หุ่นยนต์เตะฟุตบอล
SOCCER ROBOT

นายพิศาล มูลอำคา
นายพุฒิพงศ์ มีประทัง
นายจิรายุส คงโท

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมเครื่องกล
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

หุ่นยนต์เตะฟุตบอล
SOCCER ROBOT

โดย

นายพิศาล มูลอำคา

นายพุฒิพงศ์ มีประทัง

นายจิรายุส คงโท

อาจารย์ที่ปรึกษา

ดร. ณัฐวุฒิ เดไปวา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชา วิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์เตะฟุตบอล

SOCCER ROBOT

ผู้จัดทำ

1. นายพิศาล	มูลอำคา	รหัสประจำตัว 46015408
2. นายพุฒิพงศ์	มีประทัง	รหัสประจำตัว 46015409
3. นายจิรายุส	คงโท	รหัสประจำตัว 45010130

ณัฐวุฒิ เดไปวา **อาจารย์ที่ปรึกษา**

(ดร. ณัฐวุฒิ เดไปวา)

บทคัดย่อ

โปรเจกนี้เป็นการศึกษาถึงหุ่นยนต์เตะบอลโดยใช้หลักการทางด้านกลศาสตร์ในการออกแบบ ซึ่งใน การศึกษาถึงการทำงานของหุ่นยนต์เตะบอลก็เป็นอีกทางหนึ่งที่ทำให้เราเข้าใจถึงการทำงานของหุ่นยนต์ โดยในโปรเจกนี้การออกแบบโครงสร้างของหุ่นใช้หลักการทฤษฎีหาโครงสร้างของตัวหุ่น ล้อ และมอเตอร์ที่ทำให้หุ่นมีอัตราเร่งที่สูงที่สุด และอุปกรณ์ส่วนอื่น ๆ ของหุ่น อย่างเช่น ระบบการเตะบอลจะเป็นการออกแบบทางด้านกลศาสตร์ใช้ระบบโซลินอยด์ ส่วนทางด้านตัวเลี้ยงบอลจะเป็นการออกแบบทางด้านกลศาสตร์ในการหาตำแหน่งและมุมที่ดีที่สุดในการติดระบบ ส่วนทางด้านการรับคำสั่งและประมวลผลเราจะใช้ระบบไมโครคอนโทรเลอร์ แปลผลสัญญาณจากคอมพิวเตอร์โดยหุ่นยนต์ในโปรเจกนี้จะเป็นหุ่นต้นแบบซึ่งเป็นการศึกษาระบบการทำงานของหุ่นยนต์เตะบอล

SOCCER ROBOT

Mr. Pisan	Moonumca	Student ID 46015408
Mr. Phutthiphong	meeprathang	Student ID 46015409
Mr. Jirayut	kongtoo	Student ID 45010130
Dr. Nattawoot	**Depaiwa**	**Advisor**

Abstract

This project aim to comprehend robot mechanic through studying 'Robocup' by using mechanical discipline to design hardware. The project frame is to design the robot structure. Beside the movement system, the researchers also study about kicking system that use solenoid mechanism and for the dribbling mechanism we design the system to be able to calculate the best position for dribbling. For processing system, we use microcontroller system to process the signal from computer. This robot project is going to be prototype.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จลุล่วงไปได้ด้วยดี ถ้าหากปราศจากคำแนะนำ และความอุปถัมภ์ทางด้านการเงินจากอาจารย์ ดร.ณัฐวุฒิ เดไปวา อาจารย์ที่ปรึกษาปริญญานิพนธ์

ท้ายที่สุด คณะผู้จัดทำขอกราบขอบพระคุณบิดา มารดา พี่น้อง รวมถึงครอบครัวของคณะผู้จัดทำที่ให้การสนับสนุนทางด้านการศึกษา กำลังใจ และแนะแนวทางในการดำเนินชีวิต ด้วยความรักและความปรารถนาดีตลอดมา

นายพิศาล มูลอำคา
นายพุฒิพงศ์ มีประทัง
นายจิรายุส คงโท

สารบัญ

หน้าที่

สารบัญตาราง

หน้าที่

สารบัญรูปภาพ

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมา

ในปัจจุบันนี้หุ่นยนต์ที่สามารถปฏิบัติงานได้อย่างอัตโนมัติ ได้เริ่มเข้ามามีบทบาทอย่างมากในชีวิตประจำวัน ทั้งในด้านการอำนวยความสะดวกต่าง ๆ และมีประโยชน์อย่างมากในด้านอุตสาหกรรม โดยหุ่นยนต์ทั่วไปจะทำงานได้ก็ต้องมีผู้เขียนโปรแกรมการทำงานให้ เมื่อเราต้องการศึกษาและทดลองเกี่ยวกับหุ่นยนต์ เราก็จำเป็นต้องมีความรู้ในด้านการเขียนโปรแกรมโดยในโครงการนี้ได้ใช้โปรแกรมวิชวลเบสิก (Visual Basic) ในการเขียนโปรแกรม

