET C51INC=C:\C51\INC

SET C51LIB=C:\C51\LIB

ในกรณีนี้ คอมพายเลอร์อยู่ใน C:\C51

เริ่มเขียนโปรแกรมโดยใช้ อีดิตเตอร์ทั่วไป แล้วบันทึกเป็น

XX.C XX = ชื่อไฟล์ จากนั้นเรียก

C51 XX.C เพื่อทำการคอมพายเป็น XX.OBJ

L51 XX.OBJ co(2000h) xd(8000h) nool เป็นคำสั่ง Linkไฟล์ co แอสเดรสเริ่มต้นของโค้ด xd แอสเดรสเริ่มต้นของDATA

OHS51 XX แปลงโค้ดเป็น Intel Hex Format

หลังจากได้ ไฟล์ XX.HEX แล้ว จะทำการส่งข้อมูลผ่านทางพอรท์อนุกรมไปยังตัวไมโครเมาส์ โดยในตัวไมโครเมาส์จะมีโปรแกรมรอรับการส่งข้อมูลมาเก็บไว้ในหน่วยความจำของไมโครเมาส์

3.2.2 การทำงานของฟังก์ชั่นย่อย

การทำงานของฟังก์ชั่นต่างๆของไมโครเมาส์ มีดังนี้

1. ฟังก์ชั้นติดต่อกับบอร์ดคอลโทรลเลอร์

1.1 อ่านค่าสถานะของสวิท รูปแบบการใช้งาน: char READ_SWITCH(char sw)

ใช้เช็คว่า กดหรือไม่กด sw คือค่าที่จะใช้ในการตรวจสอบว่าเป็นสวิท ตัวที่ 1 หรือ 2 ค่าที่ได้ จะเป็น 1 ถ้าสวิทถูกกด และเป็น 0 เมื่อไม่ถูกกด

โปรแกรมนี้จะตรวจสอบว่าสวิทตัวไหนถูกกด ถ้าสวิทตัวที่ 1 ถูกกดที่ LED จะแสดงค่าเป็น 0xFF ถ้า
สวิท ตัวที่ 2 ถูกกดจะแสดงค่า 0xF0

```
#include "mouse.h"
void main (void)
{   while (1)
    {
        WRITE_LED(0x00);
        if (READ_SWITCH(1))      { WRITE_LED(0xFF); }
        if (READ_SWITCH(2))      { WRITE_LED(0xF0); }
    }
}
```

1.2 แสดงผลที่ LED รูปแบบการใช้งาน: void WRITE_LED(char p)

แสดงค่าตัวแปร p ออกทาง LED เป็นแบบบิต ขนาดจำนวน 8 บิต
โปรแกรมนี้จะแสดงค่า 0x0F ที่ LED จากนั้นก็แสดงค่า 0xF0

```
#include "mouse.h"
void main(void)
{
  WRITE_LED(0x0F);
  DELAY(300);
  WRITE_LED(0xF0);
  DELAY(300);
}
```

2. ฟังก์ชั่นเกี่ยวกับบิท

2.1 อ่านค่าบิท รูปแบบการใช้งาน: unsigned char BIT(int p,unsigned char v)

SET บิท รูปแบบการใช้งาน: unsigned char SET(int p,unsigned char v)

CLR บิท รูปแบบการใช้งาน: unsigned char CLR(int p,unsigned char v)

BIT เป็นการเช็คว่าบิทนั้นเป็น 0 หรือ 1 โดยดูจาก บิทที่ p ของตัวแปร v

SET เป็นการทำให้ค่าบิทที่ p ของตัวแปร v มีค่าเป็น 1 แล้วส่งค่าที่ได้ใหม่กลับออกมา

CLR เป็นการทำให้ค่าบิทที่ p ของตัวแปร v มีค่าเป็น 0 แล้วส่งค่าที่ได้ใหม่กลับออกมา

โปรแกรมนี้จะแสดงค่าแรกเป็น 0 ซึ่งเป็นตัวตรวจสอบบิท 0 ของตัวแปร ch

แสดงค่าที่ 2 เป็น 0x78 ซึ่งเป็นคำสั่งที่เคลียร์บิทที่ 1 ของตัวแปร ch

แสดงค่าที่ 3 เป็น 0x7E ซึ่งเป็นคำสั่งที่เซตบิทที่ 2 ของตัวแปร ch

```
#include "mouse.h"
void main(void)
{ unsigned char ch;
        ch = 0x7A;
        WRITE_LED(BIT(0,ch));DELAY(300);
        WRITE_LED(CLR(1,ch));DELAY(300);
        WRITE_LED(SET(0,ch));DELAY(300);
}
```

2.2 เลื่อนบิทไปทางซ้ายชุดละ 4 บิท รูปแบบการใช้งาน: unsigned char ROTL4(unsigned char v);

เลื่อนบิทไปทางขวาชุดละ 4 บิท รูปแบบการใช้งาน: unsigned char ROTR4(unsigned char v);

ROTL4 ฟังก์ชั่นนี้จะเลื่อนบิทแบบ nibbles ของตัวแปร v ไปทางซ้าย1 ครั้ง ถ้าตัวแปร v เป็น " abcd efgh " มันจะกลายเป็น " dabc hefg "

ROTR4 ฟังก์ชั่นนี้จะเลื่อนบิทแบบ nibbles ของตัวแปร v ไปทางขวา 1 ครั้ง ถ้าตัวแปร v เป็น " abcd efgh " มันจะกลายเป็น " bcda fghe "

โปรแกรมนี้จะทำการเลื่อนค่า 0x7A แบบ 4 บิทโดยค่าแรกที่ได้จะมีค่าเท่ากับ 0xE5 เป็นการเลื่อนบิท ทางซ้าย ส่วนค่าที่ 2 จะได้เป็น 0xB5 เป็นการเลื่อนบิททางขวา

```
#include "mouse.h"
void main(void)
{ unsigned ch;
    ch = 0x7A;
  WRITE_LED(ROTL4(ch));DELAY(300);
  WRITE_LED(ROTR4(ch));DELAY(300);
}
```

2.3 เลื่อนบิทไปทางซ้ายชุดละ 8 บิท รูปแบบการใช้งาน: unsigned char ROTL8(unsigned char v);

เลื่อนบิทไปทางขวาชุดละ 8 บิท รูปแบบการใช้งาน: unsigned char ROTR8(unsigned char v);

ฟังก์ชั่น **ROTR8 ()** จะทำการเลื่อนบิทของตัวแปร **v** ไปทางขวา ถ้าตัวแปร **v** มีค่าเป็น "abcdefgh" ผลที่ได้ก็จะมีค่าเป็น "habcdefg"

ฟังก์ชั่น **ROTL8 ()** จะทำการเลื่อนบิทของตัวแปร **v** ไปทางซ้าย ถ้าตัวแปร **v** มีค่าเป็น "abcdefgh" ผลที่ได้ก็จะมีค่าเป็น "bcdefgha"

โปรแกรมนี้จะทำการเลื่อนค่า 0x7A แบบ 8 บิทโดยค่าแรกที่ได้จะมีค่าเท่ากับ 0xF5 เป็นการเลื่อนบิท ทางซ้าย ส่วนค่าที่ 2 จะได้เป็น 0x3D เป็นการเลื่อนบิททางขวา

```
#include "mouse.h"
void main(void)
{ unsigned ch;
      ch = 0x7A;
      WRITE_LED(ROTL8(ch));DELAY(300);
      WRITE_LED(ROTR8(ch));DELAY(300);
}
```

2.4 กำหนดตำแหน่งเป้าหมาย รูปแบบการใช้งาน: void SET_GOAL(void);

กำหนดตำแหน่งปลายทางโดยเมื่อใช้ฟังก์ชั่นนี้แล้วรถจะรับการกดปุ่มที่ 1 ซึ่งจะมีการแสดงค่าที่ display โดยเป็นค่าตำแหน่งในแกน X เมื่อกดไปเรื่อยๆ ค่าก็จะเปลี่ยนไปเมื่อได้ค่าที่ต้องการก็กดปุ่มที่ 2 ก็จะเปลี่ยนเป็นการตั้งค่าตามแนวแกน Y เมื่อกดไปเรื่อยๆ ค่าก็จะเปลี่ยนไปแบบเดียวเหมือนกับค่า X เมื่อได้ค่า ที่ต้องการแล้วก็กดปุ่มที่ 2 อีกที ไมโครเมาส์จะเก็บตำแหน่งปลายทางนี้ไว้ในตัวแปร GOAL_X และ GOAL_Y เพื่อใช้ในการทำงานต่อไป

โปรแกรมนี้ จะทำการเซตค่าตำแหน่ง goal ของไมโครเมาส์

```
#include "mouse.h"
void main(void)
{       SET_GOAL(); }
```

3. ฟังก์ชั่นเกี่ยวกับการเคลื่อนไหว

3.1 บังคับล้อ รูปแบบการใช้ งาน : void MOTOR(char x,char y)

ควบคุม MOTOR ให้หมุนไปตามทิศทางและความเร็วต่างๆ ค่า X = LEFT หรือ RIGHT, ค่า Y = FORWARD ,REWARD,LO,MID_LO,MID_HI,HI,COMPENSATE เช่น MOTOR (LEFT,FORWARD) สั่งให้มอเตอร์ล้อซ้ายเดินหน้า

โปรแกรมข้างล่างนี้จะสั่งให้มอเตอร์ทางด้านขวาวิ่งไปข้างหน้า ส่วนข้างซ้ายจะหยุดกับที่ด้วยความเร็วระดับหนึ่ง จากนั้นก็จะให้มอเตอร์ทางขวาหยุดหมุน ส่วนมอเตอร์ซ้ายก็จะวิ่งถอยหลังด้วยความเร็วที่สูงขึ้นกว่าเดิม

```
#include "mouse.h"
void main(void)
{       MOTOR(MID_HI);
        MOTOR(RIGHT,FORWARD);
        DELAY(1000);
        MOTOR(HI);
        MOTOR(LEFT,REVERSE):
        DELAY(1000);
}
```

3.2 Clear จำนวน STEP การหมุนของล้อ รูปแบบการใช้งาน: void RESET_STEP(void)

ทำให้ค่าของการนับจำนวนการหมุนของมอเตอร์เป็น 0 ทั้ง 2 ตัว ใช้ในการ clear ค่าก่อนที่จะทำการนับ step การหมุนของมอเตอร์

โปรแกรมนี้จะสั่งให้ไมโครเมาส์วิ่งไปข้างหน้าด้วยความเร็วที่สูงขึ้น โดยความเร็วที่สูงขึ้นจะขึ้นอยู่จำนวนสเตป ของมอเตอร์คือตัวแปร STEP_HI_L เป็นสเตปของมอเตอร์ซ้าย

```
#include "mouse.h"
void main(void)
{       RESET_STEP();
        CAR(FORWARD);
        if (STEP_HI_L < 1) { CAR(MID_LO);}
        if (STEP_HI_L > 1 ) { CAR(MID_HI);}
        if( STEP_HI_L > 2 ) { CAR(HI);}
}
```

3.3 อ่านค่าจำนวน STEP การหมุนของล้อ รูปแบบการใช้งาน: unsigned int READ_STEP(char p)

อ่านค่าจำนวนสเต็ปการหมุนของมอเตอร์โดย ค่า p จะเป็น ค่า LEFT หรือ RIGHT คือ อ่านค่าของล้อซ้ายหรือขวา ค่าที่ได้จะเป็นค่าแบบ 16 บิต

โปรแกรมนี้จะแสดงค่าตามสเตปโดยค่าของสเตปมีค่าน้อยกว่า 300 จะแสดงค่าเป็น 1 สเตปมีค่ามากกว่า 300 จะแสดงค่า 2 ถ้าสเตปมีค่ามากกว่า 600 จะแสดงค่า 3

```
#include "mouse.h"
void main(void)
{ unsigned int  i;
        CAR(FORWARD);
        CAR(MID_HI);
        while (1)
        { i = READ_STEP() ;
        if (i < 300) { WRITE_LED(1); }
        if (i > 300) { WRITE_LED(2); }
        if (i > 600) { WRITE _LED(3); }
        }
}
```

3.4 หน่วงเวลา รูปแบบการใช้งาน : void DELAY(int num)

หน่วงเวลาการทำงานของโปรแกรม

3.5 อ่านสถานะของเซนเซอร์ รูปแบบการใช้งาน: char READ_SENSOR(void)

อ่านค่าเซนเซอร์ ว่ามีสถานะอย่างไรในแต่ละตำแหน่งของเซนเซอร์ ค่าที่ได้จะเป็นสถานะของเซนเซอร์ทั้ง 8 บิต

โปรแกรมนี้จะอ่านค่าจากเซนเซอร์ แล้วนำค่าไปแสดงที่ LED ให้ลองเอามือมาแตะที่เซนเซอร์ ดู LED ก็จะติดตามเซนเซอร์ตัวนั้น

```
#include "mouse.h"
void main(void)
{       while (1)
        { WRITE_LED(READ_SENSOR()); }
```

```
}
```

3.6 อ่านค่าลักษณะกำแพงรอบๆ รูปแบบการใช้งาน : char SENSOR(void)

อ่าน SENSOR ให้ O/P เป็นรูปแบบต่าง ๆ ของกำแพง โดยมีค่าดังนี้ บิท 1 จะแสดงถึงกำแพงทางด้านขวาว่ามี (1) หรือไม่มี (0) บิท 2 จะแสดงถึงกำแพงทางด้านหน้าว่ามี (1) หรือไม่มี (0)บิท 3 จะแสดง ถึงกำแพงทางด้านซ้ายว่ามี (1) หรือไม่มี (0) ทำให้มีลักษณะกำแพงได้ 8 แบบ (0-7) ดังนั้นเมื่อ มีค่าเป็น 5 หมายความว่ามีกำแพงด้านซ้ายและขวา ถ้ามีค่าเป็น 6 หมายความว่ามีกำแพงด้านซ้ายและหน้า

โปรแกรมข้างล่างนี้จะอ่านค่าลักษณะกำแพง จากนั้นนำค่าเหล่านี้ไปแสดงที่ LED

```
#include "mouse.h"
void  main(void)
{  unsigned char  ch;
   while (1)
        {  ch = SENSOR();
           WRITE_LED(ch);
        }
}
```

3.7 เปลี่ยนทิศทาง รูปแบบการใช้งาน : void DIRECTION(char d)

เปลี่ยนทิศรถตามการเลี้ยว ให้ I/P เป็น 'L', 'R', 'U' O/P เป็นค่า DIRECTION ที่ เปลี่ยนไปตามทิศที่เลี้ยว เช่น ถ้ารถอยู่ทิศเหนือแล้วสั่ง DIRECTION(L) ทิศของรถจะเปลี่ยนเป็นทิศตะวันตก

โปรแกรมข้างล่างนี้ จะกำหนดให้ทิศทางเป็นทิศเหนือก่อน ซึ่งค่า DIR มีค่าเท่ากับ 0 จากนั้นก็จะถูกเปลี่ยนทิศทางโดยคำสั่ง DIRECTION(L) ค่าของ DIR ก็จะไปแสดงที่ LED ซึ่งมันจะเปลี่ยนไปเรื่อย ๆ เป็น 0,3,2,1,0,.....

```
#include "mouse.h"
void main(void)
{       DIR = NORTH;
        while (1)
        {
        DIRECTION(L);
        WRITE_LED(DIR);
        DELAY(300);
        }
```

}

3.8 แปลงค่ากำแพง รูปแบบการใช้งาน : char CONVERSE(char w)

เปลี่ยนรูปแบบกำแพงให้หันไปทางทิศเหนือ เพื่อเก็บลงในแผนที่สนาม I/P เป็นค่ากำแพง O/P กำแพงรูปใหม่ เนื่องจากการอ่านค่ากำแพงหลายครั้งที่รถอยู่ในทิศที่ไม่เหมือนกันทำให้ลำบากในการทำแผนที่สนามจึงต้องมีการเปลี่ยนรูปแบบกำแพงให้เป็นทิศเหนือเสมอก่อนที่จะเก็บรูปแบบกำแพง ดังนั้นรูปแบบกำแพงจะมีทั้งหมด 15 แบบ (0-14) เช่น สมมุติว่ารถอยู่ทิศใต้และอ่านค่ากำแพงได้ = 6 คือมีด้านซ้ายกับด้านหน้า ถ้ามองทางทิศเหนือจะได้ค่ากำแพงเป็น =12

โปรแกรมข้างล่างนี้ เราได้กำหนดทิศทางของรถให้อยู่ที่ทิศเหนือ และสมมติให้ค่ากำแพงเป็น 6 เมื่อเราทำคำสั่ง CONVERSE มันก็จะทำการเป็นค่ากำแพงให้อยู่ในทางทิศเหนือ เราก็จะได้ค่ากำแพงเป็น 12

```
#include "mouse.h"
void main(void)
{ char ch;
   DIR = SOUTH;   WALL = 6;
   ch = CONVERSE(WALL);
   WRITE_LED(ch);
   for (;;){}
}
```

3.9 เปลี่ยนค่าตำแหน่งรถ รูปแบบการใช้งาน : void CHG_POINT(void)

เปลี่ยนค่าตำแหน่งรถ (X,Y) โดยการเปลี่ยนค่าจะดูจากทิศทางที่รถวิ่ง และ แสดงผลออก DISPLAY ทิศเหนือ X จะบวก 1 ทิศใต้ X จะลบ 1 ทิศตะวันออก Y จะบวก 1 ทิศตะวันตก Y จะลบ 1

โปรแกรมข้างนี้จะกำหนดให้ค่าทิศทางเป็นทิศเหนือ และให้ค่าตำแหน่งของไมโครเมาส์อยู่ X = 0 และ Y=0 จากนั้นเมื่อใช้คำสั่งนี้ค่าที่ได้ก็คือ X = 1 ,Y = 0 นำค่าทั้งสองแสดงที่ LED

```
#include "mouse.h"
void main(void)
{       X = Y = 0;
        DIR = NORTH;
        CHG_POINT()
        WRITE_LED(X);
        DELAY(300);
```

```
        WRITE_LED(Y);
    }
```

3.10 ควบคุมรถ รูปแบบการใช้งาน : void CAR(char f)

ควบคุมรถให้วิ่งแบบต่างๆ ค่า f = FORWARD,REWARD,LO,MID_LO,MID_HI ,HI,LEFT ,RIGHT เช่น CAR(LEFT) สั่งให้ ไมโครเมาส์เลี้ยวซ้าย CAR(HI) สั่งให้ ไมโครเมาส์ใช้ สปีดเร็ว

โปรแกรมนี้ จะสั่งให้ไมโครเมาส์วิ่งไปข้างหน้าจนครบ 1 ช่องมันก็จะหยุด

```
#include "mouse.h"
void main(void)
{       CAR(MID_HI);
        CAR(FORWARD)
        while ( END_B() == 0) {}
        CAR(STOP);
        DELAY(300);
}
```

3.11 ปรับการเอียงของรถ รูปแบบการใช้งาน : void CARLIBATE(void)

ชุดเซนเซอร์การเอียงของรถโดยการดูจากเซนเซอร์ ถ้ารถเอียงซ้ายก็เร่งล้อซ้าย ถ้ารถเอียงขวาก็เร่งล้อขวา

โปรแกรมนี้จะวิ่งไปข้างหน้า โดยการวิ่งนั้นจะมีการปรับตัวของไมโครเมาส์ให้วิ่งเป็นแนวเส้นตรง เพราะฉนั้นเราต้องต่อกำแพงให้มันด้วย จึงจะทำให้โปรแกรมนี้ทำงานได้

```
#include "mouse.h"
void main(void)
{       CAR(FORWARD);
        CAR(MID_HI);
        while (1)
          { CARLIBATE(); }
}
```

3.12 ตรวจสอบทางวิ่งต่อไป รูปแบบการใช้งาน : char NEXT_BLOCK(char t)

• ดูว่าทางที่จะไปได้เคยไปหรือยัง I/P เป็นทิศทางต่อไป O/P เป็นค่า '1' ไปแล้ว หรือ '0' ยังไม่เคยไป

โปรแกรมนี้จะตรวจสอบว่า บล็อกทางด้านขวาของบล็อกที่ X=5,Y=5 เคยไปแล้วหรือยังถ้ายังจะได้ค่าเป็น 0 ถ้าไปแล้วจะมีค่าเป็น 1 ผลของโปรแกรมนี้จะเป็น 0

```
#include "mouse.h"
void main(void)
{       X = Y = 5;
        while (1)
        {
          WRITE_LED(NEXT_BLOCK(RIGHT));
        }
}
```

3.13 การเคลื่อนที่ รูปแบบการใช้งาน : void MOVE_CAR(char d,char l)

สั่งรถให้วิ่งไปตามทิศ (X) เป็นจำนวนระยะทาง (Y) บล็อก

โปรแกรมจะสั่งให้ ไมโครเมาส์วิ่งไปข้างหน้า 5 ช่อง จากนั้นจะเลี้ยวขวาแล้ววิ่งไป 3 ช่อง จากนั้นเลี้ยวขวาแล้ววิ่งไป 4 ช่อง

```
#include "mouse.h"
void main(void)
{       MOVE_CAR(NORTH,5);
        MOVE_CAR(EAST,3);
        MOVE_CAR(SOUTH,4);
}
```

3.14 การเคลื่อนที่ 1 บล็อก รูปแบบการใช้งาน: void MOVE_BLOCK(void);

สั่งให้รถเคลื่อนที่ไปข้างหน้า 1 บล็อกแล้วหยุด จะมีการเปลี่ยนแปลงเกิดขึ้นคือ ค่าตำแหน่งของรถ X,Y จะเปลี่ยนไปและรถจะส่งอ่านค่ากำแพงส่งมาให้ในตัวแปร WALL

โปรแกรมนี้ จะสั่งให้ไมโครเมาส์วิ่งไปข้างหน้าจำนวน 1 บล็อก จากนั้นก็จะเลี้ยวขวาแล้ววิ่งไปข้างหน้า 2 บล็อกเพราะได้มีการเรียกฟังก์ชั่นนี้ 2 ครั้ง จากนั้นไมโครเมาส์ก็จะหยุด

```
#include "mouse.h"
void main(void)
```

```
{       X = Y = 0;
        CAR(MID_HI);
        CAR(FORWARD);
        MOVE_BLOCK();
        CAR(RIGHT);
        MOVE_BLOCK();
        MOVE_BLOCK();
        CAR(STOP);
}
```

3.15 การเคลื่อนที่อย่างเร็ว รูปแบบการใช้งาน : void FAST_RUN(char d,char u)

รถจะวิ่งอย่างเร็วไปทางทิศ d เป็นจำนวน u บล๊อก

โปรแกรมนี้ จะสั่งให้ไมโครเมาส์วิ่งไปข้างหน้าด้วยความเร็วสูง จำนวน 4 บล๊อก แล้วเลี้ยวขวาจากนั้นก็วิ่งไปข้างหน้าด้วยความเร็วที่สูงขึ้นจำานวน 5 บล๊อก แล้วเลี้ยวขวาจากนั้นก็วิ่งไปข้างหน้าด้วยความเร็วที่สูงขึ้นจำนวน 6 บล๊อก

```
#include "mouse.h"
void main(void)
{
        CAR(FORWARD);
        FAST_RUN(NORTH,4);
        FAST_RUN(EAST,5);
        FAST_RUN(SOUTH,6);
        CAR(STOP);
}
```

4. ฟังก์ชันเกี่ยวกับการเก็บแผนที่สนาม

4.1 แปลงค่าตำแหน่งสนาม รูปแบบการใช้งาน: int ARR(char x,char y)

ใช้ทำการแปลงค่าตำแหน่ง X,Y ของสนามไปเป็นค่าตำแหน่งของอาเรย์ของสนาม ณ จุด X,Y นั้นๆ โดยการเก็บค่าในอาเรย์จะเป็นแบบมิติเดียวแต่สามารถอ้างตำแหน่งเป็นแบบ 2 มิติได้ ในการแปลงค่านั้นจะอ้างอิงกับค่าตัวแปร MAX_X และ MAX_Y ถ้าเราเปลี่ยนค่านี้ ค่าอาเรย์ที่ได้ก็จะเปลี่ยนไปด้วย

โปรแกรมนี้จะทำการแปลงค่า X และ Y ให้เป็นค่าได้จะคิดจากสูตร (X*16) +Y ดังนั้นผลที่ได้มีค่าเท่ากับ 69 ค่าที่ได้จะแสดงที่ LED

```
#include "mouse.h"
void main(void)
{       while (1)
        { WRITE_LED(ARR(4,5)); }
        }
```

4.2 MARK สนาม รูปแบบการใช้งาน: void MARK_MAP(char x,char y)

ใช้ในการใส่ค่าในอาเรย์ของสนามว่าได้ไปแล้ว ก็ MARK สนามตำแหน่งนั้นให้มีค่าเป็น 1

โปรแกรมนี้ จะทำการ mark ที่ตำแหน่ง X= 4 , Y=3 เป็นตำแหน่งด้านบนของตำแหน่ง X,Y และ unmark ที่ตำแหน่ง X=5 Y=4 อยู่ทางขวาของตำแหน่ง X,Y จากนั้นเราจะทำการตรวจสอบว่าบล๊อกด้านขวาและด้านบนถูก unmark แล้วหรือยัง ในกรณีทางขวาถูก unmark ที่LED ก็จะแสดงค่า 0xF0

```
#include "mouse.h"
void main(void)
{ X = Y = 3;
        MARK_MAP(X+1,Y);
        UNMARK_MAP(X,Y+1);
        if (NEXT_BLOCK(RIGHT) == 0) { WRITE_LED(0xF0);
        if (NEXT_BLOCK(FRONT) == 0) { WRITE_LED(0x0F);
}
```

4.3 UNMARK สนาม รูปแบบการใช้งาน: void UNMARK_MAP(char x,char y)

ก็จะทำให้ค่าการ MARK สนามของตำแหน่งนั้นเป็น 0 ปกติตอนเริ่มต้นจะมีค่าเป็น 0 หมดทั้งสนามอยู่แล้ว

ดูตัวอย่างจาก การ MARK สนาม

4.4 บันทึกค่ากำแพง รูปแบบการใช้งาน: void WRITE_MAP(char x,char y,char w)

เป็นการแปลงค่าลักษณะกำแพงที่ได้ (w) ให้เป็นค่าที่หันไปในทิศเดียวกันก่อนที่จะเก็บค่ากำแพงนี้ลงอาเรย์ของสนามของตำแหน่ง x,y นั้นๆ

โปรแกรมนี้เราจะส่งค่ากำแพงลงใน map ที่ตำแหน่ง X = 4 ,Y = 4 ในทิศใต้ ซึ่งค่ากำแพงมีค่าเท่ากับ 7 ค่กำแพงนั้นก็จะถูกแปลงให้อยู่ในรูปแบบของกำแพงในทางทิศเหนือ (ให้ดูฟังก์ชั่น CONVERSE) จากนั้นเราก็ทำการอ่านค่ากำแพงนี้ออกมาอีกครั้งหนึ่งค่าถูกแปลงแล้วก็จะได้เป็น 15 แสดงผลที่ LED

```
#include "mouse.h"
void main(void) {
        DIR = SOUTH;
        WALL  = 7;
        WRITE_MAP(X,Y,WALL);
        ch = READ_MAP(X,Y);
        WRITE_LED(ch);
}
```

4.5 อ่านค่ากำแพงจากแผนที่สนาม รูปแบบการใช้งาน: char READ_MAP(char x,char y)

อ่านค่าลักษณะของกำแพงที่เก็บไว้ในอาเรย์ของสนาม ณ ตำแหน่ง x,y นั้นๆ
ดูตัวอย่างจากฟังก์ชั่นบันทึกค่ากำแพง

4.6 เก็บตำแหน่งทางแยก รูปแบบการใช้งาน : void PUSH_WAY(char x,char y)
อ่านตำแหน่งทางแยก รูปแบบการใช้งาน : void POP_WAY(void)

เก็บตำแหน่งทางแยกที่ยังไม่ได้ไปไว้ในอาเรย์ คือค่า x และ y เพื่อนำไปใช้ในการตรวจเช็คว่าไมโครเมาส์วิ่งไปทั่วทั้งสนามแล้วโดยจะมีการทำงานเหมือนกับสแต็กคือเมื่อมีการเก็บก็มีการใส่ค่า เมื่อมีการอ่านก็มีการลบค่าในอาเรย์และนำค่าที่ได้ไปใส่ไว้ในตัวแปร SX และ SY โดยมีตัวแปร G เป็นตัวชี้ตำแหน่งของข้อมูลเหมือน pointer top of stack

โปรแกรมนี้จะทำการเก็บตำแหน่งของทางแยก X=3 Y=5 และ X =6 Y=2 และ ทำการอ่านค่าทางแยกล่าสุด และจะมีการแสดงค่าตัวชี้ตำแหน่งทาง LED

```
#include "mouse.h"
void main(void)
{ char x,y;
        x = 3; y = 5;
        PUSH_WAY(x,y);
        WRITE_LED(G);
        DELAY(1000);
        x = 6; y = 2;
        PUSH_WAY(x,y);
        WRITE_LED(G);
        DELAY(1000);
```

```
        POP_WAY( );
        WRITE_LED(G);
        DELAY(1000);
    }
```

4.7 ส่งข้อมูลไปยังคอมพิวเตอร์ รูปแบบการใช้งาน : void send_port(char ch)

ส่งค่าที่ต้องการจะรู้ออกสู่หน้าจอคอมพิวเตอร์ ค่า ch เป็นค่าที่ต้องการแสดงผล ถ้าจะให้แสดงค่าตำแหน่ง หรือ กำแพงจะต้องส่งเป็น โปโตคอล ตามรูปแบบในในเรื่องการแสดงสถานะบนหน้าจอ

โปรแกรมนี้จะทำการส่ง ค่ากำแพง ตำแหน่ง X,Y และทิศทาง ไปยังพอร์ทอนุกรมของเครื่องคอมพิวเตอร์ เพื่อให้ไปแสดงผลบนหน้าจอ เมื่อเรา download โปรแกรมนี้ลงไมโครเมาส์แล้ว ให้เรารันโปรแกรม STATUS เพื่อดูผลการทำงาน จากนั้นก็สั่งให้ไมโครเมาส์ทำงาน โดยการกดรีเซตใหม่แล้วกด sw1 บนหน้าจอมอนิเตอร์ก็จะแสดงกำแพงพร้อมกับลูกศรที่ชี้ไปทางทิศเหนือที่ตำแหน่ง X=5 ,Y = 7

```
#include "mouse.h"
void main(void)
{       send_port(0xEE);
        send_port(0x0B)
        send_port(5);
        send_port(7);
        send_port(0);
        send_port(0xEE);
}
```

5. ฟังก์ชั่นเกี่ยวกับการค้นหาเส้นทาง

5.1 slove_path รูปแบบการใช้งาน : void slove_path(char x_start, y_start,x_goal,y_goal,limit);

หาเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดเป้าหมายโดยจะทำการคำนวณจากสนามที่เก็บไว้ในหน่วยความจำค่าที่ได้จะเป็นลักษณะทิศทางและระยะทางเป็นคู่ๆจัดเก็บเรียงลำดับอยู่ในอาเรย์ slove และมีค่า count เป็นตัวบอกจำนวนของอาเรย์ที่ใช้โดยสามารถนำค่าในอาเรย์นี้ไปใช้กับฟังก์ชั้น

MOVE_CAR ได้ ค่า limit ในฟังก์ชั่นนี้เป็นระยะทางสูงสุดของการคำนวณให้กำหนดตามความเหมาะสมของขนาดสนาม

โปรแกรมนี้ จะสมมติค่ากับแพงออกมา จากนั้นก็จะเรียกฟังก์ชั่น slove_path เพื่อทำการหาเส้นทางที่สั้นที่สุด ค่าที่ส่งกลับมาคือ ตัวแปร slove ซึ่งเก็บค่าเส้นทาง ส่วนตัวแปร count จะเก็บค่าจำนวนข้อมูลที่อยู่ในตัวแปร slove จากนั้นก็จะทำการวิ่งจากจุด 0,0 ไปยังจุด 5,5 แล้วก็จะหยุด

```
#include "mouse.h"
void main(void)
{ int  i;
temp_box[0] = 2; temp_box[1] = 1;          temp_box[2] = 5; temp_box[3] = 6;
temp_box[4] = 3; temp_box[5] = 5; temp_box[6] = 6;
temp_box[16] = 11;      temp_box[17] = 5;      temp_box[18] = 7;      temp_box[19] = 14;
temp_box[20] = 9;       temp_box[21] = 4;      temp_box[22] = 10;
temp_box[32] = 8;       temp_box[33] = 2;      temp_box[34] = 10;     temp_box[35] = 9;
temp_box[36] = 5;       temp_box[37] = 7;      temp_box[38] = 14;
temp_box[48] = 3;       temp_box[49] = 12;     temp_box[50] = 11;     temp_box[51] = 5;
temp_box[52] = 6;       temp_box[53] = 10;     temp_box[54] = 10;
temp_box[64] = 11;      temp_box[65] = 5;      temp_box[66] = 14;     temp_box[67] = 1;
temp_box[68] = 15;      temp_box[69] = 12;     temp_box[70] = 10;


temp_box[80] = 9;       temp_box[81] = 4;      temp_box[82] = 9;      temp_box[83] = 5;
temp_box[84] = 13;      temp_box[85] = 1;      temp_box[86] = 12;

slove_path(0,0,5,5,20);
for (i = 0;i < count;i++)
{ FAST_RUN(SLOVE[i],SLOVE[i+1]) }
CAR(STOP);
}
```

