หุ่นยนต์วาดภาพ

Painting Robot

โดย

นาย ทศพร แก้วมณีสกุล

นาย ทัศนัย อัศวะศิริจินดา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

PAINTING ROBOT

BY

MR. TOSAPHON KAEWMANEESAKUL

MR. THASSANAI ASSAVASIRIJINDA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2002

หัวข้อปริญญานิพนธ์	หุ่นยนต์วาดภาพ Painting Robot
นักศึกษา	นายทศพร แก้วมณีสกุล รหัสประจำตัว 42010128 นายทัศนัย อัศวะศิริจินดา รหัสประจำตัว 42010130
อาจารย์ผู้ควบคุมวิทยานิพนธ์	อาจารย์ มยุรี เลิศเวชกุล อาจารย์ บุณย์ชนะ ภู่ระหงษ์
ระดับการศึกษา	ปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2545

บทคัดย่อ

โครงงานนี้เป็นการสร้างหุ่นยนต์วาดภาพ หุ่นยนต์วาดภาพนี้สามารถพ่นสีเป็นรูปภาพหรือตัวอักษรได้ โดยจะสั่งงานด้วยคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม ส่งข้อมูลไปที่หุ่นยนต์โดยจะมีไมโครคอนโทรลเลอร์เป็นตัวรับข้อมูล แล้วทำการประมวลผลข้อมูลที่ได้รับเป็นสัญญาณควบคุมส่งไปบังคับอุปกรณ์ต่างๆบนหุ่นยนต์ เพื่อบังคับให้หุ่นยนต์เคลื่อนที่และพ่นสี ในจุดที่ได้จากการประมวลผลข้อมูลที่รับมา โดยข้อมูลที่ส่งไปประมวลผลที่ไมโครคอนโทรลเลอร์นั้นจะประกอบไปด้วย ทิศทางของการเคลื่อนที่ สี และระยะทาง เมื่อไมโครคอนโทรลเลอร์ ได้รับข้อมูลเหล่านี้แล้วก็จะนำข้อมูลเหล่านี้ไปเปรียบเทียบกับคำสั่งเพื่อควบคุมอุปกรณ์ให้ทำงานตามที่ต้องการได้

Thesis Title	Painting Robot	
Student	Tosaporn Kaewmaneesakul	No. 42010128
	Thassanai Assavasirijinda	No. 42010130
Advisor	Mayuree Lertwatechakul	
	Boonchana Purahong	
Graduate Level	Bachelor Degree of Information Engineering	
Department	Information Engineering	
Academic Year	2002	

ABSTRACT

This project is to develop a painting robot. The painting robot can paint two-tone picture or alphabet. By using a computer controls through serial ports and transfers data to a microcontroller as a receiver. The receiver data is proceeded to a control signal, is sent to control robot's devices. After that the robot moves to IIpaint on an over that is specified from the received data. The data is sent to the microcontroller, consists of movement direction, painting command and distance. When a sub-task is completed, the robot sends an acknowledge back to the computer and asks for the next data that need proceeding.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจสำเร็จลุล่วงไปได้เลย ถ้าปราศจากความร่วมมือร่วมใจกันทำงาน และบุคคลรอบๆข้าง ที่คอยให้กำลังใจ คอยให้ความรู้ คอยช่วยเหลือในด้านต่างๆ

ขอขอบคุณ คุณพ่อ คุณแม่ ผู้มีพระคุณอย่างสูงที่เลี้ยงเรามาจนถึงบัดนี้

ครู อาจารย์ทุกท่าน ที่ช่วยประสิทธิ์ประสาทความรู้ คุณธรรม จริยธรรมให้แก่เรา

เพื่อน ๆ ที่อยู่ร่วมทุกข์ ร่วมสุข คอยช่วยเหลือ ให้กำลังใจแก่กัน

รุ่นพี่ รุ่นน้อง ที่คอยช่วยเหลือสนับสนุนในบางโอกาส ให้กำลังใจอย่างสม่ำเสมอ

ภาควิชาวิศวกรรมสารสนเทศ ที่เปิดโอกาสให้เราได้ร่ำเรียน

ขอขอบคุณจากใจจริง และที่ลืมไม่ได้ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ทั้งสองท่านที่กรุณาเป็นที่ปรึกษาให้ คือ อาจารย์ มยุรี เลิศเวชกุล และ อาจารย์ บุณย์ชนะ ภู่ระหงษ์ ที่ให้ความเอาใจใส่ดูแล คอยช่วยเหลือ ให้คำแนะนำปรึกษาอย่างดี

สุดท้ายนี้คณะผู้จัดทำจะนำความรู้ที่ได้ทั้งหมดจากการทำปริญญานิพนธ์นี้ถ่ายทอดเป็นตัวอักษรเพื่อเป็นวิทยาทานแก่ผู้อื่นต่อไป จึงขอขอบคุณมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในปัจจุบันมีการนำเทคโนโลยีมาช่วยในการทำงานเพื่อให้เกิดความสะดวก รวดเร็ว ในการทำงานมากขึ้น ในโครงงานนี้ได้นำเทคโนโลยีมาช่วยในการวาดภาพหรือพิมพ์ตัวอักษรบนพื้นราบในบริเวณกว้าง โดยการสั่งงานผ่านทางคอมพิวเตอร์ เพื่อให้หุ่นยนต์สามารถทำงานแทนมนุษย์ได้โดยอัตโนมัติ เมื่อสั่งงานไปแล้วหุ่นยนต์จะทำงานได้อย่างอัตโนมัติไม่จำเป็นที่มนุษย์ต้องคอยควบคุมตลอดเวลา

ในบางครั้งอุปกรณ์อำนวยความสะดวกต่างๆไม่สามารถทำงานให้เราได้ เช่น ในเครื่องพริ้นเตอร์ไม่สามารถทำการพิมพ์ตัวอักษรลงบนพื้นราบได้ ดังนั้นในโครงงานนี้จึงได้ทำการศึกษาและออกแบบ สร้างหุ่นยนต์เพื่อให้สามารถพ่นสีตัวอักษรและวาดภาพในพื้นราบได้

ในการควบคุมหุ่นยนต์ ในตัวหุ่นจะมีไมโครคอนโทรลเลอร์ควบคุมการทำงานต่างๆอยู่ โดยจะทำการรับคำสั่งมาจากคอมพิวเตอร์ ผ่านทางพอร์ตอนุกรมโดยจะส่งสัญญาณเป็นข้อมูลคำสั่งเพื่อประมวลผลเป็นสัญญาณควบคุม ที่ส่งไปควบคุมส่วนต่างๆของหุ่นยนต์

ในโครงงานนี้จะทำการศึกษารวมทั้งการออกแบบและสร้างเครื่องต้นแบบขึ้นมา เพื่อใช้ศึกษาการทำงานของหุ่นยนต์ รวมทั้งการติดต่อสื่อสารระหว่างคอมพิวเตอร์และหุ่นยนต์ในการควบคุมหุ่นยนต์

1.2 จุดประสงค์

1.2.1 ศึกษาการสื่อสารข้อมูลระหว่างคอมพิวเตอร์และหุ่นยนต์

1.2.2 ศึกษาและพัฒนาหุ่นยนต์ให้ทำการพ่นสีได้ตามที่ต้องการ

1.2.3 ออกแบบและสร้างหุ่นยนต์ให้ทำงานได้ตามต้องการ

1.2.4 สร้างหุ่นยนต์พ่นสีวาดภาพหรือพ่นสีเป็นตัวอักษรลงบนพื้นราบโดยควบคุมจากคอมพิวเตอร์

1.2.5 นำผลจากการศึกษาไปประยุกต์ใช้งานได้จริง

1.3 ขั้นตอนการดำเนินโครงงาน

1.3.1 ศึกษาทฤษฎีต่างๆที่เกี่ยวข้อง

- ศึกษารายละเอียด ส่วนประกอบ การทำงานของมอเตอร์ที่ใช้
- ศึกษาสถาปัตยกรรม การทำงาน การสั่งงาน ของไมโครคอนโทรลเลอร์
- ศึกษาโปรแกรมที่ใช้ในไมโครคอนโทรลเลอร์เพื่อสั่งงานฮาร์ดแวร์
- ศึกษาโปรแกรมที่ใช้ติดต่อระหว่างผู้ใช้และคอมพิวเตอร์
- ศึกษาการทำงานของแอร์บรัช

1.3.2 ออกแบบส่วนต่างๆในโครงงาน

- ออกแบบระบบขั้นตอนการทำงานทั้งหมดของระบบ
- ออกแบบรูปแบบในการติดต่อกับผู้ใช้
- ออกแบบรูปแบบการสื่อสาร
- ออกแบบรูปแบบการรับและส่งข้อมูล
- ออกแบบหุ่นยนต์
- ออกแบบชุดควบคุมหุ่นยนต์(ไมโครคอนโทรลเลอร์)
- ออกแบบโปรแกรมในไมโครคอนโทรลเลอร์

1.3.3 ทดสอบส่วนต่างๆของโครงงาน

- นำมาประกอบเป็นหุ่นยนต์
- ทดสอบการทำงานของส่วนต่างๆ
- แก้ไขและปรับปรุงจุดบกพร่อง

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 ได้รับความรู้ในด้านต่างๆเกี่ยวกับการควบคุมหุ่นยนต์

1.4.2 สามารถนำหุ่นยนต์ที่สร้างขึ้นไปพัฒนาใช้ในงานจริงได้

1.4.3 สามารถนำความรู้ที่ได้ไปประยุกต์ใช้กับงานอื่นๆได้

บทที่ 2
ทฤษฎี

2.1 โปรแกรมวิชวลเบสิก

วิชวลเบสิกเป็นภาษาคอมพิวเตอร์ (Programming Language) ที่พัฒนาโดยบริษัทไมโครซอฟท์ ซึ่งเป็นบริษัทที่สร้างระบบปฏิบัติการวินโดว์ 95/98 และวินโดว์ NT ที่ใช้กันอยู่ในปัจจุบันโดยตัวภาษาเองมีรากฐานมาจากภาษาเบสิก (Basic ซึ่งย่อมาจาก Beginner's All-Purpose Symbolic Instruction) ถ้าแปลให้ได้ความหมายคือ "ชุดคำสั่งหรือภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้น" ภาษาเบสิก มีจุดเด่นคือผู้ที่ไม่มีพื้นฐานเรื่องการเขียนโปรแกรมเลย ก็สามารถเรียนรู้และนำไปใช้งานได้อย่างง่ายดายและรวดเร็วเมื่อเทียบกับการเรียนภาษาคอมพิวเตอร์อื่น ๆ

สำหรับวิชวลเบสิก ในปัจจุบันคือเวอร์ชัน 6.0 ซึ่งออกมาในปี 1998 ได้เพิ่มความสามารถในการเขียนโปรแกรมติดต่อกับเครือข่ายอินเตอร์เน็ต การเชื่อมต่อกับระบบฐานข้อมูล รวมทั้งปรับปรุงเครื่องมือและการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) ให้สมบูรณ์ยิ่งขึ้น พร้อมทั้งเพิ่มเครื่องมือต่าง ๆ อีกมากมายที่ทำให้ใช้งานง่ายและสะดวกขึ้นกว่าเดิม

Edition ของ วิชวลเบสิก และข้อแตกต่าง

บริษัทไมโครซอฟท์ได้แบ่งผลิตภัณฑ์ โปรแกรมวิชวลเบสิก ออกเป็นหลายรุ่นด้วนกัน ตามลักษณะการใช้งานและราคาขาย ดังรายละเอียดต่อไปนี้

- Learning Edition เป็นรุ่นที่เหมาะสำหรับศึกษาการใช้งานของภาษาวิชวลเบสิก จะมีส่วนประกอบต่าง ๆ น้อยที่สุดแต่ก็มีส่วนประกอบพื้นฐานเพียงพอให้โปรแกรมเมอร์สามารถสร้างโปรแกรมได้มากมายบนระบบปฏิบัติการวินโดว์ 95/98 หรือ วินโดว์ NT

- Professional Edition ในรุ่นนี้จะมีส่วนประกอบต่าง ๆ อย่างครบครัน เหมาะสำหรับโปรแกรมเมอร์มืออาชีพที่ต้องการสร้างโปรแกรม หรือพัฒนาระบบงานสำหรับลูกค้าที่ใช้ ระบบปฏิบัติการวินโดว์ โดยมีส่วนประกอบเพิ่มเติมมาจาก Learning Edition ได้แก่ ActiveX control, เพิ่มเติมการติดต่อกับเครือข่ายอินเตอร์เน็ต (Internet information Server Application Designer), เครื่องมือช่วยออกแบบฐานข้อมูล (Visual Database Tools and Data Environment), Active Data Object และเครื่องมือช่วยสร้างโฮมเพจ แบบไดนามิค HTML (Hypertext Markup Language)

- Enterprise Edition เป็นรุ่นที่มีส่วนประกอบและเครื่องมือครบถ้วนที่สุด คือจะมีทุกอย่างที่ Professional Edition มี พร้อมรวมเครื่องมือเกี่ยวกับการพัฒนา โปรแกรมสำหรับเครือข่าย

การบริหารงานพัฒนาโปรแกรมเป็นทีมรวมทั้งเครื่องมือที่ใช้เชื่อมต่อกับ ผลิตภัณฑ์ชิ้นอื่น ๆ ของไมโครซอฟท์เข้าไปด้วย เช่น MS Back Office, MS SQL Server, Microsoft Transaction, Internet Information Server, SNA Server และอื่น ๆ อีกมากมาย

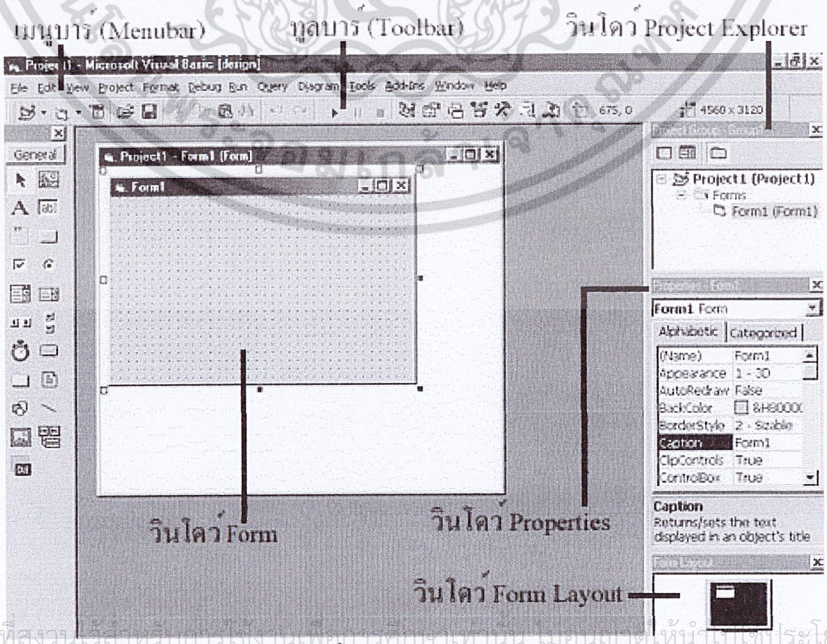
ส่วนข้อแตกต่างของแต่ละเวอร์ชัน ถ้ากล่าวคร่าว ๆ สำหรับในเวอร์ชัน 6 นั้น จะเน้นเพิ่มความสามารถของเครื่องมือที่เชื่อมต่อกับอินเตอร์เน็ต ระบบเครือข่าย และระบบฐานข้อมูล รวมทั้งทำให้การพัฒนาโปรแกรมและการใช้งานง่ายขึ้น สะดวกขึ้น รายละเอียดสามารถดูได้จากแผ่น MSDN ในหัวข้อ What's new in Visual Basic 6

วัตถุ (Object), คุณสมบัติ (Property) และ วิชวลเบสิก

ออบเจ็ค (Object) หรือวัตถุ ในวิชวลเบสิก จะหมายถึงส่วนประกอบย่อยต่าง ๆ ของโปรแกรมในวินโดว์ เช่น Form, Command Button, Option Button, Text Box และปุ่มควบคุมต่าง ๆ ส่วนพร็อพเพอร์ตี้ (Property) จะหมายถึงคุณสมบัติหรือลักษณะเฉพาะของออบเจ็คนั้น ๆ

2.1.1 IDE และส่วนประกอบต่าง ๆ ของ IDE

คำว่า IDE หรือ Integrated Development Environment หมายถึงสภาพแวดล้อมการทำงานในการพัฒนาโปรแกรมโดยใช้วิชวลเบสิก หรือจะแปลอีกอย่างคือ อุปกรณ์เครื่องมือต่าง ๆ แบบเพรียบพร้อมที่ไมโครซอฟท์ เตรียมมาให้ใช้ในการพัฒนาโปรแกรมด้วย วิชวลเบสิกเมื่อเปิดโปรแกรมวิชวลเบสิก โปรแกรมจะปรากฏหน้าจอ IDE ซึ่งมีส่วนประกอบหลักดังรูปที่ 2.1



รูปที่ 2.1 หน้าจอ IDE

ส่วนประกอบหลักของหน้าจอ Visual Basic IDE มีดังนี้

- ทูลบาร์(Toolbar)
- Toolbox
- วินโดว์ Form
- วินโดว์ Project Explorer
- วินโดว์ Properties
- วินโดว์ From Layout

2.1.2 ทูลบาร์ (Toolbar)

เมื่อพิจารณาจากภาพหน้าจอ IDE จะเห็นปุ่มต่าง ๆ ที่วางเรียงเป็นแผงควบคุม ช่วยให้สามารถเรียกใช้งานคำสั่งได้อย่างสะดวกรวดเร็ว โดยเพียงแค่คลิกเมาส์ที่ปุ่มเท่านั้น รายละเอียดของปุ่มต่าง ๆ มีดังรูปที่ 2.2

รูปที่ 2.2 ทูลบาร์

A. เปิดโปรเจ็คใหม่ขึ้นมา สามารถพัฒนาได้หลายโปรเจ็คไปพร้อม ๆ กัน

B. เพิ่มฟอร์มโมดูล หรือออบเจ็คประเภทต่าง ๆ เข้าไปในโปรเจ็คหรือโปรแกรมที่เรากำลังพัฒนาอยู่

C. เปิดวินโดว์ Menu Editor ซึ่งเป็นเครื่องมือช่วยในการสร้างเมนูของโปรแกรม

D. เปิดไฟล์โปรเจ็ค (Open)

E. บันทึกไฟล์โปรเจ็ค (Save)

F. ตัด (Cut)

G. ก็อปปี้ (Copy)

H. วาง (Paste)

I. ค้นหา (Find)

J. ยกเลิกการกระทำหรือการพิมพ์ (Undo)

K. เรียกคืนกลับสิ่งที่ Undo ไป (Redo)

L. สั่งให้โปรแกรมทำงาน (Run)

M. ให้โปรแกรมหยุดการทำงานชั่วคราว (Pause)

N. ให้โปรแกรมหยุดทำงาน (Stop)

O. เปิดวินโดว์ Project Explorer ที่แสดงฟอร์ม โมดูล และส่วนประกอบต่าง ๆ ของโปรเจ็ค

P. เปิดวินโดว์ Properties เพื่อดูและกำหนดคุณสมบัติต่าง ๆ ของออบเจ็ค

Q. เปิดวินโดว์ Form Layout เพื่อจัดตำแหน่งวินโดว์ของโปรแกรมบนจอภาพเมื่อโปรแกรมทำงาน

R. Object Browser เป็นเครื่องมือช่วยค้นหาข้อมูลรายละเอียดของออบเจ็คต่าง ๆ

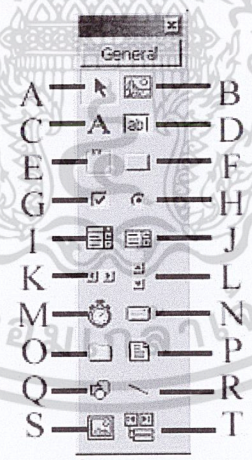
S. Toolbox เป็นที่รวมของออบเจ็คต่าง ๆ ที่จะนำมาประกอบในโปรแกรมหรือแอพพลิเคชั่น

T. เปิดวินโดว์ Data View เพื่อดูการติดต่อกับฐานข้อมูลต่าง ๆ รวมทั้งดูโครงสร้างของฐานข้อมูลที่เรากำลังติดต่ออยู่ด้วย

U. Visual Component Manager ใช้ในการช่วยค้นหา เรียบเรียง ดูแล และจัดการส่วนประกอบต่าง ๆ ที่จะนำมาใช้ในการพัฒนาโปรเจ็ค

2.1.3 Toolbox

เป็นที่รวมออบเจ็คต่าง ๆ ที่จะนำมาประกอบกันเป็นโปรแกรมหรือแอพพลิเคชัน เมื่อใช้ออบเจ็คเหล่านี้ประกอบกันจะได้เป็นหน้าตาของโปรแกรม จึงอาจเรียกให้ชัดเจนได้ว่า Control Object ซึ่งมีออบเจ็คหลักดังรูปที่ 2.3 นอกจากนี้ยังสามารถเพิ่มออบเจ็คต่าง ๆ เข้าไปใน Toolbox ได้อีกมากมาย สำหรับรายละเอียดคร่าว ๆ ของออบเจ็คหลักจะมีดังนี้



รูปที่ 2.3 Toolbox

A. Pointer ใช้ในการจัดขนาด เคลื่อนย้าย และวางตำแหน่งออบเจ็คต่าง ๆ ในฟอร์ม

B. Picture ใช้ควบคุมและแสดงข้อมูลภาพต่าง ๆ บนฟอร์ม

C. Label ใช้แสดงข้อความต่าง ๆ บนฟอร์ม เหมือนกับเป็นป้ายลาเบล หรือข้อความกำกับ

D. Text Box เป็นออบเจ็คสำหรับรับข้อความที่ผู้ใช้ป้อนเข้ามา

E. Frame ใช้จัดกลุ่มและรวบรวมออบเจ็คต่าง ๆ เข้าไว้ด้วยกัน เพื่อให้สะดวกในการควบคุมและเคลื่อนย้ายตำแหน่ง หรือจัดหน้าจอให้เป็นระเบียบเรียบร้อยและสะดวกแก่การใช้งาน

F. Command Button หรือปุ่มคำสั่ง เป็นออบเจ็คที่เป็นปุ่มกด เพื่อให้ผู้ใช้สั่งทำงาน

G. Check Box เป็นปุ่มที่ใช้เลือกว่าต้องการหรือไม่

H. Option Button บางครั้งเรียกว่า Radio Button ใช้สำหรับเลือกค่าใดค่าหนึ่งจากหลาย ๆ ค่า

I. Combo Box ผู้ใช้สามารถเลือกตัวเลือกได้จากการกดปุ่ม Drop down เพื่อแสดงทางเลือกต่าง ๆ ขึ้นมาให้ มีความสามารถเหมือนกับ List Box และ Text Box ผสมกัน

J. List Box ใช้แสดงตัวเลือกต่าง ๆ ในลักษณะของบรรทัดรายการ โดยผู้ใช้สามารถเลือกรายการใดรายการหนึ่ง หรือหลาย ๆ รายการจากลิสต์รายการที่มีอยู่ก็ได้