การรับค่าจากผู้ใช้ คอมพิวเตอร์นั้นเราสามารถรับค่าได้หลายทาง เช่น เมาส์ แป้นพิมพ์ และอื่นๆ โดยนำค่าเหล่านั้นมาวิเคราะห์อีกทีหนึ่งถึงความต้องการของผู้ใช้จะทำให้การทำงานมีค่าความแม่นยำมากยิ่งขึ้นในการใช้คอมพิวเตอร์แทนคนนั้นในงานที่ซ้ำๆ หรือต้องการความแม่นยำโดยคนเราไม่สามารถทำงานได้แต่ว่าคอมพิวเตอร์ทำแทนได้ ทั้งนี้ก็จำเป็นจะต้องมีอีกฝ่ายหนึ่งที่ต้องผลิตโปรแกรมขึ้น ในการผลิตโปรแกรมนี้จะต้องให้ผู้ใช้สามารถใช้งานในลักษณะที่ง่ายแต่การใช้ และต้องมีความผิดพลาดน้อยที่สุดจึงจำเป็นจะต้องวิเคราะห์ให้ละเอียดด้วย

1.2 วัตถุประสงค์ของโครงงาน

เพื่อศึกษาการทำงาน และ นำไปประยุกต์ใช้ ของหุ่นยนต์อัตโนมัติ โดยใช้ความรู้ทางด้าน การเขียนโปรแกรมประยุกต์ใช้งานด้วย โปรแกรมภาษาวิชวลเบสิก และการประยุกต์ใช้งานของไมโครคอนโทรลเลอร์(Microcontroller) พร้อมด้วยการเขียนโปรแกรมติดต่อกับ พอร์ต USB และนำค่าจากการควบคุมมาประมวลผลและก็ส่งออกไปยัง พอร์ตอนุกรมเพื่อสั่งให้ตัวหุ่นทำงานตามที่ต้องการได้

1.3 ขอบเขตของโครงงาน

เนื้อหาของปริญญานิพนธ์นี้ จะเริ่มการศึกษาการทำงานของหุ่นยนต์โดยใช้คนบังคับ โดยใช้ จอยร์สติก ซึ่งส่งค่าผ่านพอร์ตยูเอสบี (USB port) และใช้การคลิกของเมาส์แล้วคำนวณตำแหน่งของตัวหุ่น แล้วนำค่านั้นไปคำนวณการเคลื่อนที่ของตัวหุ่น แล้วส่งค่าออกไปที่พอร์ตอนุกรม (Serial port) และ ไมโครคอนโทรลเลอร์ (Microcontroller) จะรับค่าที่ได้แล้วตีความหมายของค่าที่ส่งมา แล้วสั่งให้มอเตอร์เคลื่อนที่ และได้ทำการพัฒนาจนสามารถประมวลผลหาตำแหน่งได้อย่างแม่นยำและหุ่นยนต์สามารถเคลื่อนที่เข้าไปยิงลูกบอลได้อย่างอัตโนมัตินอกจากนี้เรายังได้ทำการนำค่าพิกัดของเส้นทางการเคลื่อนที่แบบการใช้เมาส์คลิกจากภาพสนามแข่งเพื่อให้หุ่นเคลื่อนที่ไปยังตำแหน่งที่ต้องการได้

1.4 ขั้นตอนการดำเนินโครงงาน

1. ศึกษาหลักการทำงานของ ไมโครคอนโทรลเลอร์
2. ออกแบบและสร้างตัวหุ่นยนต์เตะฟุตบอล

3. ศึกษาโปรแกรมวิชวลเบสิก และสร้างโปรแกรมที่ใช้ในการคำนวณหาตำแหน่งพิกัดที่ได้จากการคลิกเมาส์เพื่อใช้ในโปรแกรมที่ควบคุมแบบออโตเมติกส์

4.ศึกษาโปรแกรมวิชวลเบสิก และสร้างโปรแกรมที่ใช้ในการคำนวณหาตำแหน่งพิกัดที่ได้จากการใช้จอยล์สติกเพื่อใช้ในโปรแกรมที่ควบคุมด้วยมือ

5 .ศึกษาและสร้างไมโครคอนโทรลเลอร์ที่จะใช้ในการควบคุมโปรแกรมที่ใช้ในการกำหนดการเคลื่อนที่ของตัวหุ่นเมื่อได้รับค่าต่างๆ จากคอมพิวเตอร์