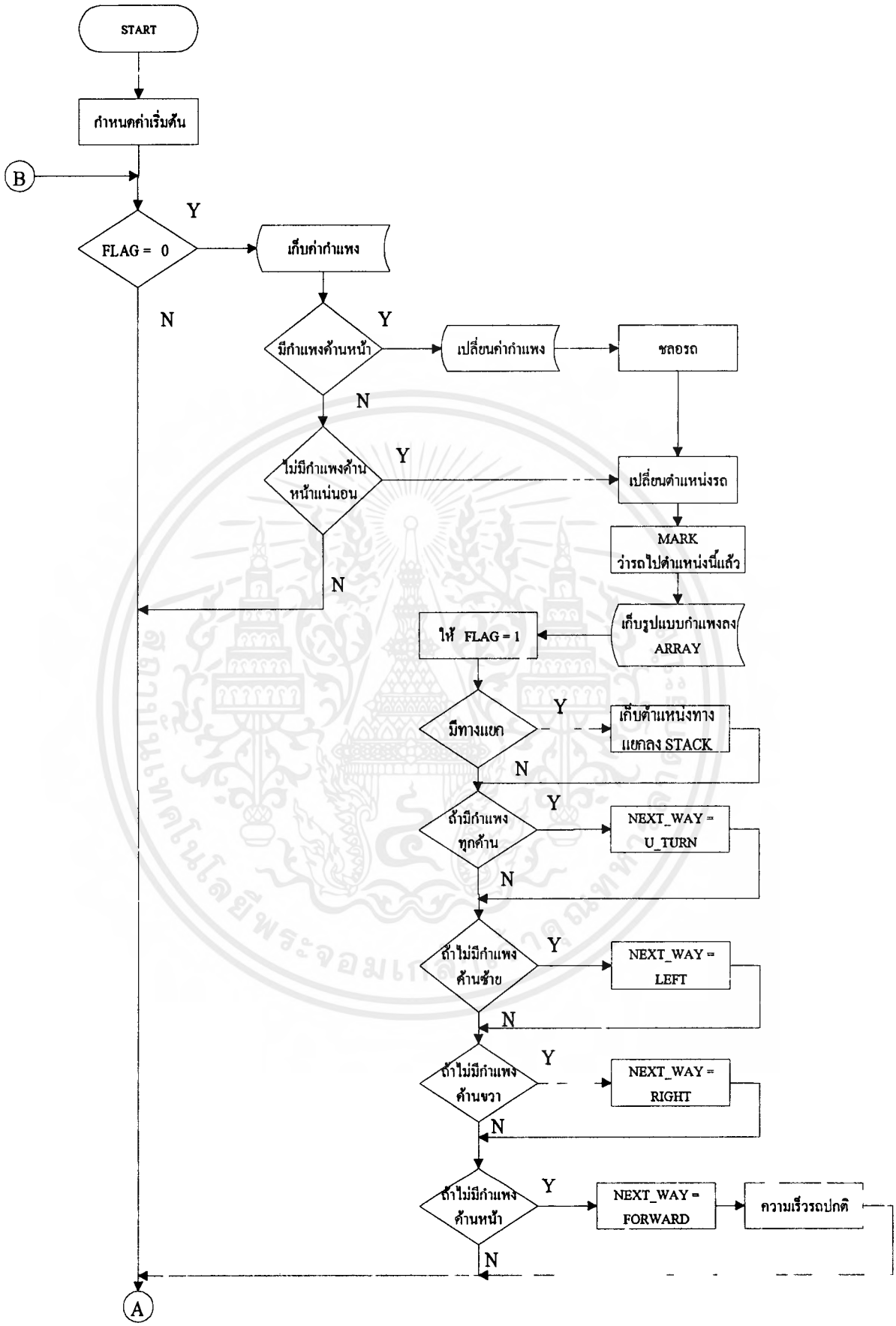
3.2.3 ตัวแปรที่สงวนไว้ใช้งาน

ในการใช้งานฟังก์ชั่นพื้นฐาน เราจะต้องรู้ถึงตัวแปรที่สงวนไว้ใช้งานและการกำหนดค่าซึ่งจะต้องไม่มีการตั้งชื่อซ้ำกัน ตัวแปรสงวนมีตัวแปรต่างๆดังนี้

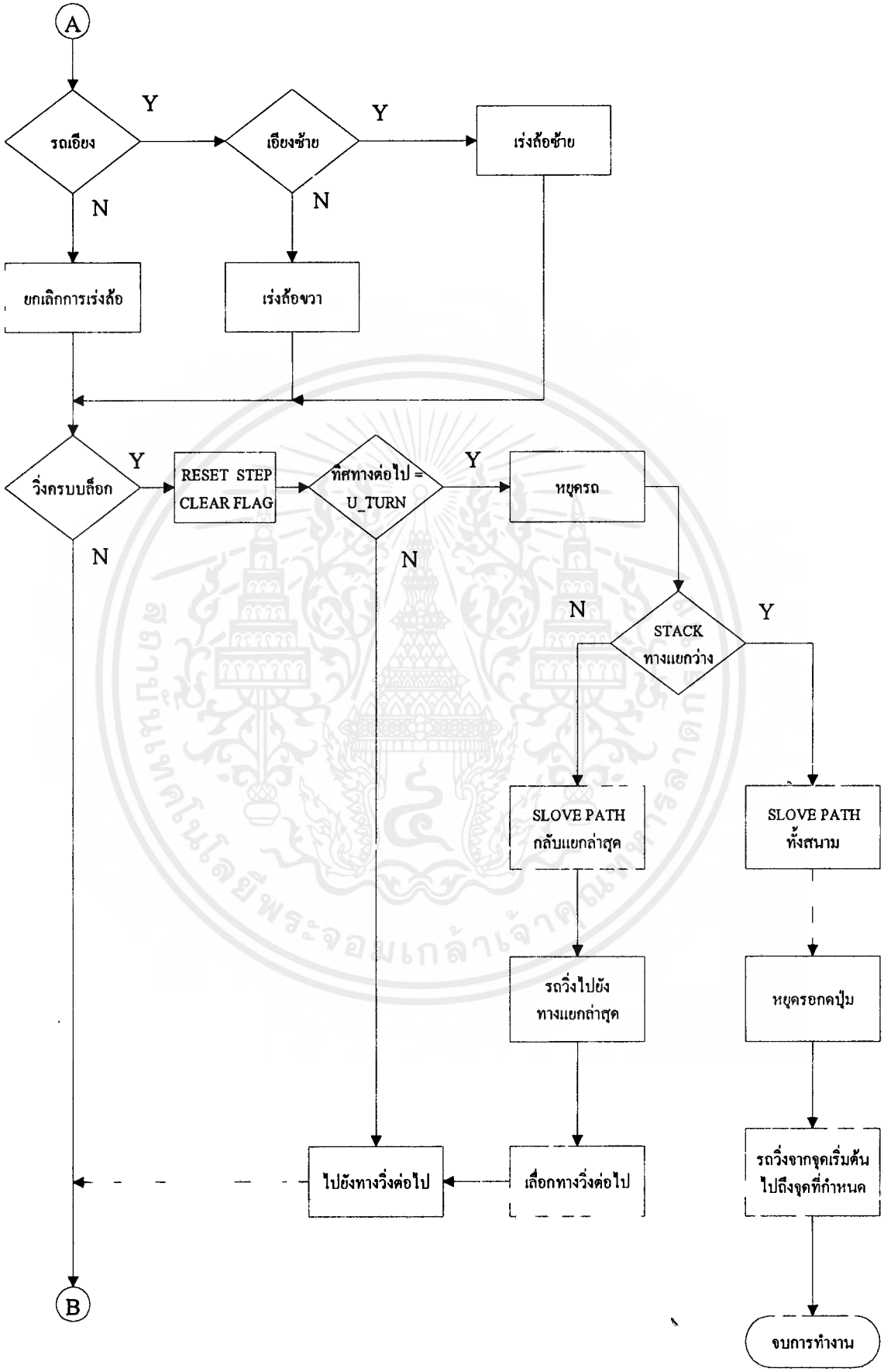
```
unsigned char xdata  *PCTRL;          ชี้ตำแหน่งของ port 8255
unsigned char xdata  *PSENSOR;        ชี้ตำแหน่งของ port SENSOR
unsigned char xdata  *PDISPLAY;       ชี้ตำแหน่งของ port DISPLAY
unsigned char idata  *SPEED_HI;       ชี้ตำแหน่งของ ค่าความเร็ว 8 บิทบน
unsigned char idata  *SPEED_LO;       ชี้ตำแหน่งของ ค่าความเร็ว 8 บิทล่าง
unsigned char idata  *STEP_HI_L;      ชี้ตำแหน่งของ ค่าจำนวน STEP 8 บิทบน ล้อซ้าย
unsigned char idata  *STEP_LO_L;      ชี้ตำแหน่งของ ค่าจำนวน STEP 8 บิทล่าง ล้อซ้าย
unsigned char idata  *STEP_HI_R;      ชี้ตำแหน่งของ ค่าจำนวน STEP 8 บิทบน ล้อขวา
unsigned char idata  *STEP_LO_R;      ชี้ตำแหน่งของ ค่าจำนวน STEP 8 บิทล่าง ล้อขวา

#define HI     0xFE                   กำหนดความเร็วต่างๆ
#define MID_HI 0xFD                   "
#define MID_LO 0xFC                   "
#define LO     0xFB                   "

#define FORWARD 'F'                   กำหนดค่าของด้านหน้า
#define REVERSE 'B'                   กำหนดค่าของด้านหลัง
#define LEFT    'L'                   กำหนดค่าของด้านซ้าย
#define RIGHT   'R'                   กำหนดค่าของด้านขวา
#define U_TURN  'U'                   กำหนดค่าของการกลับหลัง

#define COMPENSATE 'C'                กำหนดค่าของการชดเชยการวิ่ง
#define CLEAR      'E'                กำหนดค่าของการลบการชดเชย
#define STOP       'O'                กำหนดค่าของการหยุดการหมุนของมอเตอร์
#define DISABLE    'D'                กำหนดค่าของการเลิกการสั่งงานของมอเตอร์

#define NORTH  0x00                   กำหนดค่าของทิศเหนือ
#define EAST   0x01                   กำหนดค่าของทิศตะวันออก
#define SOUTH  0x02                   กำหนดค่าของทิศใต้
#define WEST   0x03                   กำหนดค่าของทิศตะวันตก

#define MAX_X  16                     กำหนดค่าสูงสุดของแกน X
#define MAX_Y  16                     กำหนดค่าสูงสุดของแกน Y
#define MAX    256                    กำหนดค่าสูงสุดของจำนวนบล็อกของสนาม
```

```
static unsigned char xdata map[MAX];        อาเรย์ของแผนที่สนาม
static unsigned char xdata m[MAX];          อาเรย์ของการ MARK สนาม
static unsigned char xdata x[70],y[70];     อาเรย์การเก็บค่าทางแยก

int  G,store[30];                           บอกถึงจำนวนทางแยก
char DIR;                                   ทิศทางของรถ
char X,Y;                                   ตำแหน่งของรถ
char  SX,SY;                                ค่าตำแหน่งที่เก็บในทางแยก
char WALL;                                  เก็บค่ากำแพง
char NEXT_WAY;                              เก็บค่าทิศทางที่จะไปต่อไป
char GOAL_X,GOAL_Y;                         ตำแหน่งเป้าหมาย
char box[256],temp_box[256];                อาเรย์สำหรับการเก็บการเปลี่ยนแปลงของสนาม
char status[4],slove[30],slove2[30];        อาเรย์สำหรับการค้นหาเส้นทาง
int  count,value,final1,point;              การนับค่าระยะทางรวม
```

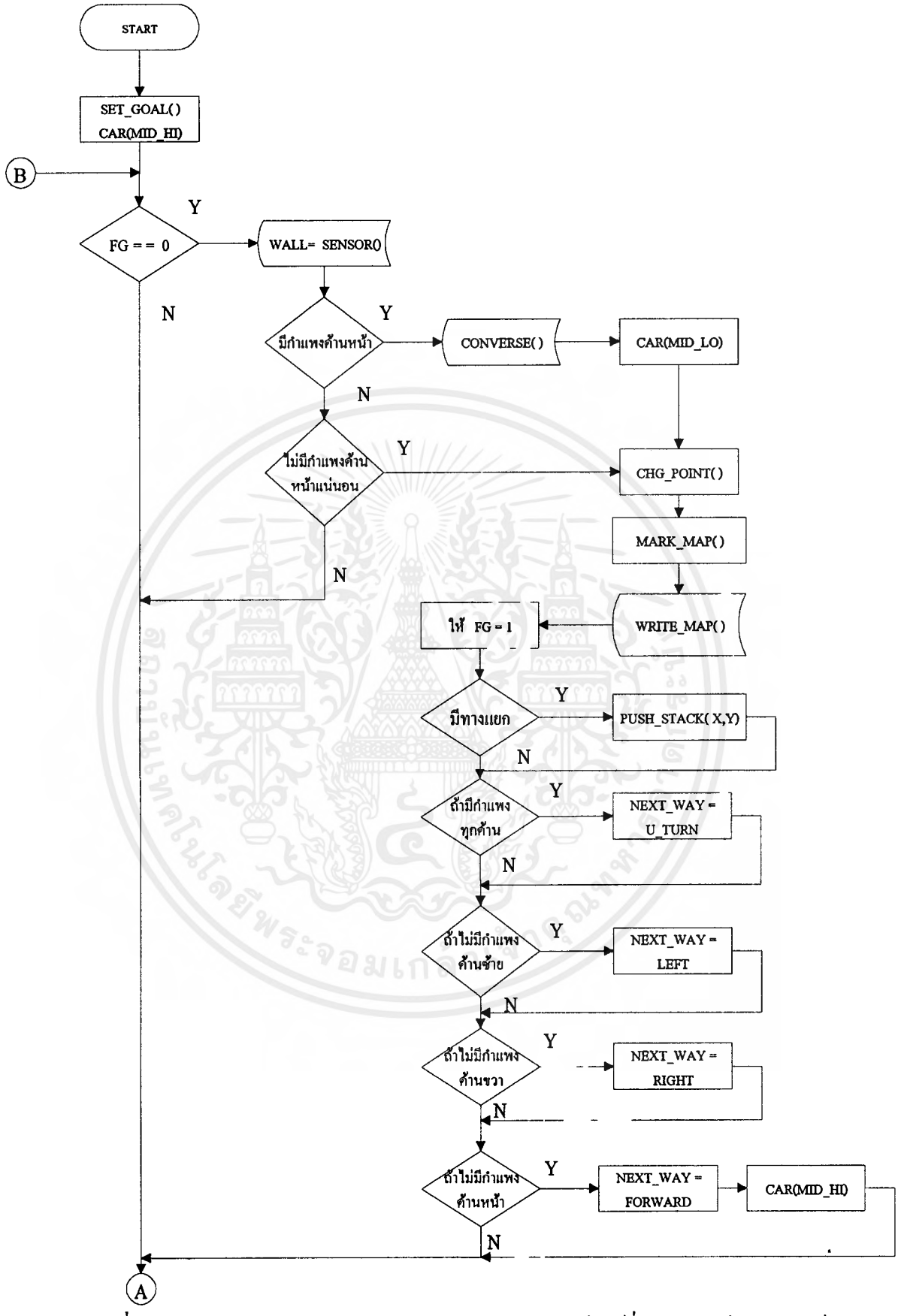
เราจะสามารถนำฟังก์ชั่นต่างๆมาใช้ในการเขียนโปรแกรมได้ดังตัวอย่างแผนภาพแสดงลำดับการทำงานของโปรแกรมดังรูป 3.8 และรูป 3.9



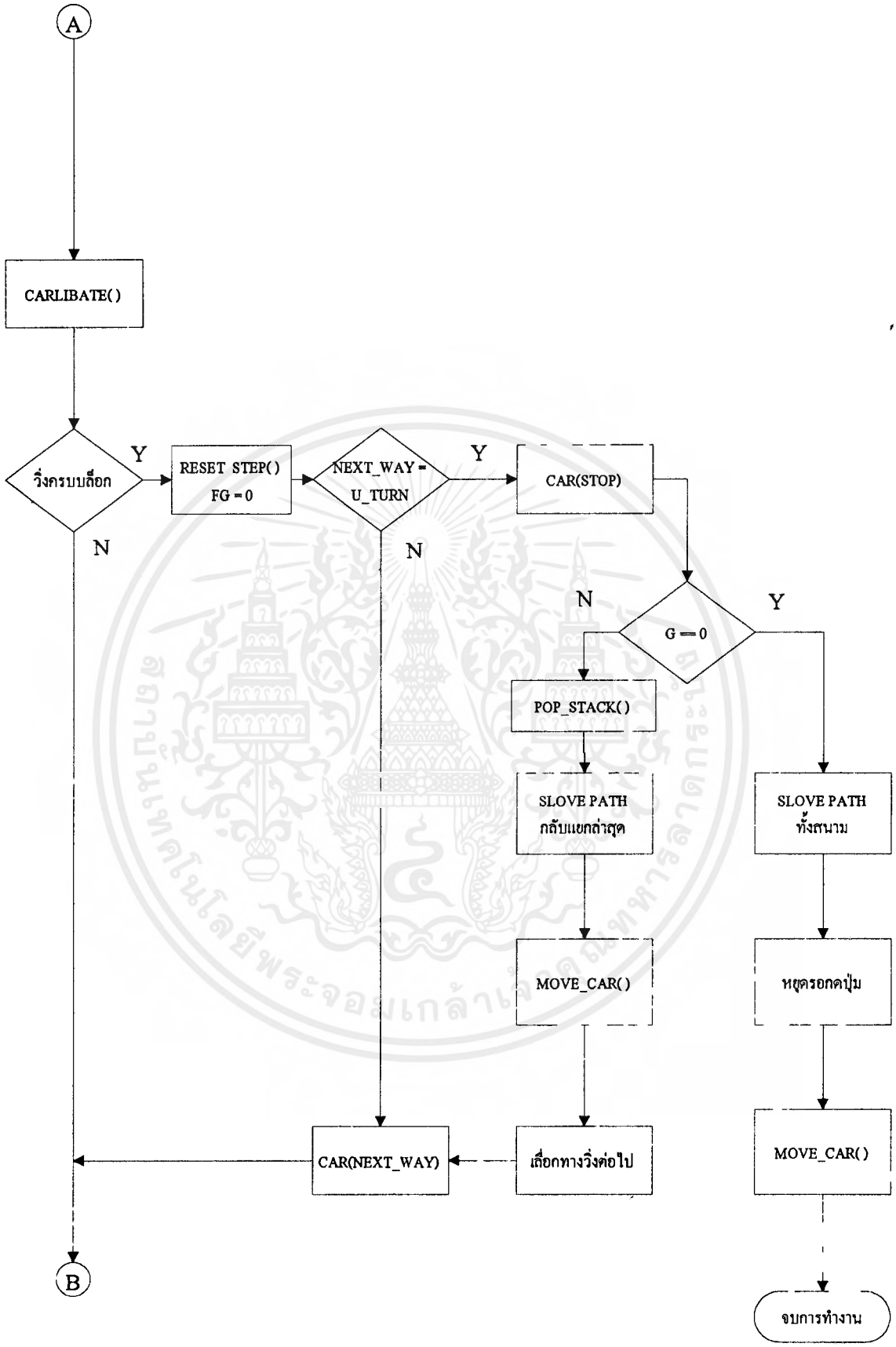
รูปที่ 3.8 ก Flowchart ตัวอย่างขั้นตอนการทำงานของไมโครเมาส์



รูปที่ 3.8 ข Flowchart ตัวอย่างขั้นตอนการทำงานของไมโครเมาส์ (ต่อ)



รูปที่ 3.8 ค Flowchart ตัวอย่างการเขียนโปรแกรมโดยเรียกใช้ฟังก์ชั่นต่างๆของไมโครเมาส์



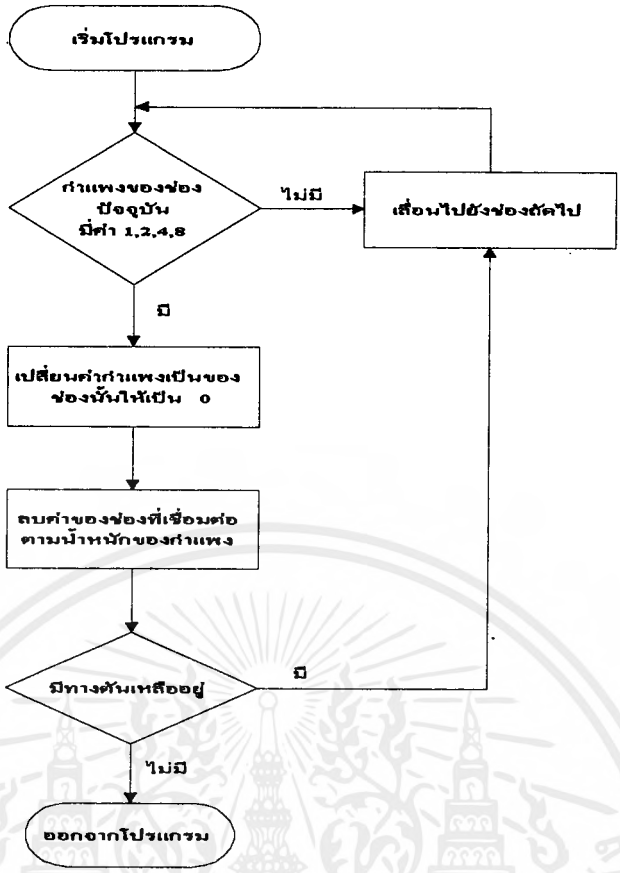
รูปที่ 3.8 ง Flowchart ตัวอย่างการเขียนโปรแกรมโดยเรียกใช้ฟังก์ชั่นต่างๆของไมโครเมาส์ (ต่อ)

จาก Flow Chart สามารถอธิบายขั้นตอนต่างๆได้ดังนี้

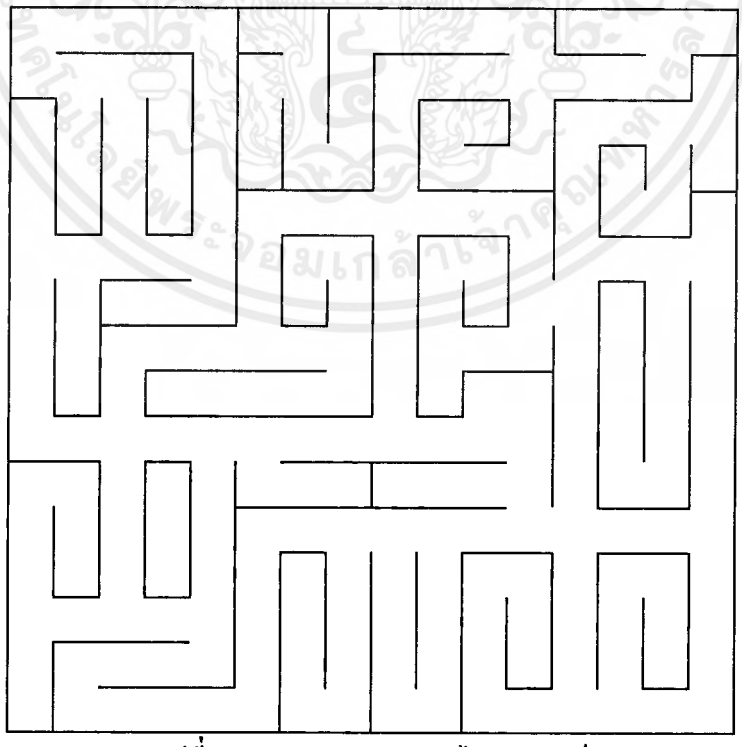
เริ่มต้นกำหนดค่าต่างๆให้กับ ไมโครเมาส์ เช่น ตำแหน่งรถ , ทิศทาง , ขนาดจำนวนบล็อก ,ฯลฯ เริ่มให้รถวิ่งตรงมีการเช็ค FLAG ว่ามีการเก็บลักษณะกำแพงของบล็อกนี้หรือยัง ถ้ายังก็มีการเก็บลักษณะกำแพง ถ้ามีกำแพงด้านหน้าก็ให้ชลอรถ ซึ่งจะต้องมีการเปลี่ยนตำแหน่งของรถตามแนวทิศทางก่อนที่จะมีการ Mark ว่าตำแหน่งนี้รถได้ไปแล้ว และ เก็บค่ากำแพงที่ได้ลงใน แผนที่สนามหลังจากนั้นก็มีการตรวจสอบลักษณะของกำแพง เพื่อเลือกเส้นทางที่วิ่งต่อไป ในขณะที่รถวิ่งนั้นจะมีการตรวจเช็คความเอียงของรถ และสั่งการปรับความเร็วของล้อเพื่อให้รถวิ่งอยู่ในแนวตรงเสมอ และเมื่อรถวิ่งครบบล็อกแล้วก็จะเคลียร์ค่า Flag และค่า step ต่างๆ เพื่อที่จะทำการเก็บค่ากำแพงในบล็อกต่อไป รถจะวิ่งไปตามทางจากค่าที่ได้จากการเลือกเส้นทาง ว่าจะเดินหน้า , เลี้ยวซ้าย , เลี้ยวขวา หรือถอยหลัง ถ้าถอยหลังก็จะตรวจสอบว่ามีการเก็บ stack ทางแยกไว้หรือเปล่าถ้ามีก็วิ่งไปยังแยกนั้นแล้วค่อยกลับไปทำการเก็บค่ากำแพงต่อไป ถ้าไม่มีค่าใน สแต็ก ก็แสดงว่ารถวิ่งครบทั้งสนาม แล้วก็ทำการ Slove path หาระยะทางที่ เร็วที่สุดจากจุดเริ่มต้นไปยังตำแหน่งที่ต้องการ

3.2.4 การทำงานของฟังก์ชั่น Compression

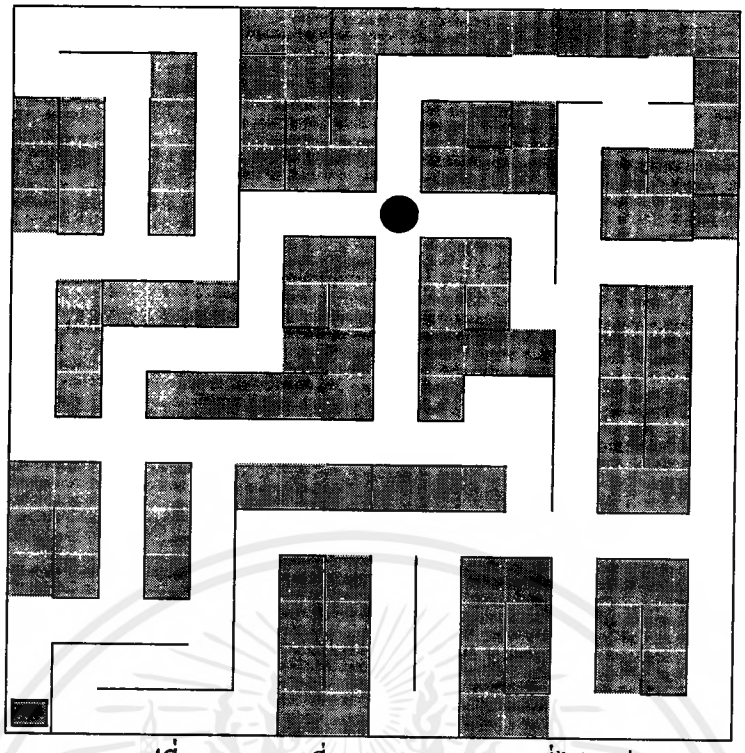
ฟังก์ชั่นนี้จะสนับสนุนการทำงานของ ฟังก์ชั่น Slove-path โดยเมื่อไมโครเมาส์ทำการวิ่งไปจนหมดทุกช่องแล้ว เราก็จะเอารูปแบบของแต่ละช่องมา ซึ่งรูปแบบของแต่ละบล็อกนั้นเราจะนำมาวิเคราะห์อีกทีสมมติว่าเราให้มีรูปแบบของสนามเป็นแบบนี้ตามรูปที่ 3.10 เราจะนำข้อมูลในสนามนี้มาทำการตรวจสอบเพื่อที่จะหาทางตันของเส้นทาง เมื่อเราพบแล้วเราก็จะทำการลบบล็อกนั้นออกไป ซึ่งนั้บล็อกนั้นจะต้องไม่เป็นตำแหน่งเริ่มต้นกับตำแหน่งปลายทาง เราจะทำอย่างนี้ไปเรื่อยๆจนไม่มีช่องไหนที่ตันอีก วิธีการนี้เราก็จะได้รูปแบบสนามใหม่ในรูปที่ 3.11 เมื่อเราได้สนามนี้มาแล้วต่อไปเราก็จะทำการลบช่องในลักษณะที่เป็นทางวน (loop) ออกซึ่งก็จะได้ดังรูปที่ 3.12 การลบบล็อกต่าง ๆ ที่ไม่สำคัญจะมีผลการทำงานอย่างมากกับการทำงานของฟังก์ชั่น Slove - Path มีผลให้มีการคำนวนได้เร็วขึ้น



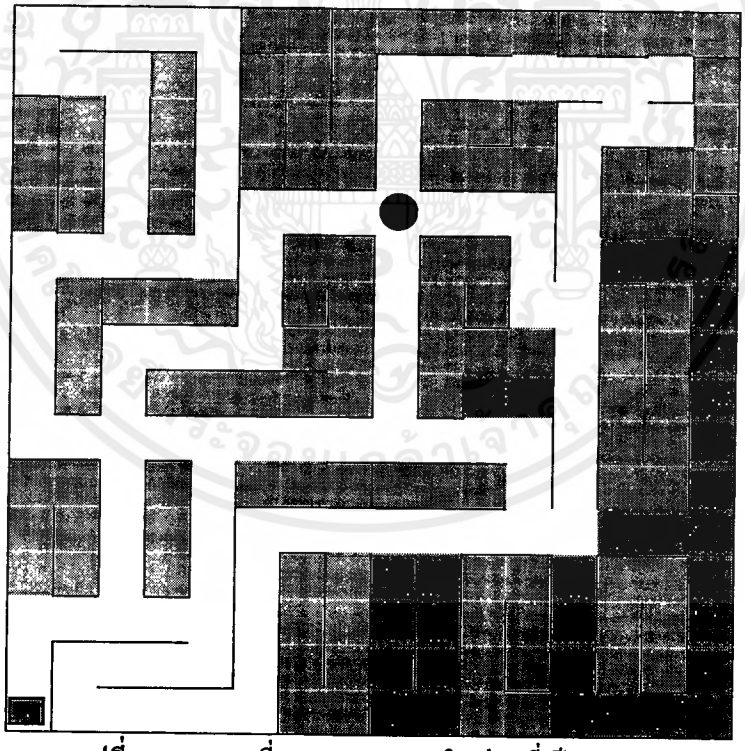
รูปที่ 3.9 Flowchart ของ ฟังก์ชั่น Compression



รูปที่ 3.10 สนามจำลองของไมโครเมาส์



รูปที่ 3.11 สนามที่ถูกการลบ BLOCK ที่ไม่จำเป็นออก



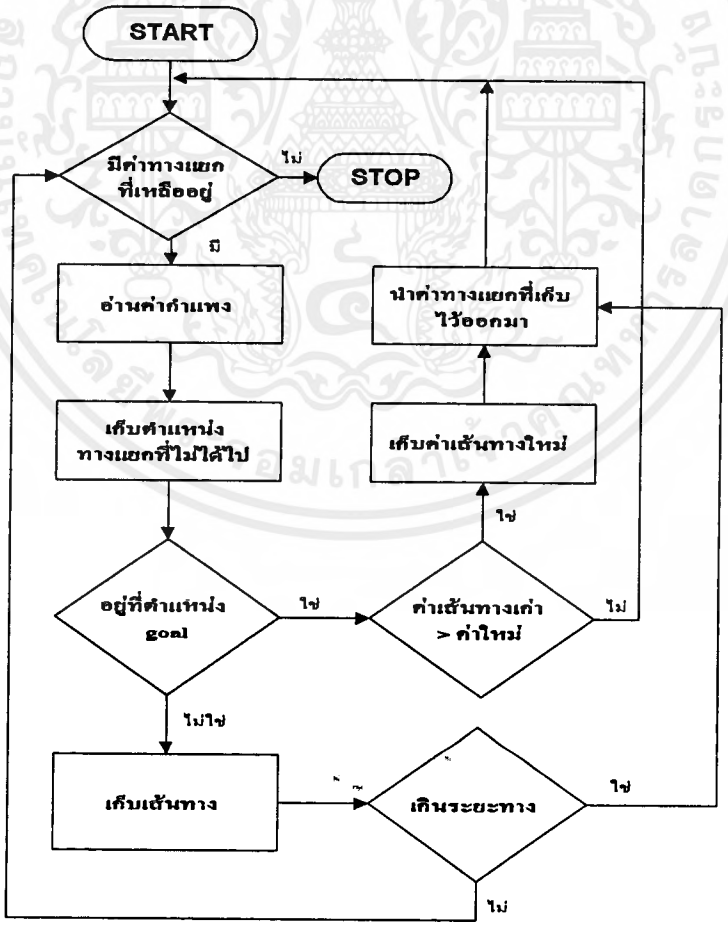
รูปที่ 3.12 สนามที่ถูกลบ BLOCK ในส่วนที่เป็นทางวน

3.2.5 การทำงานของฟังก์ชั่น Slove-Path

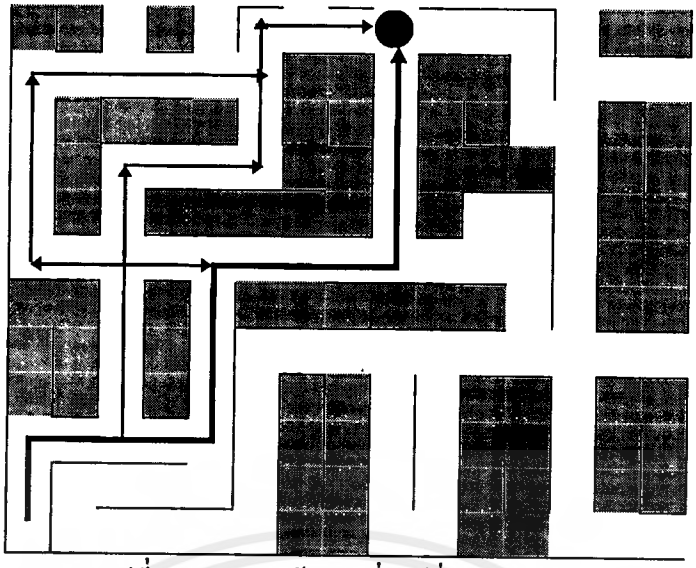
ฟังก์ชั่นนี้จะทำหน้าที่ในการหาเส้นทางจากตำแหน่งเริ่มต้นไปยังตำแหน่งเป้าหมาย ที่ใช้ระยะทางใกล้ที่สุด โดยการดูจากลักษณะของกำแพงที่ได้ไปทำการสำรวจมาก่อนแล้ว นำมาวิเคราะห์เพื่อ

หาเส้น โดยการที่ทำให้ประมวลได้เร็ว เราจงต้องมีวิธีการต่าง ๆ ที่จะทำให้ผลลัพท์ออกมาถูกต้องและรวดเร็ว