K. Horizontal Scroll Bar เป็นแถบเลื่อนทางแนวนอน ใช้เลื่อนปรับค่าโดยค่าจะเปลี่ยนไปตามตำแหน่งที่อยู่ของแถบเลื่อน (ตำแหน่งซ้ายสุดค่าจะน้อยที่สุด ตำแหน่งขวาสุดค่าจะมากที่สุด)

L. Vertical Scroll Bar เป็นแถบเลื่อนทางแนวตั้ง ใช้เลื่อนปรับค่าโดยค่าจะเปลี่ยนไปตามตำแหน่ง (ตำแหน่งบนสุดค่าจะน้อยที่สุด ตำแหน่งล่างสุดค่าจะมากที่สุด)

M. Timer ใช้ในการควบคุมเวลา และการทำงานของโปรแกรมเมื่อมีเรื่องเวลาเข้ามาเกี่ยวข้อง

N. – P. Drive List Box, Directory List Box, File List Box ใช้ในการควบคุมการติดต่อกับระบบแฟ้มข้อมูลของเครื่องคอมพิวเตอร์

Q. Shape ใช้สร้างภาพรูปทรงต่าง ๆ ลงในฟอร์ม

R. Line ใช้วาดเส้นต่าง ๆ ลงในฟอร์ม

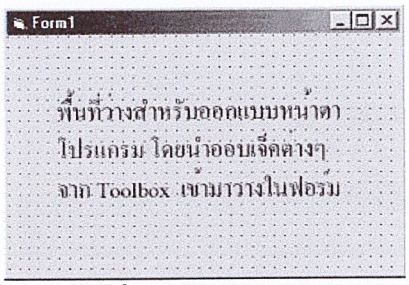
S. Image เป็นคอนโทรลที่ใช้ควบคุมข้อมูลภาพเหมือนกับ Picture เพียงแต่มีความสามารถน้อยกว่า แต่ใช้หน่วยความจำน้อยลงตามไปด้วย

T. Data Control ใช้ในการเชื่อมต่อกับฐานข้อมูล

U. OLE (Object Linked and Embedded) เป็นคอนโทรลที่นำเอาโปรแกรมสำเร็จรูปต่าง ๆ ที่มีความสามารถ OLE เข้ามาใช้เป็นออบเจ็คในโปรเจ็ค

2.1.4 วินโดว์ Form

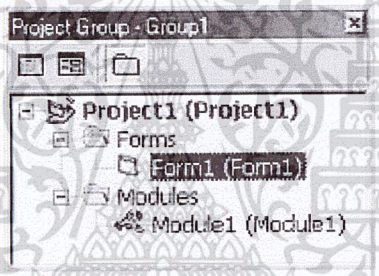
เป็นวินโดว์เปล่า ๆ หรือตัวฟอร์มเปล่าสำหรับสร้างองค์ประกอบของแอพพลิเคชันโดยการนำออบเจ็คต่าง ๆ มาใส่ในฟอร์ม หรือพูดอีกนัยคือเป็นหน้าจอของโปรแกรมที่ผู้ใช้จะเห็นเมื่อเรียกใช้งานโปรแกรมนั่นเอง เมื่อเริ่มเข้าสู่วิชวลเบสิก จะปรากฏฟอร์มเปล่าขึ้นมาให้เสมอ การเรียกดูฟอร์มสามารถใช้คีย์ Shift + F7 หรือเรียกจากเมนู View > Object



รูปที่ 2.4 วินโดว์ Form

2.1.5 วินโดว์ Project Explorer

โปรแกรมต่าง ๆ ที่พัฒนาเขียนโปรแกรมขึ้นมานั้นจะเรียกว่า โปรแกรมประยุกต์ หรือ แอพพลิเคชัน (Application) ซึ่งใน วิชวลเบสิก จะเรียกโปรแกรมที่กำลังสร้างว่า โครงการ หรือ โปรเจ็ค



รูปที่ 2.5 วินโดว์ Project Explorer

Project Explorer จะใช้ควบคุมส่วนประกอบและแฟ้มข้อมูลต่าง ๆ ที่อยู่ในโปรเจ็ค เพื่อความสะดวกในการควบคุมและเปลี่ยนการทำงานระหว่างส่วนประกอบต่าง ๆ โดยแต่ละโปรเจ็คจะประกอบด้วยแฟ้มข้อมูลมากมายหลายประเภท ซึ่งแฟ้มข้อมูลหลัก ๆ ตามตารางที่ 2.1

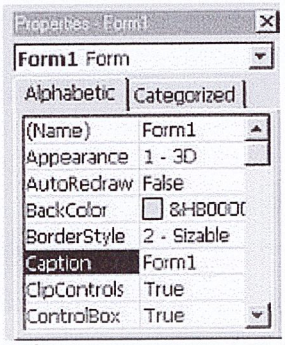
ตารางที่ 2.1 ประเภทของแฟ้มข้อมูล

ประเภทไฟล์	รายละเอียด	นามสกุลไฟล์
ไฟล์โปรเจ็ค (Project File)	เก็บข้อมูลต่าง ๆ ของโปรเจ็ค รวมทั้งรายชื่อแฟ้มที่ประกอบขึ้นมาเป็นโปรเจ็ค	.vbp
ไฟล์ฟอร์ม (Form File)	เก็บฟอร์มที่เราได้ออกแบบไว้โดยในไฟล์นี้จะรวมคำสั่งต่าง ๆ ที่เขียนโปรแกรมไว้ให้กับแต่ละออบเจ็คที่อยู่ในฟอร์มด้วย	.frm
ไฟล์ไบนารีฟอร์ม	จะเก็บข้อมูลที่เป็นแฟ้มไบนารีของฟอร์ม เช่น รูปภาพ หรือไอคอนเป็นต้น	.frx
ไฟล์โมดูลแบบ	เก็บโปรแกรมย่อย และ ตัวแปรต่าง ๆ ที่เขียนแยกออกจาก	.bas

ปกติ (Standard Module)	ฟอร์มเพื่อให้ฟอร์มหรือโมดูลอื่นสามารถเรียกใช้งานได้	
ไฟล์ Object Control	นามสกุลลงท้ายด้วย .ocx (activeX Control) หรือ .vbx เป็นออบเจ็คที่เราเพิ่มเข้าไปในโปรเจ็คนอกเหนือจากคอนโทรลพื้นฐาน ได้แก่ Internet Control Object ,Database Grid Control Object เป็นต้น	.ocx , .vbx
ไฟล์เอกสาร ActiveX	เหมือนกับ ฟอร์ม เพียงแต่ต้องเรียกดูผ่านโปรแกรมเว็บบราวเซอร์เช่น Internet Explorer	.dob
ไฟล์คลาสโมดูล	เก็บออบเจ็คต่าง ๆ ที่เราสร้างขึ้น เมื่อมีการเรียกใช้ Class Module โปรแกรมนั้นก็จะสร้างออบเจ็คนั้นขึ้นมาใหม่ (เรียกว่า Instance) แทนที่จะใช้จากโมดูลหรือออบเจ็คนั้นโดยตรง อาจกล่าวได้ว่าClass Module เปรียบเสมือนที่เก็บแผนผังหรือ Template ของออบเจ็คที่เราสร้างขึ้นใหม่นั่นเอง	.cls
ไฟล์ทรัพยากรอื่น ๆ (Resource File)	เก็บภาพBitmap(BMP), ข้อความ(Text String)หรือข้อมูลใด ๆ ที่สามารถแก้ไขได้โดยไม่ต้องไปยุ่งเกี่ยวกับโปรแกรมในโมดูลหรือฟอร์มต่าง ๆ ในโปรเจ็ค	.res

2.1.6 วินโดว์ Properties

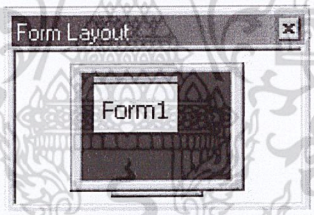
วินโดว์นี้จะแสดงคุณสมบัติทั้งหมดของออบเจ็คที่ถูกเลือกอยู่ การคลิกเลือกที่ออบเจ็คใดในฟอร์มจะทำให้คุณสมบัติที่แสดงในวินโดว์ Properties เปลี่ยนไปตามออบเจ็คที่เลือก ซึ่งการแก้ไขหรือตั้งค่าคุณสมบัติสามารถทำได้โดยตรงที่คุณสมบัติแต่ละค่า สำหรับแท็ป Alphabetic และ Categorized มีไว้เพื่อช่วยให้เราหาพร็อพเพอร์ตี้ได้ง่ายขึ้นเท่านั้น โดยแท็ป Alphabetic จะแสดงคุณสมบัติเรียงตามชื่อตัวอักษร ส่วนแท็ป Categorized จะแสดงคุณสมบัติเรียงตามลักษณะการใช้งาน การเรียกดูวินโดว์ Properties สามารถเรียกได้จากเมนู View > Properties window หรือกด F4



รูปที่ 2.6 วินโดว์ Properties

2.1.7 วินโดว์ Form Layout

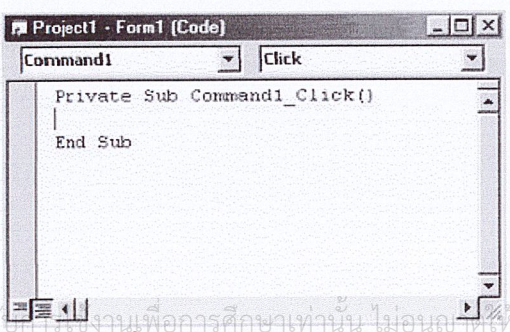
จะแสดงตำแหน่งฟอร์มของโปรแกรมที่กำลังสร้างให้ดูบนจอภาพ เพื่อกำหนดตำแหน่งสำหรับตอนที่โปรแกรมทำงานจริง ๆ การย้ายตำแหน่งทำได้โดยใช้เมาส์ลาก (drag) รูปฟอร์มตรงกลางจอภาพไปยังตำแหน่งที่ต้องการ ซึ่งสามารถทดลองได้โดยเลื่อนตำแหน่งแล้วกด F5 เพื่อรันโปรแกรม จะเห็นว่าตำแหน่งโปรแกรมของเราจะถูกเคลื่อนย้ายตามไปด้วย



รูปที่ 2.7 วินโดว์ Form Layout

2.1.8 วินโดว์ Code Editor

Code Editor เป็นเนื้อที่สำหรับเขียนโปรแกรม เรียกขึ้นมาแสดงได้โดยใช้เมนู View > Code หรือดับเบิลคลิกที่ออบเจ็คใด ๆ ในฟอร์มซึ่งวินโดว์ Code Editor จะแสดงขึ้นมาพร้อมสำหรับการป้อนโปรแกรม ให้กับเหตุการณ์หลักของออบเจ็คนั้น



รูปที่ 2.8 วินโดว์ Code Editor

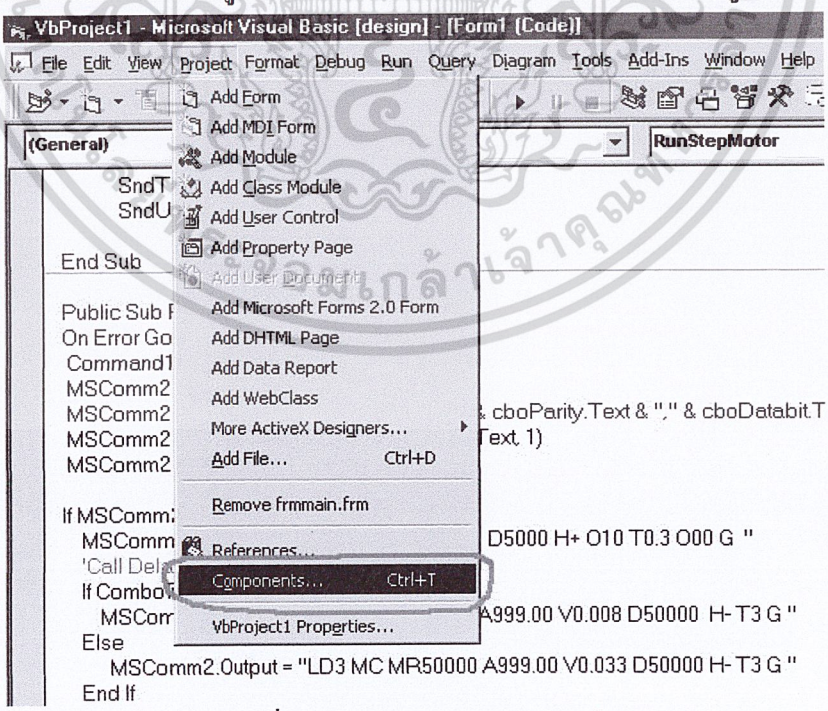
ส่วนที่สำคัญของวินโดว์นี้คือ คอมโบบ็อกซ์ (Combo box) ทั้งสองช่องที่อยู่ตรงส่วนบนของวินโดว์ ซึ่งจะเป็นตัวควบคุมการเลือกออบเจ็คและเหตุการณ์ (Event) ที่จะเกิดขึ้นกับออบเจ็คนั้น โดยโค้ดที่ปรากฏจะเป็นโปรแกรมหรือคำสั่งที่ถูกเรียกใช้งานเมื่อมีเหตุการณ์นั้น โดยโค้ดที่ปรากฏจะเป็นโปรแกรมหรือคำสั่งที่จะถูกเรียกใช้งานเมื่อมีเหตุการณ์นั้นเกิดขึ้นกับออบเจ็ค

Object list box จะแสดงชื่อออบเจ็คว่าส่วนของโปรแกรมที่กำลังแสดงอยู่ใน Code edition เป็นของออบเจ็คใด ถ้าคลิกที่ดรอปดาวน์จะปรากฏลิสต์รายการของออบเจ็คและเหตุการณ์ได้จากชื่อของโพรซีเยอร์ เช่น Command1_Click จะหมายถึงส่วนของโปรแกรมที่จะทำงานเมื่อออบเจ็คนั้นชื่อ Command1 ถูกคลิก เป็นต้น

2.1.9 Microsoft Comm Control

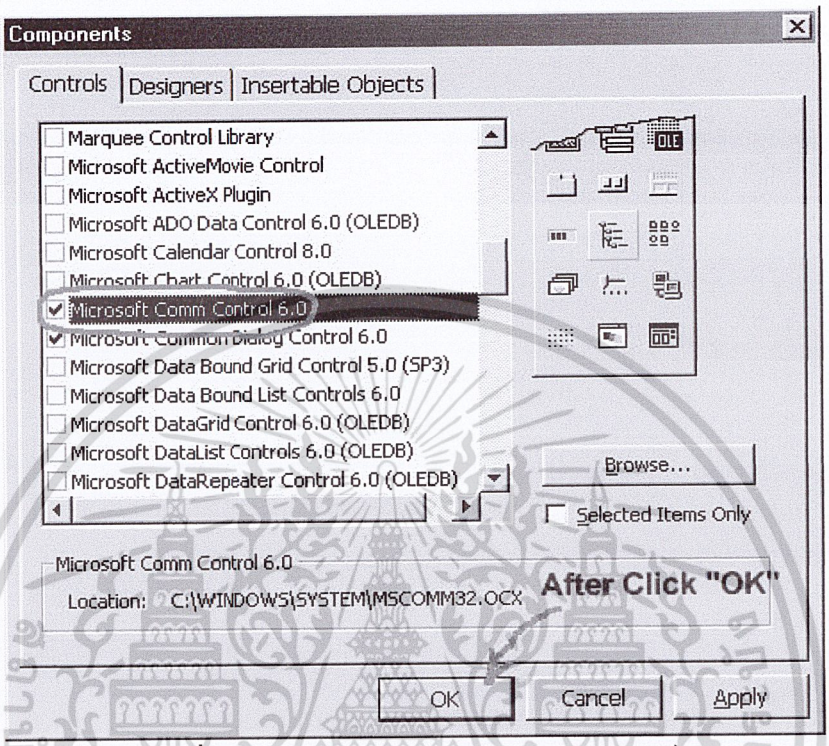
เป็น Control ที่ใช้ในการเขียนโปรแกรมติดต่อผ่านพอร์ตอนุกรม (RS-232) โดยที่ต้องกำหนด Custom Control เข้าไปที่ เมนู Project--->Components แล้วเลือกที่ช่อง MSComm ก็จะปรากฏ เป็นรูปไอคอนโทรศัพท์สีเหลือง ให้คลิกที่ไอคอนลากนำมาไว้บน Form ใน Project ของโปรแกรม เรา โดยสามารถทำตามวิธีที่กล่าวมา ได้ดังรูปต่อไปนี้

ขั้นที่ตอนแรก เลือกที่เมนูบาร์ด้านบนของโปรแกรม Visual Basic ดังรูปด้านล่าง



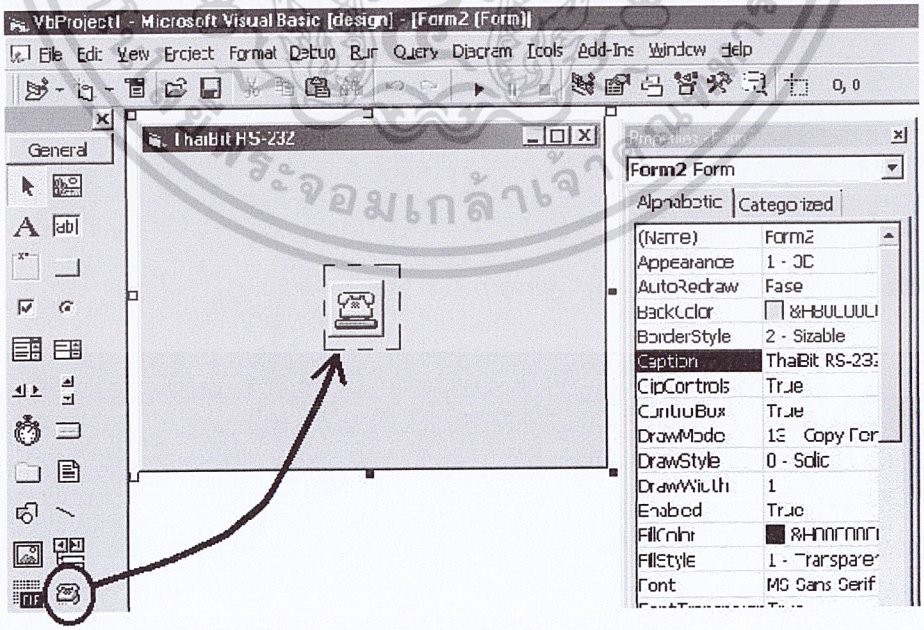
รูปที่ 2.9 การเลือกคอมโพแนนท์

ขั้นที่สอง เลือกชื่อ Control ชื่อ Microsoft Comm Control 6 ดังรูป



รูปที่ 2.10 การเลือกประเภทของคอมโพแนนท์

ขั้นที่สาม ลาก Control ชื่อ Microsoft Comm จาก ToolBox มาไว้บน Form ดังรูป



รูปที่ 2.11 การวางคอนโทรลลงบนฟอร์ม

2.1.10 การเขียนโปรแกรมติดต่อกับ Serial Port สามารถทำได้ 2 วิธี คือ

การติดต่อแบบอินเตอร์รัพต์

ขบวนการอินเตอร์รัพท์ อุปกรณ์รอบข้างเกือบทุกชิ้นจะต้องปฏิบัติงานอยู่เพื่อส่งสัญญาณไปให้แก่ซีพียูเสมอ ถ้าอุปกรณ์นั้นพร้อมที่จะรับส่ง ที่เคยเจอจากการทำโครงงานอุปกรณ์ จะส่งเป็นรหัสแอสกี้ เราจะเขียนโปรแกรมอินเตอร์รัพต์ โดยเมื่อที่ข้อมูลเข้ามาก็จะทำให้มี CommEvent กับ OnComm Event

การติดต่อแบบโพลลิ่ง

ในระบบพีซี การโพลมีบ้างที่ใช้การส่งผ่านข้อมูลระหว่าง Terminal กับ CPU กรณีข้อมูลเป็นประเภทไบท์ที่ส่งจากคีย์บอร์ด โดยวิธีการนี้จะตรวจสอบ คีย์บอร์ดว่ามีข้อมูลส่งมาหรือเปล่า โดยจะตรวจสอบตลอดเวลา ของการทำงานกับข้อมูลที่รับเข้ามาจะตรวจสอบด้วยความเร็วที่สูงกว่าอัตราความเร็วข้อมูลที่ส่งเข้ามาทาง คีย์บอร์ด การที่ CPU ส่งสัญญาณออกไปตรวจสอบพบว่ามีข้อมูลที่ต้องส่งเข้ามา เรียกว่า "Wet Poll" ซึ่งจะเสียช่วงเวลา 90 เปอร์เซ็นต์ คาบเวลาที่เสียไปนั้น เราเลี่ยงไปใช้เทคนิค การโพลแบบ "Round Robin" แทน แต่ใน Visual Basic จะใช้การตรวจสอบข้อมูลที่มาจากพอร์ตอนุกรมตลอด โดยจะใช้ Control Timer เข้ามาช่วยในการเขียนโปรแกรมซึ่งสามารถตรวจสอบได้ถึงระดับ 1 มิลลิวินาที

การตั้งค่าติดต่อกับพอร์ต

- **ComPort** คือ เราต้องกำหนดหมายเลขพอร์ตที่ใช้ต่อ RS-232 (Com1, Com2) รายละเอียดดูในเมนูด้านซ้าย Serial Port Detail
- **Setting** คือ เราต้องกำหนดอัตรา Baud, Parity, Data (จำนวนบิต), Stop ตัวอย่าง 1200, n, 8, 1 เป็นต้น
- **HandShaking** คือ เราจะกำหนดได้ 4 แบบ 1. comNone 2. comXonXoff 3. comRTS 4. comTRSXonXoff

การใช้ Buffer ในการรับส่งข้อมูล

- **InBuffersize** คือ การกำหนด Buffer ในการรับข้อมูลเข้ามา
- **OutBuffersize** คือ การกำหนด Buffer ในการส่งข้อมูลออกไป
- **Rthreshold** คือ การที่เรากำหนดการเกิด Event-driven ในการรับข้อมูลเข้ามา
- **Sthreshold** คือ การที่เรากำหนดการเกิด Event-driven ในการส่งข้อมูลออกไป
- **Inputlen** คือ จำนวนของข้อมูลที่ไปอ่านใน Buffer รับข้อมูล ถ้าเป็น 0 คือ การอ่านทุกค่าใน Buffer รับข้อมูลไปประมวลผล

- EOFEnable คือ การที่บอกว่าสิ้นสุดของไฟล์ (EOF) End of File

ด้านฮาร์ดแวร์

- ParityReplace คือ ค่าของคาแรกเตอร์ที่จะแทนในเมื่อเกิด Parity Error

- NullDiscard คือ การกำหนดให้รับหรือไม่รับ NULL CHARACTER

- RTSEnable คือ ทำให้มีสัญญาณ RTS (Request To Send)

- DTSEnable คือ ทำให้มีสัญญาณ DTR(Data Terminal Ready)

2.1.11 การกำหนดคุณสมบัติของ MSComm Control ให้สามารถติดต่อกับพอร์ตได้

Property CommPort คือ เลือกคอมพอร์ตที่เราจะต่อใช้งาน ในที่นี้เลือกจะใช้ Com1อยู่ที่ด้านหลังเครื่องคอมฯ