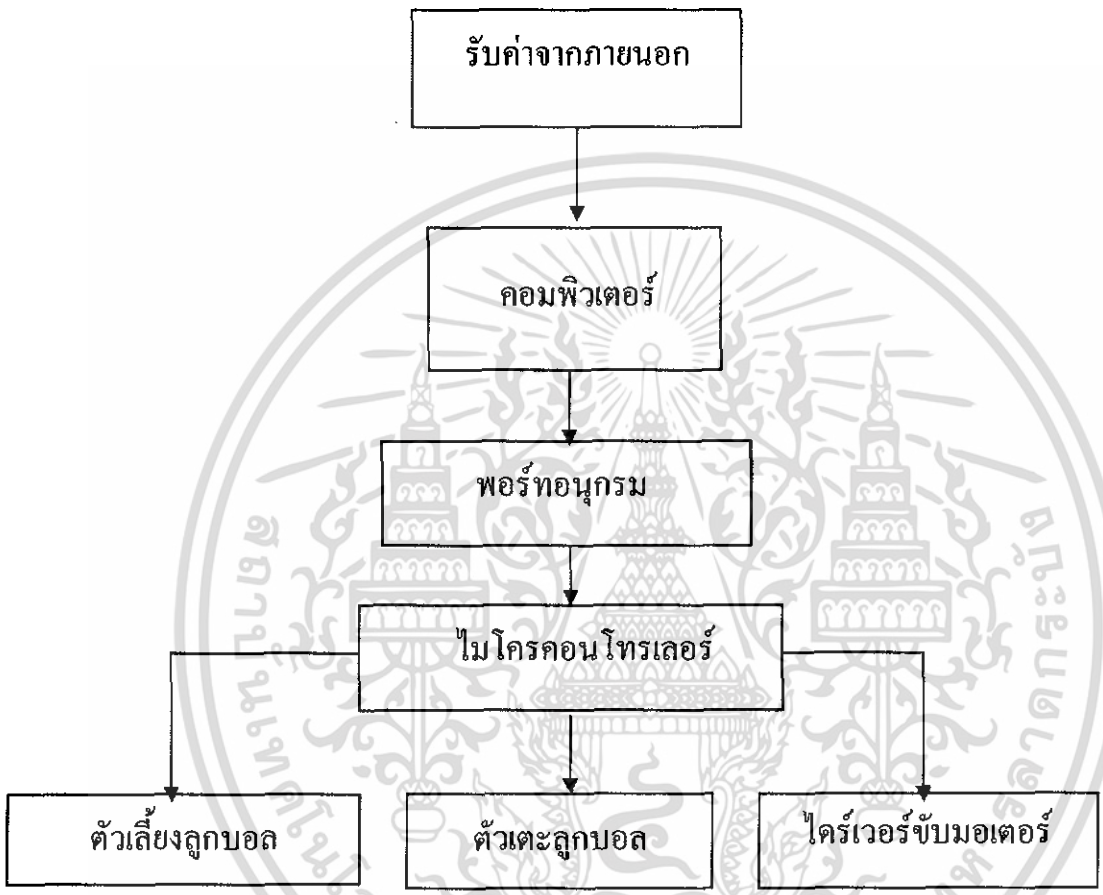
6 .ติดตั้งและทำการทดลอง

7 .สรุปผลการทดลองและข้อผิดพลาด

8 .นำเสนอผลงาน

บทที่2

อุปกรณ์และทฤษฎีที่ใช้ในหุ่นยนต์เตะฟุตบอล



รูปที่ 2.1 Block diagram แสดงหลักการทำงานของหุ่นยนต์เตะฟุตบอล

ส่วนประกอบของตัวหุนยนต์เตะฟุตบอลประกอบไปด้วยส่วนหลักๆ ทั้ง 3 ส่วนดังต่อไปนี้

2.1 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์คือ ไอซีหน่วยประมวลผลกลาง (Central Processing Unit) ประเภทหนึ่งที่สามารถรับสัญญาณข้อมูลจากอุปกรณ์ภายนอก เพื่อนำมาคำนวณ ตัดสินใจ และส่งสัญญาณออกไปยังอุปกรณ์ภายนอกเพื่อสั่งให้ทำงานตามโปรแกรม และข้อมูลที่รับเข้ามา มีคุณสมบัติความเป็น Single Chip คือ สามารถทำงานได้โดยตัวไอซีเอง ไม่

ต้องต่อวงจรเพิ่ม หรือต่อเพิ่มน้อยที่สุด ประโยชน์ที่ได้รับก็คือ การออกแบบวงจรทำได้ง่ายขึ้น ใช้อุปกรณ์ประกอบวงจรน้อยกว่าเดิม พื้นที่วงจรรวมมีขนาดเล็กลงเป็นอย่างมาก กินไฟเลี้ยงวงจรน้อยลง ทำให้ต้นทุนการผลิตและต้นทุนการทำงานลดต่ำลง นอกจากนี้ การบันทึกโปรแกรมที่เขียนลงในตัวไมโครคอนโทรลเลอร์ก็สามารถทำได้ง่ายและรวดเร็ว โดยเลือกรุ่นที่มีแฟลชเมโมรีในตัวที่สามารถเขียนโปรแกรมลงไปได้ ทำให้ไม่ต้องต่ออีพรอมเพื่อเก็บโปรแกรม

ในโครงงานนี้ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT89C1051 มีขาทั้งหมด 20 ขา โดยมีพอร์ต 2 พอร์ต คือ P1 และ P3 สามารถต่อสัญญาณนาฬิกาได้ 4-24 MHz มีหน่วยความจำภายใน 64 byte หน่วยความจำโปรแกรม 1k byte