เริ่มแรกจากการ กำหนดค่าต่างที่เราต้องใช้ในฟังก์ชั่น เช่น ตัวแปร Count ซึ่งจะเป็นตัวบอกว่ามีทางแยกที่เรายังไม่ได้ไปอยู่กี่ที่ ซึ่งถ้าค่าของ Count เป็น -1 ก็แสดงว่าได้ไปทุกเส้นทางแล้ว ให้ออกจากฟังก์ชั่นได้เลย ในตอนแรกเราจะให้มีค่าเป็น 0 ในส่วนต่อไปก็จะทำการอ่านค่ากำแพงของจุดเริ่มต้นที่กำหนดมา แล้วทำการเพิ่มค่าตัวชี้ และทำการเก็บตำแหน่งนั้นลงใน Array ที่ทำหน้าที่เป็น Stack เพื่อจะเก็บเส้นทางการวิ่ง โดยในการตรวจสอบกำแพงเราจะมี การตรวจสอบทิศทางด้วย เมื่อเราพบกำแพงที่มีลักษณะเป็นทางแยกเราจะทำการคำนวนเส้น ทางที่สามารถไปได้ว่าเส้นทางไหนใกล้เป้าหมายมากที่สุดแล้ว ให้ตัดสินใจเลือกทางนั้น ส่วนทางที่เหลือก็จะถูกเก็บลงใน Stack เมื่อเราวิ่งไปถึงเป้าหมาย จะทำการตรวจค่าของ Times ตัวใหม่ว่ามีค่าเท่าใด ถ้ามีค่าน้อยกว่าค่าเดิม ก็จะทำการก๊อปปี้เส้นทางการเดินที่เคยเก็บไว้ใน Stack ลงใน Array แกชุดหนึ่ง แล้วทำการเอาข้อมูลทางแยก ล่าสุดที่ถูกเก็บเอาไว้มาทำการหาเส้นทางอื่นต่อ เพราะไม่แน่ว่าเส้นอื่นอาจจะใกล้กว่าเส้นเดิมก็ได้ และถ้าในกรณีที่ไม่พบเป้าหมายแต่ค่า Times นั้นเกินค่าเดิมแล้ว ก้ต้องข้อมูลของทางแยกตัวล่าสุดออกมาหาเส้นทางใหม่ไปเรื่อย ๆ จนหมด เราก็จะได้เส้นทางที่ใกล้ที่สุด



รูปที่ 3.13 Flowchart ของ ฟังก์ชั่น SLOVE_PATH



รูปที่ 3.14 แสดงเส้นทางที่ใกล้ที่สุดของสนาม

3.2.6 โปรแกรมแสดงสถานะ (STATUS PROGRAM)

การเขียนโปรแกรมควบคุมการทำงานของไมโครเมาส์นั้น เราจะใช้ภาษาซีในการเขียนโปรแกรมที่เราเขียนจะถูกแปลงให้เป็นภาษาเครื่องของ MCS-51 จากนั้นก็จะถูกส่งไปให้ไมโครเมาส์เพื่อให้มันทำงานตามที่เราเขียน ในการทำงานของไมโครเมาส์นั้น อาจจะเกิดข้อผิดพลาดเกิดขึ้นได้ ถ้าเป็นข้อผิดพลาดที่ไม่ซับซ้อน เราก็สามารถที่จะวิเคราะห์ได้เองว่าข้อผิดพลาดอยู่ส่วนไหนของโปแกรม แต่ถ้าเกิดข้อลผิดพลาดที่ซับซ้อนมาก ทำให้เรายากแก่การวิเคราะห์และแก้ไขปัญหา เราจึงต้องสร้างโปรแกรมนี้ขึ้นมาเพื่อรองรับปัญหาซับซ้อนที่จะเกิดขึ้น เพราะในขณะที่ไมโครเมาส์วิ่งอยู่เราไม่สามารถรู้ได้เลยว่าตัวที่เรากำหนดในการเขียนโปรแกรมมีค่าเท่าไร

วิธีการก็คือเราจะเขียนฟังก์ชั่นตัวหนึ่งขึ้นมา เพื่อให้มีหน้าที่ในการส่งค่าต่าง ๆ ที่เราต้องการจะตรวจสอบ ในขณะที่ไมโครเมาส์กำลังวิ่ง ค่านี้จะถูกส่งไปยังพอร์ทอนุกรมของคอมพิวเตอร์ บนเครื่องคอมพิวเตอร์ เราก็จะมีโปรแกรมที่สร้างขึ้นมาเพื่อคอยรับค่าจากไมโครเมาส์ มาแสดงบนหน้าจอมอนิเตอร์ ซึ่งในโปรแกรมที่อยู่บนเครื่องคอมพิวเตอร์ เราจะบ่งการแสดงผลออกเป็น 3 ลักษณะ ดังนี้

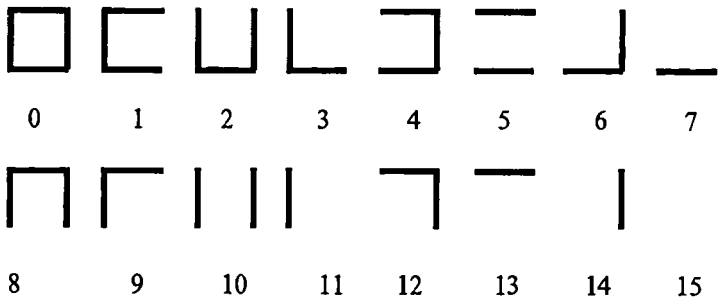
1. แสดงค่ากำแพงและทิศทางที่ตำแหน่งของช่องนั้น

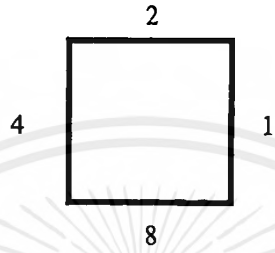
โปรโตคอลที่เรากำหนดขึ้นไว้มี ดังนี้

1 byte	1 byte	1 byte	1 byte	1 byte	1 byte
HEADER	WALL	X	Y	DIR	END

HEADER , END คือโค้ดที่บอกว่าเป็นโปรโตคอลของการแสดงค่ากำแพง มีค่าเป็น 0xEE

WALL คือลักษณะของกำแพงมี 16 รูปแบบดังต่อไปนี้





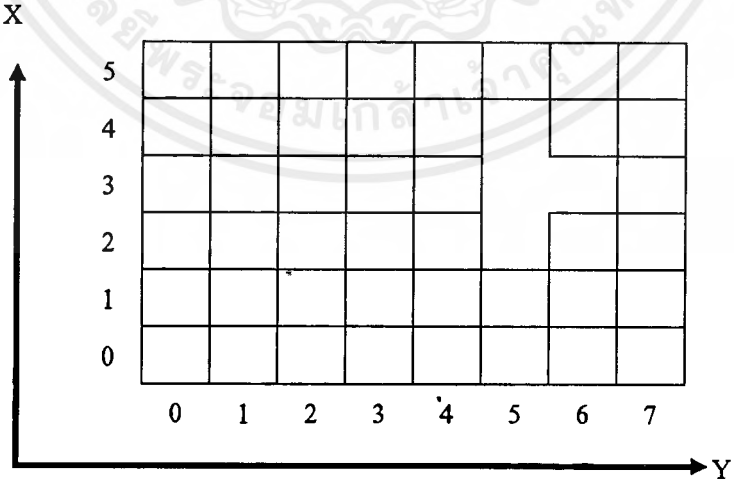
รูปที่ 3.15 รูปแสดงค่าแทนลักษณะของกำแพง

รูป 3.15 นี้จะแสดงถึงน้ำหนักของกำแพงในแต่ละด้าน ตัวอย่างเช่น ถ้ากำแพงด้านบนและด้านขวาไม่มี ค่าของกำแพงก็จะมีค่าเท่ากับ 4+2 = 6

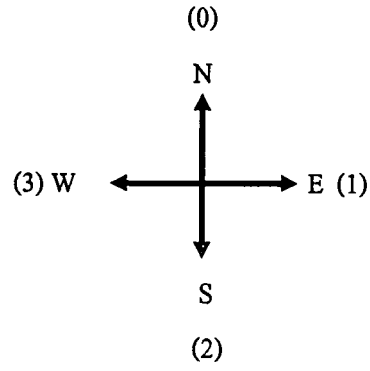
X คือค่าของตำแหน่งปัจจุบันของไมโครเมาส์ ทางด้านแนวตั้ง

Y คือค่าของตำแหน่งปัจจุบันของไมโครเมาส์ ทางด้านแนวนอน

จากรูปข้างล่างนี้ สมมติว่าเราให้ค่า X = 3 Y=5 และ ลักษณะกำแพงเป็น 11 ผลที่ได้จะเป็นดังนี้



DIR คือค่าทิศทางปัจจุบันของไมโครเมาส์ ซึ่งเราจะนำค่านี้ไปแสดงบนหน้าจอ ซึ่งมีลักษณะเป็นลูกศร เพื่อแสดงทิศทางปัจจุบันของเมาส์

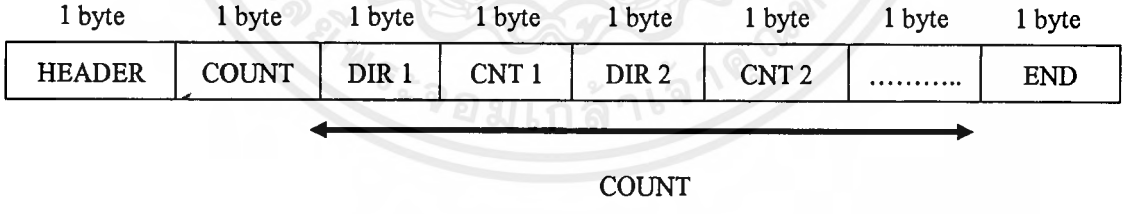


ตัวอย่างการเขียนโปรแกรม

```
send_port(0xEE);
send_port(WALL);
send_port(X);
send_port(Y);
send_port(DIR);
send_port(0xEE);
```

2. แสดงการ SLOVE PATH ของไมโครเมาส์

โปรโตคอลที่เรากำหนดขึ้นไว้มีดังนี้



HEADER , END คือโค้ดที่บอกว่าเป็นโปรโตคอลของการแสดงเส้นทางของฟังก์ชั่น SLOVE_PATH มีค่าเป็น 0xAA

COUNT เป็นตัวบอกจำนวนข้อมูลของเส้นทางที่ได้จากการ SLOVE_PATH ตัวอย่างเช่น

ผลของการ SLOVE_PATH ได้ดังนี้คือ

ไปทางทิศเหนือ 3 บล็อก ไปทางทิศตะวันตก 2 บล็อก

ไปทางทิศใต้ 1 บล็อก ไปทางทิศตะวันออก 2 บล็อก

เพราะฉะนั้นเราจะได้ข้อมูลดังนี้

0 3 3 2 2 1 1 2 จะได้ค่า COUNT = 8

DIR 1 คือ ทิศทางของชุดที่ 1

COUNT 1 คือ จำนวนบล็อกที่วิ่งไป ชองชุดที่ 1

```
send_port(0xAA);
    send_port(8);
    send_port(0);       send_port(3);
    send_port(3);       send_port(2);
    send_port(2);       send_port(1);
    send_port(1);       send_port(2);
    send_port(0xAA);
```

3. แสดงค่าตัวแปรทั่วไป

โปรโตคอลที่เรากำหนดไว้มีดังนี้

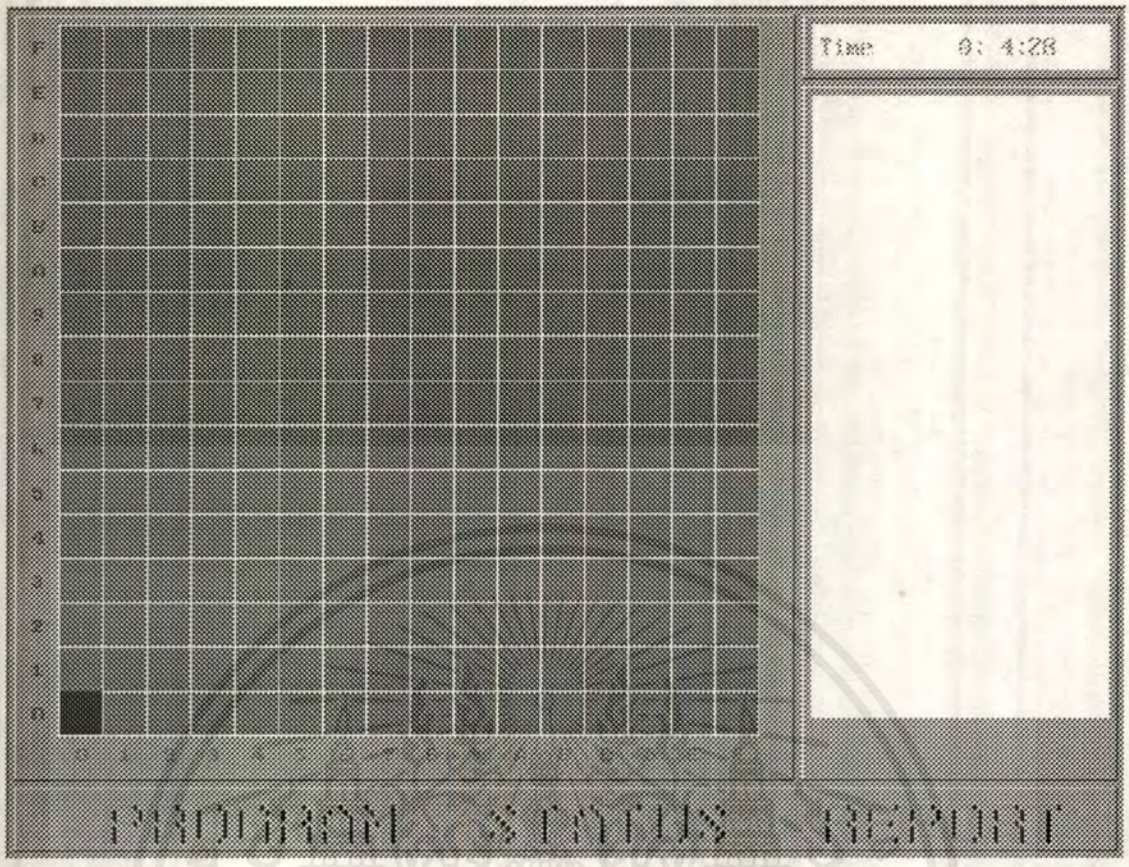
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte
HEADER	DATA	DATA	DATA	DATA	END

HEADER , END คือโค้ดที่บอกว่าเป็นโปรโตคอลของการแสดงค่าตัวแปรมีค่าเป็น 0xDD

DATA เป็นค่าที่ผู้เขียนสามารถส่งค่ามาแสดงผล สามารถส่งค่าตัวแปรที่ต้องการจะตรวจสอบ จำนวนกี่ตัวก็ได้

ฟังก์ชั่นที่เรียกใช้คือ ฟังก์ชั่น send_port(DATA) ตัวอย่างการเรียกใช้ฟังก์ชั่นนี้ สมมติว่าเราต้องการจะตรวจสอบค่าตัวแปร TOP,NEXT_WAY และฟังก์ชั่น SELECT_WAY จะสามารถเขียนโปรแกรมได้ดังนี้

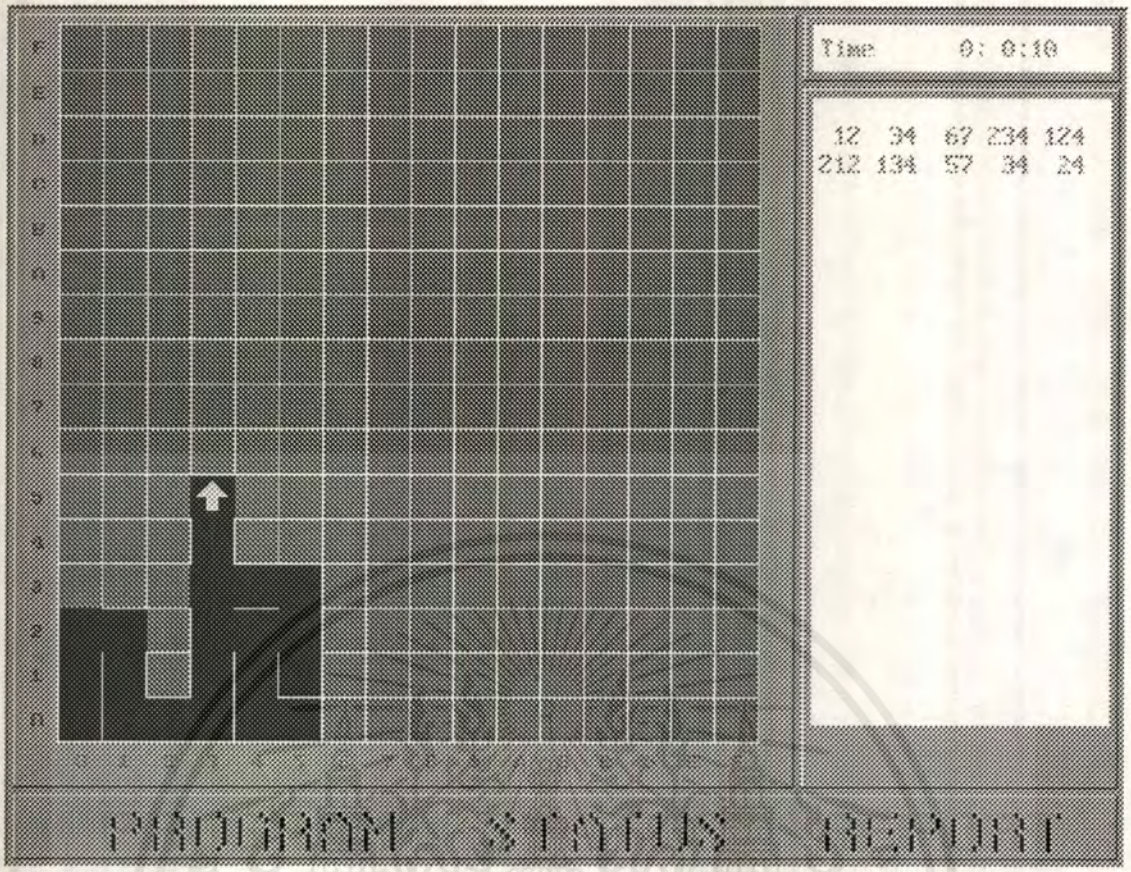
```
send_port(0xDD);
send_port(TOP);
send_port(NEXT_WAY);
send_port(SELECT_WAY());
send_port(0xDD);
```



รูปที่ 3.16 รูปแบบการแสดงผลหน้าจอของคอมพิวเตอร์

รูปนี้คือโปรแกรมที่คอยรับข้อมูลจากไมโครเมาส์ ทางด้านซ้ายมือจะแสดงเขาวงกตไว้ 16 x 16 บล็อก เมื่อไมโครเมาส์วิ่งก็จะส่งค่ากำแพงทิศทางและตำแหน่ง โปรแกรมก็จะรับมา แล้วแสดงผลบนหน้าจอมอนิเตอร์ สุดท้ายเราก็จะได้ลักษณะของกำแพงของแต่ละช่อง ที่ไมโครเมาส์วิ่งไป ในกรณีที่วิ่งจนหมดทุกช่องแล้ว ไมโครเมาส์จะทำการวิเคราะห์แล้วทำการ SLOVE_PATH เส้นทางที่ใกล้ที่สุด จากนั้นจะส่งข้อมูลมาให้โปรแกรม เพื่อแสดงเส้นทางเส้นทางบนหน้าจอมอนิเตอร์

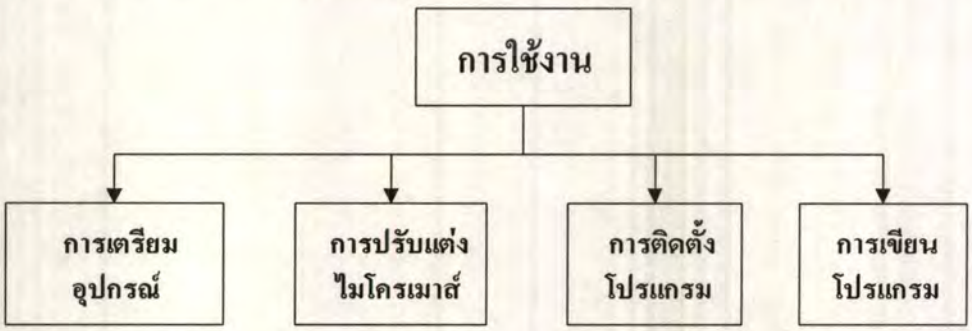
ในส่วนของทางด้านขวามือจะแสดงค่าของตัวแปร ที่เราต้องการทราบว่ามีค่าเท่าไรในขณะที่ไมโครเมาส์กำลังทำงาน ทำให้เราสามารถตรวจสอบข้อผิดพลาดได้ง่ายขึ้น



รูปที่ 3.17 รูปแบบการแสดงผลลักษณะของกำแพงขณะที่ไมโครเมาสกำลังวิ่ง

3.3 การใช้งานของไมโครเมาส์

ในการที่จะเขียนควบคุมไมโครเมาส์ ให้สามารถทำงานได้ตามที่เราต้องการนั้น จะต้องมีการเตรียมความพร้อมในส่วนต่าง ๆ ไม่ว่าจะเป็นส่วนของตัวไมโครเมาส์ การติดตั้งโปรแกรม การใช้งานของโปรแกรม การเขียนโปรแกรมถูกต้องเหมาะสม การเชื่อมต่อระหว่างตัวไมโครเมาส์กับเครื่องคอมพิวเตอร์ ทั้งหมดนี้เราจำเป็นที่ต้องเข้าใจในส่วนต่าง ๆ เหล่านี้ก่อน จึงจะสามารถควบคุมการทำงานได้อย่างมีประสิทธิภาพ สิ่งที่เราต้องมาทำเข้าใจมีดังต่อไปนี้



รูปที่ 3.18 แสดงส่วนประกอบต่างๆของการใช้งาน

3.3.1 การเตรียมอุปกรณ์

การเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครเมาส์ ผู้ที่จะเขียนจะต้องความรู้ทางด้านการเขียนโปรแกรมบนคอมพิวเตอร์ ซึ่งโครงงานนี้เราจะใช้ภาษา C ในการควบคุมการทำงาน สำหรับบุคคลที่ไม่เคยเขียนโปรแกรมหรือเขียนโปรแกรมภาษาอื่นได้ ก็จำเป็นต้องศึกษาการเขียนโปรแกรมภาษา C มาก่อน ซึ่งในท้องตลาดนั้นมีหนังสือการเขียนโปรแกรมภาษา C มากมาย เราสามารถเลือกซื้อมาศึกษาได้โดยไม่ยากนัก

ก่อนที่เริ่มเขียนโปรแกรมควบคุม เราจำเป็นจะต้องจัดเตรียมอุปกรณ์ให้ครบถ้วนเสียก่อน ซึ่งมีอุปกรณ์ที่ต้องใช้ดังนี้

- เครื่องคอมพิวเตอร์ 286 ขึ้นไป
- แหล่งจ่ายไฟจ่ายกระแสได้ 2 แอมป์
- ตัวไมโครเมาส์
- สายเชื่อมต่อกับเครื่องคอมพิวเตอร์
- โปรแกรม EDITOR
- โปรแกรมคอมไพล์เลอร์
- ไขควง คีม

3.3.2 การปรับแต่งตัวไมโครเมาส์

ในการวิ่งของไมโครเมาส์จะวิ่งได้ดีนั้น นอกจากการเขียนโปรแกรมที่ดีแล้ว ยังต้องมีการปรับแต่งตัวไมโครเมาส์ที่ถูกต้องด้วย ส่วนที่เราต้องปรับแต่งนั้นมีดังนี้

1 การปรับแต่งล้อ

ล้อที่ต้องการปรับมีสองส่วนคือส่วนที่อยู่ด้านหน้าและด้านหลัง เราจะต้องปรับล้อหน้าให้สามารถลอยจากพื้นได้เล็กน้อย เมื่อเรากดส่วนท้ายของตัวไมโครเมาส์ เพราะว่าเวลาวิ่งล้อหน้าลอยจากพื้น ทำให้เวลาวิ่งล้อหน้าจะได้ไม่ไปขัดขวางการวิ่งของมอเตอร์ทั้งสอง แต่การปรับไม่ควรให้ลอยมาเกินไป ควรปรับให้มีค่าน้อยที่สุดที่จะไม่ไปขัดขวางการวิ่ง เพราะมันมีผลต่อการตรวจสอบกำแพงของเซนเซอร์ ถ้าล้อหน้าลอยมากเกินไป เซนเซอร์จะไม่สามารถตรวจสอบกำแพงได้

2. การปรับระยะห่างระหว่างเซนเซอร์กับกำแพง

ในการปรับระยะห่างระหว่างเซนเซอร์กับกำแพง เป็นสิ่งจำเป็นเพราะแสงของตัวส่งอินฟาเรด จะไปตกกระทบกำแพงแล้วสะท้อนกลับมา ควรมีระยะทางไม่เกิน 3 - 5 mm ถ้าสูงเกินเซนเซอร์จะตรวจสอบกำแพงไม่เจอ และถ้าต่ำเกินไปเวลาที่ไมโครเมาส์วิ่งเซนเซอร์อาจจะไปชนกำแพงได้ ในกรณีที่เซนเซอร์สูงเกินไปก็สามารถหาแหวนรอง มารองชุดเซนเซอร์ให้ต่ำลงได้

จากนั้นให้เริ่มจากจ่ายไฟให้กับไมโครเมาส์ ที่แรงดัน 15 - 24 โวลท์ ระวังเรื่องขั้วไฟฟ้าต้องต่อให้ถูกขั้วด้วย ปรับกระแสของแหล่งจ่ายไฟไปที่ 2 แอมป์ถ้ามีตัวปรับกระแส เปิดสวิท ของไมโครเมาส์ ที่ LED จะสว่างสลับกัน แสดงว่าไมโครเมาส์พร้อมทำงานแล้ว ถ้ายังก็ให้ตรวจสอบสายไฟว่าต่อกัน

แน่นสนิทหรือเปล่า หรืออาจเป็นเพราะว่าแหล่งจ่ายไฟเสีย ให้ลองตรวจสอบดูว่ามีแรงดันออกมาจากขั้วของแหล่งจ่ายไฟหรือเปล่า

3.3.3 การติดตั้งโปรแกรม

สร้างไดเร็คทอรี่ชื่อ C51 โดยใช้คำสั่ง MD C51 จากนั้น copy โปรแกรมคอมไพล์เลอร์และโปรแกรมอีดิตเตอร์ และโปรแกรม STATUS ลงในไดเร็คทอรี่นี้ จากนั้นทำการต่อสาย link เข้าที่พอร์ทอนุกรมของคอมพิวเตอร์กับพอร์ทอนุกรมของไมโครเมาส์ ดูว่าสายนั้นต่ออยู่กับพอร์ทไหนของคอมพิวเตอร์ ถ้าต่ออยู่กับ com1 ให้เขียนคำสั่งตามข้างล่างนี้ เพื่อกำหนดสัญญาณในการสื่อสารระหว่างไมโครเมาส์กับคอมพิวเตอร์ให้ตรงกัน

```
c:\c51> mode com1:96,n,8
```

ถ้าต่ออยู่กับพอร์ท COM2 ให้เขียนดังนี้

```
c:\c51> mode com2:96,n,8
```

จากนั้นเขียนคำสั่งข้างล่างดังนี้ เป็นคำสั่งที่ทำหน้าเซตไดเร็คทอรี่ให้สามารถเรียกใช้คอมไพล์เลอร์ได้

```
c:\c51> autoexec
```

3.3.4 การเขียนโปรแกรม

เรียกโปรแกรม editor ขึ้นมาทดลองเขียนโปรแกรมดังนี้

```
#include "mouse.h"
void main (void)
{  unsigned char ch;
   initial();
while (1)
   {
        ch = read_sensor();
        write_led(ch);
    }
}
```

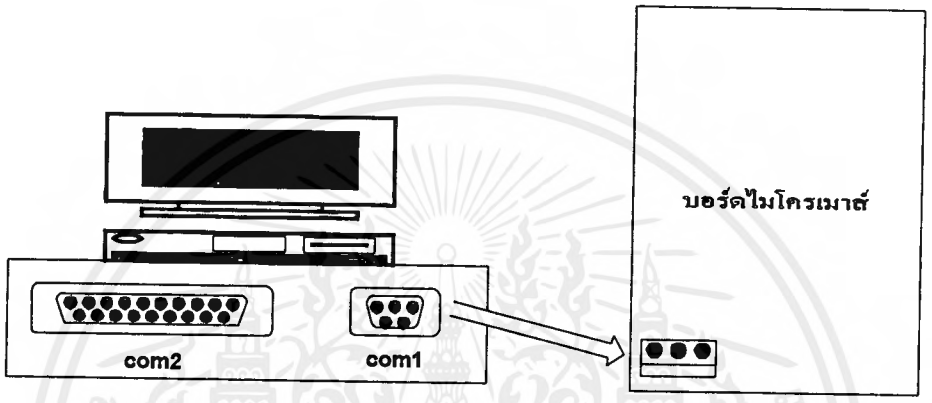
เมื่อเขียนโปรแกรมนี้เสร็จทำการบันทึก โดยตั้งชื่อไฟล์ว่า test.c จากนั้นออกจากโปรแกรม editor แล้วเขียนคำสั่งดังนี้

```
c:\c51> complie  test
```

ถ้าเราเขียนถูกต้อง โปรแกรมจะสร้างไฟล์ใหม่ให้เรา โดยเราสามารถดูไฟล์เหล่านี้ได้ โดยการใช้คำสั่งข้างล่างดังนี้ แต่ถ้าเขียนไม่ถูกต้อง โปรแกรมจะแสดงหมายเลขบรรทัดที่เขียนผิด ก็ให้เราเข้าไปแก้ไขตามนั้น

c:\c51> dir test.*

จากคำสั่งนี้เราจะเห็นไฟล์ที่ชื่อ test หลายไฟล์ ให้เราทำการ copy ไฟล์ test.hex ไปที่พอร์ท com1 ในกรณีที่สาย link ต่อที่พอร์ท com1 แต่ในกรณีที่ต่อพอร์ท com2 ให้เปลี่ยนจาก com1 เป็น com2 โดยใช้คำสั่งดังนี้



รูปที่ 3.19 แสดงการต่อไมโครเมาส์กับพอร์ตของเครื่องคอมพิวเตอร์

c:\c51> copy test.hex com1

ตอนนี้เครื่องคอมพิวเตอร์ กำลังส่งไฟล์นี้ไปให้ไมโครเมาส์ โดยผ่านทางพอร์ท com1 เมื่อส่งไฟล์นี้เสร็จ ไมโครเมาส์จะทำตามคำสั่งทันที โดยโปรแกรมนี้จะอ่านค่าที่ได้จาก ตัวเซนเซอร์มาแสดงผลที่ LED ให้เราลองเอามือไปแตะที่ด้านล่างของเซนเซอร์ดู ถ้าเราแตะตัวไหน LEDตัวนั้นก้จะสว่าง แสดงว่าเราสามารถสั่งการทำงานของไมโครเมาส์ได้สมบูรณ์แล้ว

คำแนะนำ : เวลาเรา copy ไฟล์มาที่ไมโครเมาส์ให้เรากดปุ่ม รีเซตก่อน โดยที่ LED สว่างสลับกันด้วย เพื่อให้แน่ใจว่าไมโครเมาส์พร้อมที่จะรับข้อมูลแล้ว

ในกรณีที่ต้องให้ไมโครเมาส์ เริ่มทำงานตั้งแต่ต้นให้กดปุ่มรีเซต และกดปุ่มสวิทตัวที่ 1 ไมโครเมาส์ก็ทำตามโปรแกรมที่เราส่งมาใหม่อีกครั้ง

คำเตือน : โปรแกรมที่เราส่งมาให้ไมโครเมาส์ จะหายไปทันทีถ้าเราปิดสวิทของไมโครเมาส์ ในกรณีที่มีโปรแกรมอยู่ที่ตัวไมโครเมาส์อยู่แล้ว แล้วเราทำการส่งโปรแกรมมาใหม่ โปรแกรมเก่าจะถูกทับทันที

รูปแบบในการเขียนโปรแกรม

การเขียนโปรแกรมต้องมีการประกาศ ดังนี้

```
#include "mouse.h"
void main(void)
{
        :
        :
        :
}
```

บรรทัดแรก เป็นการประกาศเพื่อเราจะใช้ฟังก์ชั่นต่างๆและตัวแปรของ MCS -51

บรรทัดที่ 2 เป็นส่วนของโปรแกรมหลัก

เมื่อเราทำการเขียนโปรแกรมลงไมโครเมาส์เรียบร้อยแล้ว ในกรณีที่เราได้มีการส่งค่าต่าง ๆ กลับมายังคอมพิวเตอร์ ซึ่งเราจะใช้ฟังก์ชั่น send_port() ในการส่ง ในกรณีนี้จึงจะต้องมีการรันโปรแกรม STATUS PROGRAM เพื่อดูสถานะการทำงานของไมโครเมาส์ โดยเราจะต้องเรียกโปรแกรมที่ชื่อว่า STATUS ที่ DOS ตามข้างล่างนี้

C :\c51> STATUS

โปรแกรมก็จะแสดงหน้าตาของโปรแกรมขึ้นมา พร้อมทั้งรายงานค่าต่าง ๆ ตามโปรโตคอลที่ส่งมา

บทที่ 4

ผลการทดลองและทดสอบ

ในส่วนที่เราต้องทำการทดสอบนั้นเราจะต้องเขียนโปรแกรมลงบนตัว 89C52 เพื่อทำการ ทดสอบวงจรในส่วนต่าง ๆ ดังนี้

4.1 วงจรแสดงผล

ซึ่งเราใช้ LED เป็นตัวแสดงผล เชื่อมต่อกับไอซี 8255 ที่ตำแหน่ง พอร์ท FF00H พอร์ท A และพอร์ทคอนโทรลอยู่ที่แอดเดรส FF03H ในการทดสอบจะเขียนโปรแกรมทดสอบได้ดังนี้

```
ORG  2000H
     MOV     DPTR,#0FF03H     ; พอร์ทคอนโทรลของ 8255
     MOV     A,#80H           ; ค่าที่สั่งให้พอร์ททั้งหมดเป็น o/p
     MOV     @DPTR,A          ; ส่งค่าคอนโทรลไปยัง 8255
     MOV     DPTR,#0FF00H     ; เซตค่า DPTR ให้ชี้ที่พอร์ท A
     MOV     A,#0AAH          ; เซตค่า A ให้มีค่าเท่ากับ
     MOV     @DPTR,A          ; ส่งค่า 0xAA ไปยังพอร์ท A ของ 8255
     JMP     $
```

เมื่อเขียนโปรแกรมทดสอบแล้วผลที่ได้ LED จะติดดับสลับกันไป

4.2 วงจรเซนเซอร์

ซึ่งเราใช้เซนเซอร์แบบ LED INFRAED เป็นตัวตรวจสอบกำแพง ซึ่งมันถูกเชื่อมต่อกับไอซี 8255 ที่ตำแหน่ง พอร์ท FF02H พอร์ท C และพอร์ทคอนโทรลอยู่ที่แอดเดรส FF03H ในการทดสอบสามารถเขียนโปรแกรมทดสอบได้ดังนี้

```
ORG  2000H
     MOV     DPTR,#0FF03H     ; เซตพอร์ทคอนโทรลของ 8255 ให้พอร์ท A เป็น
     MOV     A,#89H           ; o/p และพอร์ท C เป็น i/p
     MOV     @DPTR,A
LOOP: MOV    DPTR,#0FF02H     ;ให้ DPTR ชี้ที่พอร์ท C
     MOVX    A,@DPTR          ; อ่านค่าจากเซนเซอร์
     MOV     DPTR,#0FF00H     ; ให้ DPTR ชี้ที่พอร์ท A
     MOVX    @DPTR,A          ; นำค่าที่ได้แสดงที่พอร์ท A
     JMP     LOOP             ;
```