Property Settings คือ การตั้งค่าของการรับส่งข้อมูลซึ่งจะต้องรู้ด้วยว่าอัตราบอดเรดของอุปกรณ์ที่จะติดต่อด้วยเป็นเท่าไร มีรายละเอียดการใส่ต่าง ๆ ค่าดังนี้

MSComm1.Settings=”Baud” (อัตราการรับส่งข้อมูล)

Parity (ถ้าไม่ใช้ใส่ N, จำนวนบิตข้อมูล, บิตสต๊อบ")

Property InputLen คือ กำหนดขนาดขณะที่มีข้อมูลเข้ามาให้ไปอ่านข้อมูลทั้งหมดที่อยู่ในบัพเฟอร์

Property PortOpen คือ จะเปิดให้พอร์ตใช้งานหรือไม่ ถ้าเปิด = True ถ้าปิด = False

Property Rthreshold คือ ทำให้เกิดการกระตุ้นด้วย Event-driven เมื่อมีข้อมูลในบัฟเฟอร์รับข้อมูล (Comport) มันทำให้เกิด CommEvent ใน OnComm Event

จากรายละเอียดข้างต้นจะเขียนใน โพรซีเดอร์ Visual Basic ซึ่งจะไว้ที่ Sub Form_Load() หรือจะสร้าง Sub ขึ้นใหม่ในกรณีที่จะเรียกใช้ภายหลัง

```
Private Sub Form_Load()
        MSComm1.Settings="1200,N,8,1"
        MSComm1.CommPort=1
        MSComm1.InputLen=1
        MSComm1.PortOpen=True
        MSComm1.Rthreshold=1
End Sub
```

วิธีของการรับส่งข้อมูลจากพอร์ตอนุกรม

จากวิธีเขียนโค๊ดด้านบนเป็นการกำหนดค่าเริ่มต้นให้กับคอมพอร์ตและเปิดใช้การรับและส่งของพอร์ต RS-232 ดังนั้นก็สามารถจะรับและส่งข้อมูลทางพอร์ตได้ โดยใช้ Property ดังนี้

Output = ซึ่งจะเป็นการส่งข้อมูลไปที่พอร์ต

Input = เป็นส่วนของการรับข้อมูลจากพอร์ต แต่ในส่วนนี้จะต้องนำคำสั่งไปเขียนที่ Event Property OnCommจะอยู่ใน Sub Mscomm_OnComm ซึ่งจะอ่านข้อมูลเข้ามาทางพอร์ต RS-232

2.1.12 CommonDialog Control

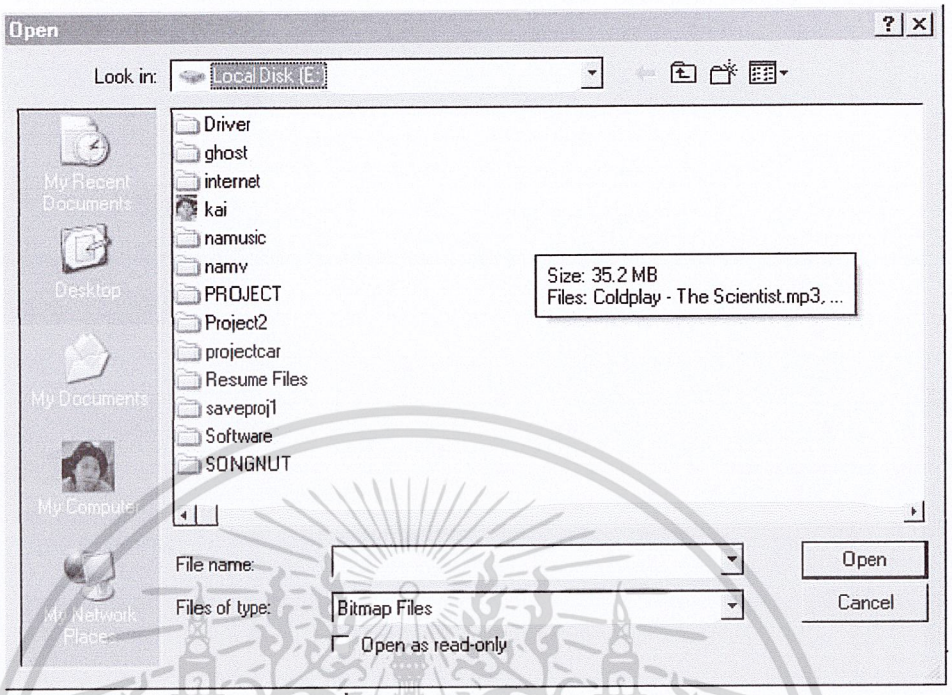
คอนโทรล CommonDialog สามารถนำไปใช้งานได้หลายด้าน เช่น การเลือกไฟล์ การเลือกสี การเลือกฟอนต์ โดยที่ต้อง กำหนด Custom Control เข้าไปที่ เมนู Project--->Components แล้วเลือกที่ช่อง Microsoft Common Dialog Control 6.0 ก็จะปรากฏ เป็นรูปไอคอน CommonDialog ให้คลิกที่ไอคอนลากนำมาไว้บน Form ใน Project ของโปรแกรมเรา
โดยจะมีเมธอดให้เลือกใช้สำหรับการใช้งานต่าง ๆ ดังนี้

CommonDialog.ShowOpen	เลือกไฟล์เพื่อเปิดไฟล์
CommonDialog.ShowSave	เลือกไฟล์เพื่อบันทึก
CommonDialog.ShowColor	เลือกสี
CommonDialog.ShowFont	เลือกฟอนต์
CommonDialog.ShowPrinter	เลือกเครื่องพิมพ์

โดยในการเลือกไฟล์จะสามารถกำหนดพร็อพเพอร์ตี้บางอย่างของไดอะล็อกได้ดังนี้

Filter กำหนดให้แสดงเฉพาะไฟล์บางประเภทมีรูปแบบการกำหนดคือ <File type>|<type> เช่น FileDialog.Filter = “Bitmap Files|*.bmp”จะแสดงเฉพาะไฟล์ที่มีนามสกุล .bmp และไดเรกทอรี่

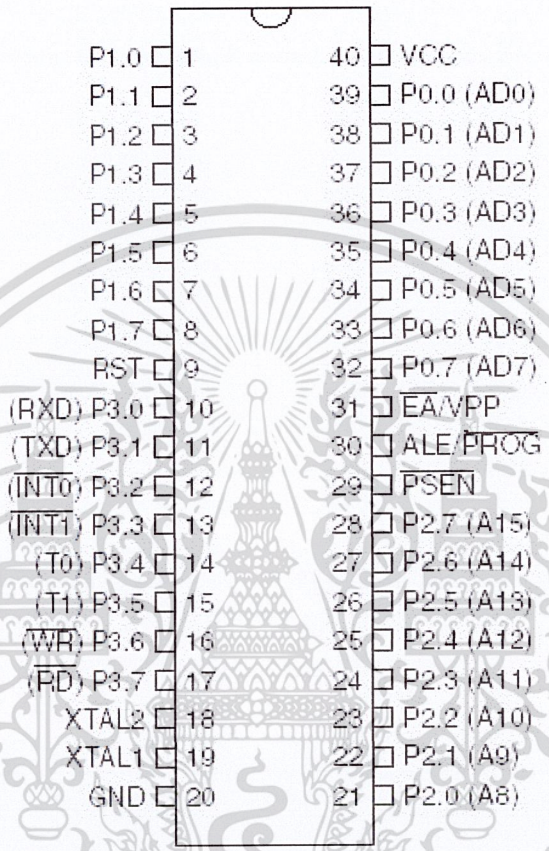
DialogTitle กำหนดข้อความส่วนหัวของไดอะล็อก เช่น FileDialog.Filter = “Open” เมื่อกำหนดพร็อพเพอร์ตี้ทั้ง 2 แล้ว ไดอะล็อกจะแสดงดังรูป เมื่อเรียกเมธอด ShowOpen



รูปที่ 2.12 การเปิดไฟล์

2.2 ไมโครคอนโทรลเลอร์ MCS-51

2.2.1 ขาต่าง ๆ ของ MCS-51



รูปที่ 2.13 ไมโครคอนโทรลเลอร์ 8051

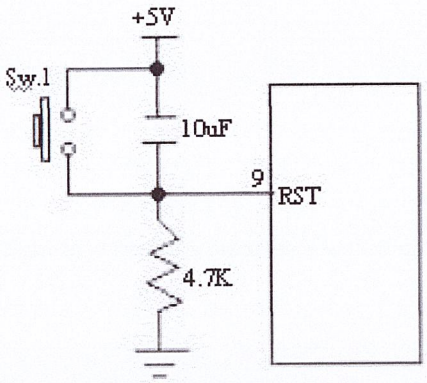
จากรูปข้างบนนั้นเป็นรูปร่างและขาต่าง ๆ ของตระกูล MCS-51 โดยมีขาแต่ละขา ดังนี้

P0.0-P0.7 (ขา 39-32) เป็น Port ใช้งานทั่วไป และพอร์ท Data และ Address

P1.0-P1.7 (ขา 1-8) เป็น Port ใช้งานทั่วไป

P2.0-P2.7 (ขา 21-28) เป็น Port ใช้งานทั่วไป

Reset (ขา 9) ขานี้มีไว้สำหรับรีเซตการทำงานของ MCS-51 ให้เริ่มต้นทำงานใหม่ ขานี้ทำงานที่ลอจิก "1" สิ่งที่เราต้องออกแบบวงจรก็คือต้อง สร้างสัญญาณพัลส์บวกที่ขา RST ในช่วงเวลาที่จ่ายไฟให้กับ MCS-51 ตอนเปิดเครื่อง จะทำการรีเซตตัวเองโดยอัตโนมัติ วงจรนี้สามารถสร้างโดยใช้ตัวต้านทานและ ตัวเก็บประจุเพื่อสร้างสัญญาณพัลส์บวกได้ดังวงจรข้างล่างนี้



รูปวงจร Reset ของ MCS-51

รูปที่ 2.14 วงจรรีเซ็ตของไมโครคอนโทรลเลอร์ 8051

การทำงานของวงจร

เมื่อเริ่มจ่ายไฟเข้าไปในวงจรตัวเก็บประจุจะเริ่มชาร์จตัวเอง ตอนนี้ตัวเก็บประจุเปรียบเสมือนช็อต แรงดันตกคร่อมที่ ตัวเก็บประจุ (Vc) = 0 โวลต์ แรงดันที่ตกคร่อมความต้านทาน (Vr) = 5 โวลต์ จากนั้นเมื่อเวลาผ่านไป ตัวเก็บประจุ ก็เริ่มชาร์ตไฟมากขึ้น ทำให้มีแรงดันตกคร่อมที่ตัวมันมากขึ้น แรงดันที่ความต้านทานก็จะน้อยลง จนในที่สุดตัวเก็บประจุก็ชาร์ตไฟจนเต็มแรงดัน (Vc) = 5 โวลต์ และแรงดันที่ความต้านทาน (Vr) = 0 โวลต์ เวลาของสัญญาณพัลส์นั้นขึ้นอยู่กับค่าตัวต้านทานและตัวเก็บประจุที่เหมาะสมโดย ตัวต้านทานมีค่าประมาณ 1-10 กิโลโอห์ม, ตัวเก็บประจุมีค่าประมารณ 1-10 ไมโครฟารัด สวิตซ์ที่ตกคร่อมตัวเก็บประจุนั้นทำหน้าที่เป็นตัวดิสชาร์จตัวเก็บประจุ และ รีเซต ให้กับ MCS-51

P3.0-3.7 (ขา 10-17) พอร์ต P3.0-P3.7 นอกจากจะเป็นพอร์ตใช้งานทั่วไปแล้วยังสามารถทำหน้าที่ได้ดังนี้

P3.0, RXD สามารถเป็นขารับสัญญาณพอร์ตอนุกรม (Serial Port)

P3.1, TXD สามารถเป็นขาส่งสัญญาณพอร์ตอนุกรม

P3.2, INT0 สามารถเป็นขารับสัญญาณอินเตอร์รัพ (Interrupt) หมายเลข 0

P3.3, INT1 สามารถเป็นขารับสัญญาณอินเตอร์รัพ หมายเลข 1

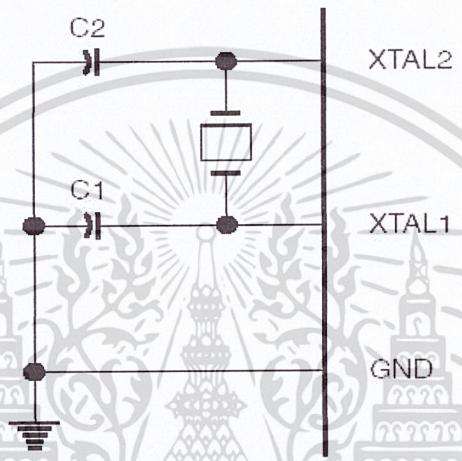
P3.4, T0 สามารถเป็นขารับสัญญาณพัลส์ หมายเลข 0 เพื่อเข้าวงจรนับ (Counter)

P3.5, T1 สามารถเป็นขารับสัญญาณพัลส์ หมายเลข 1 เพื่อเข้าวงจรนับ

P3.6, WR สามารถเป็นขาสัญญาณเขียนข้อมูลไปยังอุปกรณ์ภายนอก

P3.7, RD สามารถเป็นขาสัญญาณอ่านข้อมูลจากอุปกรณ์ภายนอก

XTAL2, 1 (ขา 18, 19) ไว้สำหรับต่อ X-TAL เพื่อสร้างสัญญาณนาฬิกาให้กับ MCS-51 ให้มันสามารถทำงานได้ เพราะถ้าไม่มีการสร้างสัญญาณ MCS-51 ก็ไม่สามารถทำงานได้ ค่าของ X-TAL ที่เราใช้จะประมาณ 11.0592 MHz การที่เราใช้ค่านี้เพราะว่าเวลาที่เราส่งข้อมูลจาก Serial Port จำเป็นต้องสร้างความถี่ในการส่งข้อมูลให้ตรงกับมาตรฐาน ค่า X-TAL ที่เราใช้นี้จะได้ความถี่ที่ตรงตามมาตรฐานพอดี เราจึงเลือกใช้ค่านี้



รูปที่ 2.15 วงจรกำเนิดความถี่ของไมโครคอนโทรลเลอร์ 8051

ตัวเก็บประจุที่เราใช้เราจะใช้เพื่อป้องกันสัญญาณฮาร์มอนิกที่เกิดขึ้นจากวงจรออสซิเลเตอร์ (Oscillator) ที่อยู่ภายใน MCS-51 ค่าที่เหมาะสมของตัวเก็บประจุคือประมาณ 6-30 พิโกฟารัด

$\overline{\text{PSEN}}$ (Program Status Enable) (ขา 29)

เป็นขาที่ทำหน้าที่บอกว่า MCS-51 ต้องการติดต่อกับ หน่วยความจำโปรแกรมภายนอก (Program Memory) ขานี้จะทำงานที่ลอจิก "0" โปรแกรมเมมโมรี่ ส่วนใหญ่ก็คือ EPROM ที่ต่ออยู่ภายนอกหรือ FLASH ROM ที่อยู่ภายในเบอร์ของ AT89Cxx นั้นเอง ในกรณีที่เราไม่ได้ต่อ EPROM ภายนอกเราก็ปล่อยลอย ๆ ไว้อย่างนั้น

ALE (Address Latch Enable) (ขา 30)

ขานี้จะใช้ในกรณีที่เรามีการต่อ EPROM, RAM, Chip Support ภายนอก ขานี้จะเป็นขาที่บอกให้รู้ว่าข้อมูลที่พอร์ต 0 เป็น Address เพราะว่า พอร์ต 0 มีข้อมูลเป็นค่า Address ขา ALE จะเป็น "1" เพื่อส่งสัญญาณให้กับไปซีที่ทำหน้าที่เป็นวงจร LATCH

$\overline{EA}$ (Enable Access) (ขา31)

MCS-51 สามารถติดต่อกับหน่วยความจำได้ 3 แบบ คือ

-หน่วยความจำภายใน (Internal Memory) เป็นหน่วยความจำภายในที่อยู่ใน MCS-51 ซึ่งจะเป็นพวกรีจิสเตอร์ต่าง ๆ

-หน่วยความจำข้อมูลภายนอก (Data Memory) สามารถอ้างข้อมูลได้ 64 กิโลไบต์ เป็นส่วนที่ทำหน้าที่เก็บข้อมูล จะเป็นหน่วยความจำแบบแรม (RAM)

-หน่วยความจำโปรแกรมภายนอก (Program Memory) สามารถอ้างข้อมูลได้ 64 กิโลไบต์ เป็นส่วนที่ทำหน้าที่เก็บคำสั่งที่เราเขียน ส่วนใหญ่ถูกเก็บไว้ที่ EPROM ภายนอก แต่ปัจจุบันนี้ได้มีการพัฒนาจนสามารถนำเอาหน่วยความจำโปรแกรมภายนอก อีกแบบหนึ่งซึ่งเรียกว่า Flash Rom เข้าไปในไมโครคอนโทรลเลอร์ได้แล้ว Flash ROM นั้นมีข้อดีกว่า EPROM เพราะสามารถลบด้วยไฟฟ้าและเขียนได้เร็วกว่าด้วย

ด้วยเหตุนี้จึงต้องมีขา EA เป็นตัวเลือกว่าจะใช่หน่วยความจำโปรแกรมภายนอกหรือภายในซึ่งในที่นี้เราใช้เบอร์ 89C51 ซึ่งมี หน่วยความจำโปรแกรมภายในที่เป็น Flash ROM เราจึงให้ขา EA ต่อ VCC ถ้าเราใช้หน่วยความจำโปรแกรมภายนอกขา EA จะต่อลงกราวด์

2.2.2 รีจิสเตอร์และหน่วยความจำ

MCS-51 สามารถติดต่อกับหน่วยความจำได้ 3 ประเภท

หน่วยความจำโปรแกรมภายนอก (Program Memory)

หน่วยความจำชนิดทำหน้าที่เก็บคำสั่งของผู้เขียนโปรแกรม ส่วนใหญ่จะใช้ EPROM เป็นตัวเก็บ โดยมีสัญญาณ PSEN เป็นขาสัญญาณเพื่อติดต่อกับ EPROM สามารถอ้างหน่วยความจำได้ 64 กิโลไบต์ ส่วนใหญ่เราออกแบบให้ หน่วยความจำโปรแกรมภายนอก เริ่มต้นที่ แอดเดรส 0000H เพราะเวลาที่ MCS-51 ถูกรีเซต MCS-51 จะกระโดดไปที่แอดเดรส 0000H ของหน่วยความจำโปรแกรมภายนอก EPROM นั้นส่วนใหญ่จะขึ้นต้นด้วย 27 แล้วตามด้วยขนาดของหน่วยความจำ เช่น 2764 หมายเลข 64 คือขนาดของหน่วยความจำ โดยเอาค่า 64/8 = 8 กิโลไบต์ เราก็จะได้ขนาดของ EPROM ตัวนี้ จากนั้นได้มีการพัฒนา MCS-51 โดยเอา EPROM ใส่เข้าไปในตัว MCS-51 แล้วตั้งชื่อเบอร์ใหม่ว่า 8751 และก็มีการพัฒนาขึ้นไปอีกโดยนำเอา Flash ROM ซึ่งสามารถลบและเขียนด้วยไฟฟ้าใส่เข้าไปใน MCS-51 แล้วตั้งชื่อใหม่ว่า 8951 จนเป็นที่นิยมในปัจจุบัน คำสั่งที่ใช้ในการเอาข้อมูลจาก หน่วยความจำโปรแกรมภายนอกนั้นคือคำสั่งพวก MOVC

หน่วยความจำข้อมูลภายนอก (Data Memory)

หน่วยความจำชนิดนี้ทำหน้าที่เก็บข้อมูลต่าง ๆ ที่เราต้องการ ซึ่งก็คือ RAM ที่ต่ออยู่ภายนอกสามารถเข้าถึงข้อมูลโดยใช้ขา RD,WR ในการเขียนและอ่าน Data จาก RAM MCS-51 สามารถอ้างหน่วยความจำข้อมูลภายนอกได้ 64 กิโลไบต์ เช่นกัน ส่วนใหญ่แล้วหน่วยความจำนี้จะถูกจัด Address ให้ถัดจาก หน่วยความจำโปรแกรมภายนอก (EPROM) RAM ที่ใช้ส่วนใหญ่จะเป็น Static RAM เบอร์จะขึ้นต้นด้วยหมายเลข 61,62,64,68 แล้วตามด้วยขนาดของหน่วยความจำเช่น 6264 หมายเลข 64 คือขนาดของหน่วยความจำ โดยเอาค่า 64/8 = 8 กิโลไบต์ เราก็จะได้ขนาดของ RAM ตัวนี้ คำสั่งที่ใช้ในการเอาข้อมูลจาก หน่วยความจำข้อมูลภายนอก นั้น คือคำสั่งพวก MOVX

หน่วยความจำภายใน Internal RAM

เป็นหน่วยความจำภายใน ซึ่งเป็น RAM ในหน่วยความจำนั้น ก็จะประกอบด้วย รีจิสเตอร์ ซึ่งจะมีแอดเดรสประจำตัว ทำหน้าที่ใช้งานแตกต่างกันไปตามการใช้งาน รูปข้างล่างนี้แสดงรีจิสเตอร์ของหน่วยความจำภายใน

แอดเดรส ของหน่วยความจำเป็น แอดเดรส ที่ทำหน้าที่เป็นรีจิสเตอร์ได้ เราสามารถแบ่งประเภทของหน่วยความจำได้ออกเป็น 4 กลุ่ม ดังนี้

- **กลุ่มรีจิสเตอร์พื้นฐาน** เป็นกลุ่มที่สามารถนำไปใช้ในคำสั่งได้โดยตรง นั้นก็คือรีจิสเตอร์ R0-R7 จะแบ่ง Address ออกเป็นช่วง ๆ หรือที่เรียกว่า Bank ตั้งแต่ Bank0-Bank3 ดังนี้

Bank0 จะให้ R0-R7 อยู่ที่ Address 00H-07H

Bank1 จะให้ R0-R7 อยู่ที่ Address 08H-0FH

Bank2 จะให้ R0-R7 อยู่ที่ Address 10H-17H

Bank3 จะให้ R0-R7 อยู่ที่ Address 18H-1FH

เมื่อเริ่มจ่ายไฟให้ MCS-51 รีจิสเตอร์ R0-R7 จะถูกกำหนดให้แอดเดรสอยู่ใน Bank0 จากนั้นเราก็สามารถจะเซตได้ว่าจะให้รีจีสเตอร์ R0-R7 อยู่ที่ Bank ไหน โดยเซตที่รีจิสเตอร์ที่ชื่อว่า PSW ซึ่งอยู่ในส่วนของกลุ่มของ SFR (Special Function Registers)