2.1.1 ความหมายของไมโครคอนโทรลเลอร์

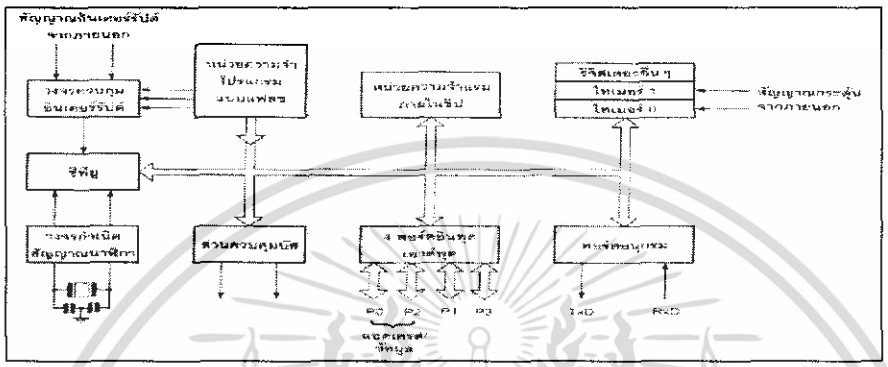
ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณเอาท์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิคที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบ ในขณะที่มีขีดความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม ดังนั้นไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ

2.1.2 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx มีคุณสมบัติดังนี้คือ

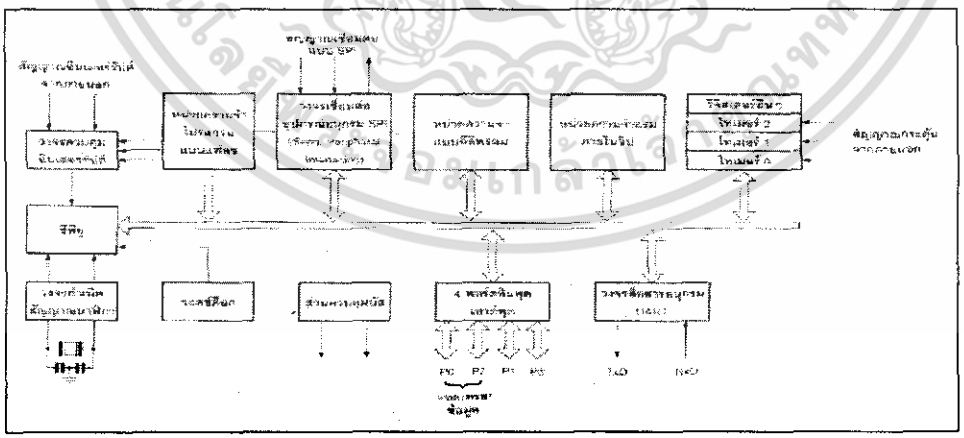
- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้เป็นพันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม และในบางเบอร์จะมีหน่วยความจำแบบอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุต-เอาท์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเทอร์รัปได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

ด้านโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx พบว่าโครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแต่แตกต่างกันเฉพาะหน่วยความจำแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ในอนุกรม87 xx หน่วยความจำภายในจะเป็นแบบอีพรอม และบางเบอร์สามารถโปรแกรมได้เพียงครั้งเดียว



รูปที่2-2 แสดงถึงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx

ในโครงสร้างพื้นฐานของอนุกรม AT89Sxx พบว่า มีส่วนประกอบที่เพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน เช่น วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์อนุกรมนี้ใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบ หรือเรียกว่า การโปรแกรมวงจร และยังมีไทม์เมอร์/เคาน์เตอร์ขนาด 16 บิต ที่เพิ่มเติมเข้ามาอีก 1 ตัว เป็นไทม์เมอร์ และยังเพิ่มวงจรว็อตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียูอีกด้วย



รูปที่2-3 แสดงถึงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Sxx

2.1.3 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังนี้

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง5+ v

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัป จากภายนอกช่อง 0 หรือขา $\overline{INT0}$

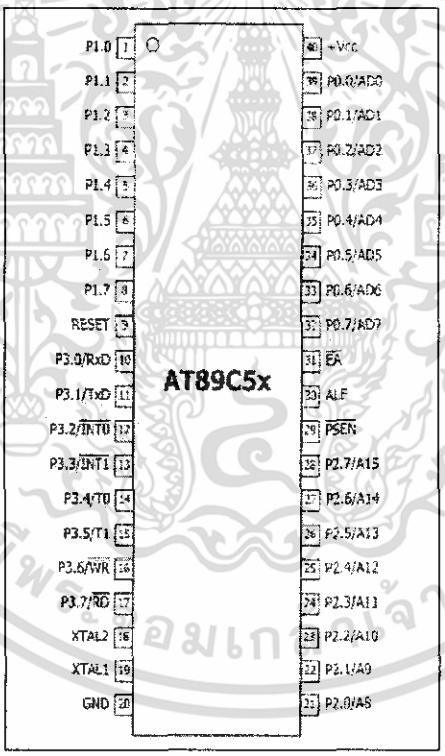
P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเทอร์รัป จากภายนอกช่อง 1 หรือขา $\overline{INT1}$