ผลที่ได้จากโปรแกรมนี้คือ เมื่อเราเอานิ้วมาผ่านบริเวณด้านหน้าของเซนเซอร์ตัวไหน LED ตัวนั้นก็จะติดตามด้วย

4.3 วงจรขับสเต็ปปิ้งมอเตอร์

เราสามารถควบคุมการทำงานของตัวสเต็ปปิ้งมอเตอร์ได้โดย ผ่านบอร์ดควบคุม ซึ่งมัถูกเชื่อมต่อกับพอร์ท 1 ของไมโครคอนโทรลเลอร์ 89C51 ในการทดสอบสามารถเขียนโปรแกรมทดสอบได้ดังนี้

```
        ORG     2000H

        MAIN:
                MOV     IE,#0A2H            ; เซตการอินเตอร์รัพท์
                MOV     RCAP2H,#0FFH        ; โหลดค่าความเร็ว ของมอเตอร์ตัวที่ 1
                MOV     RCAP2L,#00H
                MOV     TH0,#0FFH           ; โหลดค่าความเร็วให้กับมอเตอร์ตัวที่ 2
                MOV     TL0,#00H
                MOV     T2CON,#02H          ; เซตการใช้ TIMER
                MOV     TMOD,#10H           ; เซตการเลือกตัว TIMER
                SETB    TR0                 ; เซตให้ TIMER ตัวที่ 0 ทำงาน
                SETB    TR2                 ; เซตให้ TIMER ตัวที่ 2 ทำงาน
                SETB    P1.4                ; เซตให้ชุดขับมอเตอร์ทำงาน
                CLR     P1.2                ; เซตให้มอเตอร์ซ้ายวิ่งไปข้างหน้า
                CLR     P1.5                ; เซตให้มอเตอร์ขวาวิ่งไปข้างหน้า
                JMP     $
```

ผลที่ได้จากโปรแกรมนี้ตัวไมโครเมาส์จะวิ่งไปข้างหน้าทั้งสองล้อไปเรื่อย ๆ โดยเราจะทดสอบทิศทางโดยการเปลี่ยนค่าของ P1.2 และ P1.5 ให้เป็น 0 หรือ 1 ตามความต้องการ

4.4 วงจรหน่วยความจำ

เป็นวงจรที่เชื่อมต่อไมโครคอนโทรลเลอร์ ไว้สำหรับเก็บข้อมูลและคำสั่งของโปแกรมต่าง เราจะมาทดสอบว่าวงจรหน่วยความจำสามารถอ่านและเขียนได้หรือไม่ โดยการเขียนโปรแกรมดังนี้

```
        ORG     2000H
                MOV         DPTR,#0FF03H        ; ให้ DPTR ชี้ที่พอร์ทคอนโทรล
                MOV         A,#80H              ; เซตค่า A เป็น 80H
                MOVX        @DPTR,A             ; เซตให้พอร์ททั้งหมดของ 8255 เป็น o/p
```

```
MOV     A,#0FH
MOV     DPTR,#4000H     ; นำค่า 0FH ไปเก็บในหน่วยความจำที่
MOVX    @DPTR,A         ; Address 4000H
MOV     DPTR,#0FF00H    ; อ่านค่าจากหน่วยความจำที่ Address
MOVX    @DPTR,A         ; 4000H มาแสดงผลที่พอร์ท A ของ 8255
JMP     $
```

การทำงานของโปรแกรม เริ่มแรกจะทำการกำหนดให้ค่าของพอร์ท A เป็นเอาท์พุทเพราะต่อกับ LED อยู่ จากนั้นก็นำค่า 0F ไปเก็บที่ตำแหน่งของหน่วยความจำที่แอดเดรส 4000H จากนั้นทำการอ่านค่าจากตำแหน่งนี้มาเก็บไว้แล้วไปแสดงผลที่ LED เป็นค่า 0FH ถ้าเป็นไปตามนี้แสดงว่าวงจรหน่วยความจำถูกต้อง

4.5 โปรแกรมรับข้อมูลจากคอมพิวเตอร์

เป็นโปรแกรมที่รับข้อมูลจากคอมพิวเตอร์ ที่ได้จากการเขียนโปรแกรมภาษาซี แปลงเป็นไฟล์นามสกุล HEX จากนั้นก็ส่งไปยังไมโครเมาส์โดยผ่านพอร์ทอนุกรม ซึ่งโปรแกรมนี้ถูกเก็บไว้ที่ในส่วนของ FLASH ROM ที่แอดเดรส 0000 - 1FFFH โปรแกรมจะเอาข้อมูลที่ได้ไปเก็บที่หน่วยความจำที่แอดเดรส 2000 - EFFF โปรแกรมนี้อยู่ในส่วนของภาคผนวก ในการทดลองสามารถเขียนโปรแกรมบนเครื่องคอมพิวเตอร์แล้วแปลงให้เป็นไฟล์นามสกุล HEX แล้วส่งไฟล์นี้ โดยเขียนคำสั่งบน DOS ดังนี้

C:\ MODE COM1:96,N,8

C:\ COPY ชื่อไฟล์.HEX COM1

ถ้าโปรแกรมนี้ถูกต้อง คำสั่งที่เราเขียนขึ้นมาจะสามารถทำงานได้ตามที่เราเขียน

4.6 ฟังก์ชั่นต่าง ๆ

ฟังก์ชั่นที่ใช้ในโครงงานนี้ได้แสดงไว้แล้วในบทที่ 3 ซึ่งเราสามารถที่จะทดสอบการทำงานได้ โดยใช้ฟังก์ชั่นเหล่านี้ได้ตามต้องการ ซึ่งฟังก์ชั่นเหล่านี้ได้มีการทดลองการแสดงผล,การอ่านค่าเซ็นเซอร์,การเปลี่ยนทิศทาง,การเปลี่ยนตำแหน่งของรถ ปกติไม่มีปัญหาอะไร

ทดลองการปรับความเร็ว,การวิ่งตรง,การเลี้ยวซ้าย ขวา,การนับสเต็ป ต้องทดลองเปลี่ยนค่า ความเร็วและจำนวนสเต็ปการวิ่งจนเหมาะสม ทดลองการเก็บค่าลักษณะกำแพงของสนาม,การตัดสินใจในการเลือกทางวิ่งมีปัญหาเกี่ยวกับอะเรย์ดังที่กล่าวไว้ข้างต้นหลังจากแก้ไขแล้วก็ใช้งานได้เป็นปกติ

ทดลองการเลือกเส้นทางที่เร็วที่สุด ต้องทดลองกับภาษา C เพราะเราไม่สามารถดูการคำนวณค่าต่างๆจากไมโครเมาส์ได้ ผลการทดลองก็ใช้งานได้ปกติ

เราได้ทำการเปรียบเทียบฟังก์ชั่นการทำงานต่างๆระหว่างไมโครเมาส์ของเครื่องต้นแบบกับไมโครเมาส์ของโครงงานนี้ได้ผลออกมาดังนี้

1. ฟังก์ชั่นประเภทติดต่อกับผู้ใช้งาน

เครื่องต้นแบบ	**1.1 clrscr**	; ล้างหน้าจอ

	1.2 getch,getchar	; รับตัวอักษร
	1.3 gets	; รับข้อมูล
	1.4 gotoxy	; ไป ณ ตำแหน่งต่างๆบนหน้าจอ
	1.5 kbhit	; เซ็คการกดปุ่ม
	1.6 printf	; พิมพ์ข้อความบนหน้าจอ
	1.7 putchar	; ใส่ค่าตัวอักษร
	1.8 puts	; ใส่ข้อความ
	1.9 scanf	; รับข้อมูล
	1.10 putbin	; แสดงผลข้อมูลเป็นแบบบิท

เครื่องของโครงงาน ไม่มี

สรุป ในการเขียนฟังก์ชั่นประเภทนี้ตัวคอมพายเลอร์จะต้องมีส่วนของการทำงานบนเครื่องคอมพิวเตอร์ด้วยจึงจะสามารถทำงานบนคอมพิวเตอร์ได้ซึ่งคอมพายเลอร์ของเครื่องต้นแบบได้มีการออกแบบทำขึ้นมาโดยเฉพาะ แต่คอมพายเลอร์ของ 8051 โดยปกติจะไม่สามารถทำการแปลงโค๊ดให้สามารถทำงานบนเครื่องคอมพิวเตอร์ได้จึงไม่สามารถเขียนฟังก์ชั่นประเภทนี้ได้

2. ฟังก์ชั่นประเภทติดต่อกับบอร์ดคอลโทลเลอร์

เครื่องต้นแบบ	**2.1 read_switch**	; อ่านค่าสวิท
	2.1 trig	; รอการกดสวิท
	2.2 write_led	; แสดงผลที่ LED
	2.3 init_trig,init_timer	; กำหนดค่าของฟังก์ชั่น trig และ timer
	2.4 reset_timer,read_timer	; เคลียร์และอ่านค่าเวลา
	2.5 read_volt	; อ่านค่าแรงดันของตัวไมโครเมาส์
เครื่องของโครงงาน	**2.1 read_switch**	; อ่านค่าสวิท
	2.2 write_led	; แสดงผลที่ LED

สรุป บางฟังก์ชั่นไม่มีเพราะไม่จำเป็น เช่น ฟังก์ชั่น init _timer เป็นฟังก์ชั่นกำหนดค่าต่างๆ ของการใช้งานของฟังก์ชั่น timer แต่ฟังก์ชั่นของโครงงานนี้มีการกำหนดค่าเริ่มต้นไว้แล้วทั้งหมดจึงไม่ต้องมีฟังก์ชั่น กำหนดค่าเริ่มต้นต่างๆอีก ส่วนฟังก์ชั่น read_volt นั้น ต้องมีการออกแบบไว้รองรับการใช้งานด้วยแต่ฮาร์ดแวร์ของโครงงานนี้ไม่ได้ออกแบบไว้ให้ทำงานได้

3. ฟังก์ชั่นเกี่ยวกับบิท

เครื่องต้นแบบ	**3.1 bit,set,clr**	; อ่าน,เซต,เคลียร์ บิทต่างๆ
	3.2 rotr4,rotl4	; เลื่อนบิทชุดละ 4 บิท ไปทางซ้ายและขวา
	3.3 rotr8,rotl8	; เลื่อนบิทชุดละ 8 บิท ไปทางซ้ายและขวา
เครื่องของโครงงาน	**3.1 bit,set,clr**	; อ่าน,เซต,เคลียร์ บิทต่างๆ
	3.2 rotr4,rotl4	; เลื่อนบิทชุดละ 4 บิท ไปทางซ้ายและขวา
	3.3 rotr8,rotl8	; เลื่อนบิทชุดละ 8 บิท ไปทางซ้ายและขวา
	3.4 SET_GOAL	; กำหนดค่าตำแหน่งปลายทาง

สรุป ฟังก์ชั่นต่างของเครื่องโครงงานนี้สามารถทำงานได้เหมือนกับของเครื่องต้นแบบทุกประการและยังสามารถใช้งานได้กว้างกว่าด้วยเพราะฟังก์ชั่นของเครื่องต้นแบบจะใช้กับตัวแปรแบบ global เท่านั้น แต่ ฟังก์ชั่นเครื่องของโครงงานนี้จะมีการส่งค่าของในฟังก์ชั่นคืนกลับคือใช้ตัวแปรประเภทไหนก็ได้ทำให้ไม่ต้องใช้ตัวแปรแบบ global เสมอไป ส่วนฟังก์ชั่น SET_GOAL นั้นเป็นฟังก์ชั่นที่เครื่องต้นแบบไม่มีฟังก์ชั่นนี้ทำให้ผู้ใช้มีความสะดวกในการกำหนดตำแหน่งเป้าหมายให้กับไมโครเมาส์โดยไม่ต้องทำการแก้ไขโปรแกรมทุกครั้งที่กำหนดเป้าหมายใหม่

4 ฟังก์ชั่นเกี่ยวกับการเคลื่อนไหว

เครื่องต้นแบบ	**4.1 write_motor**	; สั่งงานมอเตอร์ต่างๆ
	4.2 reset_step,read_step	; เคลียร์และอ่านค่าสเต็ปมอเตอร์
	4.3 delay	; หน่วงเวลา
	4.4 read_sensor	; อ่านค่าเซ็นเซอร์
	4.5 profile	; กำหนดความเร็วที่เพิ่มขึ้นในการวิ่ง
	4.6 init_motor,init_sensor	; กำหนดค่าเริ่มต้นของ motor และ sensor
	4.7 read_motor	; อ่านค่าสถานะปัจจุบันของมอเตอร์
	4.8 load_profile	; อ่านค่าที่ได้จากการกำหนดความเร็ว
	4.9 init_move	; กำหนดค่าเริ่มต้นของฟังก์ชั่น _move
	4.10 _move	; ให้รถเคลื่อนที่ไป 1 บล็อก
	4.11 _backtrack	; กำหนดเส้นทางการเคลื่อนที่
เครื่องของโครงงาน	**4.1 motor**	; สั่งงานมอเตอร์ต่างๆ
	4.2 reset_step,read_step	; เคลียร์และอ่านค่าสเต็ปมอเตอร์
	4.3 delay	; หน่วงเวลา

	4.4 read_sensor	; อ่านค่าเซ็นเซอร์
	4.5 FAST_RUN	; เคลื่อนที่ไปยังตำแหน่งที่ต้องการอย่างเร็ว
	4.6 MOVE_CAR	; เคลื่อนที่ไปยังตำแหน่งที่ต้องการ
	4.7 SENSOR	; อ่านรูปแบบของกำแพงในขณะนั้น
	4.8 MARK_MAP	; บันทึกตำแหน่งสนาม
	4.9 UNMARK_MAP	; ยกเลิกการบันทึกตำแหน่งสนาม
	4.10 WRITE_MAP	; บันทึกลักษณะกำแพงของสนาม
	4.11 READ_MAP	; อ่านค่ากำแพงจากที่บันทึกสนามไว้
	4.12 CAR	; สั่งรถทำงานแบบต่างๆ
	4.13 CARLIBATE	; สั่งให้รถวิ่งอยู่ในแนวตรง
	4.14 NEXT_BLOCK	; เช็คดูว่าทางใดยังไม่ได้ไป
	4.15 DIRECTION	; ทิศทางการวิ่ง
	4.16 CONVERSE	; แปลงค่ากำแพงก่อนเก็บลงลักษณะกำแพง
	4.17 CHG_POINT	; เปลี่ยนตำแหน่งของรถ
	4.18 ARR	; แปลงตำแหน่งของสนามให้ตรงกับในอาเรย์
	4.19 SEND_PORT	; ส่งข้อมูลมายังคอมพิวเตอร์
	4.20 MOVE_BLOCK	; รถเคลื่อนที่ไปข้างหน้า 1 บล็อก
	4.21 PUSH_WAY	; เก็บค่าตำแหน่งสำหรับทางแยกที่ยังไม่ได้ไป
	4.22 POP_WAY	; อ่านค่าทางแยกล่าสุดที่ยังไม่ได้ไป

สรุป ฟังก์ชั่นของโครงงานไมโครเมาส์นี้ไม่ต้องมีการกำหนดค่าเริ่มต้นให้กับฟังก์ชั่นต่างๆและยังสามารถทำงานได้ครอบคุมทุกฟังก์ชั่นของเครื่องต้นแบบเว้นแต่การกำหนดค่า profile เพราะต้องมีการเรียกใช้งานโปรแกรมสำเร็จรูปที่มีหน้าที่สร้างค่า profile ขึ้นมาแต่สำหรับโครงงานนี้ยังมีฟังก์ชั่นช่วยเหลือในการเขียนโปรแกรมอีกหลายอย่าง เช่น การเก็บค่าลักษณะกำแพง การเปลี่ยนตำแหน่งรถ การมาร์คตำแหน่งของสนามซึ่งถ้าเป็นเครื่องต้นแบบผู้ใช้จะต้องทำการเขียนโปรแกรมส่วนต่างๆเหล่านี้ขึ้นเองทั้งหมด นอกจากนี้ยังมีฟังก์ชั่น SEND_PORT ใช้เพื่อตรวจสอบค่ากำแพง,ค่าตัวแปรต่างๆในขณะนั้น,ตำแหน่งและทิศทางของรถในขณะใดๆได้

5. ฟังก์ชั่นเกี่ยวกับการหาเส้นทาง

เครื่องต้นแบบ	**5.1 sloverpath**	;หาเส้นทางจากจุดเริ่มต้นไปยังจุดเป้าหมาย
	5.2 slovercost	;หาน้ำหนักของแต่ละเส้นทาง
	5.3 addwall	;เพิ่มกำแพง
	5.4 delwall	;ลบกำแพง
เครื่องของโครงงาน	**5.1 slove_path**	;หาเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดเป้าหมาย

สรุป ฟังก์ชั่นslove_path ของโครงงานนี้สามารถทำงานได้กว้างและครอบคุมการทำงานของเครื่องต้นแบบได้คือสามารถแสดงเส้นทางจากจุดหนึ่งไปยังอีกจุดหนึ่งได้โดยมีระยะทางและการใช้เวลาที่น้อยที่สุดโดยการอ่านวิเคราห์เส้นทางจากการเก็บลักษณะของสนามที่ได้มาทั้งหมด

6. ฟังก์ชั่นเกี่ยวกับการจำลองเส้นทาง

เครื่องต้นแบบ	**6.1 draw_maze**	;วาดสนาม
	6.2 dwall	;วาดกำแพง
	6.3 show	;แสดงทางวิ่ง
	6.4 init_smove	;กำหนดค่าการเคลื่อนที่
	6.5 smove	;เคลื่อนที่ไปตามทิศทางที่กำหนด
	6.6 marksq,unmarksq	;มาร์ค, ไม่มาร์คตำแหน่ง

เครื่องของโครงงาน ไม่มี

สรุป เครื่องของโครงงานนี้ไม่สามารถสร้างฟังก์ชั่นประเภทนี้ได้เนื่องจากเป็นความสามารถพิเศษของตัวคุมพายเลอร์ของต่างประเทศที่ทำขึ้นมาเฉพาะทำให้สามารถทำการจำลองการวิ่งได้

บทที่ 5
วิจารณ์และสรุป

5.1 บทสรุป

ไมโครเมาส์นั้นเป็นรถที่สามารถวิ่ง โดยมีสเต็ปปิ้งมอเตอร์ และเซนเซอร์ในการตรวจสอบกำแพง ซึ่งจะทำการเก็บค่ากำแพงของแต่ละช่องที่วิ่งผ่าน เมื่อเก็บค่าจนหมดแล้วจะทำการลบช่องที่ไม่จำเป็นออก แล้วทำการหาเส้นทางที่ไกลที่สุดจากจุดเริ่มต้นถึงปลายทาง

จากการทดลองผลที่ได้เป็นที่น่าพอใจ สามารถวิ่งไปยังทุกๆ ช่องได้อย่างถูกต้องและสามารถหาระยะทางที่สั้นที่สุดได้ เมื่อเปรียบเทียบกับเครื่องต้นแบบ ฟังก์ชั่นการทำงานของโครงงานนี้ มีประสิทธิภาพที่ทัดเทียมกับเครื่องต้นแบบได้ อีกทั้งฟังก์ชั่นการทำงานในบางหมวดของโครงงานนี้ ยังมีมากกว่าฟังก์ชั่นของเครื่องต้นแบบเสียอีก และถึงแม้บางฟังก์ชั่นของบางประเภทจะมีน้อยกว่า ทั้งนี้เนื่องจากฮาร์ดแวร์นั้นไม่สนับสนุน สาเหตุก็เนื่องมาจากเราใช้ไมโครคอนโทรลเลอร์คนละเบอร์กัน ซึ่งฟังก์เหล่านี้มีผลต่อการเขียนโปรแกรมน้อย อาจจะไม่จำเป็นต้องนำมาใช้เลยก็ได้

ในส่วนของฟังก์ชั่นที่เกียวกับการ Simulator ของเครื่องต้นแบบ โครงงานนี้ไม่ได้ทำส่วนนี้เอาไว้ เพราะว่าการจะทำฟังก์ชั่นประเภทนี้จะต้องเขียนคอมไพล์เลอร์และ Editor เอง จึงจะสามารถทำฟังก์ชั่นเหล่านี้ได้ ในการที่เราจะสร้างโปรแกรมทั้ง 2 ตัวนั้น ต้องใช้เวลาและใช้ความเชี่ยวชาญทางด้านคอมพิวเตอร์เป็นอย่างมาก จึงจะสามารถทำได้ โครงงานของเรานั้นใช้คอมไพล์เลอร์ที่ไม่ได้ยุ่งเกียวกับโปรแกรมอีดิคเตอร์ จึงไม่สามารถทำได้

5.2 ปัญหาและแนวทางแก้ไข

1. การเขียนโปรแกรมที่มีขนาดใหญ่เกินไป คอมไพเลอร์ที่เราใช้อยู่ไม่สามารถรองรับไฟล์ขนาดใหญ่ได้ จึงจำเป็นต้องนำคอมไพเลอร์ตัวใหม่มาทดแทน

2. ในช่วงแรกเมื่อเขียนโปรแกรมแล้ว การหาสาเหตุข้อบกพร่องในทำได้ยาก ทั้งนี้เนื่องจากมีตัวแปรอยู่หลายส่วน เช่น ซอฟท์แวร์ ฮาร์ดแวร์ โปรแกรมรับข้อมูล

3. วงจรไม่มีเสถียรในการทำงาน ซึ่งเกิดจากกสนามแม่เหล็กไฟฟ้าที่เกิดจากเมอเตอร์ไปรบกวนทำงานของวงจร จึงต้องใช้คาปาซิเตอร์มาต่อ เพื่อกำจัดสัญญาณรบกวนออกไป

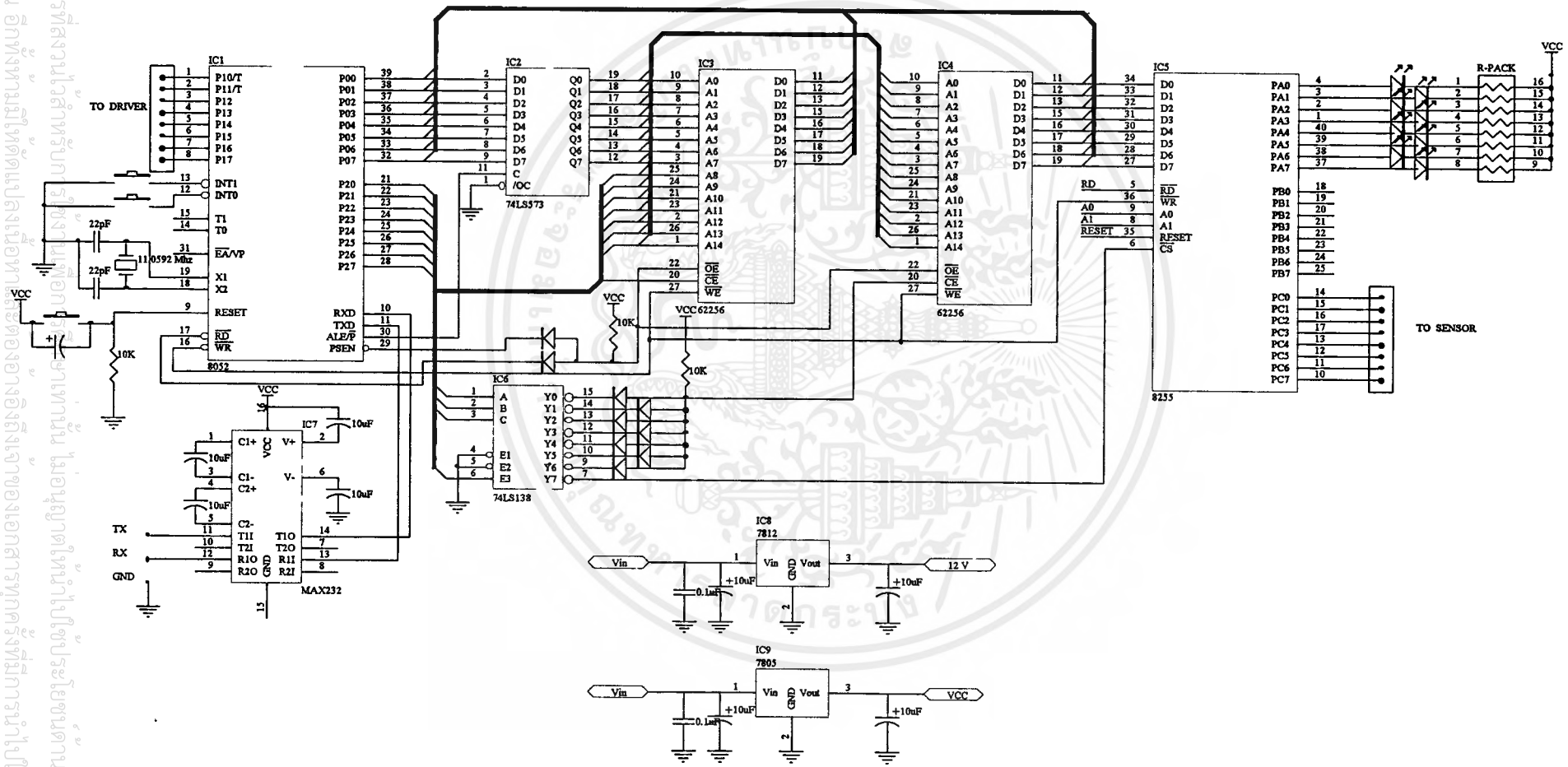
4. สนามที่ใช้ไม่ราบเรียบทำให้การวิ่งเกิดการผิดพลาด ต้องหาสนามใหม่

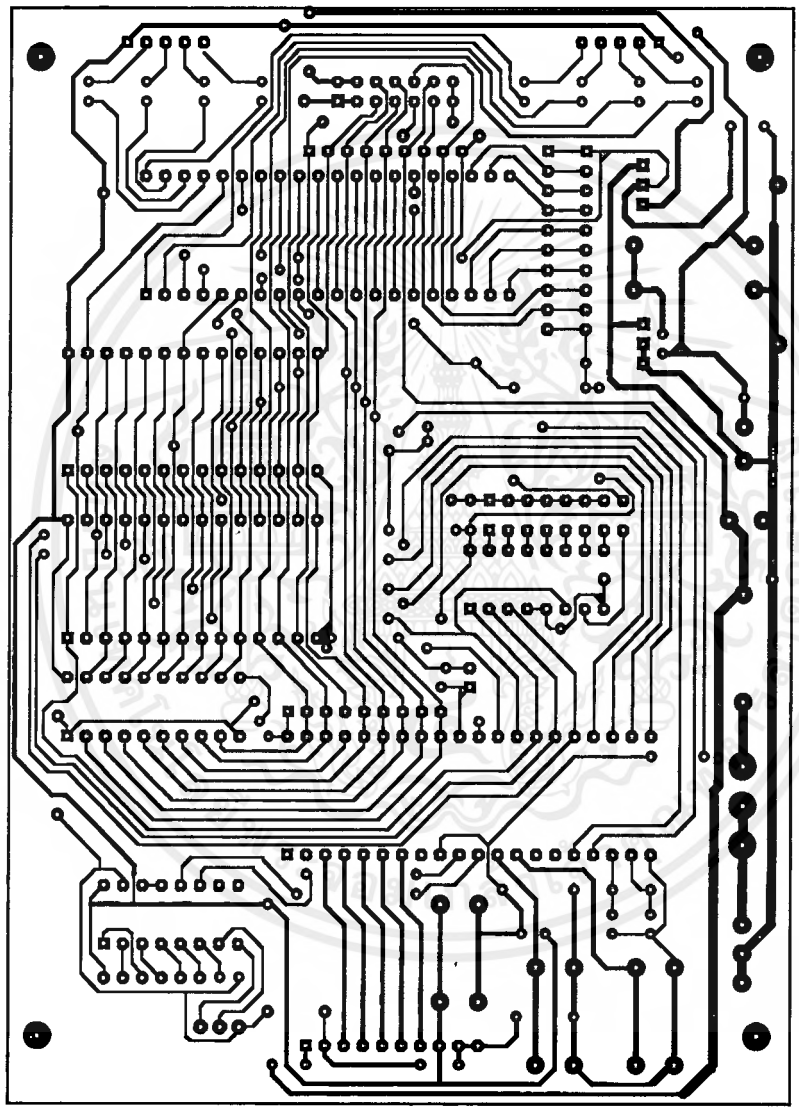
5. การเลี้ยวที่ต่อเนื่องหลาย ๆ ครั้งทำให้ไมโครเมาส์มีโอกาสวิ่งผิดพลาดได้ ทั้งนี้เนื่องจากไมโครเมาส์ไม่มีโอกาสได้ชดเชยตัวเอง เพราะฉะนั้นการลดความเร็วในการวิ่งและการให้ความกว้างระหว่างล้อแคบลง จะมีผลทำให้เกิดข้อผิดพลาดได้น้อยลงด้วย

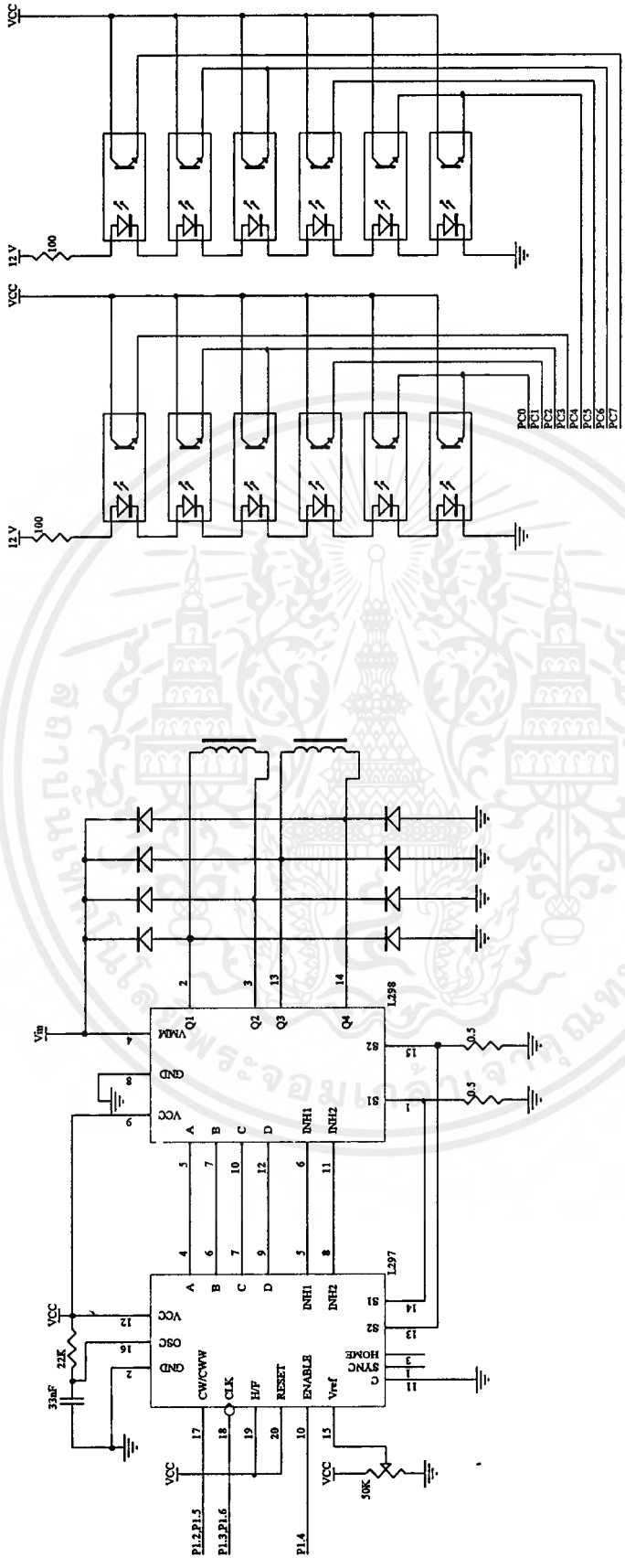
สุดท้ายนี้ทางผู้จัดทำหวังว่า โครงงานนี้จะทำให้เยาวชนหรือผู้ที่สนใจ เกิดทักษะในการเขียนโปรแกรม มีความเข้าใจการทำงานของไมโครคอนโทรลเลอร์ และให้เกิดความคิดสร้างสรรในการออกแบบสิ่งประดิษฐ์อื่น ๆ เพื่อที่จะนำสิ่งประดิษฐ์เหล่านี้ไปพัฒนาชาติของพวกเราต่อไป

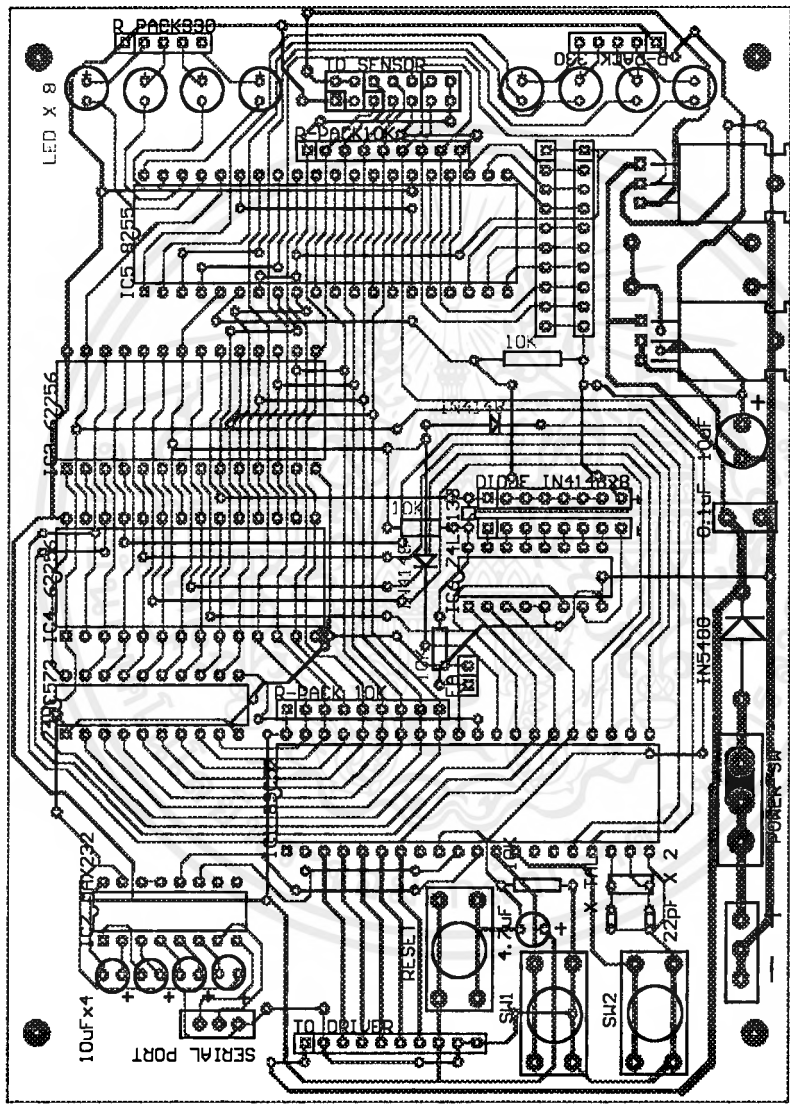
บรรณานุกรม

1. " ภาษาและการโปรแกรม C " ,ดร.วิทยา วัชระวิทยากุล ,380 หน้า ,2534
2. " ไมโครคอนโทรลเลอร์ MCS_48 MCS-51" ,พิพัฒน์ เลาหสงคราม ,450 หน้า ,2537
3. http:\\www.franklin.com
4. http:\\www.atmel.com
5. http:\\www.chipdirectory.com
6. http:\\www.maxim-ic.com