- **กลุ่มหน่วยความจำที่สามารถอ้างเป็น ไบต์ และ บิต** ในส่วนของกลุ่มนี้จะเริ่มต้นที่แอดเดรส 20H-2FH ทั้งหมดมี 16 ไบต์ = 128 บิต จะมีตำแหน่งของบิตกำกับอยู่ ซึ่งในส่วนนี้เราสามารถที่จะเซตหรือเคลียร์บิตใดบิตหนึ่งได้เลย อันนี้เป็นความสามารถเฉพาะตัวของ MCS-51

- **กลุ่มหน่วยความจำที่สามารถอ้างเป็นไบต์ได้อย่างเดียว** กลุ่มนี้จะอยู่ที่แอดเดรส 30H-7FH จะเอาไว้ใช้งานทั่วไป เช่นจะเอาไว้ค่าต่าง ๆ เวลาเราเขียนโปรแกรม ซึ่งสามารถทำคำสั่งที่เป็นไบต์ได้อย่างเดียว ไม่สามารถทำเป็นบิตได้

- กลุ่มรีจิสเตอร์พิเศษ SFR (Special Function Registers) จะประกอบด้วยรีจิสเตอร์ต่าง ๆ เช่นรีจิสเตอร์ในการคำนวณ รีจิสเตอร์สำหรับการอินเตอร์รัพ รีจิสเตอร์สำหรับวงจร Timer/Counter รีจิสเตอร์สำหรับพอร์ตอนุกรม

● รีจิสเตอร์ P0-P3 จะเป็นรีจิสเตอร์ที่เชื่อมต่อจากพอร์ต P1-P3 ถ้าจะพูดกันง่าย ๆ ก็คือถ้าข้อมูลที่รีจิสเตอร์ P0-P3 เป็นอะไร ข้อมูลที่พอร์ต P0-P3 ก็เป็นอย่างนั้น รีจิสเตอร์ทั้ง 4 ตัวนี้สามารถทำ Bit Addressing ได้ คือ สามารถเซตและเคลียร์เป็นบิตได้ รีจิสเตอร์ทั้ง 4 ตัวนี้อยู่ แอดเดรส 80H, 90H, A0H, B0H ตามลำดับ

● รีจิสเตอร์ ACC เป็นรีจิสเตอร์ที่ใช้ในการคำนวณทางคณิตศาสตร์ และทางลอจิกและใช้ในการโอนถ่ายข้อมูลต่าง ๆ เป็นรีจิสเตอร์ที่ใช้บ่อยมาก ๆ สามารถอ้างเป็นบิตได้ รีจิสเตอร์ตัวนี้อยู่ที่แอดเดรส E0H

● รีจิสเตอร์ B เป็นรีจิสเตอร์ใช้งานทั่วไปและเป็นรีจิสเตอร์ที่ทำหน้าที่หารและคูณกับรีจิสเตอร์ A เท่านั้น

● รีจิสเตอร์ PSW (Program Status Word) เป็นรีจิสเตอร์ที่แสดงสถานะของรีจิสเตอร์ A ว่าเป็นอย่างไร

● DPTR (Data Pointer) เป็นรีจิสเตอร์ที่ทำหน้าที่ชี้แอดเดรส ของหน่วยความจำภายนอก มันจึงมีขนาด 16 บิต รีจิสเตอร์ตัวนี้จะประกอบด้วยรีจิสเตอร์ DPH, DPL ขนาด 8 bit โดย DPH เป็นรีจิสเตอร์ไบต์สูง อยู่ที่ แอดเดรส 83H และ DPL เป็นรีจิสเตอร์ไบต์ต่ำอยู่ที่ แอดเดรส 82H

● SP (Stack Pointer) เป็นรีจิสเตอร์ที่ทำหน้าที่ชี้ตำแหน่งของหน่วยความจำภายในตำแหน่งมันชี้เราจะเรียกว่า Stack ทำหน้าที่เก็บข้อมูลต่าง ๆ ที่ MCS-51 ต้องการเก็บโดยมันจะมีการเก็บข้อมูลของ MCS-51 จากคำสั่งพวก Call, Push, Pop หรือจากการอินเตอร์รัพท์

● รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer :SBUF) เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่มีอยู่ในไมโครคอนโทรลเลอร์ โดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วน คือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล(transmit buffer register) และ รีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล(receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของ MCS-51

● รีจิสเตอร์ไทเมอร์ (Timer register) ขนาด 16 บิต แบ่งเป็นไบต์สูงไบต์ต่ำเช่นเดียวกับ DPTR ใช้ในการเก็บค่าของ counter เพื่อใช้ในการสร้างฐานเวลา จับเวลา หรือนับจำนวนพัลส์สัญญาณนาฬิกาภายใน บางทีเรียกรีจิสเตอร์นี้ว่า รีจิสเตอร์ไทเมอร์/เคาเตอร์

● รีจิสเตอร์ควบคุม (Control register)

รีจิสเตอร์ **PCON** เกี่ยวข้องกับการกำหนดอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรม

รีจิสเตอร์ **SCON** ใช้ในการควบคุมการทำงานของ วงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์

รีจิสเตอร์ **TCON** และ **T2CON** ใช้ในการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์ภายใน MCS-51

รีจิสเตอร์ **IE** และ **IP** เกี่ยวข้องกับการตอบสนองการอินเตอร์รัพท์ โดย IE เป็นรีจิสเตอร์สำหรับ Enable หรือใช้ในการกำหนดลักษณะของการตอบสนองอินเตอร์รัพท์ ในขณะที่ IP เป็นรีจิสเตอร์สำหรับกำหนดลำดับความสำคัญของการตอบสนองการอินเตอร์รัพท์ว่า จะให้ CPU ตอบสนองการเกิดอินเตอร์รัพท์ในลักษณะใดก่อนหรือหลัง

2.2.3 การใช้ไมโครคอนโทรลเลอร์ส่งข้อมูลผ่านพอร์ตอนุกรม (RS-232)

การรับและส่งข้อมูลจะอาศัยรีจิสเตอร์ขนาด 8 บิตสองตัว ซึ่งแยกกันทำงานสองอย่างคือใช้รับข้อมูลที่มาจากภายนอกและใช้สำหรับส่งข้อมูลออกไปภายนอก รีจิสเตอร์ทั้งสองมีตำแหน่งเดียวกันใน SFR (Special function register) มีชื่อเรียกว่า SBUF (ตำแหน่ง 99H) การเข้าถึงข้อมูลในรีจิสเตอร์ทั้งสองนั้น MCS-51 จะทราบเองว่าผู้ใช้ต้องการติดต่อกับรีจิสเตอร์ SBUF ที่ใช้รับหรือส่งจากรหัสคำสั่ง เพราะ SBUF ที่ใช้รับข้อมูลจะเป็น Operand ตัวแรกในคำสั่ง ส่วน SBUF ที่ใช้ส่งข้อมูลจะเป็น Operand ตัวที่สองในคำสั่ง การทำงานสามารถ กำหนดโหมดการทำงานได้ 4 แบบ โดยควบคุมจากค่าในรีจิสเตอร์ SCON

2.2.3.1 SCON

การใช้งานพอร์ตอนุกรมของ MCS-51 มีความคล่องตัวสูงมากทั้งนี้เนื่องจากผู้ใช้สามารถควบคุมการทำงานที่แตกต่างกันได้ถึง 4 ประเภท โดยควบคุมจากบิตต่าง ๆ ในรีจิสเตอร์ควบคุมพอร์ตอนุกรม SCON (ตำแหน่ง 99H) ซึ่งสามารถเข้าถึงได้ในระดับบิต

Bit	7	6	5	4	3	2	1	0
	SM0	SM1	SM2	REN	TB8	RB8	T1	R2

รูปที่ 2.16 รีจิสเตอร์ควบคุมพอร์ตอนุกรม SCON

SM0 บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมควบคุมได้โดยโปรแกรม

SM1 บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมควบคุมได้โดยโปรแกรม

SM0	SM1	Mode	
0	0	0	ทำงานเป็น shift register อัตราเร็วในการรับหรือส่งข้อมูลเท่ากับ OSC/12
0	1	1	8 Bit UART อัตราเร็วในการรับหรือส่งข้อมูลกำหนดเองได้
1	0	2	9 Bit UART อัตราเร็วในการรับหรือส่งข้อมูล = OSC/32 หรือ OSC/64
1	1	3	9 Bit UART อัตราเร็วในการรับหรือส่งข้อมูลกำหนดเองได้

SM2 บิตการเลือกใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 2 หรือ 3 เพื่อใช้ติดต่อระหว่างซีพียูด้วยกันเอง ควบคุมโดยใช้โปรแกรม

1 ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมติดต่อระหว่างซีพียูด้วยกันเอง เมื่อข้อมูลบิตที่ 9 ที่ได้รับมีค่าเป็น 0 (ดาต้าไบต์) บิต RI จะไม่ถูกเซตแต่หากข้อมูลที่ 9 มีค่าเป็น 1 (แอดเดรสไบต์) บิต RI จะถูกเซต

0 ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 และโหมด 3 ตามปกติในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 หากบิต SM2 ถูกเซต บิต RI จะไม่ถูกเซตจนกว่าบิตสิ้นสุดของข้อมูลจะถูกรับเข้ามา ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 บิตนี้ควรถูกเคลียร์ให้เป็น 0

REN บิตควบคุมการอนุญาตให้มีการรับข้อมูลสามารถควบคุมได้โดยโปรแกรม

1 อนุญาตให้มีการรับข้อมูล

0 ไม่อนุญาตให้มีการรับข้อมูล

TB8 ข้อมูลบิตที่ 9 ซึ่งถูกส่งออกไปในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 ควบคุมได้ด้วยคำสั่งในโปรแกรมเท่านั้น

RB8 ข้อมูลบิตที่ 9 ที่ได้รับเข้ามาจากภายนอก ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 ส่วนในการทำงานโหมด 1 ถ้าบิต SM2 = 0 บิตนี้จะเป็นบิตสิ้นสุดของข้อมูลที่รับเข้ามาได้และมาถูกกำหนดการใช้งานในโหมด 0

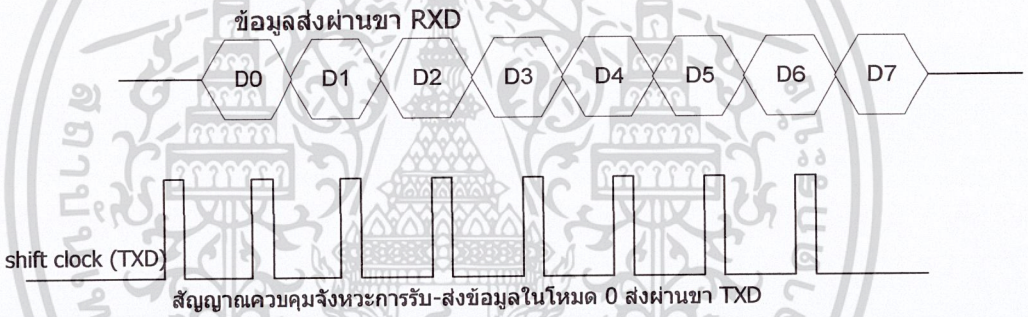
T1 บิตบอกสถานะสัญญาณอินเตอร์รัพท์ที่เกิดจากการส่งข้อมูลถูกเซตโดยฮาร์ดแวร์ เมื่อข้อมูลเริ่มส่งบิตสิ้นสุดของข้อมูลออกไปและจะต้องเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น

R1 บิตบอกสถานะสัญญาณอินเตอร์รัพท์ที่เกิดจากการรับข้อมูล ถูกเซตโดยฮาร์ดแวร์เมื่อได้รับข้อมูลบิตที่ 8 ในการทำงานโหมด 0 หรือที่จุดครึ่งทางของช่วงรับบิตสิ้นสุดของข้อมูลในการทำงานโหมดอื่นและจะต้องเคลียร์ในโปรแกรมเท่านั้น

โหมดของการรับ-ส่งข้อมูลแบบอนุกรม

โหมด 0 (Shift Register Mode)

ขา P3.0 (RXD) ถูกใช้เป็นขาสำหรับรับและส่งข้อมูล ขา P3.1(TXD) ใช้สร้างสัญญาณ Shift Clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูล การส่งข้อมูลจะเริ่มทันทีที่มีการใช้งานรีจิสเตอร์ SBUF เป็นรีจิสเตอร์ปลายทาง (destination register) ข้อมูลจะถูกเลื่อนออกไปทีละ 1 บิท โดยเริ่มจากบิทต่ำสุด (LSB) ก่อนและจะมีบิท 0 เลื่อนเข้ามาแทนที่บิทสูงสุดที่ถูกเลื่อนไป สัญญาณที่ขา TXD จะมีความถี่เท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์ ถูกส่งออกมาพร้อม ๆ กับบิทข้อมูล ดังรูป 000 การรับข้อมูลจะเกิดขึ้นได้ต่อเมื่อบิท REN = 1 และ RI = 0 โดยทีขา TXD จะมีสัญญาณควบคุมการรับข้อมูลออกมาพร้อม กับการรับข้อมูลแต่ละบิท ดังรูปที่ 2.14 อัตราการรับส่งข้อมูลถูกกำหนดไว้ที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้

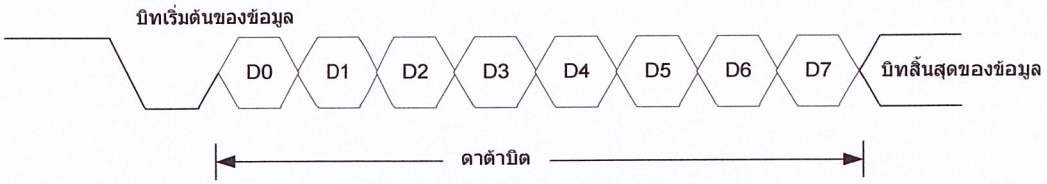


รูปที่ 2.17 สัญญาณการรับและส่งข้อมูลในโหมด 0

โหมด 1 (Standard 8 bit UART Mode)

ขา P3.0 (RXD) ถูกใช้เป็นขาสำหรับ รับข้อมูล ขา P3.1 (TXD) ถูกใช้เป็นขาสำหรับส่งข้อมูล การรับ-ส่งข้อมูลจะเป็นแบบ 10 บิท ประกอบด้วยบิทเริ่มต้น 1 บิท (เป็น 0 เสมอ) บิทข้อมูล 8 บิท (รับและส่งบิทต่ำสุดก่อน) และบิทสิ้นสุดของข้อมูล 1 บิท (เป็น 1 เสมอ) ดังรูปที่ 2.15 ในการส่งข้อมูลจะเริ่มต้นโดยคำสั่งใด ๆ ที่มีการใช้งานรีจิสเตอร์ SBUF เป็นรีจิสเตอร์ปลายทาง บิท TI จะเซตเมื่อข้อมูลถูกเลื่อนออกไปจนครบทั้ง 10 บิทแล้ว การรับข้อมูลจะเริ่มต้นทันทีที่ลอจิกที่ขา RXD เปลี่ยนสภาวะจาก 1 เป็น 0 ข้อมูลจะถูกเลื่อนเข้ามาใน Buffer สำหรับรับข้อมูล เมื่อข้อมูลเข้ามาครบทั้ง 8 บิทแล้ว จึงจะถูกโหลดไปไว้ที่รีจิสเตอร์ SBUF พร้อมกับบิทสิ้นสุดของข้อมูลถูกเก็บไว้ที่บิท RB8 ของรีจิสเตอร์ SCON บิท RI จะเซต เป็นการบอกให้รู้ว่ากระบวนการรับข้อมูลจำนวน 1 ไบท์ได้เสร็จสิ้นลงแล้ว และเริ่มรีเซตตัวเองเพื่อกลับไปรอรับการเปลี่ยนสภาวะจาก 1เป็น 0 ที่ขา RXD อีก ทั้งนี้กระบวนการรับข้อมูลจะเป็นจริงก็ต่อเมื่อ

- บิท RI = 0 และ บิท SM2 = 0 หรือถ้า SM2 = 1 บิทสิ้นสุดของข้อมูลจะต้อง = 1 ด้วย

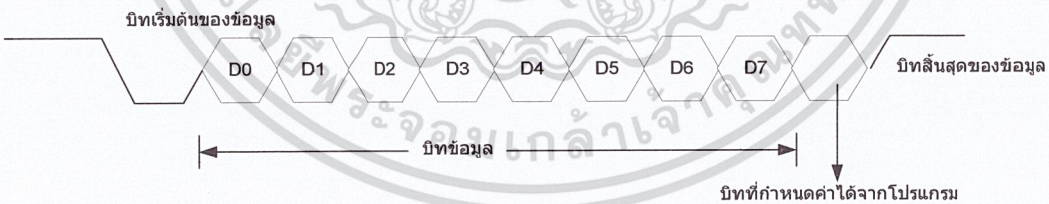


รูปที่ 2.18 การรับและส่งข้อมูลในโหมด 1

โหมด 2 และ 3 (Multiprocessor)

ขา P3.0 (RXD) ถูกใช้เป็นขาสำหรับรับข้อมูล ขา P3.1 (TXD) ถูกใช้เป็นขาสำหรับส่งข้อมูล การรับ-ส่งข้อมูลจะเป็นแบบ 11 บิท ประกอบด้วยบิทเริ่มต้น 1 บิท (เป็น 0 เสมอ) บิทข้อมูล 8 บิท (รับและส่งบิทต่ำสุดก่อน) บิทที่กำหนดได้ 1 บิท (โปรแกรมเองได้ว่าจะเป็น 1 หรือ 0) และบิทสิ้นสุด 1 บิท (เป็น 1 เสมอ) ดังรูปที่ 2.16

การส่งข้อมูลจะเริ่มต้นโดยคำสั่งใด ๆ ที่มีการใช้รีจิสเตอร์ SBUF เป็นรีจิสเตอร์ปลายทาง บิท TI จะเซต เมื่อข้อมูลถูกเลื่อนออกไปจน ครบทั้ง 11 บิทแล้ว บิทที่ 9 จะได้จากบิท TB8 ในรีจิสเตอร์ SCON ส่วนในขณะรับข้อมูลบิทที่ 9 จะอยู่ในบิท RB8 ของรีจิสเตอร์ SCON โดยไม่สนใจบิทสิ้นสุดของข้อมูล การรับข้อมูลจะใช้การตรวจสอบการเปลี่ยนแปลงสภาวะลอจิกจาก 1 เป็น 0 ที่ขา RXD และมีเงื่อนไขในการรับข้อมูลเช่นเดียวกับ โหมด 1 ในการทำงานโหมด 2 อัตราการรับและส่งข้อมูลสามารถเปลี่ยนแปลงได้



รูปที่ 2.19 การรับและส่งข้อมูลในโหมด2 และ3

2.2.3.2 PCON

เป็นรีจิสเตอร์สำหรับควบคุมการทำงานของ 8051 ซึ่งเป็นรีจิสเตอร์ที่ไม่สามารถอ้างอิงได้ ดังมีรายละเอียดดังนี้

SMOD	-	-	-	GF1	GF0	PD	IDL

รูปที่ 2.20 รีจิสเตอร์ที่ไม่สามารถอ้างอิงได้

SMOD เป็นบิตกำหนดอัตราความเร็วของการรับส่งข้อมูลแบบอนุกรมที่ใช้ Timer1 เช่นถ้าเวลาในการรับส่ง เมื่อ SMOD = 1 จะทำให้ความเร็วในการรับส่งแบบอนุกรมเป็น 2 เท่า เมื่อเลือกการใช้งานใน Mode 1, 2, 3 ของการรับส่งแบบอนุกรม เป็นบิตที่ไม่ได้ใช้งานซึ่งสงวนไว้สำหรับหน้าที่เพิ่มเติมในอนาคตของ 8051 ดังนั้นผู้ใช้จึงไม่ควรกำหนดในบิตนี้เป็น 1

GF0 – GF1 เป็นบิตอเนกประสงค์ที่ผู้ใช้สามารถกำหนดการทำงานได้

PD เป็นบิตสำหรับให้ 8051 เข้าสู่การทำงานใน Power Down Mode ซึ่งจะมีเฉพาะเบอร์ที่มีโครงสร้างภายในเป็น CMOS เท่านั้น ซึ่งจะทำงานเมื่อบิตนี้เป็น 1

IDL เป็นบิตสำหรับกำหนดให้ 8051 เข้าสู่การทำงานใน Idle Mode ซึ่งจะมีเฉพาะเบอร์ที่มีโครงสร้างภายในเป็น CMOS เท่านั้น ซึ่งจะทำงานเมื่อบิตนี้เป็น 1

ถ้าบิต PD และ IDL เป็น 1 ทั้งคู่ในเวลาเดียวกัน 8051 จะทำงานใน Power Down Mode ในการทำงานทั้งสอง Mode นี้เป็นการทำงานเพื่อประหยัดพลังงานของ 8051 โดยมีข้อแตกต่างดังนี้

Power Down Mode เมื่อมีการกำหนดให้บิตนี้เป็น 1 8051 จะทำงานอีกหนึ่งคำสั่งก่อนที่จะเข้าสู่ Power Down Mode โดยที่ส่วนสร้างความถี่อ้างอิงภายในจะหยุดทำงาน การทำงานทุกอย่างของ 8051 จะหยุด ข้อมูลต่าง ๆ ของหน่วยความจำภายในรวมทั้ง SFR จะถูกรักษาไว้ ที่สัญญาณ ALE และ PSEN จะเป็นลอจิก 0 เมื่อเข้าสู่ Power Down Mode แรงดันไฟฟ้าที่จ่ายให้กับ 8051 สามารถลดลงเป็น 2V ได้ อย่างไรก็ตามแรงดันไฟฟ้าจะต้องไม่ลดลงก่อนที่จะเข้าสู่ Power Down Mode และแรงดันไฟฟ้าจะต้องเข้าสู่ระดับปกติก่อนที่จะออกจาก Power Down Mode การออกจาก Power Down Mode จะกระทำได้เฉพาะทำการรีเซ็ต 8051 เท่านั้น ซึ่งจะทำให้ค่าของ SFR เปลี่ยนแปลง แต่ข้อมูลในส่วนของหน่วยความจำภายในไม่เปลี่ยนแปลง

Idle Mode เมื่อมีการกำหนดให้บิตนี้เป็น 1 8051 จะทำงานอีกหนึ่งคำสั่งก่อนเข้าสู่ Idle Mode โดยที่ความถี่การทำงานภายในซีพียู ของ 8051 จะหยุดทำงานยกเว้นที่ให้กับส่วนของการอินเตอร์รัพท์ Timer และการรับส่งแบบอนุกรม ค่าต่าง ๆ ของรีจิสเตอร์จะถูกเก็บรักษาไว้ และสัญญาณของ ALE และ PSEN จะอยู่ที่ลอจิก 1 ในการออก Idle Mode จะสามารถทำได้สองวิธี ได้แก่การรีเซ็ตค่าและการอินเตอร์รัพท์

2.2.3.3 TMOD

เป็นรีจิสเตอร์สำหรับควบคุม Mode การทำงานของ Timer/Counter และไม่สามารถอ้างอิงระดับบิตได้ ซึ่งจะแบ่งเป็นสองส่วน โดยที่สี่บิตแรกของรีจิสเตอร์นี้ใช้สำหรับ Timer/Couter 0 และสี่บิตบนสำหรับ Timer/Counter 1 มีรายละเอียดดังนี้