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทม์เมอร์จากภายนอกช่อง 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา T1

P3.6 ใช้เป็นขาสัญญาณ $\overline{WR}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

P3.7 ใช้เป็นขาสัญญาณ $\overline{RD}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก



รูปที่2-4 แสดงถึงการจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-5 1ในอนุกรม AT89C5x

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีนไซเคิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา $\overline{ALE}$ /PROG (Address Latch Enable/Program Pulse Input) เป็นขาที่ใช้ในการควบคุมการแลตช์ของขาพอร์ต 0เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขาสำหรับรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีพรอม

ขา $\overline{PSEN}$ (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชชีนไซเคิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆ ออกมา

ขา $\overline{EA}/V_{pp}$ (External Access Enable/Programming Voltage Input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอติดต่อกับหน่วยความจำภายนอก แต่ถ้าหากขานี้เป็น "1" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูง สำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม+12 v

ขา XTAL1 และขา XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

2.1.4 โครงสร้างและการทำงานของพอร์ตในไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ต คือ พอร์ต 0 จนถึง พอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้า และเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงแลตช์ และวงจรขับตลอดจนบัฟเฟอร์อินพุต ดังแสดงในรูปที่ 20-2

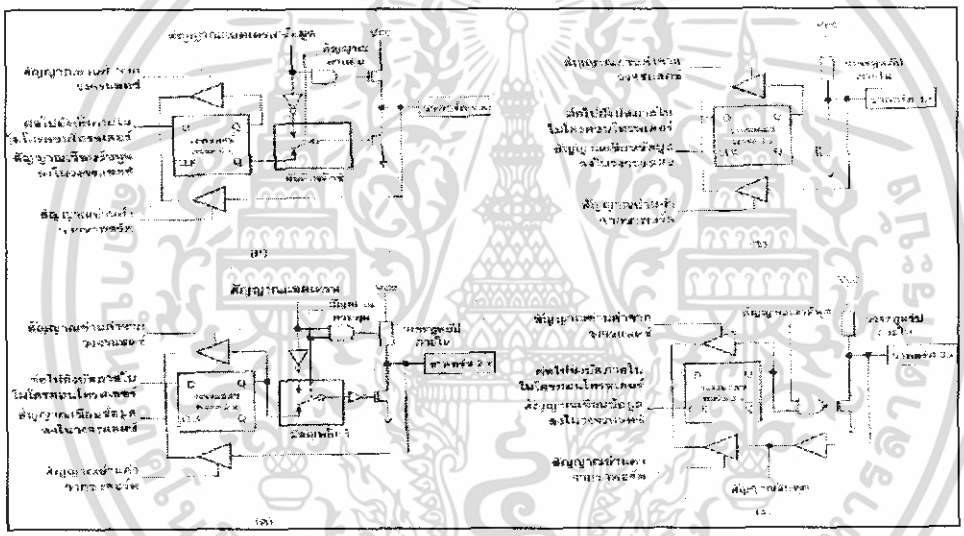
ที่พอร์ต 0 และ พอร์ต 2 จะใช้งานเป็นพอร์ตอินพุต และเอาต์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ต และพอร์ต 1 บางขา นอกจากจะใช้เป็นขาพอร์ต อินพุต-เอาต์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด

วงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช เป็นวงจรของพอร์ต 0 และวงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกจากกัน นั่นคือสัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูล

จากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิปฟลอปในขณะที่ข้อมูลจะผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิปฟลอป ที่พอร์ตนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ต ว่าต้องการใช้งานเป็นขาพอร์ตอินพุต-เอาท์พุตปกติหรือใช้ในการติดต่อกับหน่วยความจำภายนอกไมโครคอนโทรลเลอร์ เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัปภายใน หากมีการนำพอร์ต0 ไปใช้งานเป็นพอร์ตอินพุตจะต้องต่อตัวต้านทานพูลอัปภายนอกเข้าที่ขาพอร์ต 0 ทุกขาด้วย

พอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน

วงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัปเพิ่มเติมเข้ามา ส่วนวงจรภายในของพอร์ต 3 พบว่ามีลักษณะคล้ายกับพอร์ต 1 แต่มีการเพิ่มเติมวงจรบัฟเฟอร์ และวงจรอินพุต-เอาต์พุตเพื่อทำงานในฟังก์ชันพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานหน้าที่พิเศษได้ทุกขา



รูปที่2-5 แสดงถึงวงจรภายในของพอร์ตทุกพอร์ต0ก ,(พอร์ต) 1 ข(พอร์ต) 2 ค (และพอร์ต) 3 ง (ในไมโครคอนโทรลเลอร์ MCS-5 1แบบแฟลช)