IC1
8052
TO DRIVER
P10/T
P11/T
P12
P13
P14
P15
P16
P17
INT1
INT0
T1
T0
EA/VP
X1
X2
RESET
RD
WR
P00
P01
P02
P03
P04
P05
P06
P07
P20
P21
P22
P23
P24
P25
P26
P27
RXD
TXD
ALE/P
PSEN
22pF
22pF
11.0592 Mhz
10K
VCC
IC2
74LS573
D0
D1
D2
D3
D4
D5
D6
D7
C
/OC
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
IC3
62256
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
OE
CE
WE
IC4
62256
IC5
8255
RD
WR
A0
A1
RESET
CS
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
R-PACK
TO SENSOR
IC6
74LS138
A
B
C
E1
E2
E3
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
IC7
MAX232
C1+
C1-
C2+
C2-
V+
V-
T1I
T2I
R1O
R2O
T1O
T2O
R1I
R2I
TX
RX
GND
10uF
IC8
7812
Vin
Vout
12 V
0.1uF
+10uF
IC9
7805
Vin
Vout
VCC







โปรแกรมรับข้อมูลจากคอมพิวเตอร์

```
ORG 0000H
        JMP   START

P_COUNTER    EQU   2000H

ORG 000BH
        JMP   MOTOR_LEFT
ORG 002BH
        JMP   MOTOR_RIGHT

MOTOR_LEFT:   CLR   EA
        CLR   TR0
        CLR   TF0
        CPL   P1.3
        PUSH  ACC
        MOV   TH0,07EH
        MOV   TL0,07FH
        INC   07DH      ; count step_low value of left motor
        MOV   A,07DH
        CJNE  A,#00H,INC_1 ; count step_hi  value of left motor
        INC   07CH
INC_1:     POP   ACC
        SETB  TR0
        SETB  EA
        RETI

MOTOR_RIGHT:  CLR   EA
        CLR   0CFH
        CLR   0CAH
        CPL   P1.6
        PUSH  ACC
        PUSH  DPH
        PUSH  DPL
        MOV   DPTR,#0F002H
        MOVX  A,@DPTR
        MOV   DPTR,#0EF00H
        MOVX  @DPTR,A
        INC   07BH      ; count step_low value of right motor
        MOV   A,07BH
        CJNE  A,#00H,INC_2
        INC   07AH      ; count step_hi  value of right motor
INC_2:     POP   DPL
        POP   DPH
        POP   ACC
        SETB  0CAH
        SETB  EA
        RETI
```

```
START:
        MOV   P1,#0H
        MOV   R2,#0FFH
WAIT:     MOV   R3,#0FFH
        DJNZ  R3,$
        DJNZ  R2,WAIT

        MOV   DPTR,#0F003H   ; initial 8255
        MOV   A,#80H
        MOVX  @DPTR,A
        MOV   A,#0AAH      ; display to ready
        MOV   DPTR,#0F000H
        MOVX  @DPTR,A

        MOV   A,#0FDH      ; set baud rate = 9600
        MOV   TH1,A
        MOV   TL1,A

        MOV   TMOD,#00100001B ; set timer/count 0,1 is mode 1
        MOV   TH0,#0FDH
        MOV   TL0,#00H
        SETB  EA          ; set interupt active
        CLR   ET1
        SETB  TR1
        MOV   SCON,#01010000B

RE_RX:      CLR   ES
        CLR   TI
        CLR   RI
RECEIVE:    JB    P3.2,REC
        JMP   P_COUNTER
REC:        JNB   RI,RECEIVE

        MOV   A,SBUF
        CLR   RI
        CJNE  A,#3AH,RECEIVE ; check colon code

        CALL  DECODE_CH
        MOV   R6,A      ; get counter

        CALL  DECODE_CH  ; get hi_address
        MOV   DPH,A

        CALL  DECODE_CH  ; get lo_address
        MOV   DPL,A

        CALL  DECODE_CH  ; check end
        CJNE  A,#1H,READ_DATA
        JMP   P_COUNTER

READ_DATA:  INC   R6
```

```
READ_T:     DJNZ  R6,READ    ;get data from computer to ram
            JMP   CHECK_SUM
READ:       CALL  DECODE_CH
            MOVX  @DPTR,A
            INC   DPL
            MOV   R5,DPL
            CJNE  R5,#0H,INC_DPH
            INC   DPH
INC_DPH:    JMP   READ_T

CHECK_SUM:  CALL  DECODE_CH
            JMP   RECEIVE

;=============================================================
;***********  DECODE  CHARECTER TO HEXAMAL  *********
;=============================================================
; translate ascii is hexamal

DECODE_CH:
            CLR   RI
            JNB   RI,$
            MOV   A,SBUF
            CLR   RI
            CALL  DEC_CHAR
            MOV   R4,A

            JNB   RI,$
            MOV   A,SBUF
            CLR   RI
            CALL  DEC_CHAR
            MOV   R3,A
            MOV   A,R4
            SWAP  A
            ORL   A,R3
            RET

DEC_CHAR:
            CLR   C
            SUBB  A,#30H
            MOV   R2,A
            CLR   C
            SUBB  A,#0AH
            JC    EXIT_DEC
            MOV   A,R2
            SUBB  A,#7H
            RET
EXIT_DEC:   MOV   A,R2
            RET

END
```

โปรแกรมแสดงสถานะ

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>

unsigned char wall,s_val,temp,stack,ch,ch2,dir,final,empty;
short int s_count,count;
int x,y,i,j,gd,gm,x1,y1,x2,y2,box_y[16],box_x[16];
unsigned char *ch1[1],u,v;
void INITIAL_PORT();
void CHECK_DATA(void);
unsigned char GET_PORT();
char STATUS_PORT();
void RECIEVE();
void draw_slove();
void write_list();
void time();
void reset_time();

unsigned char get_port();

char ARR_N[17][17] = {
 "DDDDDDDDCDDDDDDDD",
 "DDDDDDDCCCDDDDDDD",
 "DDDDDDCCCCCDDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDCCCCCCCCCDDDD",
 "DDDCCCCCCCCCCCDDD",
 "DDCCCCCCCCCCCCCDD",
 "DCCCCCCCCCCCCCCCD",
 "CCCCCCCCCCCCCCCCC",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD"};

char ARR_S[17][17] = {
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "DDDDDCCCCCCCDDDDD",
 "CCCCCCCCCCCCCCCCC",
```

```
    "DCCCCCCCCCCCCCCCD",
    "DDCCCCCCCCCCCCCCDD",
    "DDDCCCCCCCCCCCCDDD",
    "DDDDCCCCCCCCCCDDDD",
    "DDDDDCCCCCCCCDDDDD",
    "DDDDDDCCCCCCDDDDDD",
    "DDDDDDDCCCCCDDDDDDD",
    "DDDDDDDDCDDDDDDDD"};

  char ARR_W[17][17] = {
    "DDDDDDDDCDDDDDDDD",
    "DDDDDDDCCDDDDDDDD",
    "DDDDDDCCCDDDDDDDD",
    "DDDDDCCCCDDDDDDDD",
    "DDDDCCCCCDDDDDDDD",
    "DDDCCCCCCCCCCCCCC",
    "DDCCCCCCCCCCCCCCC",
    "DCCCCCCCCCCCCCCCC",
    "CCCCCCCCCCCCCCCCC",
    "DCCCCCCCCCCCCCCCC",
    "DDCCCCCCCCCCCCCCC",
    "DDDCCCCCCCCCCCCCC",
    "DDDDCCCCCDDDDDDDD",
    "DDDDDCCCCDDDDDDDD",
    "DDDDDDCCCDDDDDDDD",
    "DDDDDDDCCDDDDDDDD",
    "DDDDDDDDCDDDDDDDD"};
  char ARR_E[17][17] = {
    "DDDDDDDDCDDDDDDDD",
    "DDDDDDDDCCDDDDDDD",
    "DDDDDDDDCCCDDDDDD",
    "DDDDDDDDCCCCDDDDD",
    "DDDDDDDDCCCCCDDDD",
    "CCCCCCCCCCCCCCDDD",
    "CCCCCCCCCCCCCCCDD",
    "CCCCCCCCCCCCCCCCD",
    "CCCCCCCCCCCCCCCCC",
    "CCCCCCCCCCCCCCCCD",
    "CCCCCCCCCCCCCCCDD",
    "CCCCCCCCCCCCCCDDD",
    "DDDDDDDDCCCCCDDDD",
    "DDDDDDDDCCCCDDDDD",
    "DDDDDDDDCCCDDDDDD",
    "DDDDDDDDCCDDDDDDD",
    "DDDDDDDDCDDDDDDDD"};

  main()
  {
        gd = VGA;
        gm = VGAHI;
        initgraph(&gd,&gm," ");
```

```
        gotoxy(59,5);
        setfillstyle(1,7);
        bar(0,0,639,479);
        REC(1,1,639,479,8,15,1);
        REC(5,5,450,435,15,8,1);
        setfillstyle(1,3);
        bar(30,10,430,410);
        setcolor(15);
        for(i=0;i<16;i++)
         { x1=i*25+30;x2 = x1+25;
          for (j=0;j<16;j++)
           {  y1 = j*25+10;
            y2 = y1+25;
            rectangle(x1,y1,x2,y2);
            box_x[15-j] = y1;
           }
           box_y[i] = x1;
         }
        for(i=0;i<16;i++)
         { x1 = i*25+40;
          y1 = i*25;
          switch(i)
           {  case 0:ch1[0]="0";break;  case 1 :ch1[0]="1";break;case 2:ch1[0]="2";break;
            case 3:ch1[0]="3";break;  case 4 :ch1[0]="4";break;case 5:ch1[0]="5";break;
            case 6:ch1[0]="6";break;  case 7 :ch1[0]="7";break;case 8:ch1[0]="8";break;
            case 9:ch1[0]="9";break;  case 10:ch1[0]="A";break;case 11:ch1[0]="B";break;
            case 12:ch1[0]="C";break;case 13:ch1[0]="D";break;case 14:ch1[0]="E";break;
            case 15:ch1[0]="F";
           }
          setcolor(12);
          settextstyle(0,0,1);
          outtextxy(x1,418,*ch1);
          setcolor(9);
          outtextxy(15,395-y1,*ch1);
         }
        REC(5,440,635,475,15,8,1);
        settextstyle(0,0,3);
setcolor(0);
        outtextxy(57,450,"PROGRAM STATUS REPORT ");
        setcolor(13);
        outtextxy(55,450,"PROGRAM STATUS REPORT ");
                REC(455,5,635,40,15,8,1);
                REC(455,45,635,435,15,8,1);
                setfillstyle(0,9);
                bar(460,10,630,35);
                bar(460,50,630,400);

        outport(0x3f8,67);
        INITIAL_PORT();
        x = 0;y = 0;wall=2;
        draw_box(wall);
```

```
                reset_time();
                j = 0;
                outport(0x3f8,j);
                delay(1000);
                gotoxy(0,0);
                for (;;)
                 {    s_val = get_port();
                              time();
                              if ( (s_val==0xEE) )
                {  temp = 0;
                                        while(temp != 4)
                                         { get_port();
                                          temp++;
                                          switch(temp)
                                          {
                                          case 1 : wall = s_val & 0x0f;break;
                                          case 2 : x    = s_val & 0x0f;break;
                                          case 3 : y    = s_val & 0x0f;break;
                                          case 4 : dir  = s_val & 0x0f;break;
                                          }
                                          if (temp == 4)
                                           { draw_box(wall);
                                                draw_arr(box_y[y],box_x[x],dir);}
                                          }
                                   }
                              if ( (s_val==0xAA) ) {draw_slove();}
                              if ( (s_val==0xDD) ) {}
                }
}

void reset_time()
{ union REGS regs;
 regs.h.ah = 0x2d;
 regs.x.dx = 0x00;
 regs.x.cx = 0x00;
 int86(0x21,&regs,&regs);
}
void time()
{ union REGS regs;
 regs.h.ah = 0x2c;
 int86(0x21,&regs,&regs);
 gotoxy(59,2);
 printf("Time    %2d:%2d:%2d",regs.h.cl,regs.h.dh,regs.h.dl);
}
draw_arr(x,y,ch)
{char i,j;
  x = x+4;
  y = y+4;
   for (i=0;i<17;i++)
                { for(j=0;j<17;j++)
                  { if (ch ==0)
```

```
                                        break;
                    }
                  i++;
                 }
      }
    }
    void write_list()
    { unsigned char temp,x,y;
      temp = 90;
      x = 59;y=4;
      gotoxy(x,y);
      while (temp < 255)
      { if (( temp%5 ) == 0){y++;gotoxy(x,y);}
       if(y==25){y=5;gotoxy(x,y);temp = temp/4;}
       getch();
       printf("%3d ",temp);
       temp++;
      }
    }

    unsigned char get_port()
    { char u;
      if ( STATUS_PORT()&0x01)
        { s_val = inport(0x3f8);
              u = 0;
              while (u == 0)
              { if ( STATUS_PORT()&0x40)
                 { u = 1;
                        outport(0x3f8,s_val);
                 }
              }
             }
    return(s_val);
    }
    void CHECK_DATA(void)
    { if (STATUS_PORT() & 0x01)
      { if(GET_PORT() == 0xAA)
        { if (STATUS_PORT() & 0x01)
          { if(GET_PORT() == 0xAA)
              { if (STATUS_PORT() & 0x01)
                { wall = GET_PORT();
                  if(STATUS_PORT() & 0x01)
                        { dir = GET_PORT();}
                }
              }
        }
      }
    }
  else if (STATUS_PORT() & 0x01)
    { if(GET_PORT() == 0xEE)
      { if (STATUS_PORT() & 0x01)
```

```
        { if(GET_PORT() == 0xEE) {}
         }
        }
       }
      }

      void INITIAL_PORT()
      { union REGS regs;
       regs.h.ah = 0x00;
       regs.h.al = 0xE3;
       regs.x.dx = 0x00;
       int86(0x14,&regs,&regs);
      }
      char WRITE_PORT(ch)
      { union REGS regs;
       regs.h.ah = 0x01;
       regs.h.al = ch;
       regs.x.dx = 0x00;
       int86(0x14,&regs,&regs);
      }

      unsigned char GET_PORT()
      { union REGS regs;
       regs.h.ah = 2;
       regs.x.dx = 0x00;
       int86(0x14,&regs,&regs);
       return(regs.h.ah);
      }

      char STATUS_PORT()
      { union REGS regs;
       regs.h.ah = 0x03;
       regs.x.dx = 0x00;
       int86(0x14,&regs,&regs);
       return(regs.h.ah);
      }
      void reset_time()
      { union REGS regs;
       regs.h.ah = 0x2d;
       regs.x.dx = 0x00;
       regs.x.cx = 0x00;
       int86(0x21,&regs,&regs);
      }
      void time()
      { union REGS regs;
       regs.h.ah = 0x2c;
       int86(0x21,&regs,&regs);
       gotoxy(59,2);
       printf("Time    %2d:%2d:%2d",regs.h.ch,regs.h.dh,regs.h.dl);
      }
```

```
REC(X1,Y1,X2,Y2,A,B,N)
{ int I;
for(I=0;I<N;I++)
 { setcolor(A);
  line(X1+I,Y1+I,X2-I,Y1+I);
  line(X1+I,Y1+I,X1+I,Y2-I);
  setcolor(B);
  line(X2-I,Y1+I,X2-I,Y2-I);
  line(X1+I,Y2-I,X2-I,Y2-I);
 }
}

draw_box(waller)
{ int wall;
 wall = waller;
 if(wall>=8) {wall = wall-8;del_box(8);}
 if(wall>=4) {wall = wall-4;del_box(4);}
 if(wall>=2) {wall = wall-2;del_box(2);}
 if(wall>=1) {wall = wall-1;del_box(1);}
}

del_box(wal)
{
 setfillstyle(1,0);
 switch(wal)
 { case 1:bar(box_y[y]+1,box_x[x]+1,box_y[y]+25,box_x[x]+24);break;
   case 2:bar(box_y[y]+1,box_x[x],box_y[y]+24,box_x[x]+24);break;
   case 4:bar(box_y[y],box_x[x]+1,box_y[y]+24,box_x[x]+24);break;
   case 8:bar(box_y[y]+1,box_x[x]+1,box_y[y]+24,box_x[x]+25);break;
 }
}

REC2(X1,Y1,X2,Y2,A,B,N)
{ REC(X1,Y1,X2,Y2,A,B,N);
 X1 = X1+N+2;X2 = X2-N-2;
 Y1 = Y1+N+2;Y2 = Y2-N-2;
 REC(X1,Y1,X2,Y2,B,A,N);
}
```

โปรแกรม mouse.h

```
#include <stdio.h>
#include <reg52.h>   /* Define 8052 registers */
unsigned char xdata *PCTRL;
unsigned char xdata *PSENSOR;
unsigned char xdata *PDISPLAY;
unsigned char idata *SPEED_HI;
unsigned char idata *SPEED_LO;
unsigned char idata *STEP_HI_L;
unsigned char idata *STEP_LO_L;
unsigned char idata *STEP_HI_R;
unsigned char idata *STEP_LO_R;
#define HI     0xFE
#define MID_HI 0xFD
#define MID_LO 0xFC
#define LO     0xFB
#define FORWARD 'F'
#define REVERSE 'B'
#define LEFT   'L'
#define RIGHT  'R'
#define U_TURN 'U'
#define COMPENSATE 'C'
#define CLEAR    'E'
#define START    'S'
#define STOP     'O'
#define DISABLE  'D'
#define NORTH  0x00
#define EAST   0x01
#define SOUTH  0x02
#define WEST   0x03
#define MAX_X 16
#define MAX_Y 16
#define MAX   256

static unsigned char xdata map[MAX];
static unsigned char xdata m[MAX];
static unsigned char xdata x[70],y[70];

int  G,store[30];              /* pointer of top of stack */
char DIR,X,Y,WALL,NEXT_WAY,GOAL_X,GOAL_Y;          /* direction NORTH,EAST,SOUTH,WEST */
char box[256],temp_box[256],status[4],slove[30],slove2[30];
int  count,value,final1,point;

unsigned char BIT(int p,unsigned char v);
unsigned char SET(int p,unsigned char v);
unsigned char CLR(int p,unsigned char v);
unsigned char ROTL4(unsigned char v);
unsigned char ROTR4(unsigned char v);
unsigned char ROTL8(unsigned char v);
unsigned char ROTR8(unsigned char v);
```

```
void SET_GOAL(void);
char SENSOR(void);
char READ_SWITCH(char sw);
char READ_SENSOR(void);
void WRITE_LED(char p);
void MARK_MAP(char x,char y);
void UNMARK_MAP(char x,char y);
void WRITE_MAP(char x,char y);
char READ_MAP(char x,char y);
void MOTOR(char x,char y);
void RESET_STEP(void);
unsigned int READ_STEP(char p);
void CAR(char f);
void CARLIBATE(void);
void MOVE_CAR(char d,char l);
void END_B(void);
void MOVE_BLOCK(void);
void FAST_RUN(char d,char l);
char NEXT_BLOCK(char t);
char SELECT_NEXT (void);
void DIRECTION(char d);
char CONVERSE(char w);
void CHG_POINT(void);
void DELAY(int num);
int  ARR(char x,char y);
void send_port(char ch);

void delete_box(void);
void check_stack(int cnt);
void compress_box(void);
void slovepath(char x_start,y_start,x_goal,y_goal,direction,limit);
void slove_path(char x_start,y_start,x_goal,y_goal,limit);
void calculate(int y1,y2,y3);
void check_stack(int cnt);
void swap(int count);

/*---------- Main Program ----------*/

void initial(void)
{ unsigned char temp,U;
  char d;
 DELAY(100);  P1   = 0x00;  T2CON  = 0x04;  TR0   = 0;  G    = 0;
 DIR   = NORTH;    /* start at direction north */
 X     = 0x00;     /* START POINT OF MAP  */
 Y     = 0x00;     /* "----------"  */
 SPEED_HI = 0x7E;
 SPEED_LO = 0x7F;
 STEP_HI_L = 0x7A;  STEP_LO_L = 0x7B;  STEP_HI_R = 0x7C;  STEP_LO_R = 0x7D;
 *SPEED_HI = MID_HI;    /* set start speed motor RIGHT */
 *SPEED_LO = 0x00;
 RCAP2H  = MID_HI;    /* set start speed motor LEFT  */
```

```
  RCAP2L = 0x00;
  P1_4 = 0; IE = 0xA2;
  PCTRL = 0xF003; PSENSOR = 0xF002; PDISPLAY = 0xF000; *PCTRL = 0x89;
  for(U=0;U<MAX;U++) { m[U] = 0; map[U] = 0; box[U] = 0; temp_box[U] = 0; }
  for(U=0;U<70;U++) { x[U] = y[U] = 0; }
  map[0] = 0x0d; box[0] = 2; m[0] = 1;
  GOAL_X=GOAL_Y= 0;
  main();
}

unsigned char ROTL8(unsigned char v)
{ if(v&0x80) { v = v << 1; v = v | 0x01; }
    else { v = v << 1; v = v & 0xfe; }
  return(v);
}

unsigned char ROTR8(unsigned char v)
{ if(v&0x01) { v = v >> 1; v = v | 0x80; }
    else { v = v >> 1; v = v & 0x7f; }
    return(v);
}

unsigned char ROTL4(unsigned char v)
{ if(v&0x80)
  { if(v&0x08) { v = v << 1; v = v | 0x11; }
    else { v = v << 1; v = v | 0x10; v = v & 0xfe; }
    }
  else { if(v&0x08) { v = v << 1; v = v & 0xef; v = v | 0x01; }
      else { v = v << 1; v = v & 0xee; }
    }
return(v);
}

unsigned char ROTR4(unsigned char v)
{ if(v&0x10)
  { if(v&0x01) { v = v >> 1; v = v | 0x88; }
    else { v = v >> 1; v = v | 0x80; v = v & 0xf7; }
    }
  else { if(v&0x01) { v = v >> 1; v = v & 0x7f; v = v | 0x08; }
      else { v = v >> 1; v = v & 0x77; }
    }
return(v);
}

unsigned char SET(int p,unsigned char v)
{ if(p==0) { v = v|0x01; }
  else if(p==1) { v = v|0x02; }
  else if(p==2) { v = v|0x04; }
  else if(p==3) { v = v|0x08; }
  else if(p==4) { v = v|0x10; }
  else if(p==5) { v = v|0x20; }
```

```
    else if(p==6)   { v = v|0x40;  }
    else if(p==7)   { v = v|0x80;  }
   return(v);
 }

 unsigned char CLR(int p,unsigned char v)
 { if(p==0)   { v = v&0xfe;  }
   else if(p==1)   { v = v&0xfd;  }
    else if(p==2)   { v = v&0xfb;  }
    else if(p==3)   { v = v&0xf7;  }
    else if(p==4)   { v = v&0xef;  }
    else if(p==5)   { v = v&0xdf;  }
    else if(p==6)   { v = v&0xbf;  }
    else if(p==7)   { v = v&0x7f;  }
   return(v);
 }

 unsigned char BIT(int p,unsigned char v)
 { char b;
   if(p==0)   { b = v&0x01;  }
   else if(p==1)   { b = v&0x02;  }
    else if(p==2)   { b = v&0x04;  }
    else if(p==3)   { b = v&0x08;  }
    else if(p==4)   { b = v&0x10;  }
    else if(p==5)   { b = v&0x20;  }
    else if(p==6)   { b = v&0x40;  }
   else if(p==7)   { b = v&0x80;  }
  if(b==0)  { return(0); }
   else { return(1); }
 }

 void WRITE_LED(char p){  *PDISPLAY = ~p; }

 char READ_SWITCH(char sw)
 {   if (sw == 1)
      { if(INT0 == 0)     { return(1); }
        else { return(0); }
      }
     if (sw==2)
      { if(INT1 == 0)     { return(1); }
        else { return(0); }
      }
 }

 char READ_SENSOR(void)
 {  char c;
    c = *PSENSOR;
    return(c);
 }

 char SENSOR(void)
```

```
{ char l,r,c,s,w;
    s = *PSENSOR;     l = s & 0xE0;        r = s & 0x07;        c = s & 0x10;
    if (l && c && r)    w = 0x07;
    else if (l && c)    w = 0x06;
    else if (l && r)    w = 0x05;
    else if (c && r)    w = 0x03;
    else if (l)    w = 0x04;
    else if (c)    w = 0x02;
    else if (r)    w = 0x01;
    else if (s == 0x00)    w = 0x00;
    return(w);
}

void MARK_MAP(char x,char y) { m[ARR(X,Y)] = 1; }

void UNMARK_MAP(char x,char y) { m[ARR(X,Y)] = 0; }

void WRITE_MAP(char x,char y)
{ char b,e;
        b = CONVERSE(WALL);
   map[ARR(X,Y)] = b;
        e = ~b;
        e = e & 0x0f;
temp_box[ARR(X,Y)] = e;
}

char READ_MAP(char x,char y)
{ char b;
  b=map[ARR(X,Y)];
  return(b);
}

void END_B(void)
{  char s,r,b,e,temp,FG,FG2;
  FG = FG2 = 0;
 while ( FG2 == 0)
 {    if ((*STEP_HI_L==1)&&(*STEP_LO_L > 80) && (FG == 0) )
       { FG = 1;       WALL=SENSOR();      }
    if ( (FG == 1) && (*STEP_HI_L == 4) && (*STEP_LO_L > 0))
      { RESET_STEP();   FG2 = 1; CHG_POINT();    }
  else if (*PSENSOR & 0x10)
    {    CAR(MID_LO);      CHG_POINT();       r = *PSENSOR;
     while ((r & 0x08) == 0)      { r = *PSENSOR; CARLIBATE();}
        RESET_STEP();
        FG2 = 1;
    }
   else if (FG == 1)
    { temp = SENSOR();
      if ( ((temp&0x02)== 0) && (WALL != temp))  {  RESET_STEP();   CHG_POINT();   FG2 = 1; }
    }
   CARLIBATE();
```

```
    }
  }

void MOVE_CAR(char d,char l)
{ char t,i,temp;
    while(d != DIR)
    { t = DIR-1;
      if(t == 0xFF)    { t = 0x03; }
      if(t == d)
        { DIRECTION(LEFT);
          RCAP2L = 0x00;
         *SPEED_LO = 0x00;
          MOTOR(LEFT,REVERSE);
          MOTOR(RIGHT,FORWARD);
          MOTOR(LEFT,START);
          MOTOR(RIGHT,START);
          RESET_STEP();
          temp = 0;
          while( temp == 0 )  { if (*STEP_HI_L == 0x01)    { temp = 1; }      }
          temp = 0;
         *STEP_LO_L = 0;
          while( temp == 0 )  { if (*STEP_LO_L == 0xBB)   { temp = 1; }     }
       MOTOR(LEFT,STOP);
       MOTOR(RIGHT,STOP);
       RESET_STEP();
      }
      else {
           DIRECTION(RIGHT);
           RCAP2L = 0x00;
           *SPEED_LO = 0x00;
           MOTOR(LEFT,FORWARD);
           MOTOR(RIGHT,REVERSE);
           MOTOR(LEFT,START);
           MOTOR(RIGHT,START);
           RESET_STEP();
           temp = 0;
           while( temp == 0 )   { if (*STEP_HI_R == 0x01)     { temp = 1; }        }
           temp = 0;
           *STEP_LO_R = 0;
          while( temp == 0 )     { if (*STEP_LO_R == 0xAA)      { temp = 1; }      }
          MOTOR(LEFT,STOP);
          MOTOR(RIGHT,STOP);
          RESET_STEP();
        }
       }
   RESET_STEP();
   for(i=1;i<=l;i++)   { CAR(FORWARD);     CAR(HI);     END_B();    }
    CAR(STOP);
}

unsigned int READ_STEP(char p)
```

```
{ unsigned int v;
  char t;
    if (p == LEFT)
     { t = *STEP_HI_L;
       v = t << 8;
       t = *STEP_LO_L;
       v = v | t;
      }
    if (p == RIGHT)
     { t = *STEP_HI_R;
       v = t << 8;
       t = *STEP_LO_R;
       v = v | t;
      }
  return(v);
}

void MOVE_BLOCK(void)
{  char s,r,b,e,temp,FG,FG2;
  FG = FG2 = 0;
 CAR(FORWARD);
 RESET_STEP();
 while ( FG2 == 0)
 {
  if ((*STEP_HI_L==1)&&(*STEP_LO_L > 80) && (FG == 0) )
    { FG = 1;       WALL=SENSOR();     }
  if ( (FG == 1) && (*STEP_HI_L == 4) )
     {  CHG_POINT();     RESET_STEP();     FG2 = 1;    }
  else if (*PSENSOR & 0x10)
    {     CAR(MID_LO);
          WALL = WALL | 0x02;
          CHG_POINT();
          r = *PSENSOR;
          while ((r & 0x08) == 0)     { r = *PSENSOR;CARLIBATE();}
          RESET_STEP();
          FG2 = 1;
    }
  else if (FG == 1)
    { temp = SENSOR();
      if ( ((temp & 0x02)==0) && (WALL != temp))
         {  RESET_STEP();
           CHG_POINT();
           FG2 = 1;
         }
    }
  CARLIBATE();
 }
CAR(STOP);
}

void CAR(char f)
```

```
{   char temp;
  P1_4 = 1;
   if (f == FORWARD)
    {  P1_5 = 0;   P1_2 = 0;   TR2 = 1;   TR0 = 1;    }
   else if (f == STOP)
    {  TR2 = TR0 = 0;     }
   else if (f == DISABLE)
    {  P1_4 = 0;     }
   else if (f == LEFT)
    {  DIRECTION(LEFT);
   RCAP2L = 0x00;
   *SPEED_LO = 0x00;
   P1_5 = 1;
   P1_2 = 0;
   TR2 = TR0 = 1;
   RESET_STEP();
   temp = 0;
   while( temp == 0 )
     {  if (*STEP_HI_L == 0x01)
      {  temp = 1;  }
     }
   temp = 0;
   *STEP_LO_L = 0;
   if(*SPEED_HI >= MID_HI)
     {
     while( temp == 0 )      {  if (*STEP_LO_L == 0xC6)      {  temp = 1; }      }
     }
    else {       while( temp == 0 )  {  if (*STEP_LO_L == 0xC6)      {  temp = 1; }  }
          }
   TR2 = TR0 = 0;
   }
  else if (f == RIGHT)
   {  DIRECTION(RIGHT);
    RCAP2L = 0x00;
    *SPEED_LO = 0x00;
    P1_5 = 0;
    P1_2 = 1;
   TR2 = TR0 = 1;
    RESET_STEP();
    temp = 0;
   while( temp == 0 )    {  if (*STEP_HI_R == 0x01)     {  temp = 1;  }     }
  temp = 0;
  *STEP_LO_R = 0;
  if(*SPEED_HI >= MID_HI)
   {       while( temp == 0 )     {  if (*STEP_LO_R == 0xC6)     {  temp = 1; }      }
   }
  else {
    while( temp == 0 )   {  if (*STEP_LO_R == 0xC6)      {  temp = 1; }      }
   }
 TR2 = TR0 = 0;
 }
```

```
        else if (f == U_TURN)
         { DIRECTION(U_TURN);
        RCAP2L = 0x00;
        *SPEED_LO = 0x00;
        P1_5 = 1;
        P1_2 = 0;
        TR2 = TR0 = 1;
        RESET_STEP();
        temp = 0;
        while( temp == 0 )   { if (*STEP_HI_R == 0x03)    { temp = 1; }     }
        temp = 0;
        *STEP_LO_R = 0;
        while( temp == 0 )   { if (*STEP_LO_R == 0xA8)    { temp = 1; }      }
        TR2 = TR0 = 0;
        RESET_STEP();
        }
        else if (f == LO)    { RCAP2H = LO;    *SPEED_HI = LO;    }
        else if (f == MID_LO)    { RCAP2H = MID_LO;    *SPEED_HI = MID_LO;    }
        else if (f == MID_HI)    { RCAP2H = MID_HI;    *SPEED_HI = MID_HI;    }
        else if (f == HI)    { RCAP2H = HI;    *SPEED_HI = HI;    }
    }

    void CARLIBATE(void)
    {  char g;
       g = *PSENSOR;

       if ((g & 0x20)||(g & 0x01))   { *SPEED_LO = 0x00;    MOTOR(LEFT,COMPENSATE);      }
       else if ((g.& 0x80)||(g & 0x04))  { RCAP2L = 0x00;    MOTOR(RIGHT,COMPENSATE);      }
       else    { *SPEED_LO = 0x00;      RCAP2L  = 0x00;     }
    }

    void RESET_STEP()
    {  *STEP_LO_L = 0;   *STEP_LO_R = 0;   *STEP_HI_L = 0;   *STEP_HI_R = 0;  }

    void MOTOR(char x,char y)
     {  P1_4 = 1;
       if (x == LEFT)
       { if  (y == FORWARD)          P1_5 = 0;
         else if (y == REVERSE)         P1_5 = 1;
         else if (y == DISABLE)         P1_4 = 0;
         else if (y == LO)            RCAP2H = LO;
         else if (y == MID_LO)     RCAP2H = MID_LO;
         else if (y == MID_HI)      RCAP2H = MID_HI;
         else if (y == HI)          RCAP2H = HI;

         else if (y == COMPENSATE)
         {   if (RCAP2H >= MID_HI)
           RCAP2L = 0xB0;
           else RCAP2L = 0x60;
         }
         else if ( y == CLEAR)  RCAP2L = 0x00;
```