Timer 1				Timer 0			
GATE	C/T	M1	M0	GATE	C/T	M1	M0

รูปที่ 2.21 รีจิสเตอร์ควบคุม Timer/Counter

GATE เป็นบิตใช้สำหรับการควบคุม Timer/Counter ถ้าเป็นลอจิก 1 Timer/Counter เมื่อขาสัญญาณที่ขา INTx เป็นลอจิก 1 เท่านั้น และบิต TRx ใน TCON ต้องเป็นลอจิก 1 ด้วย ซึ่งจะเป็นการควบคุมด้วยฮาร์ดแวร์ เมื่อเป็นลอจิก 0 การควบคุม Timer/Counter นั้น ๆ จะเป็นการควบคุมด้วยโปรแกรมที่บิต TRx ของรีจิสเตอร์ TCON เท่านั้น

C/T เป็นบิตสำหรับเลือกการทำงานของ Timer/Counter เป็น Timer หรือ Counter โดยที่ถ้าเป็นลอจิก 1 จะทำหน้าที่เป็น Counter โดยรับสัญญาณจากขา Tx ของ 8051 และลอจิก 0 เป็น Timer โดยความถี่ที่นับได้จากความถี่ภายในของ 8051 M0, M1

M0, M1เป็นบิตสำหรับเลือกการทำงานของ Timer/Counter ดังแสดงในตารางต่อไปนี้

ตารางที่ 2.2 การเลือกการทำงานของ Timer/Counter

M0	M1	การทำงาน
0	0	เป็น Timer ขนาด 13 บิต เมื่อการทำงานของตระกูล 48
0	0	เป็น Timer/Counter ขนาด 16 บิต
1	0	เป็น Timer/Counter ขนาด 8 บิต ที่สามารถตั้งค่าใหม่อัตโนมัติ
1	1	Timer/Counter 0 จะถูกแบ่งออกเป็น Timer/Counter 2ชุด ขนาด 8 บิต โดยที่ TL0 จะเป็น Timer/Counter ขนาด 8 บิต ที่ถูกควบคุมของ Timer/Counter 0 และTH0 จะเป็น Timer/Counter ขนาด 8 บิต ที่ควบคุมโดยชุดควบคุมของ Timer/Counter1
1	1	Timer/Counter 1 จะหยุดทำงาน

2.2.4 การคำนวณความเร็วการรับและส่งข้อมูลแบบอนุกรม (Generating Baud Rate)

การกำหนดความเร็วการรับส่งข้อมูลแบบอนุกรมสามารถแบ่งออกตาม Mode การทำงานดังนี้

Mode 0 ความเร็วในการรับส่งข้อมูลแบบอนุกรมใน Mode นี้จะกำหนดอัตราการรับส่งตายตัวเท่ากับ 1/12 ของความถี่ของชุดกำเนิดความอ้างอิงของ 8051 และจะไม่ใช้ Timer/Counter ดังนั้นเพียงกำหนดที่ รีจิสเตอร์ SCON ก็เพียงพอ จะได้

Baud Rate = Osc. Freq / 12

Mode 1 ในการกำหนดความเร็วการรับส่งข้อมูลแบบอนุกรมใน Mode 1 นี้จะใช้ Timer 1 เป็นฐานเวลาของการทำงานของ Timer 1ใน Mode 2 (Auto-Reload) โดยสามารถคำนวณได้ดังนี้

Baud Rate = [K * Osc. Freq] / [32*12*(256-(TH1))]

K = 1 เมื่อ SMOD ในรีจิสเตอร์ PCON = 0

K = 2 เมื่อ SMOD ในรีจิสเตอร์ PCON= 1

ส่วนมากแล้ว ผู้ใช้จะทราบค่าของ Baud Rate ที่จะส่งได้ค่านั้น จะได้ค่าของ Time 1 สำหรับ Reload ได้เป็น

TH1 = 256 – [(K * Osc .Freq) / (384 * Baud Rate)]

จากตารางต่อไปนี้ แสดง Baud Rate ต่าง ๆ และค่า Reload ของ Time 1

ตารางที่ 2.3 Baud Rate ต่าง ๆ และค่า Reload ของ Time 1

Baud Rate	f_{osc}	SMOD	Timer1		
			C/T	Mode	Reload Value
Mode 0 Max :1 MHz	12 MHz	X	X	X	X
Mode 2 Max : 375 K	12 MHz	1	X	X	X
Modes 1.3 : 62.5 K	12MHz	1	0	2	FFH
19.2 K	11.059 MHz	1	0	2	FDH
9.6 K	11.059 MHz	0	0	2	FDH
4.8 K	11.059 MHz	0	0	2	FAH
2.4 K	11.059 MHz	0	0	2	F4H
1.2 K	11.059 MHz	0	0	2	E8H
137.5	11.986 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

Mode 2 ความเร็วการรับส่งใน Mode นี้จะเป็นค่าคงที่ซึ่งมี 2 ค่า ขึ้นกับค่า SMOD ในรีจิสเตอร์ PCON ดังนี้

เมื่อ SMOD = 1 Baud Rate = 1/ 32 Osc. Freq

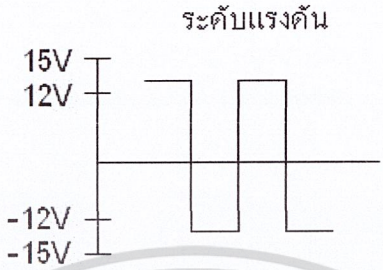
เมื่อ SMOD = 0 Baud Rate = 1/ 64 Osc. Freq

Mode 3 การกำหนดความเร็วการรับส่งใน Mode 3 จะคิดเช่นเดียวกับการคิดใน Mode 1สำหรับค่าเริ่มต้นของรีจิสเตอร์ SFR เมื่อ 8051 ถูกรีเซ็ต จะมีค่าดังแสดงในตารางต่อไปนี้

ตารางที่ 2.4 ค่าเริ่มต้นของรีจิสเตอร์ SFR เมื่อ 8051 ถูกรีเซ็ต

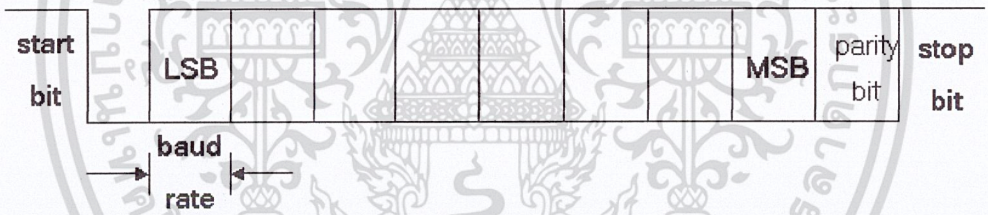
SFR Name	Reset Value
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
IP (8051)	XXX0000B
IP (8052)	XX000000B
IE (8051)	0XX00000B
IE (8052)	0X000000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
TH2 (8052)	00H
TL2 (8052)	00H
RCAP2H (8052)	00H
RCAP2L (8052)	00H
SCON	00H
SBUF	Indeterminate
PCON (HMOS)	0XXXXXXXXB
PCON (CHMOS)	0XXX0000B

2.3 พอร์ตอนุกรม (RS-232)



รูปที่ 2.22 ระดับแรงดันในการเชื่อมต่อ

จากรูปแสดงถึงระดับแรงดันในการเชื่อมต่อ โดยลักษณะสัญญาณในการเชื่อมนั้น จะเป็นแบบ RS-232 ซึ่งมีระดับสัญญาณลอจิก "1" ระดับแรงดัน –3V ถึง –12V และลอจิก "0" ที่แรงดัน 3V ถึง 15V



รูปที่ 2.23 ลักษณะสัญญาณที่ใช้ในการติดต่อผ่านมาตรฐาน RS-232

รูปข้างบนแสดงถึงลักษณะของสัญญาณที่ใช้ในการติดต่อผ่านมาตรฐาน RS-232 เพื่อให้อุปกรณ์ตัวหนึ่งกับอีกตัวหนึ่งสามารถสื่อสารกันได้ จึงต้องกำหนดมาตรฐานขึ้นมา โดยลักษณะของสัญญาณนั้นจะประกอบด้วย 4 ส่วน คือ

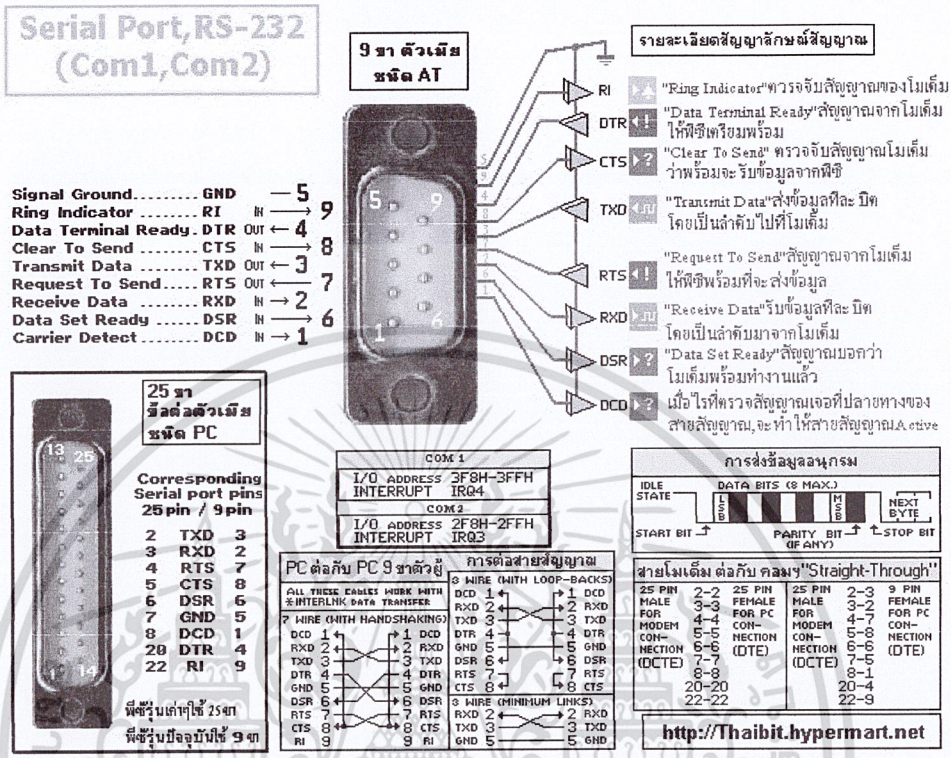
Start bit เป็นบิทสำหรับการเริ่มต้นในการติดต่อสื่อสาร

Data bits เป็นบิทสำหรับเป็นข้อมูล มีข้อมูลทั้งหมด 8 บิท โดยเริ่มต้นด้วยบิทต่ำก่อน

Parity bit เป็นที่บอกจำนวนของบิทข้อมูลที่เป็น "1" ว่าเป็นจำนวนคู่หรือคี่

Stop bit เป็นบิทสำหรับการจบในการติดต่อสื่อสาร

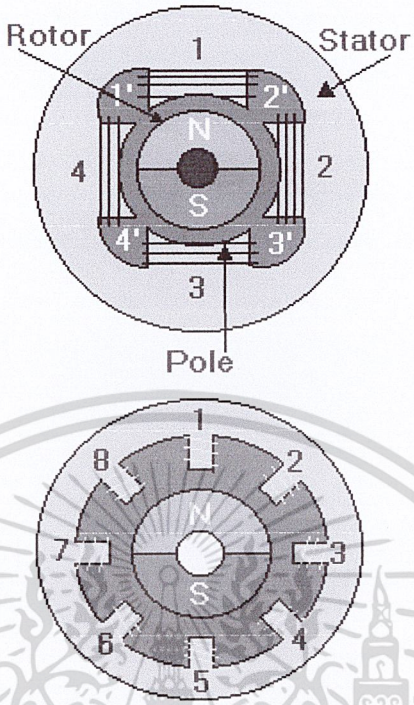
ในแต่ละบิทจะมีการกำหนดความถี่ขึ้นมา เพื่อให้สามารถสื่อสารกันได้อย่างถูกต้อง โดยเราจะกำหนดให้ความกว้างของแต่ละบิทมีความถี่ค่าหนึ่ง เรียกว่า Baud Rate หรืออัตราในการส่งข้อมูล เช่น Baud Rate 9600 หมายความว่าใช้ความถี่ในการติดต่อสื่อสารที่ 9600 Hz หรือสามารถส่งข้อมูลได้สูงสุด 9600 Bit/Sec นั้นเอง



รูปที่ 2.24 การต่อใช้งานพอร์ตอนุกรม RS-232

2.4 สเต็ปมอเตอร์

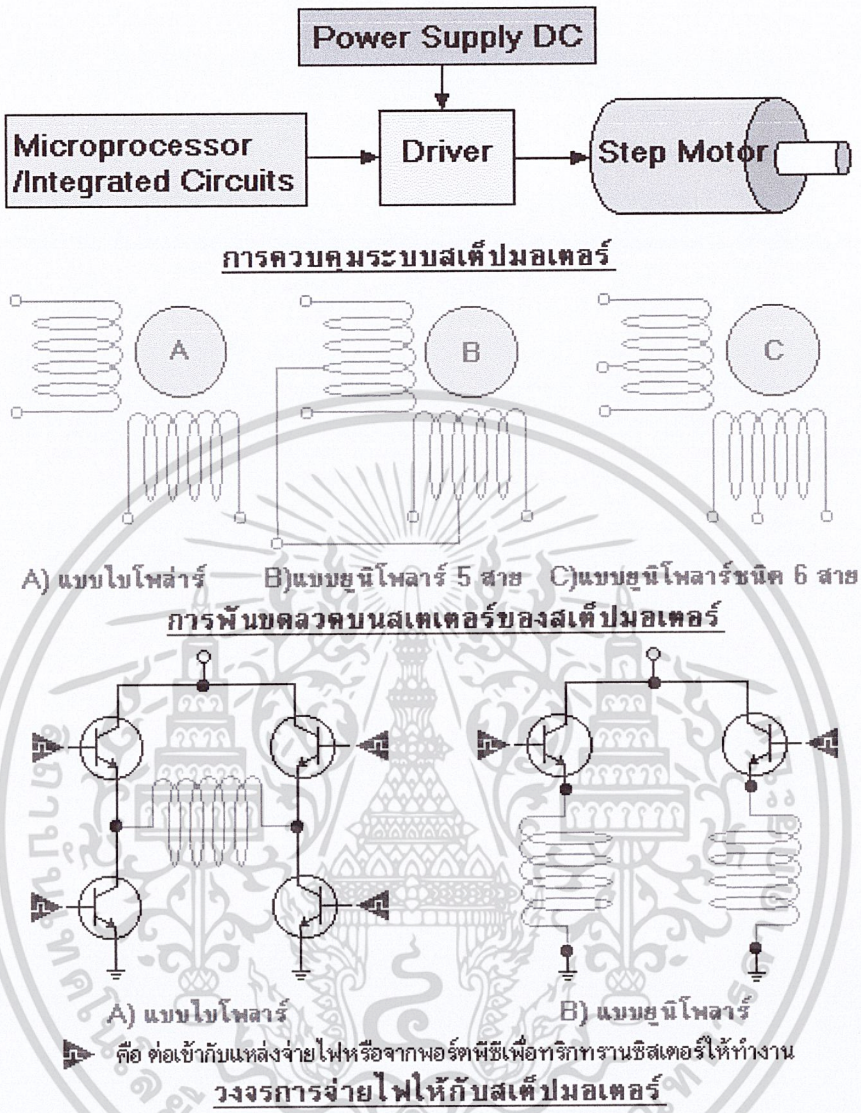
สเต็ปมอเตอร์เป็นมอเตอร์ที่มีลักษณะเมื่อเราป้อนไฟฟ้าให้กับมอเตอร์ทำให้หมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งต่าง จากมอเตอร์ ทั่วไปที่จะหมุนทันทีและตลอดเวลาเมื่อป้อนแรงดันไฟฟ้าข้อดีของสเต็ปมอเตอร์ สามารถกำหนด ตำแหน่งของการหมุนด้วยตัวเลข (องศาหรือระยะทาง) ได้อย่างละเอียดโดย ใช้คอมพิวเตอร์หรือ ไมโครคอนโทรลเลอร์เป็น เครื่องกำหนดและจัดเก็บตัวเลข โครงสร้างของขั้วแม่เหล็กบนสเตเตอร์ทำมาจากแผ่นเหล็กวงแหวนที่มีซี่ยื่นออกมาประกอบกันเป็นชั้น ๆ โดยที่แต่ละซี่นั้นจะมีคอยล์(ขดลวด)พันสวมอยู่ เมื่อมีการป้อนกระแสผ่านคอยล์ทำให้เกิดสนามแม่ เหล็กไฟฟ้า(Electromagnetic) ดูดังรูปที่ 2.14



รูปที่ 2.25 โครงสร้างภายในสเต็ปมอเตอร์

ในที่นี้ซึ่งถ้าเราเพิ่มจำนวนของขั้วแม่เหล็กมากขึ้นจะเพิ่มจำนวนของสเต็ปต่อวงจรรอบมากขึ้นตาม ด้วยลองดูตามรูปด้านบน

ลักษณะการนำไปใช้งาน สเต็ปมอเตอร์ ใช้งานลักษณะ Open Loop System แปลเป็นภาษาไทย ระบบเปิด คือ สเต็ปมอเตอร์สามารถทำงานได้โดยไม่ต้องมีการ ป้อนค่าพารามิเตอร์กลับมา (Feed back) แต่ทุกวิธีที่ต้องการกำหนดตำแหน่งที่แน่นอนนั้นล่ะ จะต้องการป้อนกลับไปยังระบบและตัวบอก ตำแหน่งว่าถูกต้องหรือผิดพลาดให้รับทราบ ดังเช่นวิธีที่ใช้กับสเต็ปมอเตอร์ คือ เรานำลิมิทสวิตช์ ติดตามตำแหน่งที่จะตรวจจับ เมื่อสเต็ปมอเตอร์ เริ่มหมุนแล้วหมุนไปจนถึงตำแหน่งของสวิตช์ตรวจจับสัญญาณ สวิตช์ทำงานก็จะป้อนกลับไปสู่ระบบ ซึ่งก็จะทำให้รู้การทำงานของสเต็ปมอเตอร์ตลอด ตัววงจรไมโครคอนโทรลเลอร์เองจะมีจุดอ้างอิง ไว้ให้เริ่มต้นการทำงานและอ้างอิงตำแหน่งได้ถูกต้อง



รูปที่ 2.26 วงจรการจ่ายไฟให้กับสเต็ปมอเตอร์

โดยแนวทางสเต็ปมอเตอร์เป็นอุปกรณ์จำพวกเชิงกลทางไฟฟ้า โดยมีกรุ๊ปของไบนารี่โวลท์เตทเป็นอินพุตและการเคลื่อนที่ แบบเชิงมุมเป็นเอาท์พุต หรือว่าหมุนทีละสเต็ปซึ่งอยู่ระหว่าง 0.1 - 30 องศา อยู่ที่โครงสร้างของสเต็ปมอเตอร์ โดยตามสัญญาณ พัลซ์ที่จ่ายให้กับขดสเตเตอร์ทำให้เกิดแรงผลักแก่โรเตอร์หมุนไป สเต็ปมอเตอร์มีขดลวดหลายชุดในที่นี้เราเรียกว่า Phase (เฟส) ดังนั้นสัญญาณที่ต่อเนื่องเป็น ซีเควน (Sequence) ลักษณะของไบนารี่ (Binary) ซึ่งจะต้องไปผ่านวงจร ไดรเวอร์ (Driver) ก็จะ ทำให้โรเตอร์หมุนไปอย่างต่อเนื่อง ที่กล่าวมาสามารถดูได้จากรูปด้านบน

การพันขดลวดบนสเตเตอร์ของสเต็ปมอเตอร์ จะเห็นว่าการพันมีด้วยกัน 2 วิธี คือ

แบบ Bipolar

จะมีการพันขดลวดหนึ่งขด(จะกี่รอบก็แล้วแต่ สเป็กใช้งาน)ในแต่ขั้วแม่เหล็กของสเตเตอร์ โดยขั้วแม่เหล็กที่เกิดขึ้น ที่สเตเตอร์จะถูกกำหนดโดยทิศทางของการไหลของกระแสไฟฟ้า ซึ่งสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงกันข้ามได้เพียง การกลับทิศทางของการไหลในกระแสไฟฟ้า โดยมาจากการควบคุมของวงจรสวิทชิ่งให้กลับขั้วไฟฟ้า

แบบ Unipolar

แบบนี้มี 2 ขด บนแต่ละขั้วแม่เหล็กของสเตเตอร์ ทำให้แต่ละขดลวดเกิดขั้วแม่เหล็กในทิศทางตรงกันข้าม เช่นกัน การกลับทิศทางขั้วแม่เหล็กทำได้โดยใช้วงจรสวิทชิ่งให้สลับหนึ่งไปยังอีกขั้วหนึ่งแทนกัน พื้นฐานการสวิทชิ่งดูรูปด้านบนที่ชื่อวงจรการจ่ายไฟให้กับสเต็ปมอเตอร์ การพันขดลวดทั้ง 2 แบบที่กล่าวมา มันต่างกันที่ แบบยูนิโพล่าร์จะทำให้เกิดแรงบิดน้อยกว่าแบบไบโพล่าร์ แล้วก็จะต้องมีคำถามตามมาอีกว่าแล้วถ้าไปซื้อหรือหามาใช้งานจะรู้ได้จากตรงไหน ก็สังเกตุจาก สายไฟที่ต่อมาจากตัวสเต็ปมอเตอร์ซึ่งแบบไบโพลาร์จะมี 4 สาย ส่วนเป็นแบบยูนิโพล่าร์จะมี 5 สายหรือ 6 สายการสั่งงานควบคุม การหมุนของสเต็ปมอเตอร์

การควบคุมและสั่งงานให้สเต็ปมอเตอร์ทำงาน ไปทีละสเต็ปสามารถทำได้โดยการจ่ายกำลังไฟไปยังขดลวด ในแต่ละขอบนสเตเตอร์ โดยการป้อนจะทำในลักษณะเป็นลำดับหรือเรียกว่าซีเควนเวียลในลูปที่ถูกต้อง ซึ่งจะแบบ ได้เป็น 3 รูปแแบ คือ แบบเวฟ(wave) แบบ 2 เฟส(2 phase) และแบบครึ่งสเต็ป (half step) ทั้ง 3 แบบนี้ก็จะมีข้อดีและข้อเสียต่างกันออกไปแบบเวฟ(wave)

จะเป็นการกระตุ้นแบบที่ง่ายที่สุด ซึ่งจะทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่ง ๆ เรียงกันไป ตัวอย่างเช่น ขดที่ 1 , 2 , 3 , 4 ,1, 2 , 3 , 4 เป็นลำดับอย่างนี้ หรือ ขด 1 , 4 , 3 , 2 , 1 , 4 , 3 , 2 เป็นลำดับกันไป ทั้งนี้ขึ้นอยู่กับทิศทางที่เราต้องให้มอเตอร์หมุนไป วงจรที่นำมากระตุ้นนั้นจะมีราคาค่อนข้างจะถูกกว่าและง่ายกว่า ดังในรูปของวงจรการจ่ายไฟ ที่อยู่ด้านบนนั้นเราสามารถเขียนขั้นตอนการทำงานเป็นตารางออกมาได้ดังนี้