2.1.5 การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตและเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุตต้องเริ่มต้นด้วยการเขียนข้อมูล “1” มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้นๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น “1” สามารถรับสัญญาณลอจิก “0” จาก

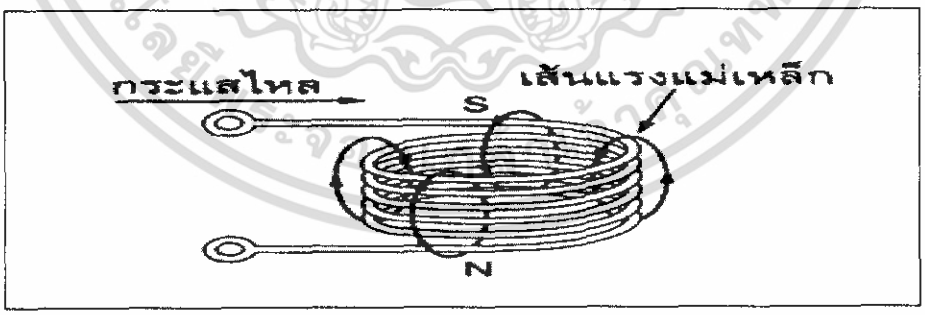
อุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนี้ อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ควงกำหนดให้ทำงานในสภาวะลอจิก "0" จะดีและสะดวกที่สุด ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก "0" แล้ว

2.1.6 การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล "0" ออกไปทางเอาต์พุตก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปยังเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก "0" ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล "1" ออกไป ก็ให้เขียนข้อมูล "1" ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

2.2 โซลินอยด์ไฟฟ้า

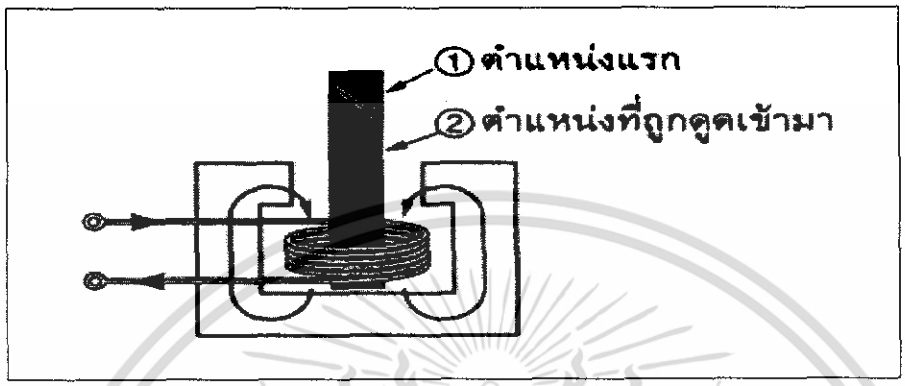
โซลินอยด์ไฟฟ้ามีหลักการทำงานดังนี้ คือ เมื่อมีกระแสไฟฟ้าไหลในขดลวดตัวนำใดๆก็ตาม จะก่อให้เกิดสนามแม่เหล็กขึ้นรอบๆตัวนำนั้น และหากนำเส้นลวดมาขดเป็นวงๆ หลายๆวง ก็จะเกิดลักษณะของขดลวดขึ้น โดยสนามแม่เหล็กที่เกิดจากขดลวดแต่ละขดจะอยู่ในทิศทางเสริมกัน และก่อกำเนิดเป็นเส้นแรงของสนามแม่เหล็กถาวรแท่งหนึ่ง ซึ่งพร้อมที่จะดูดสารแม่เหล็กทันที แต่เนื่องจากสภาพรอบๆ ขดลวดอาจเป็นอากาศ เส้นแรงแม่เหล็กจึงไม่เข้มข้นมากนัก



รูปที่2-6 แสดงถึงทิศทางของสนามแม่เหล็กที่เกิดขึ้นในขดลวดที่มีกระแสไหล

เพื่อไม่ให้สนามแม่เหล็กที่เกิดขึ้นกระจัดกระจาย จึงใส่แกนเหล็กอ่อนรูปตัวซีล้อมรอบๆขดลวดไว้ เพื่อให้สนามแม่เหล็กที่เกิดขึ้นกระจัดกระจายออกไป และเพื่อให้สนามแม่เหล็กมีความเข้มมากขึ้น จากจุดนี้หากนำเอาแกน

กระทุ้ง (Plunger) มาใส่เข้าไปตรงกลางวงของขดลวดในตำแหน่งที่ 1 แกนกระทุ้งจะถูกดูด ให้ลึกเข้ามาจนสนิทในตำแหน่งที่ 2 ความสัมพันธ์ของแรงกับระยะช่วงชักของโซลินอยด์ ในช่วงชักไกลๆจะมีแรงน้อยมาก แต่ในทางตรงกันข้ามช่วงชักที่มีระยะใกล้ๆก็จะมีแรงมากขึ้นเป็นทวีคูณ