```
        else if (y == START)        TR2 = 1;
        else if (y == STOP)         TR2 = 0;
      }
      else if (x == RIGHT)
      { if (y == FORWARD)       P1_2 = 0;
        else if (y == REVERSE)      P1_2 = 1;
        else if (y == DISABLE)      P1_4 = 0;
        else if (y == LO)           *SPEED_HI = LO;
        else if (y == MID_LO)       *SPEED_HI = MID_LO;
        else if (y == MID_HI)       *SPEED_HI = MID_HI;
        else if (y == HI)           *SPEED_HI = HI;
        else if (y == COMPENSATE)
        {  if (*SPEED_HI >= MID_HI)
           *SPEED_LO = 0xB0;
           else *SPEED_LO = 0x60;
        }

      else if ( y == CLEAR)         *SPEED_LO = 0x00;
        else if (y == START)        TR0 = 1;
        else if (y == STOP)         TR0 = 0;
      }
    }

    void CHG_POINT()
    {
      char d;
      if (DIR == NORTH)    { X++ ;}
      if (DIR == SOUTH)    { X-- ;}
      if (DIR == EAST)     { Y++ ;}
      if (DIR == WEST)     { Y-- ;}
    }

    int ARR(char x,char y)
    { int f,c;
      c = x * MAX_Y;    f = (c+y);    return(f);
    }

    char SELECT_NEXT(void)
    { char c,w;
     int i,j;
     c=0;
      w = map[ARR(X,Y)];
      if (DIR == NORTH)
       { if (((w & 0x04)==0)&&(m[ARR(X,Y-1)]==0))
         { NEXT_WAY = LEFT; c++; }
        if (((w & 0x02)==0)&&(m[ARR(X+1,Y)]==0))
         { NEXT_WAY = FORWARD; c++;  }
        if (((w & 0x01)==0)&&(m[ARR(X,Y+1)]==0))
         { NEXT_WAY = RIGHT; c++;   }
        }
       else if (DIR == EAST)
```

```
        { if (((w & 0x02)==0)&&(m[ARR(X+1,Y)]==0))
          { NEXT_WAY = LEFT; c++; }
         if (((w & 0x01)==0)&&(m[ARR(X,Y+1)]==0))
          { NEXT_WAY = FORWARD; c++; }
         if (((w & 0x08)==0)&&(m[ARR(X-1,Y)]==0))
          { NEXT_WAY = RIGHT; c++; }

        }
       else if (DIR == SOUTH)
        {  if (((w & 0x01)==0)&&(m[ARR(X,Y+1)]==0))
          { NEXT_WAY = LEFT; c++; }
         if (((w & 0x08)==0)&&(m[ARR(X-1,Y)]==0))
          { NEXT_WAY = FORWARD; c++; }
         if (((w & 0x04)==0)&&(m[ARR(X,Y-1)]==0))
          { NEXT_WAY = RIGHT; c++; }
        }
       else if (DIR == WEST)
        {  if (((w & 0x08)==0)&&(m[ARR(X-1,Y)]==0))
          { NEXT_WAY = LEFT; c++; }
         if (((w & 0x04)==0)&&(m[ARR(X,Y-1)]==0))
          { NEXT_WAY = FORWARD; c++; }
         if (((w & 0x02)==0)&&(m[ARR(X+1,Y)]==0))
          { NEXT_WAY = RIGHT; c++; }
        }
    return(c);
}

char NEXT_BLOCK(char t)
{ char n;
   if (t == FORWARD)
          {  if (DIR == NORTH)   { n = m[ARR(X+1,Y)]; }
            else if (DIR == EAST)  { n = m[ARR(X,Y+1)]; }
            else if (DIR == SOUTH) { n = m[ARR(X-1,Y)]; }
            else if (DIR == WEST)  { n = m[ARR(X,Y-1)]; }
           }
          else if (t == LEFT)
           {  if (DIR == NORTH)  { n = m[ARR(X,Y-1)]; }
            else if (DIR == EAST)  { n = m[ARR(X+1,Y)]; }
            else if (DIR == SOUTH) { n = m[ARR(X,Y+1)]; }
            else if (DIR == WEST)  { n = m[ARR(X-1,Y)]; }
           }
          else if (t == RIGHT)
           {  if (DIR == NORTH)  { n = m[ARR(X,Y+1)]; }
            else if (DIR == EAST)  { n = m[ARR(X-1,Y)]; }
            else if (DIR == SOUTH) { n = m[ARR(X,Y-1)]; }
            else if (DIR == WEST)  { n = m[ARR(X+1,Y)]; }
           }
          return(n);
}
char CONVERSE(char w)
{
```

```
    if (w == 0x01)
            { if ( DIR == EAST)         w = 0x08;
              else if ( DIR == SOUTH)   w = 0x04;
              else if ( DIR == WEST)    w = 0x02;
            }
    else if (w == 0x02)
            { if ( DIR == EAST)         w = 0x01;
              else if ( DIR == SOUTH)   w = 0x08;
              else if ( DIR == WEST)    w = 0x04;
            }
    else if (w == 0x04)
            { if ( DIR == EAST)         w = 0x02;
              else if ( DIR == SOUTH)   w = 0x01;
              else if ( DIR == WEST)    w = 0x08;
            }
    else if (w == 0x05)
            { if ( DIR == EAST)         w = 0x0A;
              else if ( DIR == SOUTH)   w = 0x05;
              else if ( DIR == WEST)    w = 0x0A;
            }
    else if (w == 0x03)
            { if ( DIR == EAST)         w = 0x09;
              else if ( DIR == SOUTH)   w = 0x0C;
              else if ( DIR == WEST)    w = 0x06;
            }
           else if (w == 0x06)
            { if ( DIR == EAST)         w = 0x03;
              else if ( DIR == SOUTH)   w = 0x09;
              else if ( DIR == WEST)    w = 0x0C;
            }
           else if (w == 0x07)
            { if ( DIR == EAST)         w = 0x0B;
              else if ( DIR == SOUTH)   w = 0x0D;
              else if ( DIR == WEST)    w = 0x0E;
            }

    return(w);
}

void DIRECTION(char d)
{
  if ( d == LEFT)
    { DIR -= 0x01;
            if (DIR == 0xFF)            DIR = WEST;
    }
  else if ( d == RIGHT)
    { DIR += 0x01;
            if (DIR == 0x04)            DIR = NORTH;
    }
  else if ( d == U_TURN)
    { DIR += 0x02;
```

```
                if (DIR == 0x04)            DIR = NORTH;
                else if (DIR == 0x05)    DIR = EAST;
               }
  /*  *PDISPLAY = ~DIR;  */
 }


void DELAY(int num)
{  int i,j;
   for(j=0;j <= num;j++)
    {          for(i=0;i < 255;i++)          {  }
    }
}

void  delete_box()
{       int  y;
     char wall,ch;
  ch = 'a';
  while (ch == 'a')
  {    ch = 'b';
  for (y=0;y<=255;y++)
  {   wall = box[y];
    if ( !((y==0)||(y==point)||(y==final1)))
    {  switch(wall)
      {
      case 1 : box[y] = 0; box[y+1] = box[y+1] - 4; ch = 'a';break;
      case 2 : box[y] = 0; box[y+16] = box[y+16] - 8; ch = 'a';break;
      case 4 : box[y] = 0; box[y-1] = box[y-1] - 1; ch = 'a';break;
      case 8 : box[y] = 0; box[y-16] = box[y-16] - 2; ch = 'a';break;
      }
    }
   }
}
}

void compress_box()
{   char  i1,wall,i2,i3,flag;
    int   i,temp_y,y;
  delete_box();
for (y=0;y<=255;y++)
   {   if ((y!=0)&&(y!=point)&&(y!=final1)&&(y+1!=final1)&&
     (box[y]==9)&&(box[y+1]==12)&&(box[y-16]%2 == 1 ))
       { box[y] = box[y+1] =0;
     box[y-16] = box[y-16] - 2;
     box[y-15] = box[y-15] - 2;}
   if ((y!=0)&&(y!=point)&&(y!=final1)&&(y+1!=final1) &&
     (box[y]==3)&&(box[y+1]==6)&&(box[y+16]%2 == 1 ))
       { box[y] = box[y+1] =0;
     box[y+16] = box[y+16] - 8;
     box[y+17] = box[y+17] - 8;}
      if ((y!=0)&&(y!=point)&&(box[y]==13)||(box[y]==15))
```

```
{ temp_y = y;
 y    = y-16;
 i1   = 0;
 wall  = box[y];
 while(wall==10)
   { y = y-16;
 i1++;
 wall=box[y];}
 if (wall==3)
  { y++;wall=box[y];i2 = 0;
   while(wall==5)
    { y++;wall=box[y];i2++; }

    if (wall==6)
    { y=y+16;wall=box[y];i3=0;
   while(wall==10)
    { y=y+16;wall=box[y];i3++;}
   if(wall>11)
     {
      y = temp_y;
      flag = 'T';
      for(i=0;i<=i1;i++)
     {y=y-16;
if((y==point)||(y==final1))
       { flag = 'F';}
     }
     for(i=0;i<=i2;i++)
{   y++;   if((y==point)||(y==final1))
       {flag = 'F';}
     }
     for(i=0;i<=i3-1;i++)
     {y=y+16;
if((y==point)||(y==final1))
       { flag ='F';}
     }
     if(flag =='T')
     {
     y = temp_y;
     box[y]= box[y]-8;
     for (i=0;i<=i1;i++)          { y= y-16;box[y]= 0;}
     box[y] = 0;
     for (i=0;i<=i2;i++)          { y++;box[y]=0;}
     for (i=0;i<=i3-1;i++)        { y= y+16;box[y]=0;}
     box[y+16] = box[y+16]-8;
    }
   }
  }
 }
 y = temp_y;
 }
}
```

```
delete_box();
}

void calculate(int y1,y2,y3)
 {
  short int  temp1,temp2,temp3,temp4;
  char  final_x,final_y,x1,x2,x3;
  x1 = y1/16; y1=y1%16;
  x2 = y2/16; y2=y2%16;
  x3 = y3/16; y3=y3%16;
  final_x = final1/16;
  final_y = final1%16;

  if (x1 < final_x) temp1 = final_x-x1; else  temp1 = x1 - final_x;
  if (y1 < final_y) temp2 = final_y-y1; else  temp2 = y1 - final_y;
  temp1 = temp1 + temp2;
  if (x2 < final_x) temp2 = final_x-x2; else  temp2 = x2 - final_x;
  if (y2 < final_y) temp3 = final_y-y2; else  temp3 = y2 - final_y;
  temp2 = temp2+temp3;

  if (x3 < final_x) temp3 = final_x-x3; else  temp3 = x3 - final_x;
  if (y3 < final_y) temp4 = final_y-y3; else  temp4 = y3 - final_y;
  temp3 = temp3 + temp4;
  if ((temp1 < temp2) && (temp1 < temp3))
    { status[1] = 1; if (temp2 < temp3) { status[2] = 2; status[3] = 3; }
        else            { status[3] = 2; status[2] = 3; }}
  if ((temp2 < temp3) && (temp2 < temp1))
    { status[2] = 1; if (temp3 < temp1) { status[3] = 2; status[1] = 3; }
        else            { status[1] = 2; status[3] = 3; }}
  if ((temp3 < temp1) && (temp3 < temp2))
    { status[3] = 1; if (temp2 < temp1)  { status[2] = 2; status[1] = 3;}
        else            { status[1] = 2; status[2] = 3;}}
  if ((temp1 > temp2) &&(temp2 == temp3)) { status[2] = 1; status[3] = 2;status[1] = 3;}
  if ((temp2 > temp3) &&(temp3 == temp1)) { status[1] = 1; status[3] = 2;status[2] = 3;}
}

void check_stack(int cnt)
{ short int i,j,y,y1;
   for (i=cnt-1;i>=0;i--)
 {   for (j=cnt-2;j>=0;j--)
    {  y = store[i];
    y1 = store[j];
   if (i != j)    {      if (y==y1) {count--; j = -1;}    }
  }
 }
}

void swap()
{  int i;
  if (value > count)
    { for (i=0;i<count;i++)     { slove[i]=slove2[i];}
```

```
        value = count;
      }

}

void  slove_path(char x_start,y_start,x_goal,y_goal,limit)
{   value = 100;
   slovepath (x_start,y_start,x_goal,y_goal,'N',limit);
   swap();
   slovepath (x_start,y_start,x_goal,y_goal,'S',limit);
   swap();
   slovepath (x_start,y_start,x_goal,y_goal,'W',limit);
   swap();
   slovepath (x_start,y_start,x_goal,y_goal,'E',limit);
   swap();
   count = value;
}

void slovepath(char x_start,y_start,x_goal,y_goal,direction,limit)
{   int  i,y;
   char  wall,dir;
   char  flag,c_w,c_way,status[4];
   int   step[20];
   char  temp,stack,ch,ch2,wdir[50],way_dir[50],dir_s[50],empty;
   int   yy[40],w[40],way[40],s_time,times;
   int   value,point,final1,goal,start;
   int   s_count;
    goal  = (x_goal*16)+y_goal;
    start = (x_start*16)+y_start;
if (goal != start)
  { final1 = goal;
    point = start;
    for (i=0;i<256;i++)
   { box[i]=temp_box[i];}
    delete_box();
    empty = 'Y';
    dir   = direction;
    dir_s[0]=way_dir[0]=wdir[0] = direction;
    s_count = s_time = 100;
    times  = count = 0;
    y=start;
    final1 = goal;
    while (count >= 0)
    {
    status[1] = 1; status[2] = 2;status[3] = 3;
    wall  = box[y];
     switch(dir)
     {
     case 'N' :     switch(wall)
       {
       case 0 : y=y-16;dir='S';break;
```

```
case 1 : y++;dir = 'E';break;
case 2 : y=y+16; break;
case 4 : y--;dir='W';break;
case 5 : y++;dir='E';break;
case 3 : y=y+16; break;
case 6 : y=y+16;break;
case 7 : y=y+16;break;
case 8 : y=y-16; dir = 'S';break;
case 9 : y++; dir = 'E';break;
case 10 : y=y+16; break;
case 11 : stack = 'Y';
calculate(y+1,y+16,300);
if (status[1] == 1)
{ store[count] = y+16;
 dir_s[count] = dir;
 step[count] = times;
 dir = 'E'; y++; }
if ( status[2] == 1)
{ store[count] = y+1;
 dir_s[count] = 'E';
 step[count] = times;
 y=y+16;}
 count++;break;

case 12 : y--; dir = 'W';break;
case 13 : stack = 'Y';
calculate(y+1,300,y-1);
if (status[1] == 1)
{ store[count] = y-1;
 dir_s[count] = 'W';
 step[count] = times;
dir = 'E';y++;}
if (status[3] == 1)
{ store[count] = y+1;
 dir_s[count] = 'E';
 step[count] = times;
 y--; dir ='W';}
 count++;break;
case 14 : stack = 'Y';
calculate(300,y+16,y-1);
if (status[2] == 1)
{ store[count] = y-1;
 dir_s[count] = 'W';
 step[count] = times;
 y=y+16;}
 if (status[3] == 1)
{
 store[count] = y+16;
 dir_s[count] = 'N';
 step[count] = times;
 y--; dir ='W';}
```

```
count++;break;

 case 15 :  stack = 'Y';
 calculate(y+1,y+16,y-1);
 if (status[1] == 1)
{ store[count] = y-1;
 dir_s[count]  = 'W';
 step[count]   = times;
 count++;
 store[count] = y+16;
 dir_s[count]  = 'N';
 step[count]   = times;
 dir = 'E';y++;}

 if (status[3] == 1)
{ store[count] = y+1;
 dir_s[count]  = 'E';
 step[count]   = times;
 count++;
 store[count] = y+16;
 dir_s[count]  = 'N';
 step[count]   = times;
 dir = 'W';y--;}
 if (status[2] == 1)
{ store[count] = y+1;
 dir_s[count]  = 'E';
 step[count]   = times;
 count++;
 store[count] = y-1;
 dir_s[count]  = 'W';
 step[count]   = times;
 y=y+16;}count++;break;}break;

case 'S' :   switch(wall) {
 case 0 :  y=y+16; dir ='N';break;
 case 1 :  y++;dir='E';break;
 case 2 :  y=y+16; dir = 'N';break;
 case 3 :  y++; dir = 'E';break;
 case 4 :  y--; dir = 'W';break;
 case 5 :  y++; dir = 'E';break;
 case 6 :  y--; dir = 'W';break;
 case 8 :  y=y-16;break;
 case 9 :  y=y-16;break;
 case 12:  y--;dir='W';break;
 case 13:  y++;dir='E';break;
 case 7 :  stack = 'Y';
calculate(y+1,300,y-1);
if (status[1] == 1)
  { store[count] = y-1;
dir_s[count]  = 'W';
step[count]   = times;
```

```
dir ='E';y++;}
if (status[3] == 1)
 { store[count] = y+1;
dir_s[count]  = 'E';
step[count]   = times;
dir ='W';y--;}count++;break;
 case 10 :  y=y-16;break;
 case 11 :  stack = 'Y';
calculate(y+1,y-16,300);
if (status[1] == 1)
 { store[count]  = y-1;
dir_s[count]  = 'S';
step[count]   = times;
dir = 'E'; y++; }
if (status[2] == 1)
 { store[count] = y+1;
dir_s[count]  = 'E';
step[count]   = times;
y=y-16;}count++;break;

 case 14 : stack= 'Y';
  calculate(300,y-16,y-1);
  if (status[3] == 1)
  {store[count] = y-16;
  dir_s[count]  = 'S';
  step[count]  = times;
  y--; dir = 'W';}else
  if (status[2] == 1)
  {store[count] = y-1;
  dir_s[count]  = 'W';
  step[count]   = times;
  dir = 'S';y=y-16;}count++;break;

case 15 : stack = 'Y';
  calculate(y+1,y-16,y-1);
  if (status[1] == 1)
  {store[count] = y-1;
  dir_s[count]  = 'W';
  step[count]   = times;
  count++;
  store[count] = y-16;
  dir_s[count]  = 'S';
  step[count]   = times;
  dir = 'E';y++;}
  if (status[3] == 1)
 {store[count] = y+1;
  dir_s[count]  = 'E';
  step[count]   = times;
  count++;
  store[count] = y-16;
  dir_s[count]  = 'S';
```

```
        step[count]   = times;
        dir = 'W';y--;}
        if ( status[2] == 1)
        {store[count] = y+1;
        dir_s[count]  = 'E';
        step[count]   = times;
        count++;
        store[count] = y-1;
        dir_s[count]  = 'W';
        step[count]   = times;
        y=y-16;}count++;break;}
        break;

      case 'E' :   switch(wall) {
       case  0 :  y--;dir='W';break;
       case  2 :  y=y+16;dir='N';break;
       case  1 :  y++;break;
       case  3 :  y++;break;
       case  4 :  y--; dir = 'W';break;
       case  5 :  y++;break;
       case  6 :  y=y+16; dir = 'N';break;
       case  8 :  y=y-16; dir ='S';break;
       case  9 :  y++;dir='E';break;
       case  11:  y++;dir='E';break;
       case  7 :  stack = 'Y';
      calculate(y+1,y+16,300);
       if(status[1] == 1)
         { store[count] = y+16;
       dir_s[count]  = 'N'; step[count]   = times;
       y++;}
       if(status[2] == 1)
         {store[count] = y+1;
      dir_s[count]  = 'E';
      step[count]   = times;
      dir ='N';y=y+16;}
      count++;break;

       case 10 :  y=y-16;dir='S'; break;
       case 12 :  y=y-16;dir = 'S';break;
       case 13 :  stack = 'Y';
      calculate(y+1,y-16,300);
      if (status[1] == 1)
        {store[count] = y-16;
      dir_s[count]  = 'S';
      step[count]   = times;
      y++;}
      if (status[2] == 1)
        {store[count] = y+1;
      dir_s[count]  = 'E';
      step[count]   = times;
      dir ='S';y = y-16;}count++;break;
```

```
 case 14 :  stack = 'Y';
 calculate(y-16,y+16,300);
 if (status[1] == 1)
   {store[count] = y+16;
 dir_s[count] = 'N'; step[count]   = times;
 dir = 'S';y=y-16;}
 if (status[2] == 1)
   {store[count] = y-16;
 dir_s[count]  = 'S'; step[count]   = times;
 dir = 'N';y=y+16;}count++;break;

 case 15 :  stack = 'Y';
 calculate(y+1,y-16,y+16);
 if (status[1] == 1 )
  { store[count] = y-16;
 dir_s[count]  = 'S';
 step[count]   = times;
 count++;
 store[count] = y+16;
 dir_s[count]  = 'N';
 step[count]   = times;
 y++;}
 if (status[2] == 1 )
  { store[count] = y+1;
 dir_s[count]  = 'E';
 step[count]   = times;
 count++;
 store[count] = y+16;
 dir_s[count]  = 'N';
 step[count]   = times;
 y=y-16; dir = 'S';}
 if (status[3] == 1)
  { store[count] = y+1;
 dir_s[count]  = 'E';
 step[count]   = times;
 count++;
 store[count] = y-16;
 dir_s[count]  = 'S';
 step[count]   = times;
 dir = 'N'; y=y+16; }count++;break;}
 break;
case 'W' :   switch(wall) {
 case 0 : y++;dir = 'E';break;
 case 1 : dir = 'E';y++; break;
 case 2 : dir = 'N';y=y+16;break;
 case 3 : dir = 'N';y=y+16; break;
 case 5 : y--; break;
 case 6 : y--; break;
 case 4 : y--; break;
 case 10: y=y+16;dir='N';break;
 case 14: y=y+16;dir='N';break;
```

```
case 12: y--; break;
case 8 : y=y-16;dir='S';break;
case 7 : stack = 'Y';
 calculate(y-1,y+16,300);
 if (status[1] == 1)
 {store[count] = y+16;
 dir_s[count]  = 'N';
 step[count]   = times;
 y--;}
 if (status[2] == 1)
 {store[count] = y-1;
 dir_s[count]  = 'W';
 step[count]   = times;
 dir ='N';y=y+16;}count++;break;

case 9 : y=y-16;dir ='S';break;
case 11: stack = 'Y';
 calculate(y+16,y-16,300);
 if (status[1]==1)
{ store[count] = y-16;
 dir_s[count]  = 'S';
 step[count]   = times;
 y=y+16; dir = 'N';}
 if (status[2] == 1)
{ store[count] = y+16;
 dir_s[count]  = 'N';
 step[count]   = times;
 y=y-16;dir = 'S';}count++;break;
case 13: stack = 'Y';
 calculate(y-16,y-1,300);
 if ( status[1] == 1 )
 { store[count] = y-1;
 dir_s[count]  = 'W';
 step[count]   = times;
 dir='S';y=y-16;}
 if ( status[2] == 1)
 { store[count] = y-16;
 dir_s[count]  = 'S'; step[count]   = times;
 y--;}count++;break;

case 15:  stack = 'Y';
 calculate(y-1,y-16,y+16);
 if (status[1] == 1)
{store[count] = y-16;
 dir_s[count]  = 'S';
 step[count]   = times;
 count++;
 store[count] = y+16;
 dir_s[count]  = 'N'; step[count]   = times;
 y--;}
```

```
        if (status[2] == 1)
        { store[count] = y-1;
          dir_s[count]  = 'W'; step[count]   = times;
          count++;
          store[count] = y+16;
          dir_s[count]  = 'N'; step[count]   = times;
          y=y-16;dir='S';}
        if (status[3] == 1)
        { store[count] = y-1;
          dir_s[count]  = 'W'; step[count]   = times;
          count++;
          store[count] = y-16;
          dir_s[count]  = 'S'; step[count]   = times;
          y=y+16;dir='N';}count++;break;}break;
       }
      if (stack == 'Y') {
      check_stack(count); }
      way[times] = y;
      way_dir[times] = dir;

      stack = 'N';
      if ((times > limit) ||(times > s_time))
       {
      y    = store[count-1];
      dir  = dir_s[count-1];
      times = step[count-1];
      way[times] = y;
      way_dir[times] = dir;
      count--;
       }

      if ((y==final1)&&(times <= s_time))
      {
       for (i= 0;i<=count-1;i++)
       {yy[i]  = store[i];}
       ch = ch2 = wdir[0];
       c_w = c_way = 0;
       for (i=0;i<=times;i++)
       { if ( wdir[i] != ch )   { c_w++;   ch = wdir[i];}
        if ( way_dir[i] != ch2) { c_way++; ch2 = way_dir[i]; }
       }
       if  ((times< s_time)||(empty == 'Y')||(c_way < c_w)) {
       for (i=0;i<=times;i++)
       {
        w[i] = way[i];
        wdir[i] = way_dir[i];
        } }

        empty = 'N';
        y    = store[count-1];
        s_time = times;
```

```
        s_count = count;
        count--;
        times   = step[count];
        way[times] = y;
        dir     = dir_s[count];
        way_dir[times] = dir;
       }
        times++;
       }
       temp =  wdir[0];
       s_count = count = 0;
        wdir[s_time+1] = 'F';
       for (i=0;i<=s_time+1;i++)
       {
      if (wdir[i] == temp)
      { switch(wdir[i])
        { case  'N' : dir = 0;break;
       case  'E' : dir = 1;break;
       case  'S' : dir = 2;break;
       case  'W' : dir = 3;break;
      }
      s_count++;
      }
       else {
      slove2[count]   =  dir;
      slove2[count+1] =  s_count;
      count = count +2;
      s_count  = 0;
      temp = wdir[i];
      i--;
      }
     }

  } else { count = 0;}
  }

 void send_port(char ch)
 { char u,v,temp;
   u = 0;
   while (u == 0)
    { if (TI == 1 )
      {
       SBUF =  ch;
       v = TI= 0;
       while(v == 0)
       { if (RI== 1)
         {  v = 1;
          u = 1;
          RI = 0;
          ch = SBUF;
          *PDISPLAY = ~ch;
```

```
            }
          }
        }
      }
    }

    void FAST_RUN(char d,char l)
    { char t,j,s,r,b,e,FG,FG2,temp;
      while(d != DIR)
      { t = DIR-1;
        if(t == 0xFF)
          { t = 0x03; }
        if(t == d)
          { DIRECTION(LEFT);
            RCAP2L = 0x00;
            *SPEED_LO = 0x00;
            MOTOR(LEFT,REVERSE);
            MOTOR(RIGHT,FORWARD);
            MOTOR(LEFT,START);
            MOTOR(RIGHT,START);
            RESET_STEP();
            temp = 0;
            while( temp == 0 )
            { if (*STEP_HI_L == 0x01)
                { temp = 1; }
            }
            temp = 0;
            *STEP_LO_L = 0;
             while( temp == 0 )
            { if (*STEP_LO_L == 0xBB)
                { temp = 1; }
            }
           MOTOR(LEFT,STOP);
           MOTOR(RIGHT,STOP);
           RESET_STEP();
          }
          else {
              DIRECTION(RIGHT);
              RCAP2L = 0x00;
              *SPEED_LO = 0x00;
              MOTOR(LEFT,FORWARD);
              MOTOR(RIGHT,REVERSE);
              MOTOR(LEFT,START);
              MOTOR(RIGHT,START);
              RESET_STEP();
              temp = 0;
              while( temp == 0 )
               { if (*STEP_HI_R == 0x01)
                  { temp = 1; }
               }
              temp = 0;
```

```
                { switch(ARR_N[j][i])
                        {
                        case 'D': putpixel(i+x,j+y,0);break;
                        case 'C': putpixel(i+x,j+y,14);break;
                        }
                }
                if (ch ==2)
                {    switch(ARR_S[j][i])
                 {
                        case 'D': putpixel(i+x,j+y,0);break;
                        case 'C': putpixel(i+x,j+y,14);break;
                 }
                }
                if (ch == 1)
                 { switch(ARR_E[j][i])
                        {case 'D': putpixel(i+x,j+y,0);break;
                        case 'C': putpixel(i+x,j+y,14);break;
                        }
                 }
                if (ch == 3)
                 {  switch(ARR_W[j][i])
                        {
                        case 'D': putpixel(i+x,j+y,0);break;
                        case 'C': putpixel(i+x,j+y,14);break;
                        }
                 }
               }
             }
}
void draw_slove()
{ char temp,i,s,x,y,j;
  x=y=7;
  temp =get_port() & 0x0f;
  while (get_port() != 0xAA)
   { for (i=3;i<temp;i++)
             {  switch(i)
                { case 0 : s = get_port();
                                for (j=0;j<s;j++)
                                { y++;draw_arr(box_y[x]+5,box_x[y]+5,'N');}
                                break;
                        case 1 : s = get_port();
                                for (j=0;j<s;j++)
                                { x++;draw_arr(box_y[x]+5,box_x[y]+5,'E');}
                                break;
                        case 2 : s = get_port();
                                for (j=0;j<s;j++)
                                { y--;draw_arr(box_y[x]+5,box_x[y]+5,'S');}
                                break;
                        case 3 : s = get_port();
                                for (j=0;j<s;j++)
                                { x--;draw_arr(box_y[x]+5,box_x[y]+5,'W');}
```

```
            *STEP_LO_R = 0;
            while( temp == 0 )
              { if (*STEP_LO_R == 0xAA)
                {  temp = 1; }
              }
            MOTOR(LEFT,STOP);
            MOTOR(RIGHT,STOP);
            RESET_STEP();
          }
         }
    RESET_STEP();
    for(i=1;i<=l;i++)
              { CAR(FORWARD);
     if(i==1){ CAR(MID_HI);}
     FG = FG2 = 0;
     while ( FG2 == 0)
    {
     if ((*STEP_HI_L==1)&&(*STEP_LO_L > 80) && (FG == 0) )
        { FG = 1; if(i==1){ CAR(HI);}
          WALL=SENSOR();
        }
     if ( (FG == 1) && (*STEP_HI_L == 3) && (i == l)) { CAR(MID_LO);}