ตารางที่ 2.5 ขั้นตอนการทำงานของมอเตอร์แบบ 1 เฟส

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2		ON		
3			ON	
4				ON
5	ON			
6		ON		

แบบ 2 เฟส(2 Phase)

แบบนี้ก็จะคล้ายกับการกระตุ้นในแบบเวฟแต่จะต่างกันตรงที่ แบบ 2 เฟส จะกระตุ้นทีละ 2 ขด ที่อยู่ใกล้กันใน เวลาเดียวกัน และจะเรียงลำดับกันไป ดังเช่นแบบเดียวกับแบบเวฟ จะยกตัวอย่างการกระตุ้นขดลวดในลักษณะ ซีเควนให้ดังนี้ 12,23,34.41,12,23,34,41 เรียงลำดับกันไปเรื่อย ๆ หรือจะเป็น 14, 43, 32, 21, 14, 43, 32, 21 เรียงกันไปเรื่อย ๆ เช่นกัน ถ้าจะมากล่าวถึงข้อดีข้อเสียของแบบ 2 เฟส แล้วมีดังนี้

ข้อดี การที่เราจะเพิ่มจำนวนขดลวดที่ถูกกระตุ้นจะทำให้แรงบิดได้มากกว่า แบบเวฟ ซึ่งโรเตอร์จะหมุนด้วยแรง ดึงแบเต็ม ๆ แรงจาก ทั้ง 2 ขดลวดที่กระตุ้นพร้อมกัน

ข้อเสีย แบบ 2 เฟส จะกระตุ้นขดลวดนั้นต้องใช้กำลังไฟมากขึ้นเป็น 2 เท่าของแบบเวฟ ก็เป็นไปตามธรรมชาติ ได้อย่างก็ต้องเสียอย่างเราสามารถเขียนลำดับการกระตุ้นของขดลวดแบบ 2 เฟส ได้ดังในตารางต่อไปนี้

ตารางที่ 2.6 ขั้นตอนการทำงานของมอเตอร์แบบ 2 เฟส

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON	ON		
2		ON	ON	
3			ON	ON
4	ON			ON
5	ON	ON		
6		ON	ON	

แบบครึ่งสเต็ป (half-step)

แบบนี้แบบรูปแบบผสมผสานของการกระตุ้นระหว่าง แบบเวฟ กับ แบบ 2 เฟส เพื่อให้จำนวนรอบของสเต็ปให้ มากขึ้นเป็น 2 เท่าซึ่งในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเรื่อย ๆ เป็นลำดับ ดังจะยกตัวอย่างต่อไปนี้ 1, 12, 2, 23, 3, 34, 4, 41, 1, 12, 2, 23, 3, 34, 4, 41, 1 เป็นลำดับอยู่อย่างนี้เรื่อยไป ถ้าเราจะกลับทิศทางการหมุนก็จะได้เป็นดังนี้ 1, 41, 4, 43, 3, 32, 2, 21, 1, 41, 4, 43, 3, 32, 2, 21, 1 เป็นลำดับ กันไปเราจะมาพูดกันถึงข้อดีและข้อเสียของการกระตุ้นแบบครึ่งสเต็ปกัน

ข้อดี การกระตุ้นแบบนี้จะให้แรงบิดที่เพิ่มมากขึ้น เนื่องจากช่วงสเต็ปที่มีระยะสั้นลงอีกประการหนึ่งแต่ละ สเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกันเป็นผลให้ค่าตำแหน่งความถูกต้องมากขึ้นไปด้วย

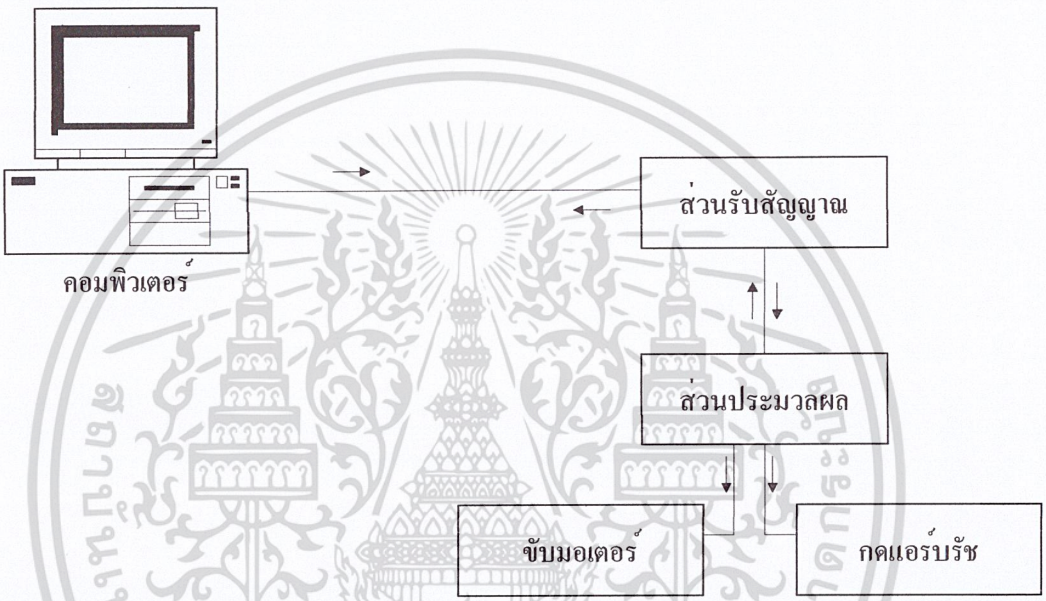
ข้อเสีย ก็คงจะเช่นเดียวกับแบบ 2 เฟส ที่ต้องจ่ายกำลังไฟเป็น 2 เท่าของแบบเวฟหรือจะใช้เท่ากับแบบ 2เฟส นั้นเอง ดังนั้นเราสามารถนำลำดับการทำงานของ แบบครึ่งเฟส ในรูปของตารางได้ดังนี้

ตารางที่ 2.7 ขั้นตอนการทำงานของมอเตอร์แบบครึ่ง เฟส

Step No.	Phase 1	Phase 2	Phase 3	Phase 4
1	ON			
2	ON	ON		
3		ON		
4		ON	ON	
5			ON	
6			ON	ON
7				ON
8	ON			ON
9	ON			
10	ON	ON		

บทที่ 3
หลักการทำงานและการออกแบบ

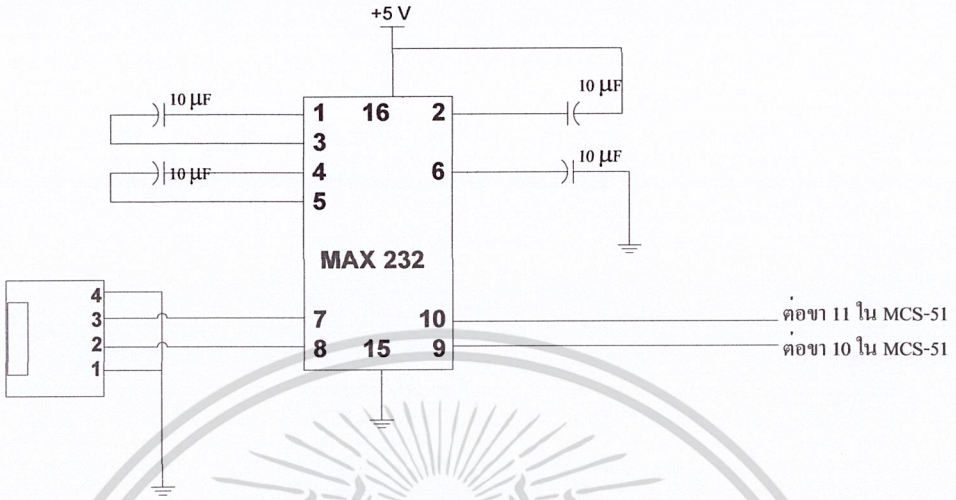
3.1 การออกแบบการทำงานโดยรวม



รูปที่ 3.1 Block Diagram การทำงานโดยรวม

จากรูปที่ 3.1 แสดงให้เห็นถึงหลักการทำงานของระบบ ซึ่งมีหลักการทำงานโดยรวมคือ คอมพิวเตอร์ทำหน้าที่แปลงภาพเป็นข้อมูลตัวเลข และทำการส่งข้อมูลทีละ 8 บิต โดยใช้โปรแกรมวิชวลเบสิก ไปที่หุ่นยนต์โดยผ่านทางพอร์ตอนุกรม ผ่านวงจรแปลงแรงดันไฟส่งไปประมวลผลที่ไมโครคอนโทรเลอร์ เมื่อหุ่นยนต์ได้รับข้อมูลแล้วจะนำข้อมูลที่ได้มาประมวลผลแล้ว จะส่งสัญญาณควบคุมไปสั่งให้หุ่นยนต์เคลื่อนที่และพ่นสีตามที่ประมวลผลได้จากไมโครคอนโทรลเลอร์ เมื่อทำงานเสร็จ แล้วจะทำการส่งสัญญาณกลับไปที่คอมพิวเตอร์ เพื่อร้องขอให้ส่งข้อมูลชุดต่อไปมาทำการประมวลผลต่อจนจบกระบวนการ

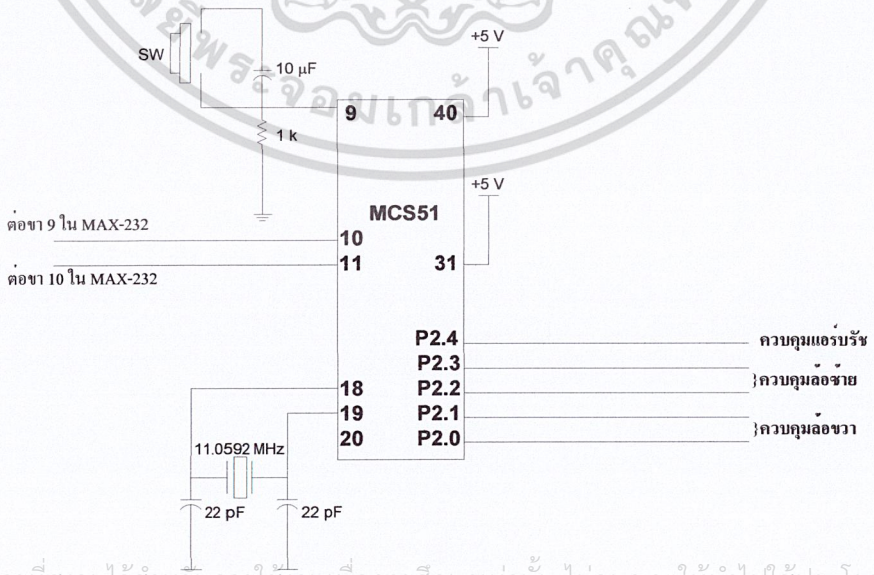
3.1.1 ส่วนรับสัญญาณ



รูปที่ 3.2 วงจรแปลงแรงดันไฟฟ้า

วงจรที่ใช้ในการรับสัญญาณจากพอร์ตอนุกรมของคอมพิวเตอร์ใช้ ใช้ไอซี MAX 232 มาต่อเป็นวงจรสำหรับแปลงแรงดันให้สามารถใช้กับไมโครคอนโทรลเลอร์ได้ เพราะความต่างศักย์ที่ออกมาจากพอร์ตอนุกรมของคอมพิวเตอร์จะมีความต่างศักย์ -15 ถึง +15 โวลต์ ซึ่งไมโครคอนโทรลเลอร์ไม่สามารถใช้งานที่ความต่างศักย์นี้ได้ ดังนั้นวงจรนี้จึงทำหน้าที่แปลงสัญญาณให้เป็น 0 ถึง 5 โวลต์ ซึ่งสามารถใช้ในไมโครคอนโทรลเลอร์ได้

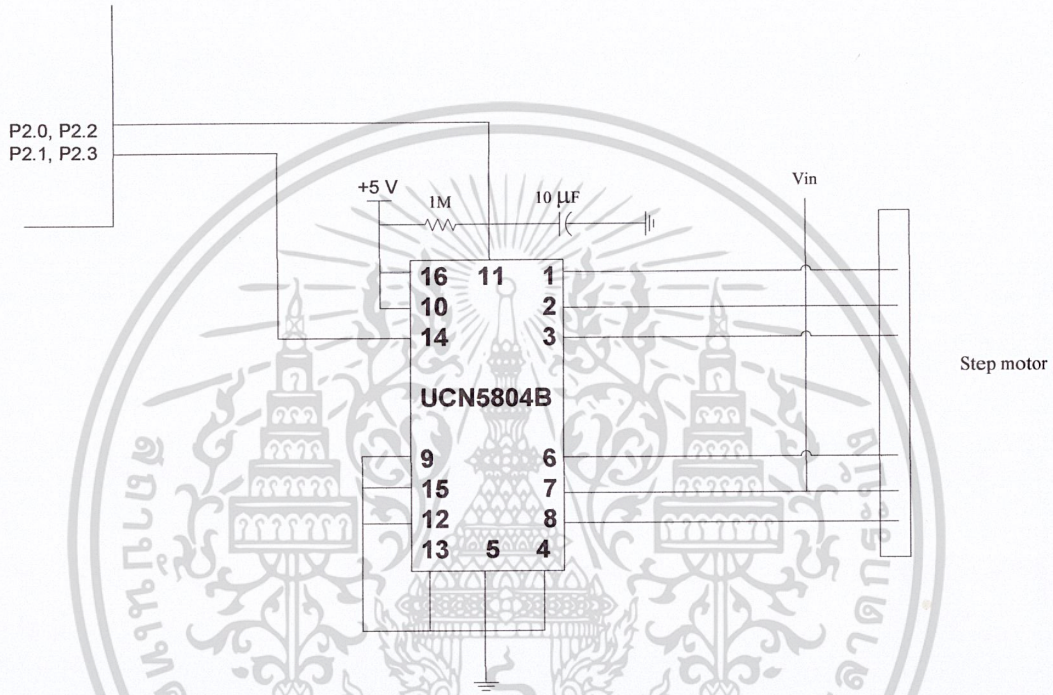
3.1.2 ส่วนประมวลผลสัญญาณ



รูปที่ 3.3 วงจร MCS-51

วงจรนี้ไว้ใช้สำหรับประมวลผลข้อมูลโดยรับข้อมูลเข้าทางขา 10 แล้วทำการประมวลผลในไมโครคอนโทรลเลอร์แล้วส่งสัญญาณควบคุมออกทางขา P2.0 และ P2.2 ควบคุมการหมุนของล้อ, P2.1 และ P2.3 ควบคุมทิศทางการหมุนของล้อ และ P2.4 ควบคุมแอร์บรัช

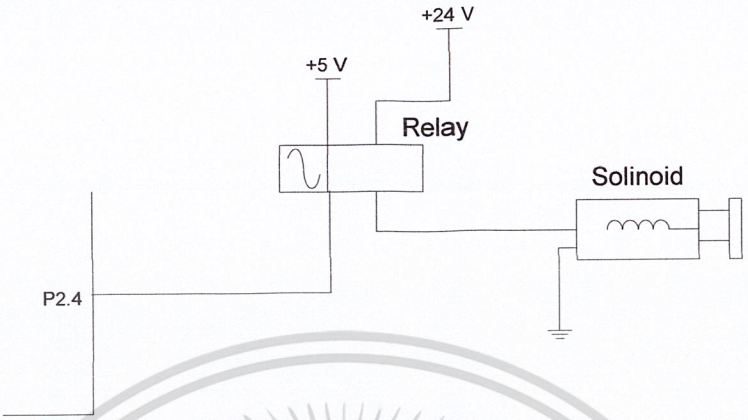
3.1.3 ส่วนขับมอเตอร์



รูปที่ 3.4 วงจรขับสเต็ปมอเตอร์

วงจรขับสเต็ปมอเตอร์ ใช้ไอซี UCN5804B ซึ่งเป็นไอซีขับมอเตอร์ จะทำงานเมื่อได้รับโดยการพัลส์สัญญาณ ซึ่งสามารถกำหนดได้ว่าจะให้หมุนซ้ายหรือขวาก็ได้โดยจ่ายไฟ 5 โวลต์ เมื่อต้องการให้มอเตอร์หมุนซ้าย และจ่ายไฟ 0 โวลต์ เมื่อต้องการให้มอเตอร์หมุนขวา

3.1.4 ส่วนควบคุมแอร์บรัช



รูปที่ 3.5 วงจรกดแอร์บรัช

วงจรควบคุมแอร์บรัชนั้นใช้โซลินอยด์ในการกดแอร์บรัช และใช้รีเลย์ทำหน้าที่เป็นสวิตซ์ไฟฟ้าทำการจ่ายไฟให้โซลินอยด์ เมื่อจ่ายไฟให้กับโซลินอยด์แล้วขดลวดภายในจะทำการเหนี่ยวนำทำให้แท่งเหล็กภายในเกิดการเคลื่อนที่ ไปกดแอร์บรัชให้ทำงาน

3.2 ส่วนของซอฟต์แวร์

ใช้โปรแกรมวิชวลเบสิก ในการเขียนโปรแกรมการแปลงภาพเป็นข้อมูลตัวเลข ซึ่งมีขั้นตอนการเขียนโปรแกรมดังนี้

3.2.1 การแปลงภาพเป็นข้อมูลตัวเลข

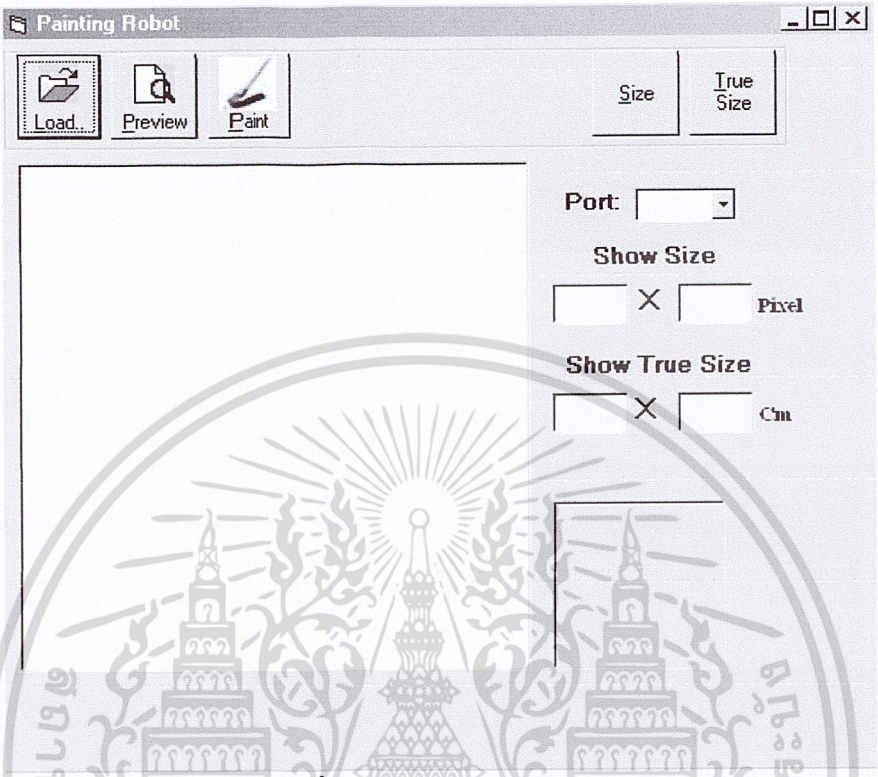
ข้อมูลตัวเลขที่จะนำไปใช้นั้นเป็นตัวเลขที่จะนำไปประมวลผลที่ไมโครคอนโทรลเลอร์เลย ดังนั้นจึงต้องกำหนดข้อมูลเพื่อที่จะสามารถนำไปใช้ได้เลย

ซึ่งข้อมูลตัวเลขจะแบ่งเป็น 3 ส่วนดังนี้

- ทิศทางของรถ เป็นการกำหนดการเคลื่อนที่ของรถโดยมี 2 ทิศทาง คือ
  - เคลื่อนที่ไปทางซ้าย ให้มีค่าเป็น 255
  - เคลื่อนที่ไปทางขวา ให้มีค่าเป็น 0
- สี หุ่นยนต์จะพ่นสีออกมาเป็นสีขาวดำเท่านั้น ดังนั้นจึงมีสีเพียง 2 สี คือ
  - สีดำ ให้แทนด้วยเลข 253
  - สีขาว ให้แทนด้วยเลข 254
- ระยะทาง จะแทนจำนวนของจุดที่มีสีเดียวกันที่อยู่ติดกันว่ามีเป็นจำนวนกี่จุด มีค่าตั้งแต่ 1 เป็นต้นไป

ดังนั้น ให้การเคลื่อนที่ครั้งหนึ่งๆ หุ่นยนต์ต้องการข้อมูล 3 อย่างประกอบกันในการประมวลผล

3.2.2 ส่วนการติดต่อและแสดงผลทางคอมพิวเตอร์

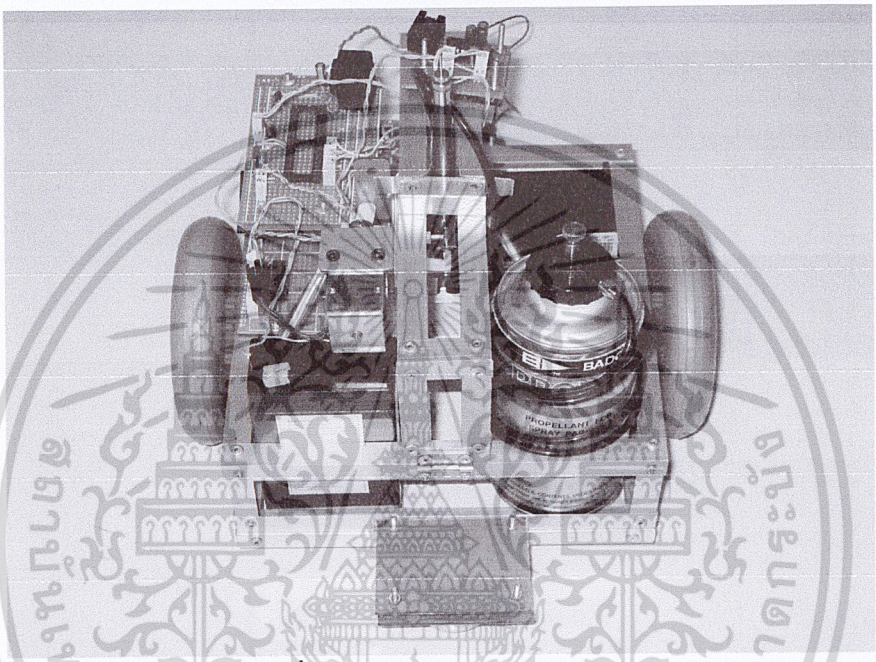


รูปที่ 3.6 หน้าจออินเตอร์เฟส

ในส่วนหน้าจออินเตอร์เฟส นั้นเป็นการนำภาพหรือตัวอักษรที่ต้องการวาดมาแสดงให้ดู และสามารถสั่งงานให้หุ่นยนต์วาดภาพได้ ผ่านทางหน้าจออินเตอร์เฟสนี้ โดยการทำงานของโปรแกรมเริ่มด้วย ผู้ใช้ทำการโหลดภาพที่เป็นไฟล์ บิตแมพ (.BMP) มาดูที่หน้าจออินเตอร์เฟส ก่อนที่จะสั่งให้หุ่นยนต์วาดภาพ ผู้ใช้จะสามารถดูภาพพรีวิวก่อนได้ โดยที่จะสามารถทราบความสูง ความกว้างของรูปภาพที่โปรแกรมคำนวณได้แสดงอยู่ที่หน้าจอด้วย