รูปที่2-7 แสดงถึงการเพิ่มเหล็กอ่อนเข้ามาเพื่อเพิ่มความเข้มของสนามแม่เหล็ก

2.2.1 ประเภทของโซลินอยด์ไฟฟ้า

โซลินอยด์ไฟฟ้าสามารถแบ่งได้เป็น 2 ประเภท คือ โซลินอยด์ที่ใช้ไฟฟ้ากระแสตรง และโซลินอยด์ที่ใช้ไฟฟ้ากระแสสลับ ข้อแตกต่างระหว่างโซลินอยด์ที่ใช้ไฟฟ้ากระแสตรง และโซลินอยด์ที่ใช้ไฟฟ้ากระแสสลับ คือ โซลินอยด์ที่ใช้ไฟกระแสตรงจะก่อให้เกิดกระแสที่ไหลในขดลวดค่อนข้างคงที่ไม่เปลี่ยนแปลง ไม่ว่าแกนกระทุ้งจะอยู่ในตำแหน่งใดก็ตาม แต่โซลินอยด์ที่ใช้ไฟกระแสสลับจะมีกระแสในขณะที่แกนกระทุ้งอยู่นอกขดลวดค่อนข้างสูง และเมื่อแกนกระทุ้งถูกดูดเข้ามาจนสุดขดลวด กระแสจะลดต่ำลง ด้วยเหตุนี้เองที่ทำให้ต้องระวังอย่าให้เกิดการกระทุ้งในโซลินอยด์ไฟสลับ เพราะจะทำให้เกิดกระแสมากๆไหลค้างอยู่ ทำให้ขดลวดร้อนขึ้น และ อาจจะไหม้เสียหายได้

2.2. 2 การเลือกใช้โซลินอยด์ไฟฟ้า

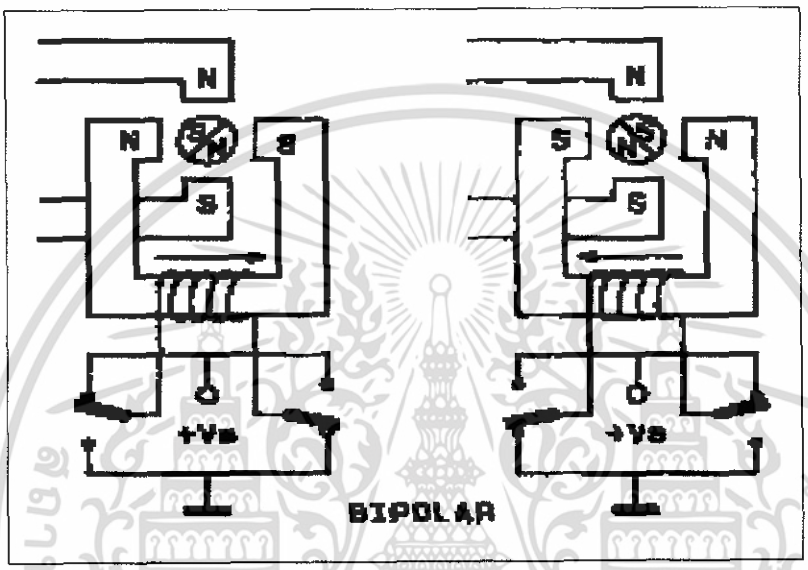
การเลือกใช้โซลินอยด์ไฟฟ้าควรพิจารณาถึงองค์ประกอบต่างๆดังนี้ คือ

1. แรงดันใช้งานซึ่งไม่ว่าจะเป็นไฟฟ้ากระแสตรง หรือไฟฟ้ากระแสสลับก็ตาม จำเป็นจะต้องดูความถี่ใช้งานให้ตรงตามความต้องต้องการด้วย ซึ่งในการออกแบบและสร้างเครื่องเจาะพลาสติกเพื่อใช้เป็นแบบสำหรับทอพรมนี้ใช้โซลินอยด์ที่มีความถี่ 55 เฮิร์ซ

2. ช่วงชักในการใช้งาน (Operating Stroke) ของโซลินอยด์จะต้องเคลื่อนที่เป็นระยะทางเท่าใด มักจะกำหนดเป็นมิลลิเมตร และการสร้างตัวหุ่นนี้จะใช้โซลินอยด์ขนาดเล็กและมีแรงขับที่พอเหมาะด้วย

3 .เป็นการใช้งานอย่างต่อเนื่องหรือไม่ ซึ่งการใช้งานอย่างต่อเนื่องในที่นี้ หมายถึง การใส่แรงดันไฟเข้าขดลวดค้างไว้โดยขดลวดไม่ไหม้ หรือเป็นการใส่แรงดันแบบเป็นจังหวะๆ เพื่อใช้ในการเตะลูกฟุตบอลจะต้องเลือกคือต้องเล็กและมีแรงในการเตะที่แรงด้วย