     if ( (FG == 1) && (*STEP_HI_L == 4) && (*STEP_LO_L > 0))
        { RESET_STEP();
          FG2 = 1;CHG_POINT();
        }
     else if (*PSENSOR & 0x10)
       {   CAR(MID_LO);
           CHG_POINT();
           r = *PSENSOR;
           while ((r & 0x08) == 0)
           { r = *PSENSOR;CARLIBATE();}
           RESET_STEP();
           FG2 = 1;
       }
      else if (FG == 1)
       { temp = SENSOR();
         if ( ((temp&0x02)== 0) && (WALL != temp))
           {  RESET_STEP();
              CHG_POINT();
              FG2 = 1;
           }
       }
      CARLIBATE();
    }
     }
      CAR(STOP);
      CAR(MID_LO);
  }
```

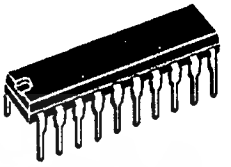
```
void SET_GOAL(void)
{ char d;
 GOAL_X=GOAL_Y= 0;
 d= 0;
  while( d == 0)
   { if (INT0 == 0) { GOAL_X++;}
    if (INT1 == 0) { d = 1;}
    if(GOAL_X==16){GOAL_X = 0;}
    *PDISPLAY = ~GOAL_X;
    DELAY(50);
  }
d = 0;
while ( d==0)
{ if(INT1==1) { d = 1;}}
   *PDISPLAY = 0x0F;
   DELAY(100);
   d = 0;
   while( d == 0)
    { if (INT0 == 0) { GOAL_Y++;}
     if (INT1 == 0) { d = 1;}
     if(GOAL_Y==16){GOAL_Y = 0;}
      *PDISPLAY = ~GOAL_Y;
      DELAY(50);
    }
}
/*---------- End Program ----------*/
```

SGS-THOMSON MICROELECTRONICS

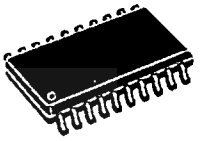
L297/A/D

STEPPER MOTOR CONTROLLERS

- NORMAL/WAWE DRIVE
- HALF/FULL STEP MODES
- CLOCKWISE/ANTICLOCKWISE DIRECTION
- SWITCHMODE LOAD CURRENT REGULATION
- PROGRAMMABLE LOAD CURRENT
- FEW EXTERNAL COMPONENTS
- RESET INPUT & HOME OUTPUT
- ENABLE INPUT
- STEP-PULSE DOUBLER (L297A only)



DIP20



SO20

ORDERING NUMBERS: L297 - L297A (DIP20)
L297D (SO20)

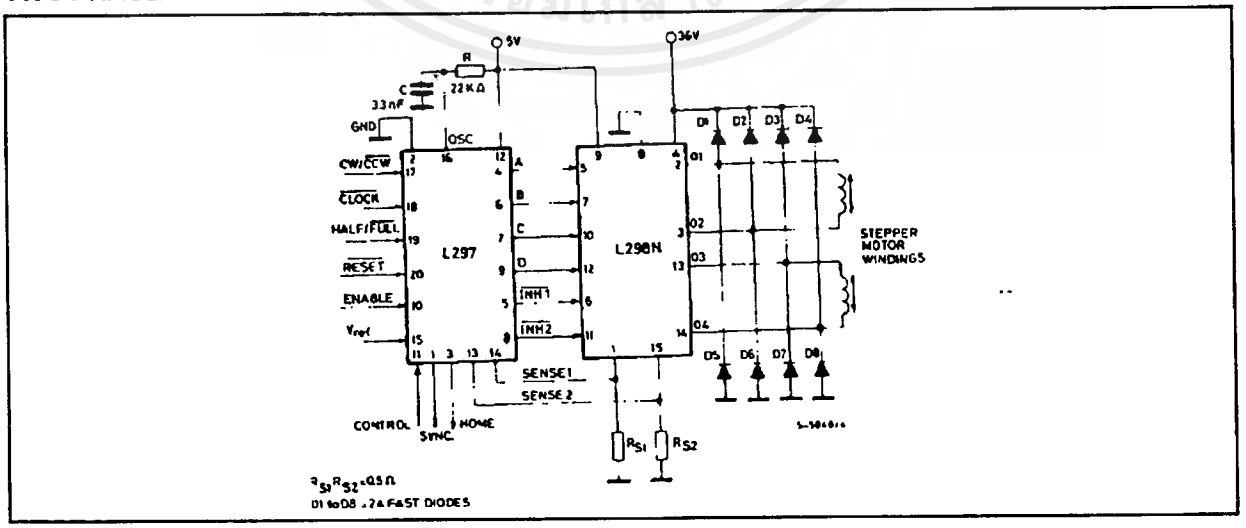
DESCRIPTION

The L297/A/D Stepper Motor Controller IC generates four phase drive signals for two phase bipolar and four phase unipolar step motors in microcomputer-controlled applications. The motor can be driven in half step, normal and wawe drive modes and on-chip PWM chopper circuits permit switchmode control of the current in the windings. A feature of this device is that it requires only clock, direction and mode input signals. Since the phase are generated internally the burden on the microprocessor, and the programmer, is greatly reduced. Mounted in DIP20 and SO20 packages, the L297 can be used with monolithic bridge drives such as the L298N or L293E, or with discrete transistors

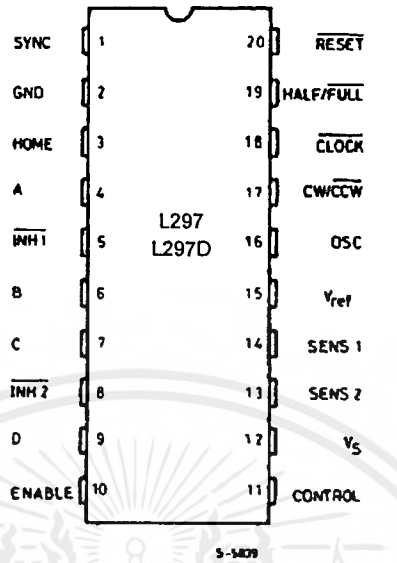
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply voltage	10	V
V_i	Input signals	7	V
P_{tot}	Total power dissipation (T_{amb} = 70°C)	1	W
T_{stg}, T_j	Storage and junction temperature	-40 to + 150	°C

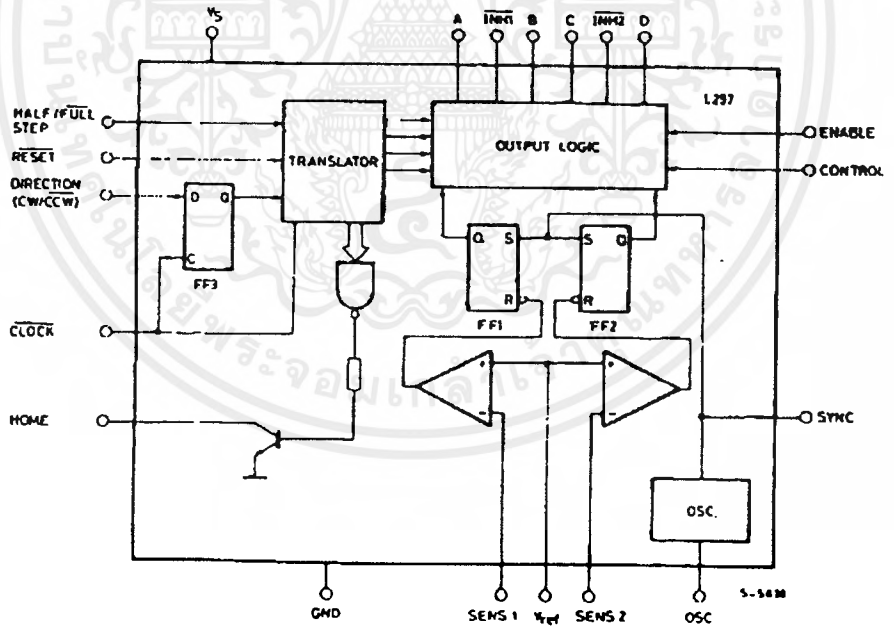
TWO PHASE BIPOLAR STEPPER MOTOR CONTROL CIRCUIT



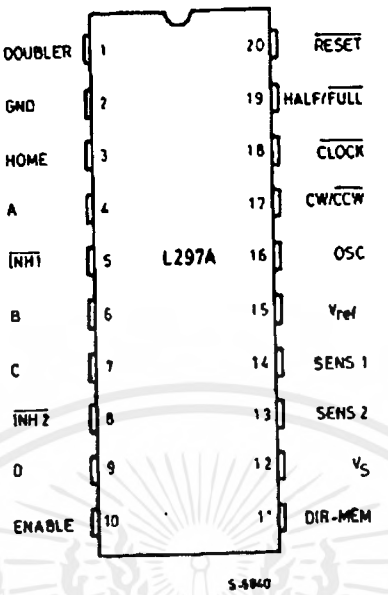
PIN CONNECTION (L297)



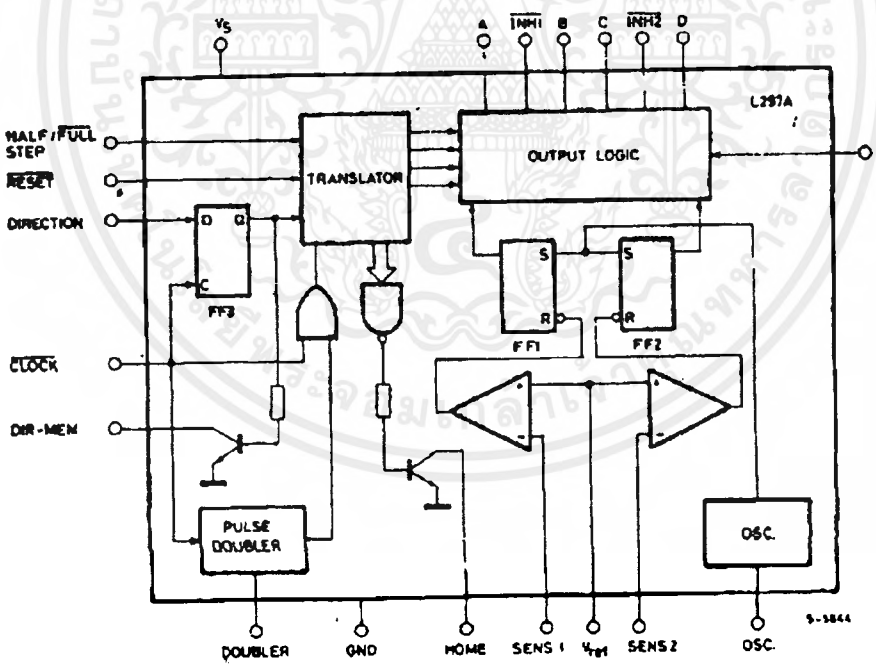
BLOCK DIAGRAM (L297/L297D)



PIN CONNECTION (L297A)



BLOCK DIAGRAM (L297A)



THERMAL DATA

Symbol	Parameter		DIP20	SO20	Unit
$R_{th\text{-}j\text{-}amb}$	Thermal resistance junction-ambient	max	80	100	°C/W

PIN FUNCTIONS - L297/L297D

N°	NAME	FUNCTION
1	SYNC	Output of the on-chip chopper oscillator. The SYNC connections The SYNC connections of all L297s to be synchronized are connected together and the oscillator components are omitted on all but one. If an external clock source is used it is injected at this terminal.
2	GND	Ground connection.
3	HOME	Open collector output that indicates when the L297 is in its initial state (ABCD = 0101). The transistor is open when this signal is active.
4	A	Motor phase A drive signal for power stage.
5	$\overline{INH1}$	Active low inhibit control for driver stage of A and B phases. When a bipolar bridge is used this signal can be used to ensure fast decay of load current when a winding is de-energized. Also used by chopper to regulate load current if CONTROL input is low.
6	B	Motor phase B drive signal for power stage.
7	C	Motor phase C drive signal for power stage.
8	$\overline{INH2}$	Active low inhibit control for drive stages of C and D phases. Same functions as INH1.
9	D	Motor phase D drive signal for power stage.
10	ENABLE	Chip enable input. When low (inactive) INH1, INH2, A, B, C and D are brought low.
11	CONTROL	Control input that defines action of chopper. When low chopper acts on INH1 and INH2; when high chopper acts on phase lines ABCD.
12	V_s	5V supply input.
13	$SENS_2$	Input for load current sense voltage from power stages of phases C and D.
14	$SENS_1$	Input for load current sense voltage from power stages of phases A and B.
15	V_{ref}	Reference voltage for chopper circuit. A voltage applied to this pin determines the peak load current.
16	OSC	An RC network (R to V_{CC}, C to ground) connected to this terminal determines the chopper rate. This terminal is connected to ground on all but one device in synchronized multi - L297 configurations. $f \cong 1/0.69\ RC$
17	CW/$\overline{CCW}$	Clockwise/counterclockwise direction control input. Physical direction of motor rotation also depends on connection of windings. Synchronized internally therefore direction can be changed at any time.
18	$\overline{CLOCK}$	Step clock. An active low pulse on this input advances the motor one increment. The step occurs on the rising edge of this signal.

PIN FUNCTIONS - L297/L297D (continued)

N°	NAME	FUNCTION
19	HALF/$\overline{\text{FULL}}$	Half/full step select input. When high selects half step operation, when low selects full step operation. One-phase-on full step mode is obtained by selecting FULL when the L297's translator is at an even-numbered state. Two-phase-on full step mode is set by selecting FULL when the translator is at an odd numbered position. (The home position is designate state 1).
20	$\overline{\text{RESET}}$	Reset input. An active low pulse on this input restores the translator to the home position (state 1, ABCD = 0101).

PIN FUNCTIONS - L297A
Pin function of the L297A are identical to those of the L297/L297D except for pins 1 and 11.

N°	NAME	FUNCTION
1	DOUBLER	An RC network connected to this pin determines the delay between an input clock pulse and the corresponding ghost pulse.
11	DIR-MEM	Direction Memory. Inverted output of the direction flip flop. Open collector output.

CIRCUIT OPERATION

The L297(A) is intended for use with a dual bridge driver, quad darlington array or discrete power devices in step motor driving applications. It receives step clock, direction and mode signals from the systems controller (usually a microcomputer chip) and generates control signals for the power stage.

The principal functions are a translator, which generates the motor phase sequences, and a dual PWM chopper circuit which regulates the current in the motor windings. The translator generates three different sequences, selected by the HALF/FULL input. These are normal (two phases energised), wave drive (one phase energised) and half-step (alternately one phase energised/two phases energised). Two inhibit signals are also generated by the L297 in half step and wave drive modes. These signals, which connect directly to the L298's enable inputs, are intended to speed current decay when a winding is de-energised. When the L297 is used to drive a unipolar motor the chopper acts on these lines.

An input called CONTROL determines whether the chopper will act on the phase lines ABCD or the inhibit lines INH1 and INH2. When the phase lines are chopped the non-active phase line of each pair (AB or CD) is activated (rather than interrupting the line then active). In L297 + L298 configurations this technique reduces dissipation in the load current sense resistors.