3.3 ออกแบบตัวรถ

การขับเคลื่อนของหุ่นยนต์จะใช้สเต็ปมอเตอร์ 2 ตัวเป็นตัวขับล้อทั้ง 2 ล้อ เพื่อขับเคลื่อนหุ่นยนต์ ในส่วนแอร์บรัชจะวางไว้กลางลำตัวรถ เพื่อตัวรถจะได้มีขนาดกระทัดรัด ขนาดของตัวถังมีขนาด 8 x 8 นิ้ว มีแหล่งจ่ายไฟขนาด 24 โวลต์ จ่ายไฟเลี้ยงโซลินอยด์ ไฟขนาด 6 โวลต์ จ่ายไฟเลี้ยงมอเตอร์ และไฟขนาด 9โวลต์ 2 ชุด จ่ายไฟเลี้ยงวงจรและรีเลย์



รูปที่ 3.7 โครงสร้างของหุ่นยนต์

บทที่ 4

ผลการทดลอง

4.1 ทดลองการเคลื่อนที่ของหุ่นยนต์

4.1.1 เคลื่อนที่เป็นเส้นตรง

ทดลองโดยการสั่งงานหุ่นยนต์ผ่านทางคอมพิวเตอร์ให้เดินเป็นเส้นตรงระยะทาง 10 หน่วย โดยให้ 1 หน่วยเท่ากับ 1.8 เซนติเมตร

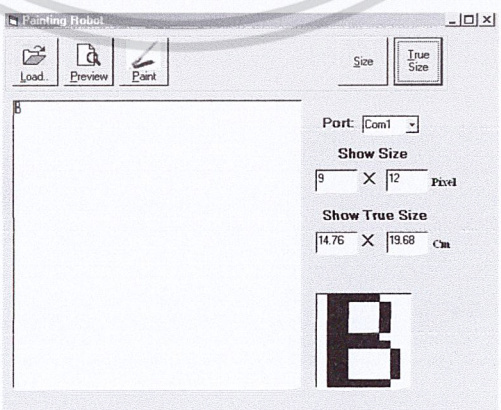
สังเกตระยะทางและทิศทางว่าตรงตามความต้องการหรือไม่

ตารางที่ 4.1 ผลการทดลองการเคลื่อนที่ของหุ่นยนต์

ครั้งที่ 1	ผลการทดลอง
1	เคลื่อนที่ได้ตามแนวเส้นตรง ออกนอกทิศทางเล็กน้อย เคลื่อนที่ได้ 16 เซนติเมตร
2	รถไม่เคลื่อนที่ แต่เกิดแรงสั่นที่ล้อทั้งสองข้าง
3	เคลื่อนที่ตรงทิศทาง เคลื่อนที่ได้ 16 เซนติเมตร
4	เคลื่อนที่ตรงทิศทาง เคลื่อนที่ได้ 17 เซนติเมตร
5	เคลื่อนที่ออกนอกทิศทางเล็กน้อยเคลื่อนที่ได้ 15.5 เซนติเมตร

4.1.2 การเขียนตัวอักษร

ทดลองโดยการสั่งงานให้หุ่นยนต์เขียนเป็นตัวอักษร “B” ดังรูป



รูปที่ 4.1 รูปที่สั่งพิมพ์

ผลการทดลองที่ได้เป็นดังรูป

รูปที่ 4.2 ภาพที่หุ่นยนต์ทำการวาด

สรุปผลการทดลอง

จากการทดลองผลที่ได้เป็นไปตามรูปที่สั่งให้วาด แต่มีความผิดพลาดเล็กน้อยคือตัวอักษรในแนวตั้งที่วาดออกมาจะเบนไปทางซ้ายซึ่งเกิดจากการทำงานที่ไม่เที่ยงตรงของฮาร์ดแวร์ แต่สามารถแก้ได้โดยการปรับแก้ในส่วนของโปรแกรมในไมโครคอนโทรลเลอร์ได้

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

5.1 สรุปผลการดำเนินการโครงงาน

การทำหุ่นยนต์วาดภาพ นั้นจะต้องใช้ความรู้ในหลายด้าน ไม่ว่าจะเป็นการศึกษาโปรแกรมวิชวลเบสิก การศึกษาโปรแกรมแอสเซมบลี การควบคุมมอเตอร์ การออกแบบโครงสร้างของรถ

จากการทำโครงงานนี้ เมื่อทำการทดลองจะเห็นว่าได้ผลเป็นที่น่าพอใจ และการทำงานของหุ่นยนต์ ที่ประมวลผลและสั่งงานอุปกรณ์ต่ างๆให้ทำงานตามที่สั่งงานได้

5.2 ปัญหาที่พบในระหว่างการดำเนินการโครงงาน

1. ความรู้ที่จะนำมาใช้ในโครงงาน เนื่องจากโครงงานเน้นหนักไปทางด้าน ฮาร์ดแวร์ ทำให้ประสบปัญหาค่อยข้างเยอะ เนื่องจากไม่มีความรู้ทางด้านนี้เพียงพอ

2. การหาอุปกรณ์ มีปัญหาในการรวบรวม ค้นหาอุปกรณ์ ที่จะนำมาใช้กับหุ่นยนต์รวมถึงการออกแบบ ตัวรถ จึงทำให้เกิดความล่าช้าอย่างมากกว่าจะรวบรวมอุปกรณ์ได้ครบ

3. วงจรอิเล็กทรอนิกส์เกิดปัญหาบ่อยครั้ง ทำให้การดำเนินโครงงานไม่เป็นไปตามที่วางไว้

4. งบประมาณ เนื่องจากอุปรณ์ที่ใช้ในการประกอบเป็นตัวรถมีราคาสูงมาก ทำให้มีปัญหาเรื่องงบประมาณมากพอสมควร

5. การทำงานที่ผิดพลาดของหุ่นยนต์ที่สร้าง เนื่องจากสิ่งแวดล้อมมีผลต่อการทำงานของหุ่นยนต์เป็นอย่างมาก ทำให้หุ่นยนต์ทำงานผิดพลาดบ่อยครั้ง

5.3 แนวทางในการพัฒนาโครงงานต่อ

1. ควรปรับปรุงการส่งข้อมูลให้เป็นในรูปแบบอื่น เนื่องจากมีเทคโนโลยีอื่นๆอีกมากที่มีวิธีการส่งข้อมูลที่หลากหลาย ที่สามารถลดข้อจำกัดในการส่งแบบใช้สายได้เนื่องจากการใช้สายส่งมีข้อจำกัดเรื่องระยะทาง

2. ควรพัฒนาอัลกอริทึ่ม ให้การสั่งงานหุ่นยนต์ให้มีประสิทธิภาพมากขึ้น

3. นำไปปรับใช้กับงานจริง

บรรณานุกรม

1. สัจจะ จรัสรุ่งรวีวร, “คู่มือการสร้างแอพพลิเคชันด้วย Visual Basic 6.0 ฉบับสมบูรณ์ สำนักพิมพ์ อินโฟเพรส”, 2542
2. กิตติ ภักดีวัฒนะกุล, จำลอง ครูอุตสาหะ, “Visual Basic 6 ฉบับโปรแกรมเมอร์”, สำนักพิมพ์เคทีพี คอมพ์ แอนด์ คอนซัลท์, พิมพ์ครั้งที่ 9, 2544
3. วันสุระ ศรีใสดี, “ประยุกต์/อินเตอร์เฟส ไมโครคอนโทรลเลอร์ MCS-51 ภาคปฏิบัติ”, สำนักพิมพ์ดวงกมล, 2542
4. วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล, “เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช”, อินโนเวตีฟ เอ็กเพอริเมนต์ บจ.

ภาคผนวก ก

โปรแกรมวิชวลเบสิก

```
Dim l As Integer
Dim color, str, direction As String
Dim k, y, x As Integer
Dim l_color As Variant

Private Sub Combo1_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
End If
Select Case Combo1.Text
Case "Com1"
   MSComm1.CommPort = 1
   MSComm1.PortOpen = True

Case "Com2"
   MSComm1.CommPort = 2
   MSComm1.PortOpen = True
End Select
End Sub

Private Sub Command1_Click()
CommonDialog1.Filter = "Bitmap Files|*.bmp"
CommonDialog1.ShowOpen
Picture1.Picture = LoadPicture(CommonDialog1.FileName)
Image1.Picture = LoadPicture(CommonDialog1.FileName)
Image2.Picture = LoadPicture(CommonDialog1.FileName)
End Sub

Private Sub Command2_Click()
For Y1 = Picture1.ScaleTop To Picture1.ScaleTop + Image1.Height - 1
   For X1 = Picture1.ScaleLeft To Picture1.ScaleLeft + Image1.Width - 1
   If Picture1.Point(X1, Y1) <> RGB(255, 255, 255) Then
      Picture1.PSet (X1, Y1)
      End If
   Next X1
   Next Y1
End Sub

Private Sub Command3_Click()
Text1.Text = Image1.Width * 1.64
Text2.Text = Image1.Height * 1.64
End Sub

Private Sub Command4_Click()
Text5 = Image1.Width
```

```
Text6 = Image1.Height
End Sub

Private Sub Command5_Click()
y = Picture1.ScaleTop
x = Picture1.ScaleLeft
direction = "right"
If Picture1.Point(x, y) = RGB(255, 255, 255) Then
color = "white"
Else: color = "black"
End If
l_color = Picture1.Point(x, y)
l = 1
x = x + 1
z = 0
While l_color = Picture1.Point(x, y) And z < Image1.Width - 1
   If Picture1.Point(x, y) = RGB(255, 255, 255) Then
   color = "white"
   Else: color = "black"
   End If
   l = l + 1
   x = x + 1
   z = z + 1
Wend
If z = Image1.Width - 1 Then
y = y + 1
End If
l_color = Picture1.Point(x, y)
x = x + 1
str = Chr(10)
Text8.Text = str

End Sub

Private Sub Form_Load()
MSComm1.Settings = "57600,N,8,1"
MSComm1.RThreshold = 1
MSComm1.InputLen = 0

End Sub

Private Sub MSComm1_OnComm()
str = MSComm1.Input
Text8.Text = str

End Sub
```

```
Private Sub Text8_Change()
If l Mod 252 = 0 And l \ 252 = k Then
str = Chr(13)
k = 0
End If
If str = Chr(10) Then
   If direction = "right" Then
   MSComm1.Output = Chr(0)
   Else
   MSComm1.Output = Chr(255)
   End If
End If
If str = Chr(11) Then
   If color = "black" Then
   MSComm1.Output = Chr(253)
   Else: MSComm1.Output = Chr(254)
   End If
End If
If str = Chr(12) Then
   If l \ 252 <> 0 Then
      k = k + 1
      While l \ 252 >= k
      MSComm1.Output = Chr(252)
      Exit Sub

      Wend
      MSComm1.Output = Chr(l Mod 252)
      k = 0
   Else
      MSComm1.Output = Chr(l Mod 252)
   End If
End If

If str = Chr(13) Then
l = 1

While y <= Image1.Height + Picture1.ScaleTop - 1
If y Mod 2 = 0 Then

While x <= Picture1.ScaleLeft + Image1.Width - 1

   If x = Picture1.ScaleLeft Then
      If Picture1.Point(x, y) = RGB(255, 255, 255) Then
        color = "white"
        Else: color = "black"
      End If
      l_color = Picture1.Point(x, y)
```

```
      l = 1
      x = x + 1
    Else
      If l_color = Picture1.Point(x, y) Then
        If Picture1.Point(x, y) = RGB(255, 255, 255) Then
          color = "white"
          Else: color = "black"
        End If
        l = l + 1
        x = x + 1
      Else
        If Picture1.Point(x - 1, y) = RGB(255, 255, 255) Then
          color = "white"
          Else: color = "black"
        End If
        l_color = Picture1.Point(x, y)
        x = x + 1

        direction = "right"
        str = Chr(10)
        Text8.Text = str
        Exit Sub
        'GoTo nong:
      End If
    End If
Wend
      If Picture1.Point(x - 1, y) = RGB(255, 255, 255) Then
        color = "white"
        Else: color = "black"
      End If
y = y + 1
x = x - 1

      direction = "right"
      str = Chr(10)
      Text8.Text = str
      Exit Sub

Else

While x >= Picture1.ScaleLeft

    If x = Image1.Width + Picture1.ScaleLeft - 1 Then
      If Picture1.Point(x, y) = RGB(255, 255, 255) Then
        color = "white"
        Else: color = "black"
      End If
```

```
      l_color = Picture1.Point(x, y)
      l = 1
      x = x - 1
    Else
      If l_color = Picture1.Point(x, y) Then
        If Picture1.Point(x, y) = RGB(255, 255, 255) Then
          color = "white"
          Else: color = "black"
        End If
        l = l + 1
        x = x - 1
      Else
      If Picture1.Point(x + 1, y) = RGB(255, 255, 255) Then
        color = "white"
        Else: color = "black"
      End If
        l_color = Picture1.Point(x, y)
        x = x - 1

        direction = "left"
        str = Chr(10)
        Text8.Text = str
        Exit Sub
      End If
    End If
Wend
      If Picture1.Point(x + 1, y) = RGB(255, 255, 255) Then
        color = "white"
        Else: color = "black"
      End If
y = y + 1
x = x + 1
      direction = "left"
      str = Chr(10)
      Text8.Text = str
      Exit Sub

End If

Wend

End If

End Sub
```

โปรแกรมแอสเซมบลี

```
        ORG 0000H

        MOV    TMOD,#20H
        MOV    SCON,#50H
        MOV    PCON,#80H
        MOV    TH1,#0FFH
        MOV    TL1,#0FFH
        CLR    TF1
        SETB   TR1
        MOV    R0,#00H
        MOV    R1,#00H

MAIN:       CALL   RECEIVE

VALUE_0:    CJNE   A,#00H,VALUE_255
        MOV    R1,A
        CLR    TI
        MOV    SBUF,#0BH
        JNB    TI,$
        JMP    MAIN
VALUE_255:  CJNE   A,#0FFH,VALUE_253
        MOV    R1,A
        CLR    TI
        MOV    SBUF,#0BH
        JNB    TI,$
        JMP    MAIN
VALUE_253:  CJNE   A,#0FDH,VALUE_254
        MOV    R2,A
        CLR    TI
        MOV    SBUF,#0CH
        JNB    TI,$
        JMP    MAIN
VALUE_254:  CJNE   A,#0FEH,LENGTH
        MOV    R2,A
        CLR    TI
        MOV    SBUF,#0CH
        JNB    TI,$
        JMP    MAIN
LENGTH:     MOV    R4,A
        MOV    R3,A
        JMP    TEST_TURN_CAR
TEST_TURN_CAR: MOV    20H,R0
        PUSH   ACC
        MOV    A,R1
```

```
          CJNE   A,20H,TURN_CAR
          POP    ACC
          CJNE   R1,#00H,SKIP
          JMP    GO_RIGHT
SKIP:       JMP    GO_LEFT
TURN_CAR:   POP    ACC
          CJNE   R1,#0FFH,TURN_CAR2
          JMP    TURN_CAR1
;-----TURN CAR FIRST RIGHT---------------------
TURN_CAR1:    PUSH   05H
          SETB   P2.4
          MOV    R5,#37
STEP_1:     DJNZ   R5,LOOP_1
          MOV    R5,#21
          SETB   P2.1
          CLR    P2.3
          JMP    STEP_2
LOOP_1:     CALL   RUN_1STEP
          JMP    STEP_1
STEP_2:     DJNZ   R5,LOOP_2
          MOV    R5,#46
          JMP    STEP_3
LOOP_2:     CALL   LEFT_1STEP
          JMP    STEP_2
STEP_3:     DJNZ   R5,LOOP_3
          MOV    R5,#21
          SETB   P2.3
          CLR    P2.1
          JMP    STEP_4
LOOP_3:     CALL   RUN_1STEP
          JMP    STEP_3
STEP_4:     DJNZ   R5,LOOP_4
          POP    05H
          SETB   P2.1
          CLR    P2.3
          CALL   RUN_1STEP
          ;CALL   RUN_PIXEL
          MOV    A,R1
          MOV    R0,A
          JMP    GO_LEFT
LOOP_4:     CALL   LEFT_1STEP
          JMP    STEP_4
;----------------------------------------
;---RUN CAR ONLY 1 STEP LEFT ONLY----------
LEFT_1STEP:   SETB   P2.2
          CALL   DELAY
          CLR    P2.2
```

```
          CALL   DELAY
          RET
;---------------------------------------
;----------TURN CAR SECOND LEFT------------
TURN_CAR2:    PUSH   05H
          SETB   P2.4
          MOV    R5,#33
STEP1:       DJNZ   R5,LOOP1
          MOV    R5,#21
          SETB   P2.3
          CLR    P2.1
          JMP    STEP2
LOOP1:       CALL   RUN_1STEP
          JMP    STEP1
STEP2:       DJNZ   R5,LOOP2
          MOV    R5,#41
          JMP    STEP3
LOOP2:       CALL   LEFT_1STEP
          JMP    STEP2
STEP3:       DJNZ   R5,LOOP3
          MOV    R5,#21
          SETB   P2.1
          CLR    P2.3
          JMP    STEP4
LOOP3:       CALL   RUN_1STEP
          JMP    STEP3
STEP4:       DJNZ   R5,LOOP4
          POP    05H
          SETB   P2.3
          CLR    P2.1
          ;CALL   RUN_PIXEL
          MOV    A,R1
          MOV    R0,A
          JMP    GO_RIGHT
LOOP4:       CALL   LEFT_1STEP
          JMP    STEP4
;-----------------------------------------
GO_RIGHT:     SETB   P2.3
          CLR    P2.1
          JMP    RUN_STEP
GO_LEFT:      SETB   P2.1
          CLR    P2.3
          JMP    RUN_STEP
AIR_BRUSH_RUN: CLR   P2.4
          CALL   DELAY
          CALL   DELAY
          CALL   DELAY
```

```
        CALL   DELAY
        SETB   P2.4
        RET
AIR_BRUSH_STOP: SETB   P2.4
        RET
;---RUN CAR ONLY 1 STEP-----------
RUN_1STEP:    SETB   P2.0
        SETB   P2.2
        CALL   DELAY
        CLR    P2.0
        CLR    P2.2
        CALL   DELAY
        RET
;--------------------------------
;---RUN CAR ONLY 1 PIXEL----------
RUN_PIXEL:    PUSH   05H
        MOV    R5,#0AH  ;NUMBER OF STEP PER PIXEL
RUN_1PIXEL:   DJNZ   R5,NOT_1PIXEL
        POP    05H
        RET
NOT_1PIXEL:   CALL   RUN_1STEP
        JMP    RUN_1PIXEL
;-------------------------------
RUN_STEP:     INC    R3
RUN_CAR:      DJNZ   R3,RUN_STEP1
        SETB   P2.4
        MOV    A,#00H
        MOV    R1,#00H
        MOV    R2,#00H
        MOV    R3,#00H
        CJNE   R4,#0FCH,NEXT  ;0FC=252
       ; CALL   DELAY
        CLR    TI
        MOV    SBUF,#0AH   ; 0A=10
        JNB    TI,$
        JMP    MAIN
NEXT:       CLR    TI
        MOV    SBUF,#0DH   ;0D=13
        JNB    TI,$
        JMP    MAIN
RUN_STEP1:    CJNE   R2,#0FDH,STOP_BRUSH
        CALL   AIR_BRUSH_RUN
        CALL   RUN_PIXEL
        JMP    RUN_CAR
STOP_BRUSH:   CALL   AIR_BRUSH_STOP
        CALL   RUN_PIXEL
        JMP    RUN_CAR
```

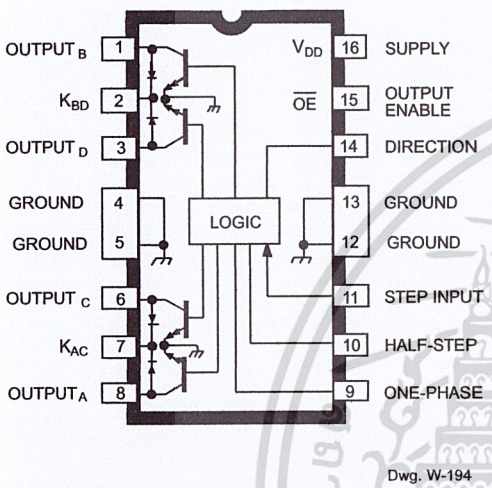
```
DELAY:  PUSH 02H
        PUSH 03H
        MOV  R2,#00H
DLY:    MOV  R3,#00H
        DJNZ R3,$
        DJNZ R2,DLY
        POP  03H
        POP  02H
        RET
RECEIVE:        JNB    RI,$
                MOV    A,SBUF
                CLR    RI
                RET
END;
```

ภาคผนวก ข

5804

Data Sheet
26184.12C

BiMOS II UNIPOLAR STEPPER-MOTOR TRANSLATOR/DRIVER



Dwg. W-194

Note that the UCN5804B (dual in-line package) and UCN5804LB (small outline IC package) are electrically identical and share a common pin number assignment.

ABSOLUTE MAXIMUM RATINGS

Output Voltage, V_{CE} ..... **50 V**
Output Sustaining Voltage, $V_{CE\ (sus)}$ ..... **35 V**
Output Sink Current, I_{OUT} ..... **1.5 A**
Logic Supply Voltage, V_{DD} ..... **7.0 V**
Input Voltage, V_{IN} ..... **7.0 V**
Package Power Dissipation, P_D ..... **See Graph**
Operating Temperature Range, T_A ..... **-20°C to +85°C**
Storage Temperature Range, T_S ..... **-55°C to +150°C**

Combining low-power CMOS logic with high-current and high-voltage bipolar outputs, the UCN5804B and UCN5804LB BiMOS II translator/drivers provide complete control and drive for a four-phase unipolar stepper-motor with continuous output current ratings to 1.25 A per phase (1.5 A startup) and 35 V.

The CMOS logic section provides the sequencing logic, DIRECTION and OUTPUT ENABLE control, and a power-ON reset function. Three stepper-motor drive formats, wave-drive (one-phase), two-phase, and half-step are externally selectable. The inputs are compatible with standard CMOS, PMOS, and NMOS circuits. TTL or LSTTL may require the use of appropriate pull-up resistors to ensure a proper input-logic high.

The wave-drive format consists of energizing one motor phase at a time in an A-B-C-D (or D-C-B-A) sequence. This excitation mode consumes the least power and assures positional accuracy regardless of any winding inbalance in the motor. Two-phase drive energizes two adjacent phases in each detent position (AB-BC-CD-DA). This sequence mode offers an improved torque-speed product, greater detent torque, and is less susceptible to motor resonance. Half-step excitation alternates between the one-phase and two-phase modes (A-AB-B-BC-C-CD-D-DA), providing an eight-step sequence.