A common on-chip oscillator drives the dual chopper. It supplies pulses at the chopper rate which set the two flip-flops FF1 and FF2. When the current in a winding reaches the programmed peak value the voltage across the sense resistor (connected to one of the sense inputs $SENS_1$ or $SENS_2$) equals V_{ref} and the corresponding comparator resets its flip flop, interrupting the drive current until the next oscillator pulse arrives. The peak current for both windings is programmed by a voltage divider on the V_{ref} input.

Ground noise problems in multiple configurations can be avoided by synchronising the chopper oscillators. This is done by connecting all the SYNC pins together, mounting the oscillator RC network on one device only and grounding the OSC pin on all other devices.

The L297A includes a pulse doubler on the step clock line which is intended to simplify the implementation of multiple stepping. A ghost pulse is generated automatically after each input pulse, delayed by the time $0.75\ R_d\ C_d$.

The RC network should be dimensioned to place the ghost pulse roughly halfway between clock pulses. If pin 1 (DOUBLER) is grounded the doubler function is disabled.

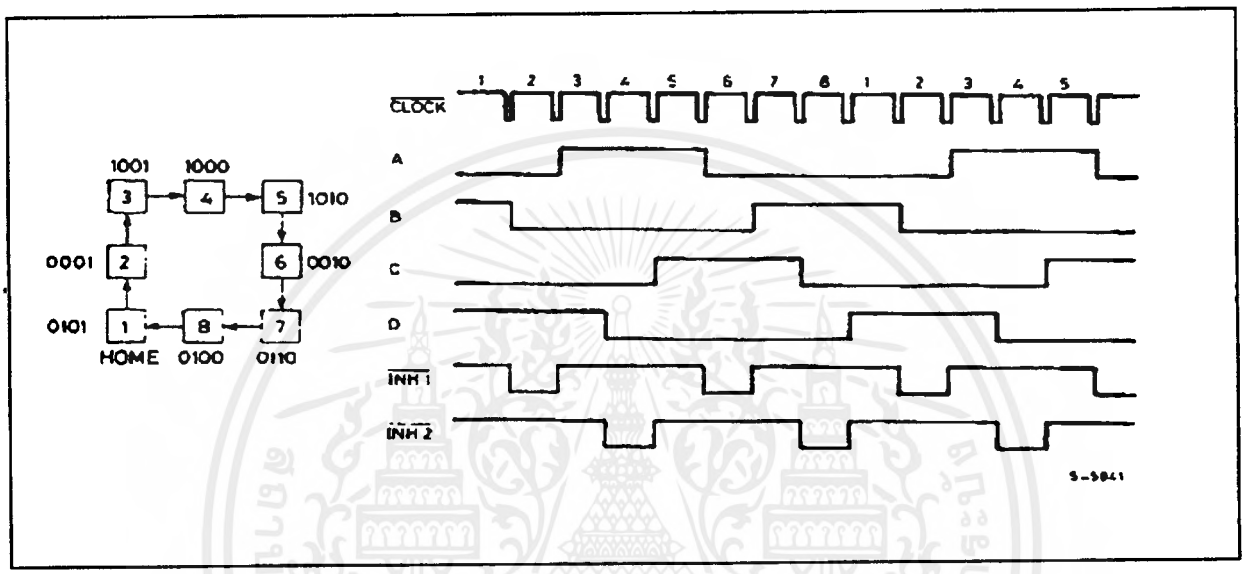
MOTOR DRIVING PHASE SEQUENCES

The L297's translator generates phase sequences for normal drive, wave drive and half step modes. The state sequences and output waveforms for these three modes are shown below. In all cases the translator advances on the low to high transistion of CLOCK.

Clockwise rotation is indicate; for anticlockwise rotation the sequences are simply reversed RESET restores the translator to state 1, where ABCD = 0101.

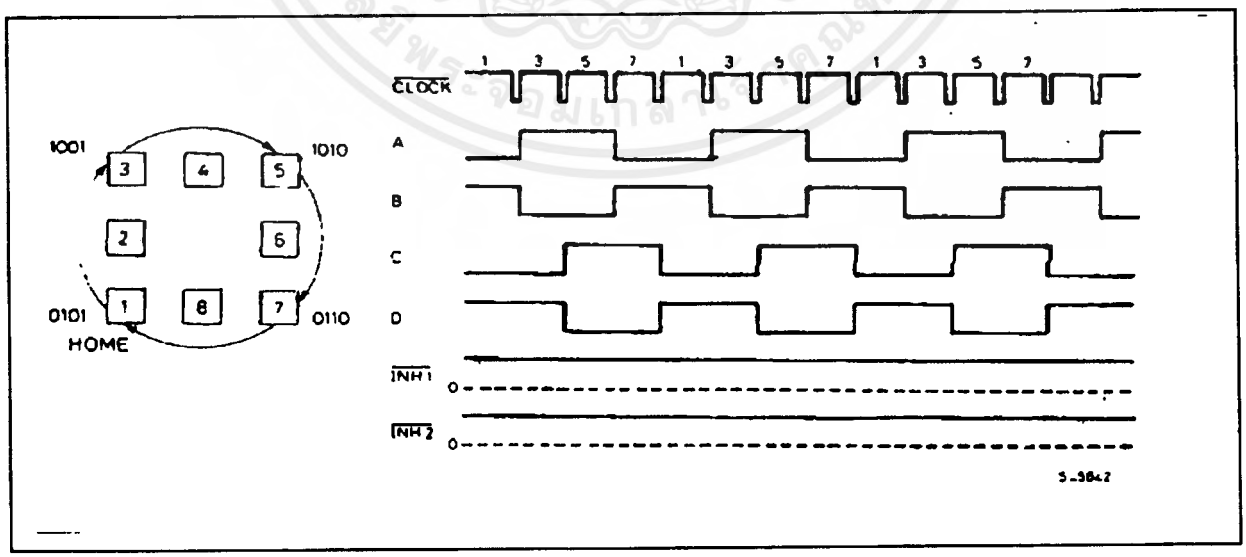
HALF STEP MODE

Half step mode is selected by a high level on the HALF/FULL input.



NORMAL DRIVE MODE

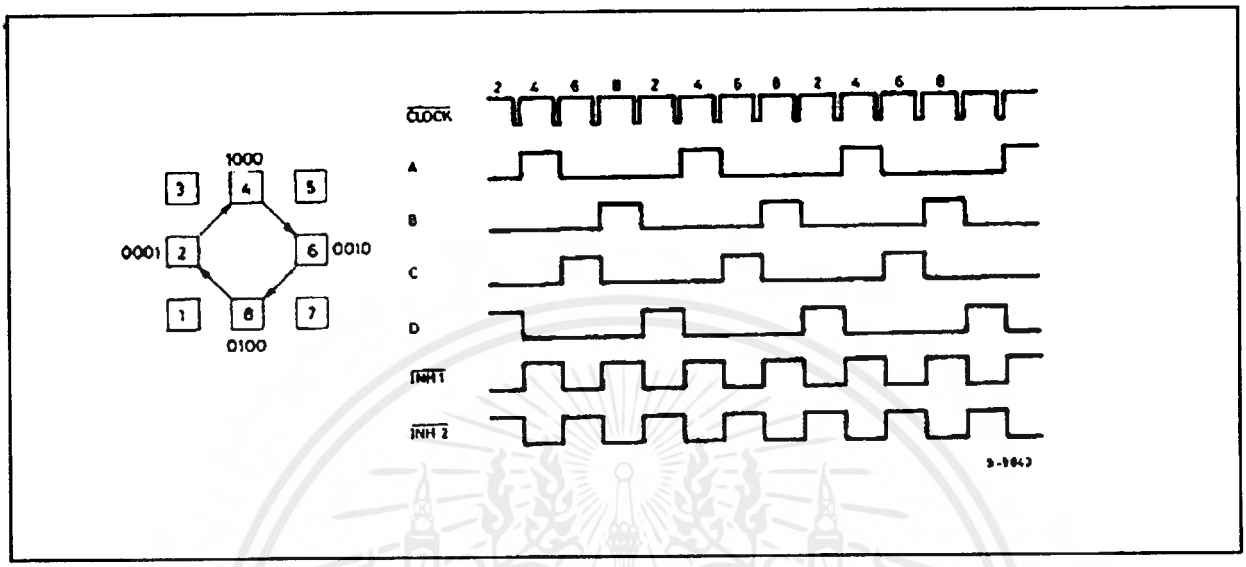
Normal drive mode (also called "two-phase-on" drive) is selected by a low level on the HALF/FULL input when the translator is at an odd numbered state (1, 3, 5 or 7). In this mode the INH1 and INH2 outputs remain high throughout.



MOTOR DRIVING PHASE SEQUENCES (continued)

WAVE DRIVE MODE

Wave drive mode (also called "one-phase-on" drive) is selected by a low level on the HALF/$\overline{FULL}$ input when the translator is at an even numbered state (2, 4, 6 or 8).



ELECTRICAL CHARACTERISTICS (Refer to the block diagram T_{amb} = 25°C, V_s = 5V unless otherwise specified)

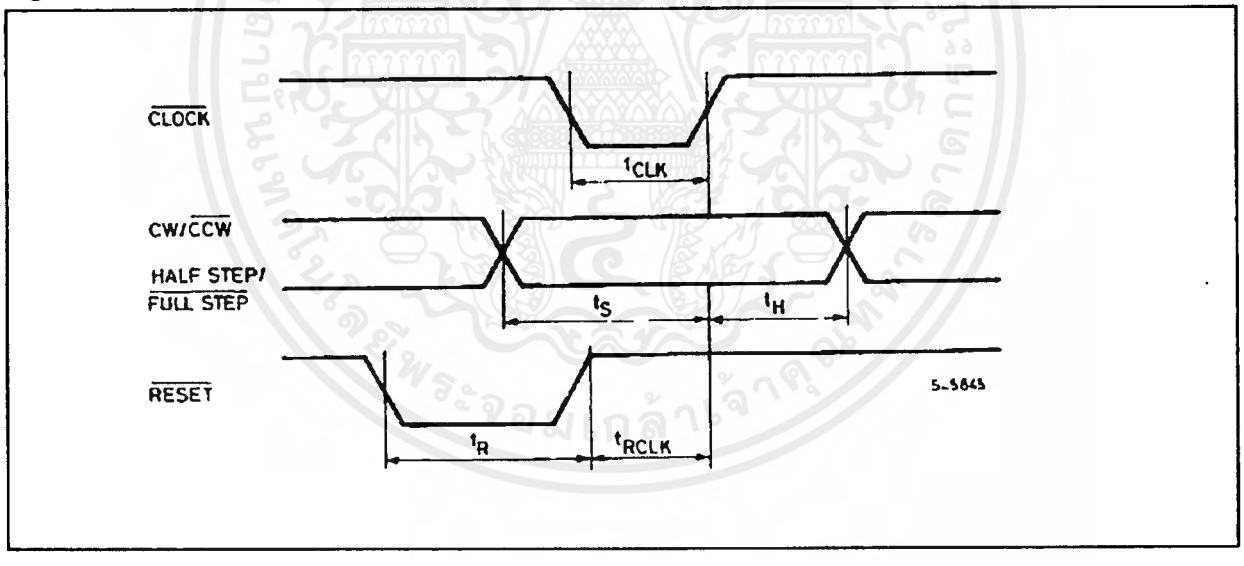
Symbol	Parameter	Test conditions		Min.	Typ	Max.	Unit
V_s	Supply voltage (pin 12)			4.75		7	V
I_s	Quiescent supply current (pin 12)	Outputs floating			50	80	mA
V_i	Input voltage (pin 11, 17, 18, 19, 20)		Low			0.6	V
			High	2		V_s	V
I_i	Input current (pin 11, 17, 18, 19, 20)		V_i = L		100	µA	
			V_i = H			10	µA
V_{en}	Enable input voltage (pin 10)		Low			1.3	V
			High	2		V_s	V
I_{en}	Enable input current (pin 10)		V_{en}= L			100	µA
			V_{en} = H			10	µA
V_o	Phase output voltage (pins 4, 6, 7, 9)	I_o = 10mA	V_{OL}			0.4	V
		I_o = 5mA	V_{OH}	3.9			V
V_{inh}	Inhibit output voltage (pins 5, 8)	I_o = 10mA	$V_{inh\ L}$			0.4	V
		I_o = 5mA	$V_{inh\ H}$	3.9			V
V_{SYNC}	Sync Output Voltage	I_o = 5mA	$V_{SYNC\ H}$	3.3			V
		I_o = 5mA	$V_{SYNC\ V}$			0.8	

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test conditions	Min.	Typ	Max.	Unit
I_{leak}	Leakage current (pin 3, 11*)	V_{CE}= 7 V			1	µA
V_{sat}	Saturation voltage (pins 3, 11*)	I = 5 mA			0.4	V
V_{off}	Comparators offset voltage (pins 13, 14, 15)	V_{ref} = 1 V			5	mV
I_o	Comparator bias current (pins 13, 14, 15)		-100		10	µA
V_{ref}	Input reference voltage (pin 15)		0		3	V
t_{CLK}	Clock time		0.5			µs
t_S	Set up time		1			µs
t_H	Hold time		4			µs
t_R	Reset time		1			µs
t_{RCLK}	Reset to clock delay		1			µs

* L297A only

Figure 1.



APPLICATION INFORMATION

TWO PHASE BIPOLAR STEPPER MOTOR CONTROL CIRCUIT

This circuit drives bipolar stepper motors with winding currents up to 2A. The diodes are fast 2A types.

Figure 2.

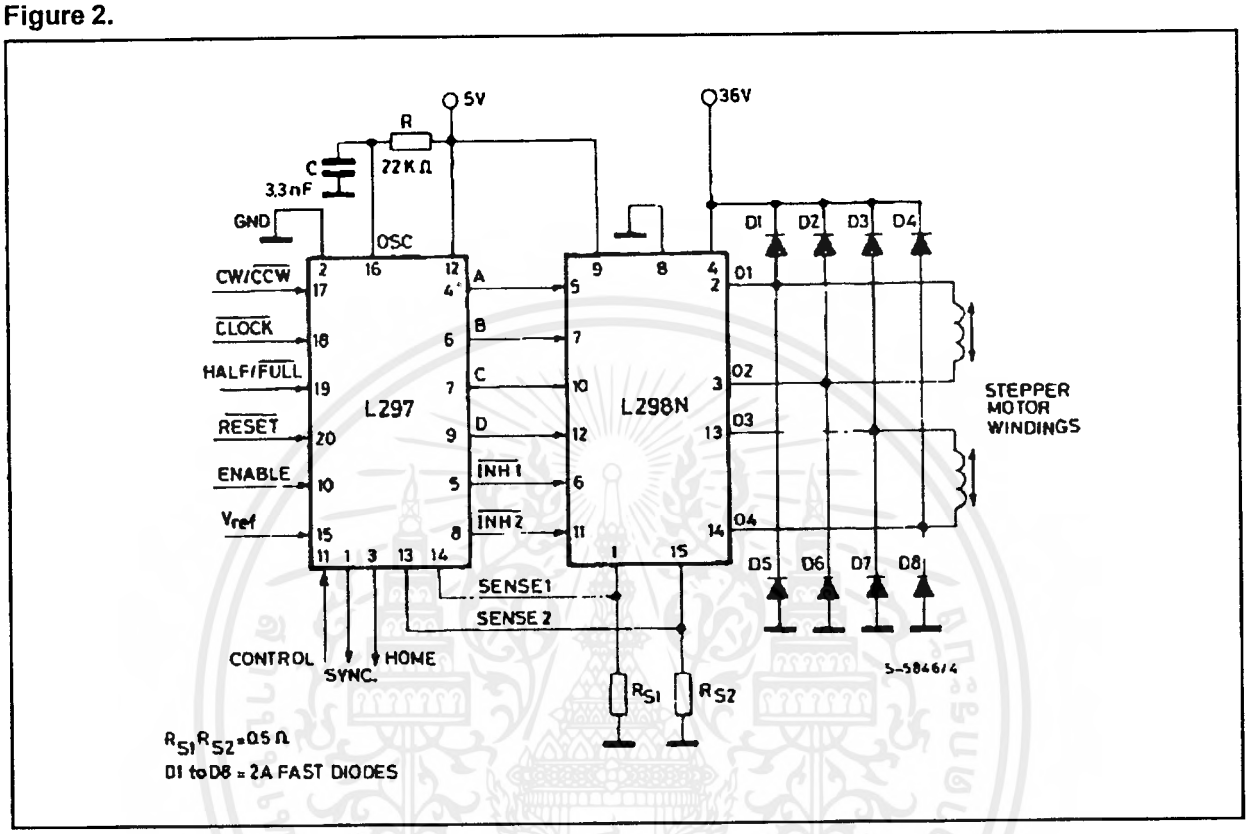


Figure 3 : Synchronising L297s

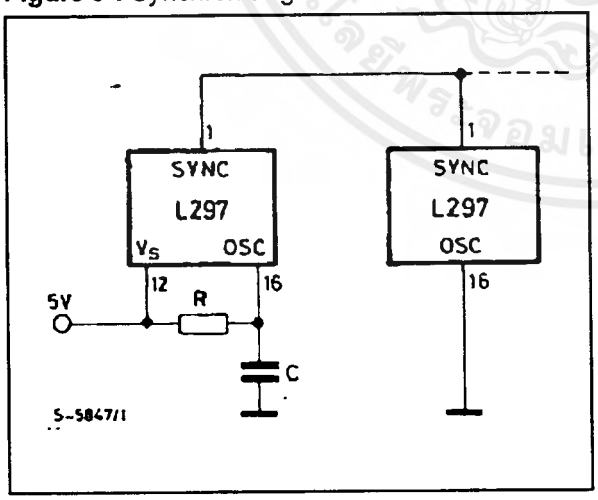
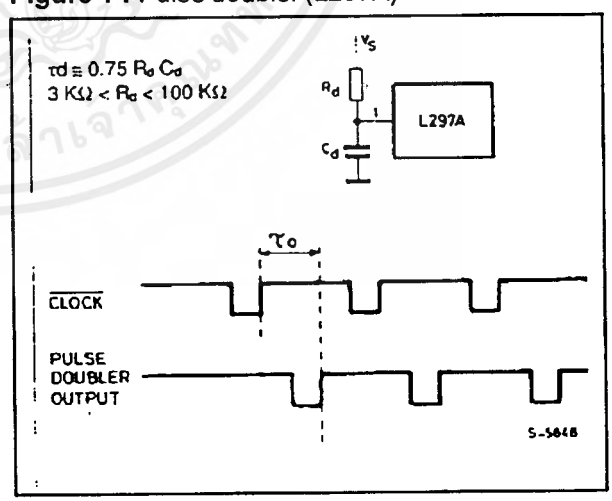
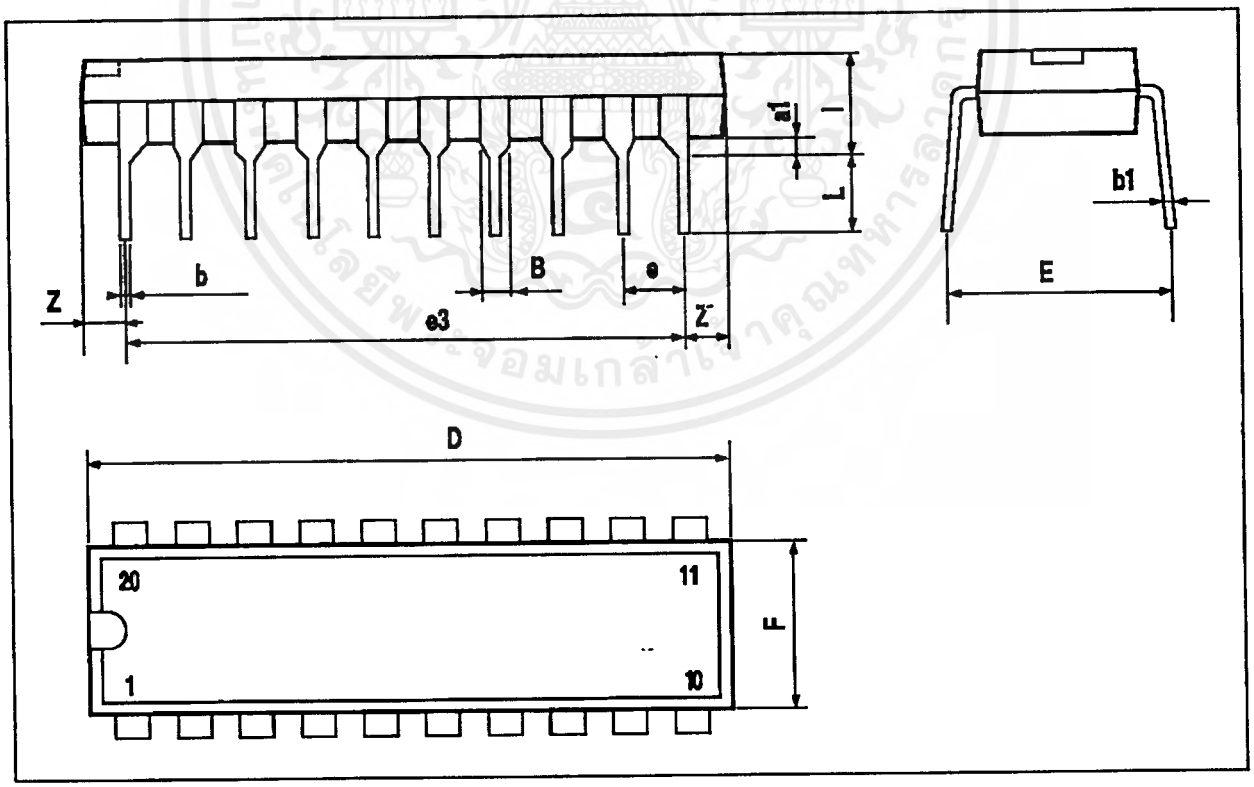


Figure 4 : Pulse doubler (L297A)



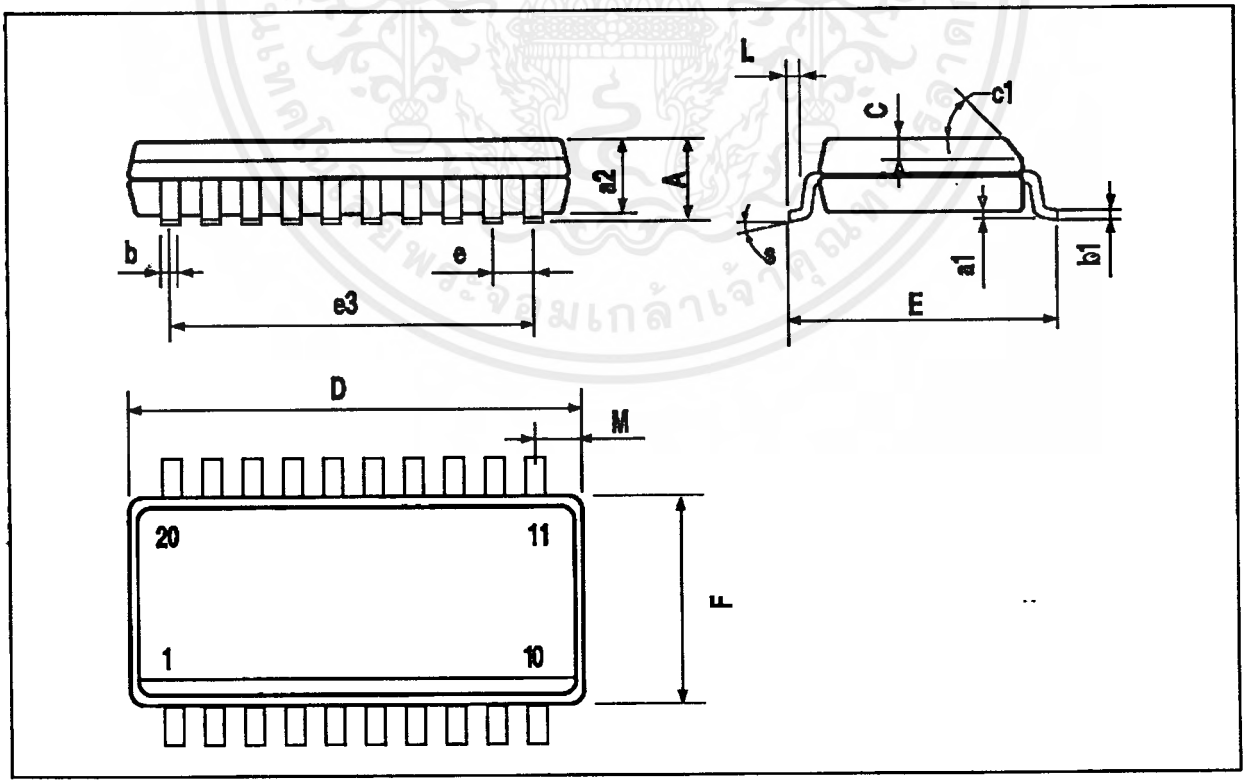
DIP20 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.254			0.010		
B	1.39		1.65	0.055		0.065
b		0.45			0.018	
b1		0.25			0.010	
D			25.4			1.000
E		8.5			0.335	
e		2.54			0.100	
e3		22.86			0.900	
F			7.1			0.280
I			3.93			0.155
L		3.3			0.130	
Z			1.34			0.053



SO20 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			2.65			0.104
a1	0.1		0.3	0.004		0.012
a2			2.45			0.096
b	0.35		0.49	0.014		0.019
b1	0.23		0.32	0.009		0.013
C		0.5			0.020	
c1	45 (typ.)					
D	12.6		13.0	0.496		0.512
E	10		10.65	0.394		0.419
e		1.27			0.050	
e3		11.43			0.450	
F	7.4		7.6	0.291		0.299
L	0.5		1.27	0.020		0.050
M			0.75			0.030
S	8 (max.)					

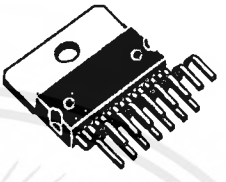


SGS-THOMSON MICROELECTRONICS

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

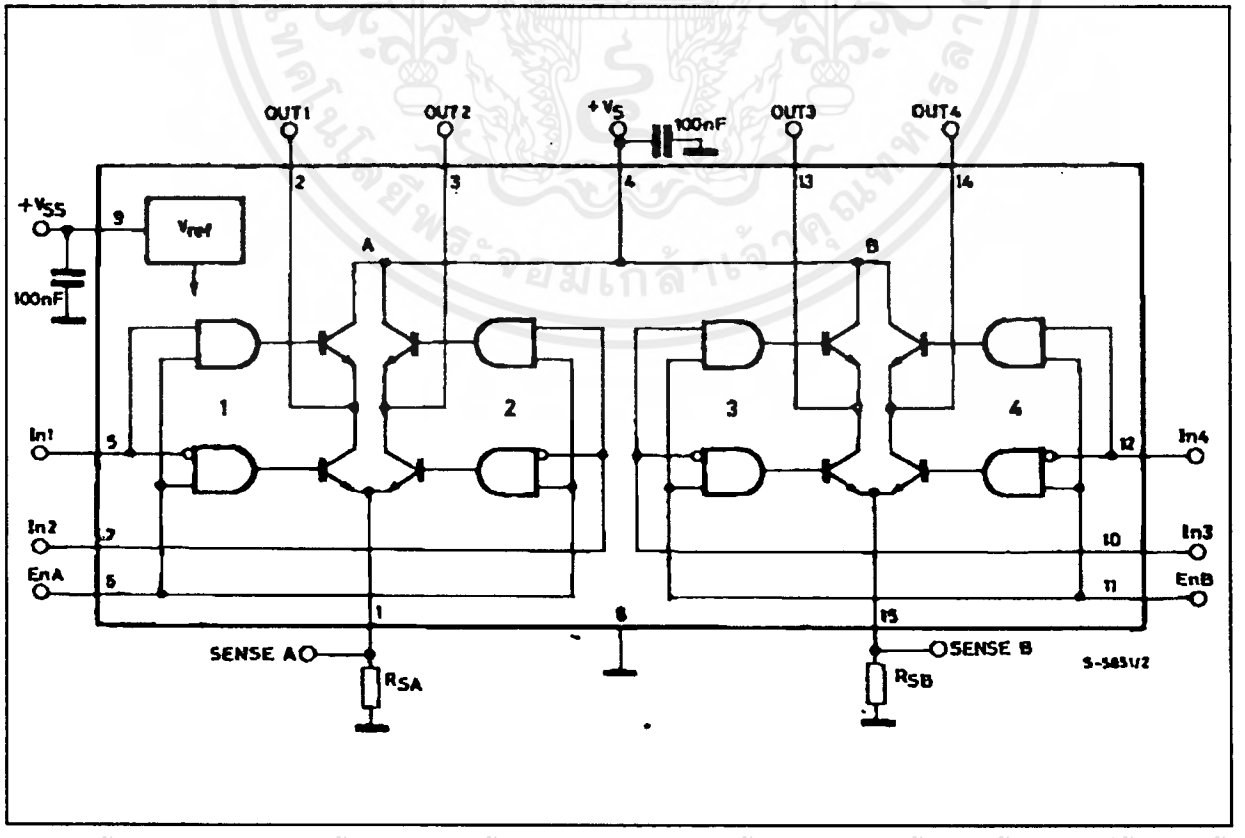
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

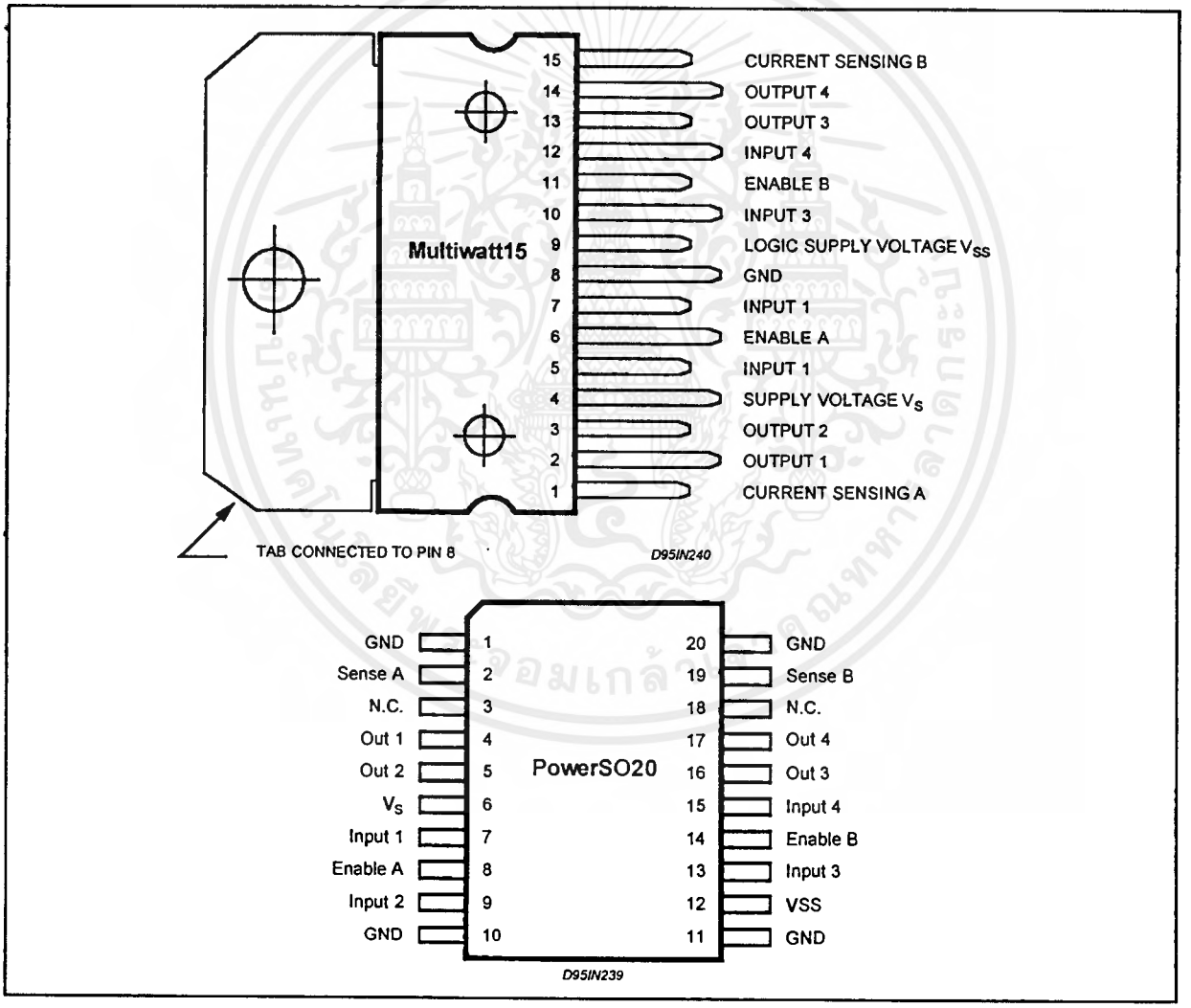
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH}+2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en}= H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS}(pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{ih}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	L = 1A I_L = 2A		1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	L = 1A (5) I_L = 2A (5)		1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)			3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs
fc (V_{en})	Commutation Frequency	I_L = 2A		1		KHz

1) 1)Sensing voltage can be –1 V for ≤ 50 µsec; in steady state V_{sens} min ≥ –0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.
5) PIN 1 and PIN 15 connected to GND.

Figure 1 : Typical Saturation Voltage vs. Output Current.

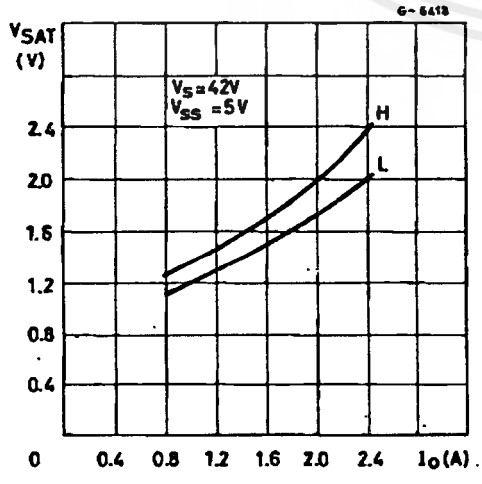
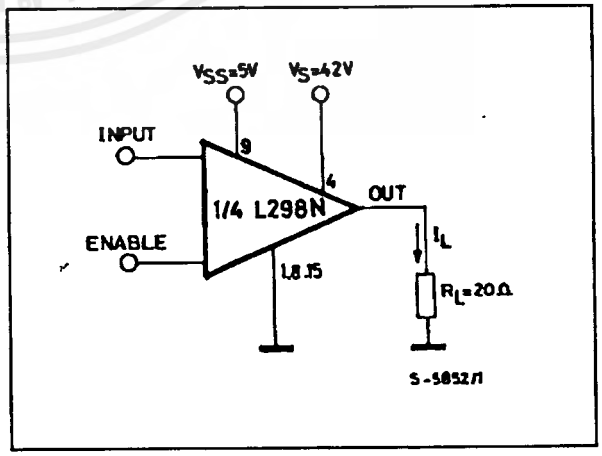


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

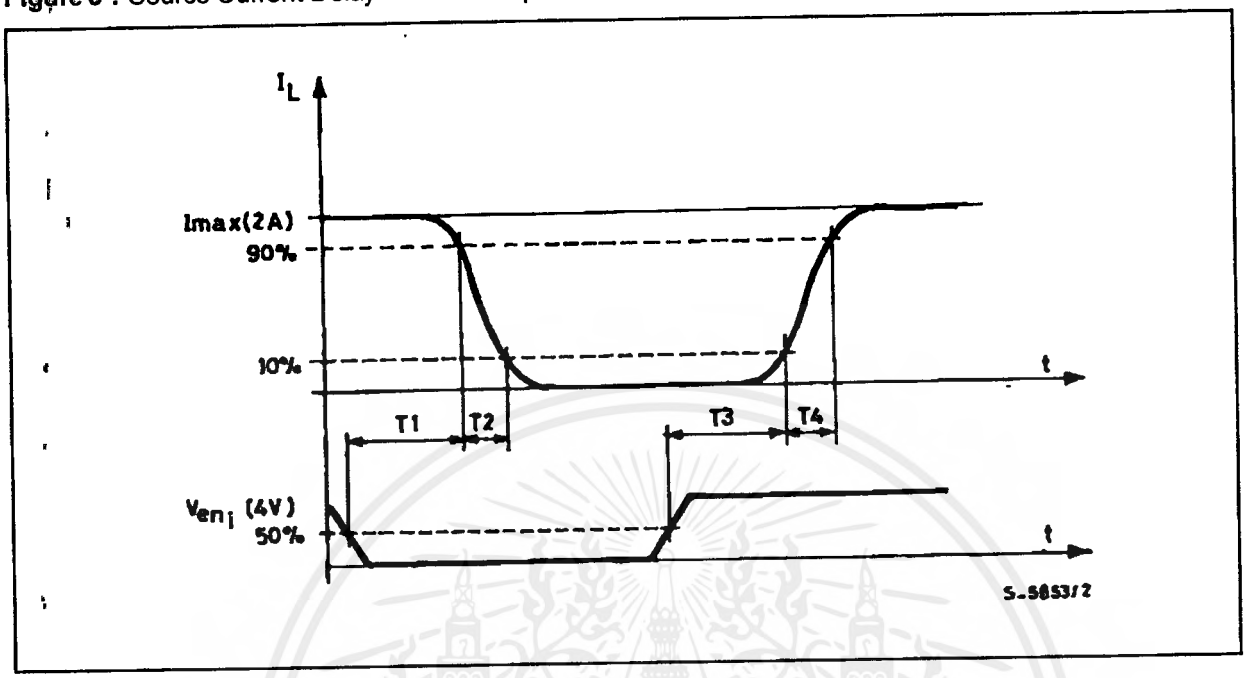
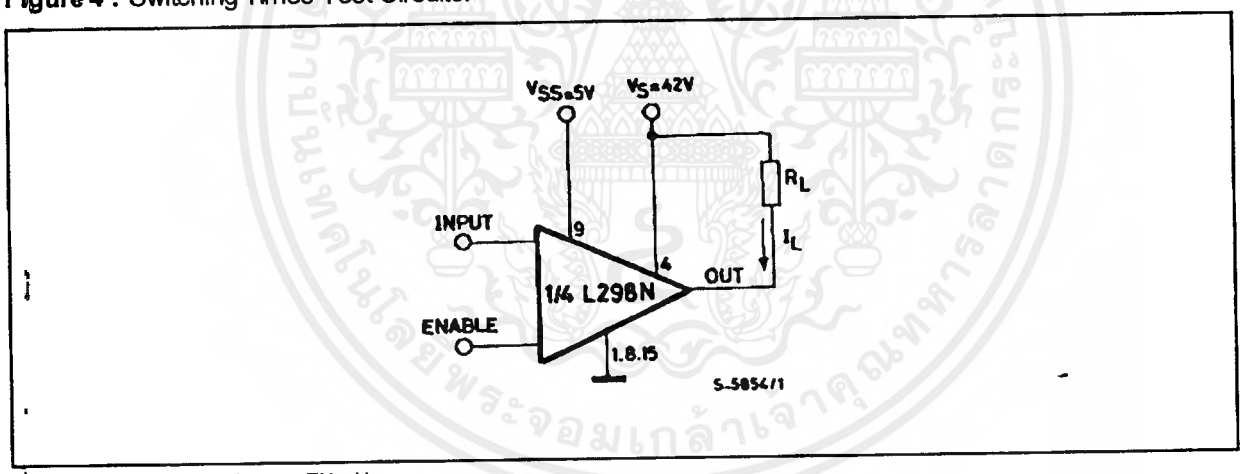


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

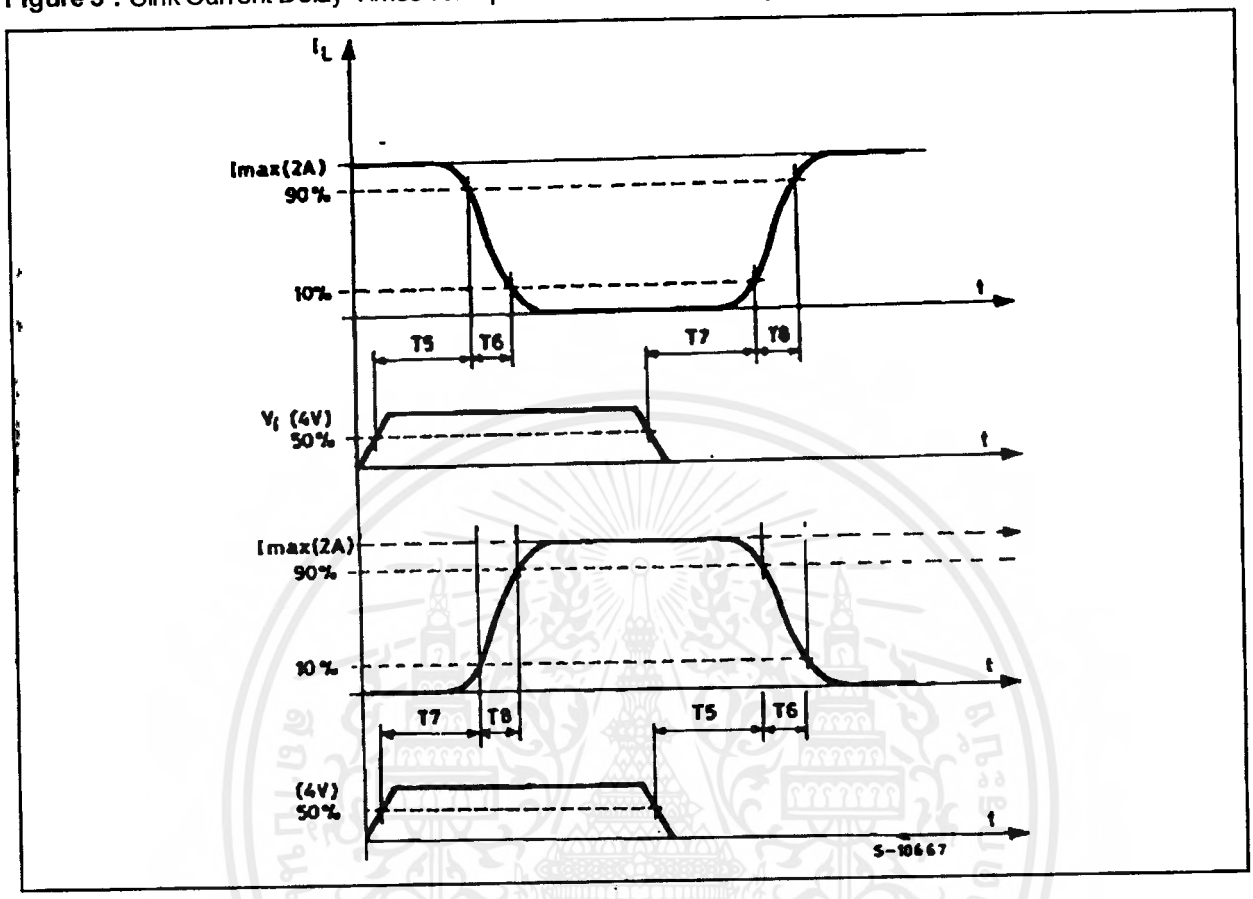
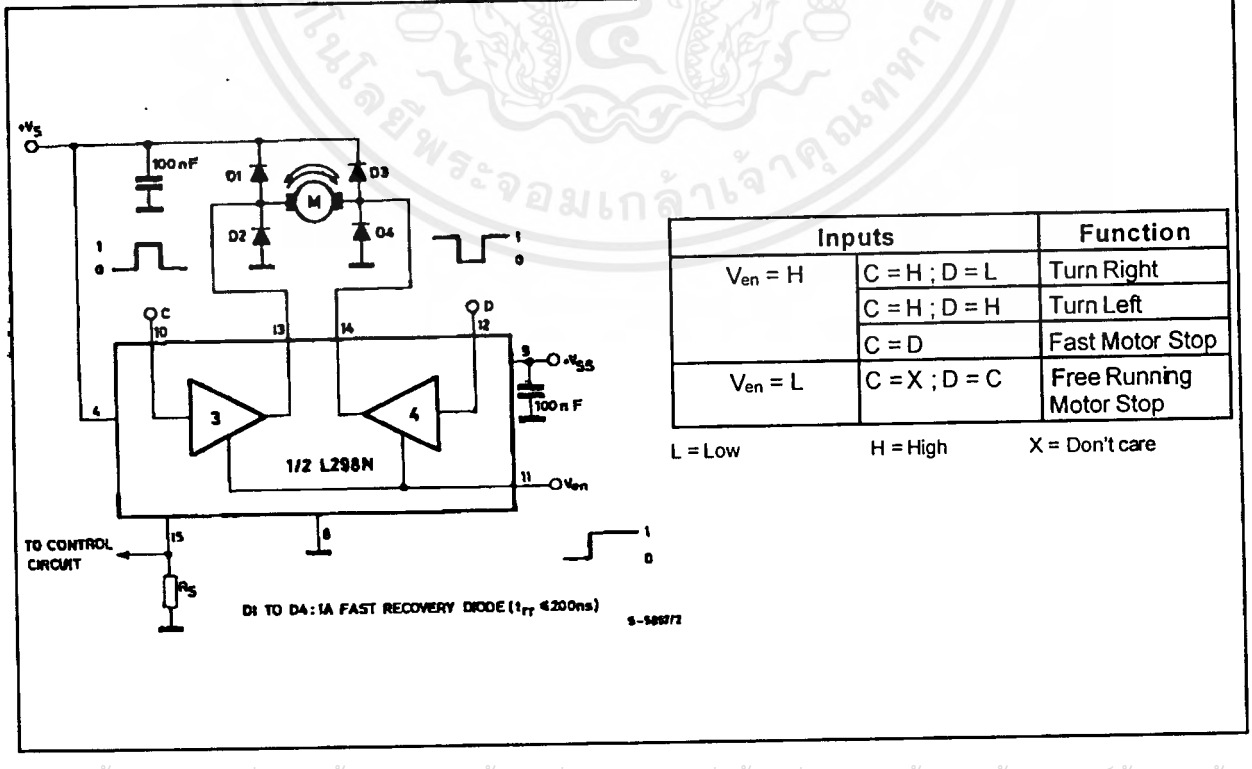


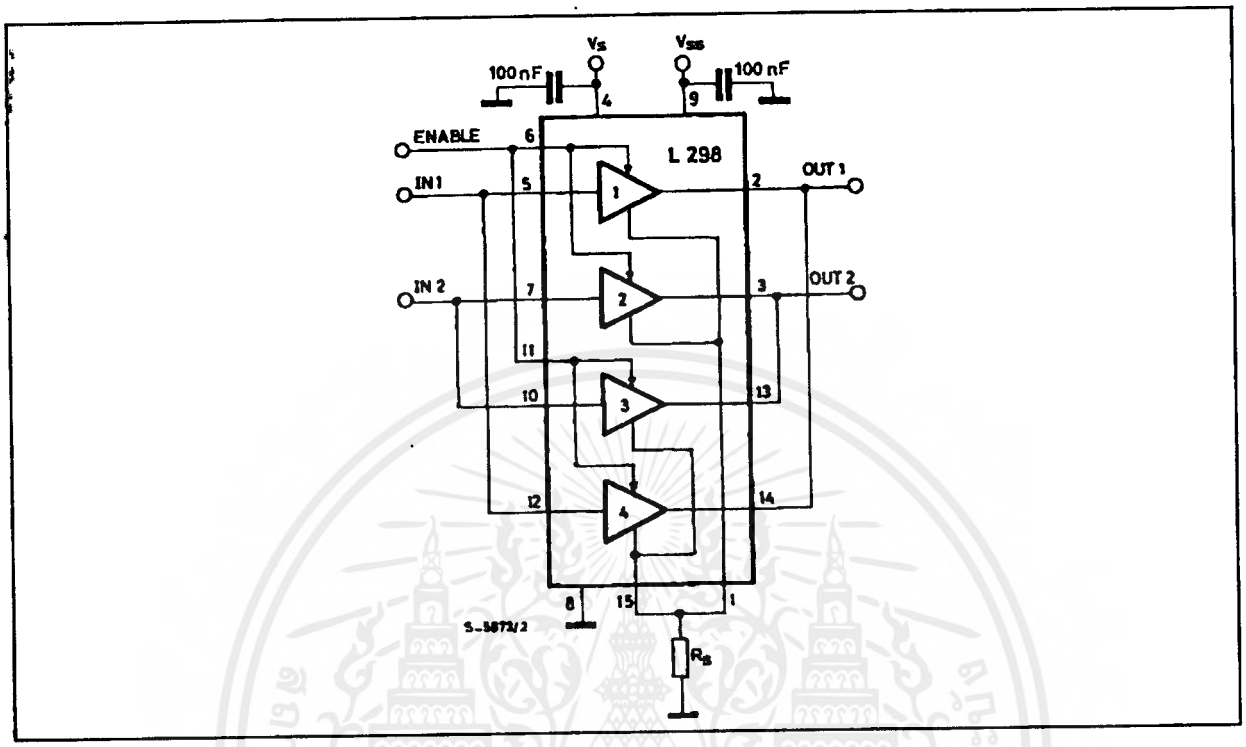
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Turn Right
	C = H ; D = H	Turn Left
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = C	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output: an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements (trr $\leq$ 200 nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

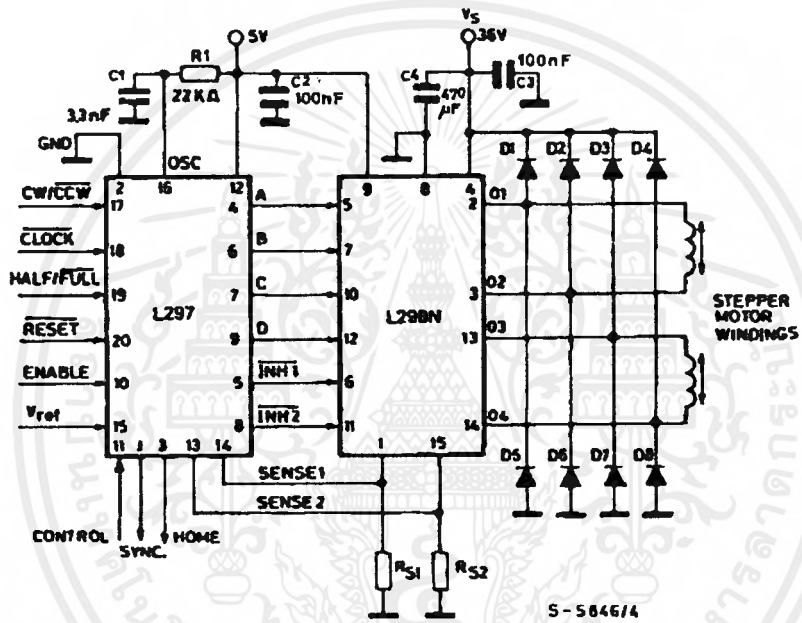
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A ; trr ≤ 200 ns

Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

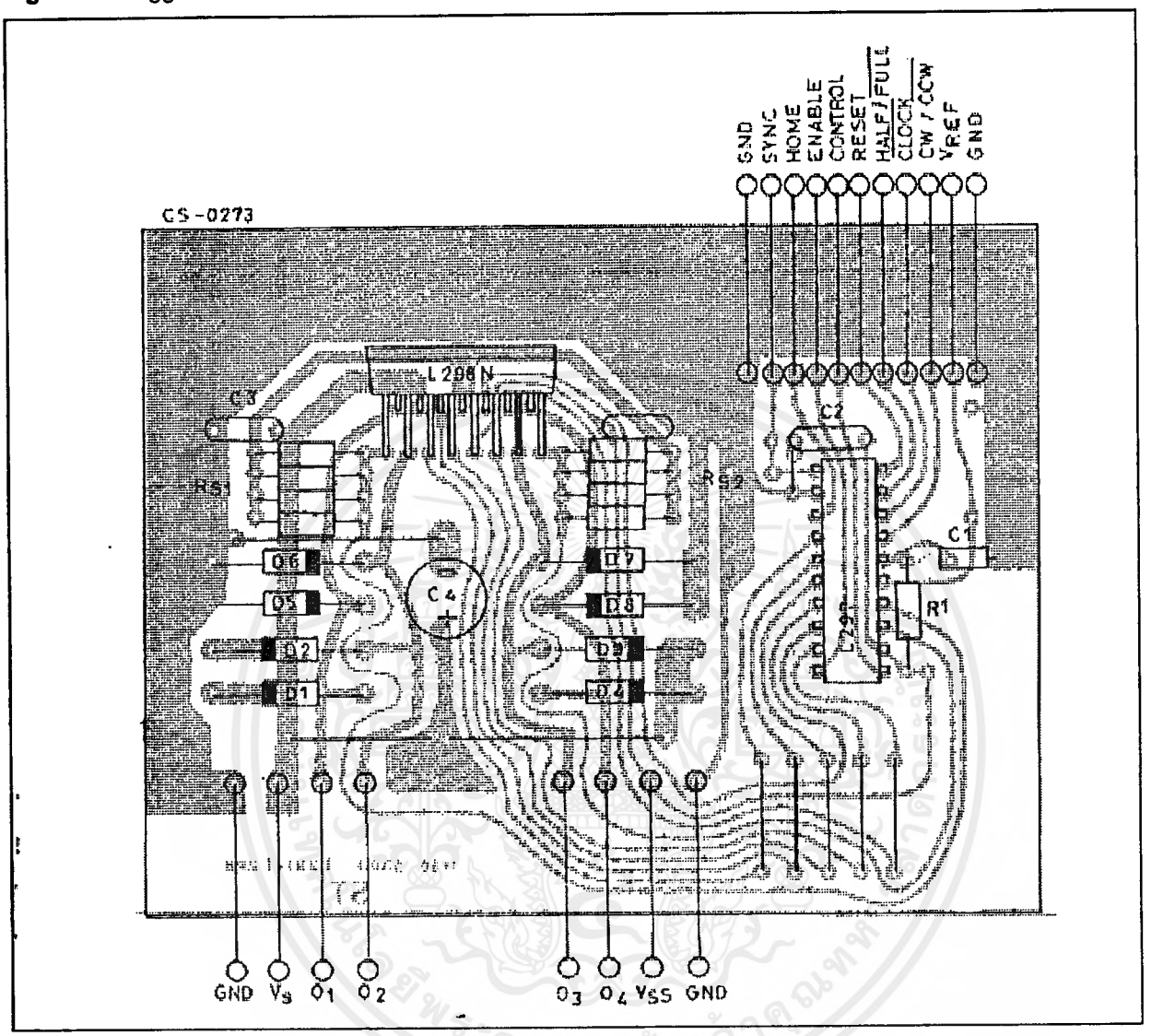
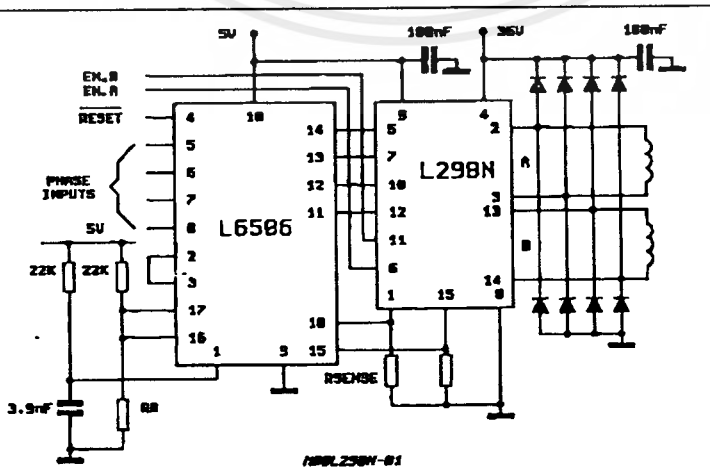


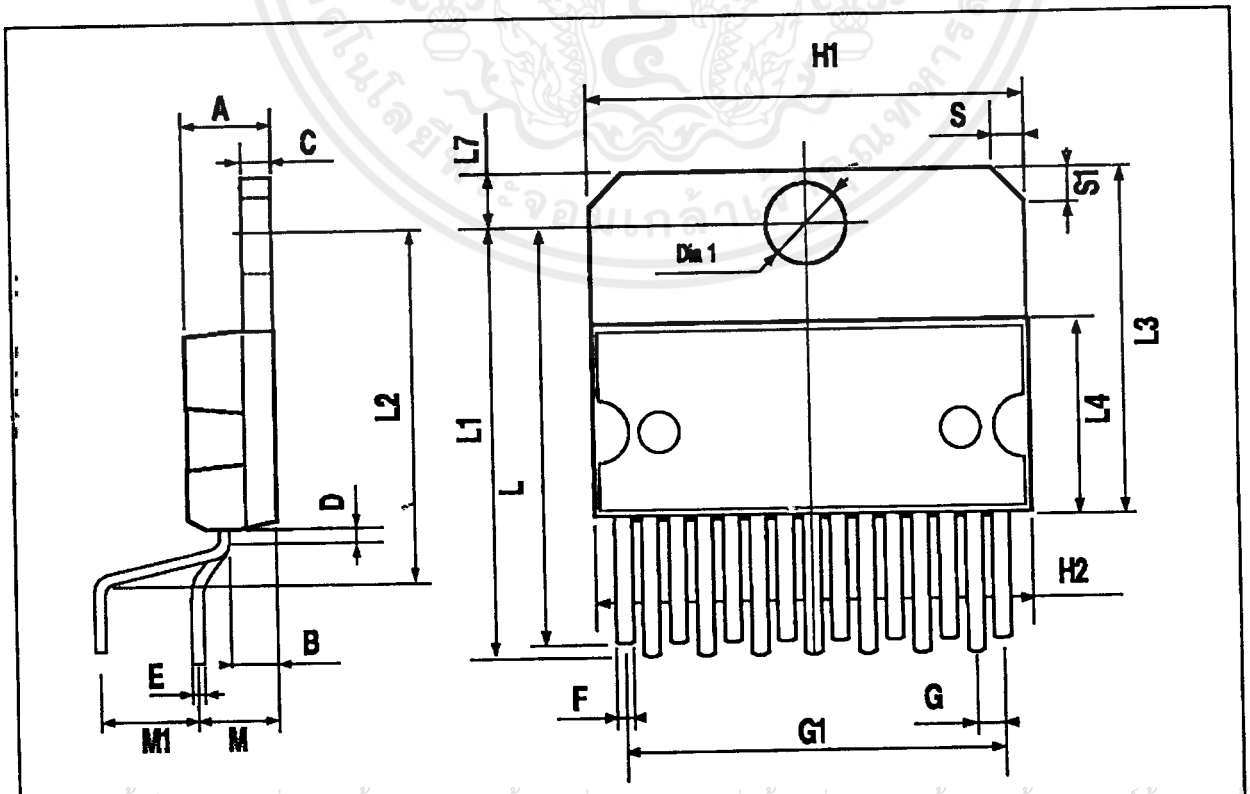
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



R_R and R_{sense} depend from the load current

MULTIWATT15 (VERTICAL) PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L	22.1		22.6	0.870		0.890
L1	22		22.5	0.866		0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.2	4.3	4.6	0.165	0.169	0.181
M1	4.5	5.08	5.3	0.177	0.200	0.209
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152



PowerSO20 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.60			0.1417
a1	0.10		0.30	0.0039		0.0118
a2			3.30			0.1299
a3	0		0.10	0		0.0039
b	0.40		0.53	0.0157		0.0209
c	0.23		0.32	0.009		0.0126
D (1)	15.80		16.00	0.6220		0.6299
E	13.90		14.50	0.5472		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.90		11.10	0.4291		0.437
E2			2.90			0.1141
G	0		0.10	0		0.0039
h			1.10			
L	0.80		1.10	0.0314		0.0433
N	10° (max.)					
S	8° (max.)					
T		10.0			0.3937	

(1) "D and E1" do not include mold flash or protrusions
- Mold flash or protrusions shall not exceed 0.15mm (0.006")