The bipolar outputs are capable of sinking up to 1.5 A and withstanding 50 V in the OFF state (sustaining voltages up to 35 V). Ground-clamp and flyback diodes provide protection against inductive transients. Thermal protection circuitry disables the outputs when the chip temperature is excessive.

Both devices are rated for operation over the temperature range of -20°C to +85°C. The UCN5804B is supplied in a 16-pin dual in-line plastic batwing package with a copper lead frame and heat-sinkable tabs for improved power dissipation capabilities; the UCN5804LB is supplied in a 16-lead plastic SOIC batwing package with a copper lead frame and heat-sinkable tabs.

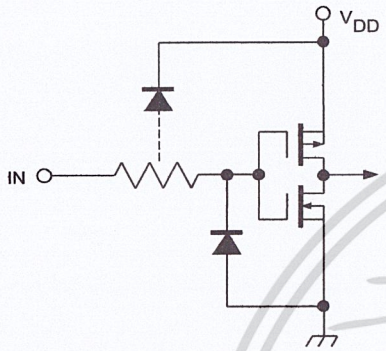
FEATURES

- 1.5 A Maximum Output Current
- 35 V Output Sustaining Voltage
- Wave-Drive, Two-Phase, and Half-Step Drive Formats
- Internal Clamp Diodes
- Output Enable and Direction Control
- Power-ON Reset
- Internal Thermal Shutdown Circuitry

Always order by complete part number, e.g., **UCN5804B** .

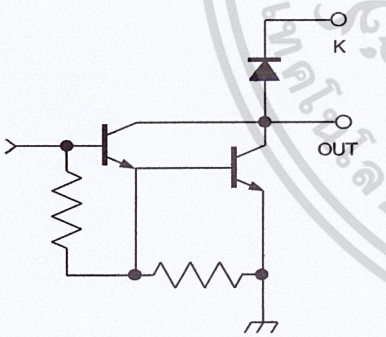
Allegro® MicroSystems, Inc.

TYPICAL INPUT CIRCUIT

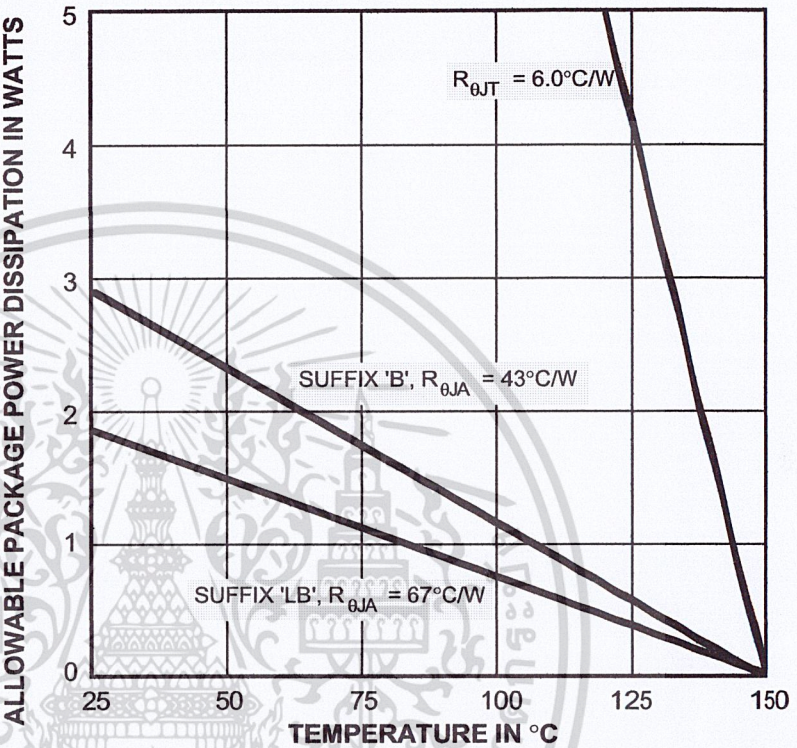


Dwg. EP-010-5

TYPICAL OUTPUT DRIVER



Dwg. EP-021-4



Dwg. GP-049-2

TRUTH TABLE

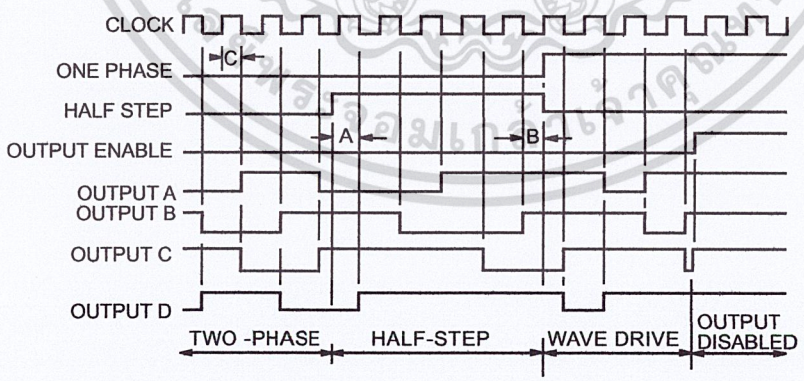
Drive Format	Pin 9	Pin 10
Two-Phase	L	L
One-Phase	H	L
Half-Step	L	H
Step-Inhibit	H	H

Allegro® MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000

ELECTRICAL CHARACTERISTICS at T_A = 25°C, $T_J \leq$ 150°C, V_{DD} = 4.5 V to 5.5 V (unless otherwise noted).

Characteristic	Symbol	Test Conditions	Limits			Units
			Min.	Typ.	Max.	
Output Leakage Current	I_{CEX}	V_{OUT} = 50 V	—	10	50	µA
Output Sustaining Voltage	$V_{CE(sus)}$	I_{OUT} = 1.25 A, L = 3 mH	35	—	—	V
Output Saturation Voltage	$V_{CE(SAT)}$	I_{OUT} = 700 mA	—	1.0	1.2	V
		I_{OUT} = 1 A	—	1.1	1.4	V
		I_{OUT} = 1.25 A	—	1.2	1.5	V
Clamp Diode Leakage Current	I_R	V_R = 50 V	—	10	50	µA
Clamp Diode Forward Voltage	V_F	I_F = 1.25 A	—	1.5	3.0	V
Input Current	$I_{IN(1)}$	$V_{IN} = V_{DD}$	—	0.5	5.0	µA
	$I_{IN(0)}$	V_{IN} = 0.8 V	—	-0.5	-5.0	µA
Input Voltage	$V_{IN(1)}$	V_{DD} = 5 V	3.5	—	5.3	V
	$V_{IN(0)}$		-0.3	—	0.8	V
Supply Current	I_{DD}	2 Outputs ON	—	20	30	mA
Turn-Off Delay	t_{ON}	50% Step Inputs to 50% Output	—	—	10	µs
Turn-On Delay	t_{OFF}	50% Step Inputs to 50% Output	—	—	10	µs
Thermal Shutdown Temperature	T_J		—	165	—	°C

TIMING CONDITIONS



Dwg. W-110A

A. Minimum Data Set Up Time . . . . . **100 ns**
B. Minimum Data Hold Time . . . . . **100 ns**
C. Minimum Step Input Pulse Width . . . . . **3.0 µs**

5804
BiMOS II UNIPOLAR STEPPER-MOTOR TRANSLATOR/DRIVER

APPLICATIONS INFORMATION

Internal power-ON reset (POR) circuitry resets $OUTPUT_A$ (and $OUTPUT_D$ in the two-phase drive format) to the ON state with initial application of the logic supply voltage. After reset, the circuit then steps according to the tables.

The outputs will advance one sequence position on the high-to-low transition of the STEP INPUT pulse. Logic levels on the HALF-STEP and ONE-PHASE inputs will determine the drive format (one-phase, two-phase, or half-step). The DIRECTION pin determines the rotation sequence of the outputs. Note that the STEP INPUT must be in the low state when changing the state of ONE-PHASE, HALF-STEP, or DIRECTION to prevent erroneous stepping.

All outputs are disabled (OFF) when OUTPUT ENABLE is at a logic high. If the function is not required, OUTPUT ENABLE should be tied low. In that condition, all outputs depend only on the state of the step logic.

During normal commutation of a unipolar stepper motor, mutual coupling between the motor windings can force the outputs of the UCN5804B below ground. This condition will cause forward biasing of the collector-to-substrate junction and source current from the output. For many L/R applications, this substrate current is high enough to adversely affect the logic circuitry and cause misstepping. External series diodes (Schottky are recommended for increased efficiency at low-voltage operation) will prevent substrate current from being sourced through the outputs. Alternatively, external ground clamp diodes will provide a preferred current path from ground when the outputs are pulled below ground.

Internal thermal protection circuitry disables all outputs when the junction temperature reaches approximately 165°C. The outputs are enabled again when the junction cools down to approximately 145°C.

WAVE-DRIVE SEQUENCE

Half Step = L, One Phase = H				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	ON

TWO-PHASE DRIVE SEQUENCE

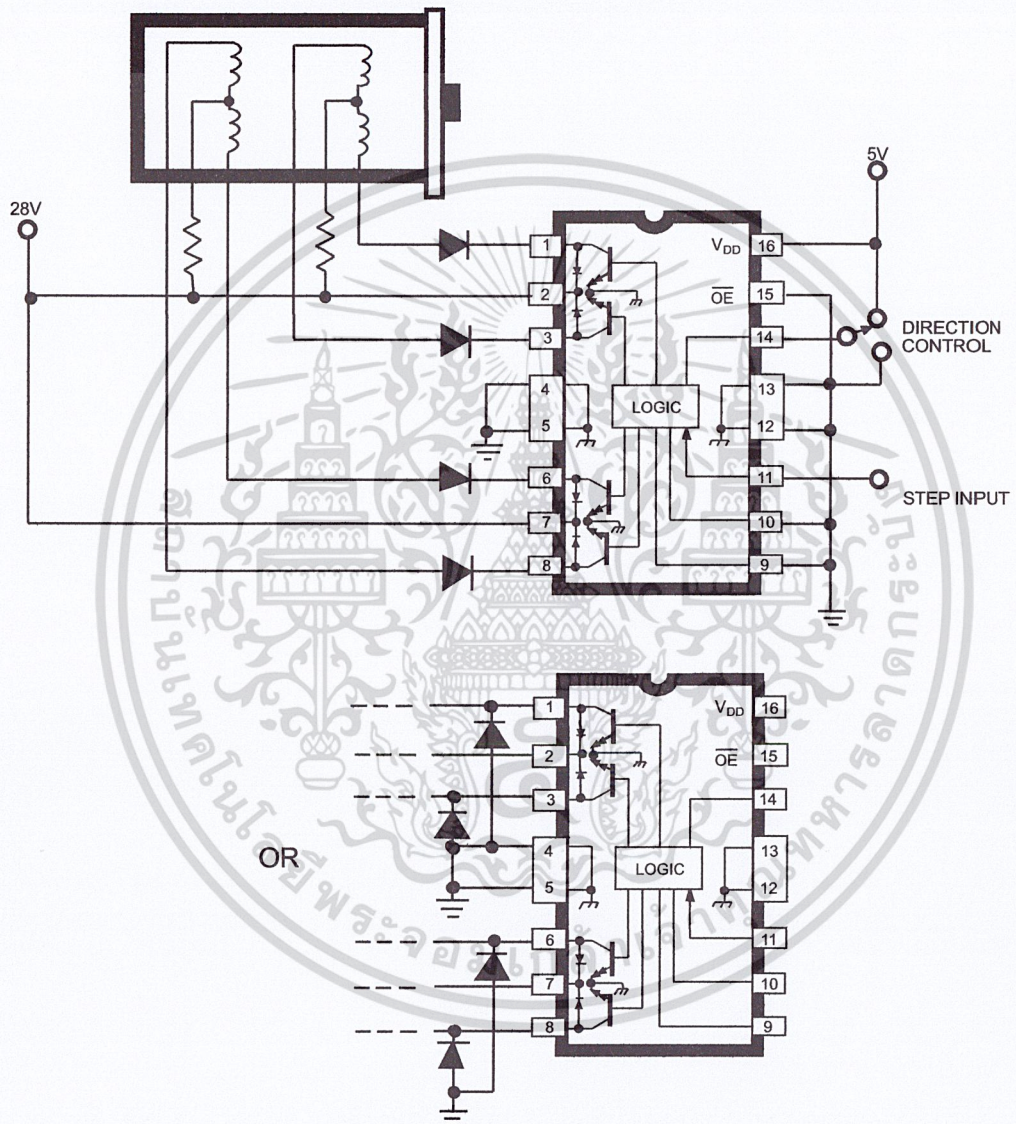
Half Step = L, One Phase = L				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	ON
1	ON	OFF	OFF	ON
2	ON	ON	OFF	OFF
3	OFF	ON	ON	OFF
4	OFF	OFF	ON	ON

HALF-STEP DRIVE SEQUENCE

Half Step = H, One Phase = L				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	ON	ON	OFF	OFF
3	OFF	ON	OFF	OFF
4	OFF	ON	ON	OFF
5	OFF	OFF	ON	OFF
6	OFF	OFF	ON	ON
7	OFF	OFF	OFF	ON
8	ON	OFF	OFF	ON

Allegro® MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000

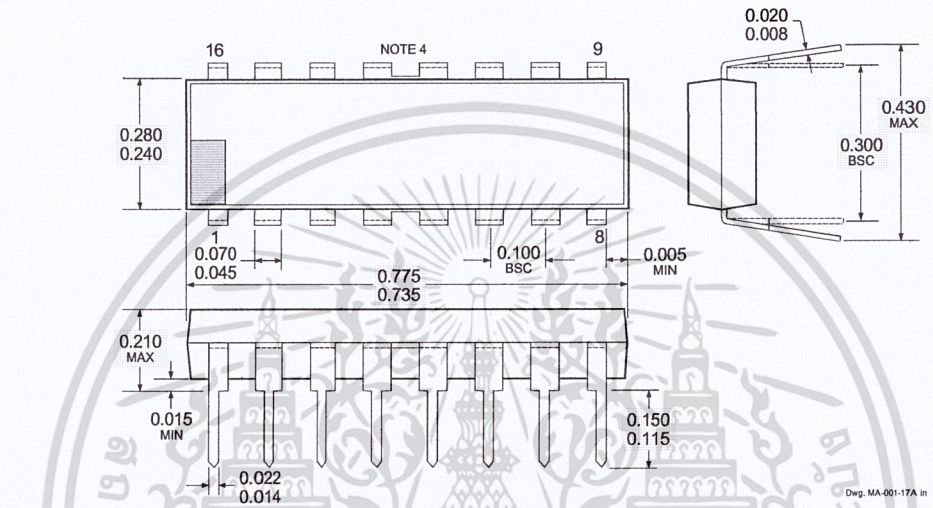
TYPICAL APPLICATION
L/R Stepper-Motor Drive



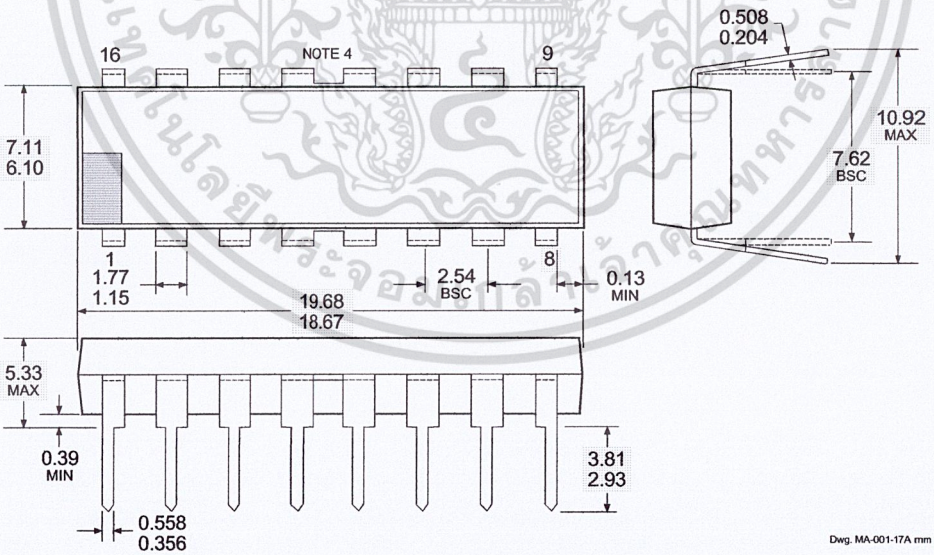
Dwg. EP-029A

UCN5804B

Dimensions in Inches
(controlling dimensions)



Dimensions in Millimeters
(for reference only)

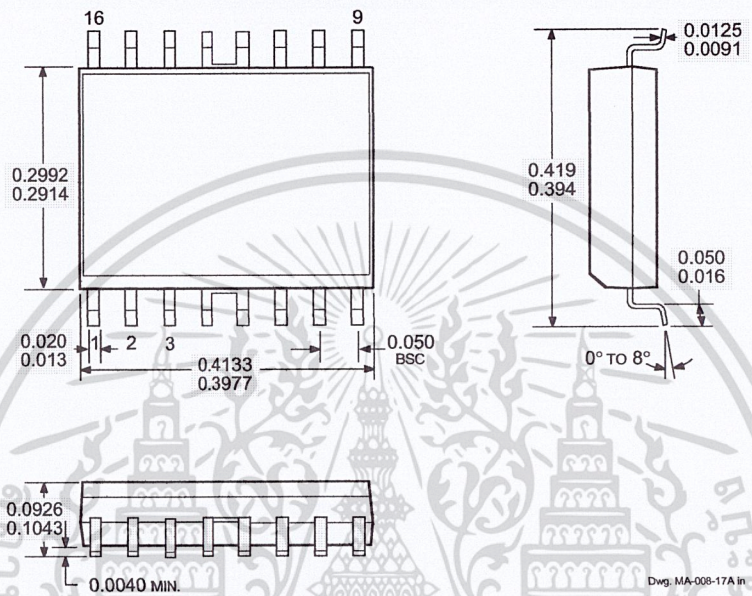


NOTES: 1. Exact body and lead configuration at vendor's option within limits shown.
2. Lead spacing tolerance is non-cumulative
3. Lead thickness is measured at seating plane or below.
4. Webbed lead frame. Leads 4, 5, 12, and 13 are internally one piece.

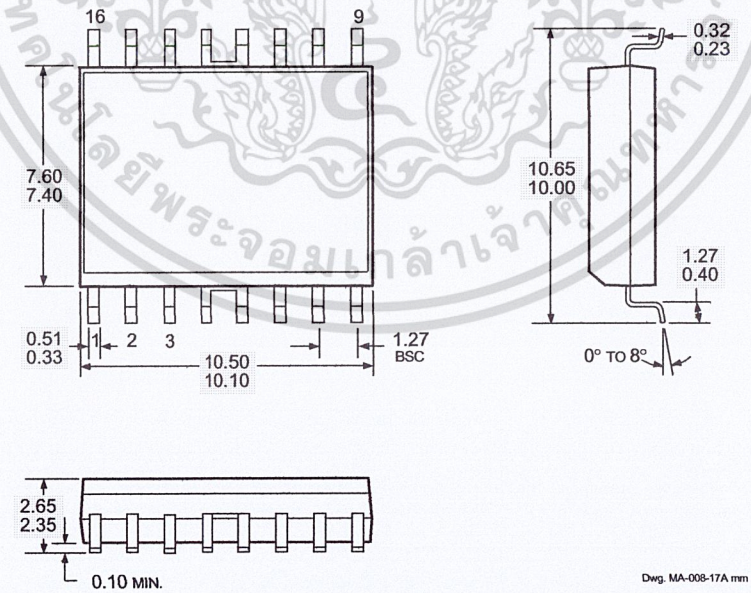
Allegro MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000

UCN5804LB

Dimensions in Inches
(for reference only)



Dimensions in Millimeters
(controlling dimensions)



NOTES: 1. Exact body and lead configuration at vendor's option within limits shown.
2. Lead spacing tolerance is non-cumulative
3. Lead thickness is measured at seating plane or below.
4. Webbed lead frame. Leads 4, 5, 12, and 13 are internally one piece.

5804
BiMOS II UNIPOLAR STEPPER-MOTOR TRANSLATOR/DRIVER

MOTOR DRIVERS SELECTION GUIDE

Function	Output Ratings *		Part Number †
INTEGRATED CIRCUITS FOR BRUSHLESS DC MOTORS			
3-Phase Controller/Drivers	±2.0 A	45 V	2936 and 2936-120
Hall-Effect Latched Sensors	10 mA	24 V	3175 and 3177
2-Phase Hall-Effect Sensor/Controller	20 mA	25 V	3235
Hall-Effect Complementary-Output Sensor	20 mA	25 V	3275
2-Phase Hall-Effect Sensor/Driver	900 mA	14 V	3625
2-Phase Hall-Effect Sensor/Driver	400 mA	26 V	3626
Hall-Effect Complementary-Output Sensor/Driver	300 mA	60 V	5275
3-Phase Back-EMF Controller/Driver	±900 mA	14 V	8902–A
3-Phase Controller/DMOS Driver	±4.0 A	14 V	8925
3-Phase Back-EMF Controller/Driver	±1.0 A	7 V	8984
INTEGRATED BRIDGE DRIVERS FOR DC AND BIPOLAR STEPPER MOTORS			
PWM Current-Controlled Dual Full Bridge	±750 mA	45 V	2916
PWM Current-Controlled Dual Full Bridges	±1.5 A	45 V	2917 and 2918
PWM Current-Controlled Dual Full Bridge	±750 mA	45 V	2919
Dual Full-Bridge Driver	±2.0 A	50 V	2998
PWM Current-Controlled Full Bridge	±2.0 A	50 V	3952
PWM Current-Controlled Full Bridge	±1.3 A	50 V	3953
PWM Current-Controlled Microstepping Full Bridges	±1.5 A	50 V	3955 and 3957
PWM Current-Controlled Dual Full Bridge	±800 mA	33 V	3964
PWM Current-Controlled Dual Full Bridge	±650 mA	30 V	3966 and 3968
PWM Current-Controlled Dual Full Bridge	±750 mA	45 V	6219
OTHER INTEGRATED CIRCUIT & PMCM MOTOR DRIVERS			
Unipolar Stepper-Motor Quad Driver	1.8 A	50 V	2544
Unipolar Stepper-Motor Translator/Driver	1.25 A	50 V	5804
Unipolar Stepper-Motor Quad Drivers	1 A	46 V	7024 and 7029
Unipolar Microstepper-Motor Quad Driver	1.2 A	46 V	7042
Voice-Coil Motor Driver	±500 mA	6 V	8932–A
Voice-Coil Motor Driver	±800 mA	16 V	8958
Voice-Coil (and Spindle) Motor Driver	±350 mA	7 V	8984

* Current is maximum specified test condition, voltage is maximum rating. See specification for sustaining voltage limits or over-current protection voltage limits. Negative current is defined as coming out of (sourcing) the output.

† Complete part number includes additional characters to indicate operating temperature range and package style.

Allegro MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000