2.3 มอเตอร์



รูปที่2-8 แสดงวงจรของมอเตอร์เมื่อทิศทางการหมุนที่ต่างกัน

2.3.1 ประเภทของมอเตอร์

ตามปกติโดยทั่วแล้ว Stepper Motor ที่มีอยู่ในปัจจุบันนี้สามารถจำแนกออกตามลักษณะโคลงสร้างได้ 3 แบบใหญ่ๆด้วยกันคือ

1.) แบบแม่เหล็กถาวร (Permanent Magnet Stepper Motor หรือ PM)

Stepper Motor แบบ PM นี้จะใช้ส่วนของ Stator สำหรับพันขดลวดไว้หลายๆ โพล (Pole) และจะมี Rotor เป็นรูปทรงกระบอกฟันเลื่อย ซึ่งส่วนของ Rotor นี้จะทำด้วยแม่เหล็กถาวร โดยเมื่อเราป้อนไฟฟ้ากระแสตรงให้กับขดลวดที่พันอยู่บนส่วนของแกน Stator จะทำให้เกิดสนามแม่เหล็กไฟฟ้าขึ้นในขดลวด Stator และสนามแม่เหล็กไฟฟ้าที่เกิดขึ้นนี้จะทำให้เกิดแรง ดัน-ผลัก กับขั้วแม่เหล็กของส่วน Rotor จึงส่งผลให้แกน Rotor ของมอเตอร์สามารถเคลื่อนที่ไปได้ ซึ่ง Stepper Motor แบบ PM นี้จะมีแรงฉุดยึดให้ส่วน Rotor หยุดอยู่กับที่ได้ ถึงแม้ว่าจะยังไม่ได้จ่ายไฟให้กับขดลวดของมอเตอร์ก็ตามข้อดีของมอเตอร์แบบนี้คือให้แรงบิดสูง แต่ระยะการเคลื่อนที่ไม่ละเอียดมากนัก

2.) แบบแปรผันค่ารีลัคแตนซ์ (Variable Reluctance Stepper Motor หรือ VR)

Stepper Motor แบบ VR นี้ ส่วนของ Rotor สามารถเคลื่อนที่ได้อย่างอิสระ โดย Rotor จะทำขึ้นจากสารแม่เหล็กกำลังอ่อน (Fource Low Magnetic) มีลักษณะเป็นทรงกระบอกฟันเลื่อย ซึ่งจะมีความสัมพันธ์โดยตรงกับจำนวนโพล (Pole) ของขดลวดใน Stator จะทำให้เกิดแรงบิดเพื่อไปหมุน Rotor ให้เคลื่อนที่ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่า Reluctance ต่ำที่สุด จึงทำให้มอเตอร์แบบนี้มีความเร็วสูงและมีตำแหน่งของการเคลื่อนที่มีความแม่นยำสูงและมีเสถียรภาพดีแต่มีแรงบิดน้อยกว่าแบบ PM

3.) แบบผสม (Hybrid Stepper Motor หรือ H)

Stepper Motor แบบนี้ เป็นการนำเอาข้อดีของมอเตอร์ แบบ PM และ VR มารวมเข้าด้วยกันโดยจะออกแบบให้มีส่วนของ Stator มีลักษณะคล้ายกับของมอเตอร์แบบ VR สำหรับในส่วนของ Rotor จะมีหมวกหุ้มปลายทำด้วยสารแม่เหล็กกำลังสูง โดยการออกแบบจะควบคุมขนาดของรูปร่างของหมวกแม่เหล็กอย่างดี เพื่อให้ได้มุมการหมุนของมอเตอร์ในแต่ละครั้งที่ละเอียดและแม่นยำที่สุดข้อดีของมอเตอร์แบบนี้คือให้แรงบิดสูง มีขนาดกระทัดรัด มีระยะการหมุนที่ละเอียดแม่นยำ มีแรงฉุดยึดให้ แกน Rotor หยุดนิ่งอยู่กับที่ได้ดี ถึงแม้ว่าจะยังไม่ได้ ถึงแม้ว่าจะยังไม่ได้จ่ายกระแสไฟฟ้าให้กับขดลวดของมอเตอร์เลยก็ตาม แต่มอเตอร์แบบนี้มักมีราคาสูงกว่ามอเตอร์แบบนี้มักมีราคาสูงกว่ามอเตอร์แบบอื่นๆ

2.3.2 การเลือกใช้มอเตอร์

จะเห็นได้ว่า Stepper Motor ทั้ง 3 ประเภทนี้ ต่างก็มีข้อดีข้อเสียต่างกันออกไป การที่เราจะตัดสินใจเลือกใช้ Motor แบบใดนั้นคงต้องพิจารณาจากจุดแระสงค์หลักนการใช้งานที่ต้องการแรงบิดสูงๆ ก็ควรเลือกใช้มอเตอร์แบบแม่เหล็กถาวร (PM) แต่ถ้าต้องการใช้กับงานที่ต้องการความแม่นยำในการเคลื่อนที่มากๆก็ควรเลือกใช้แบบแปรผันค่ารีลัคแตนซ์ (VR) และในทำนองเดียวกัน ถ้าต้องการทั้งแรงบิดและความละเอียดแม่นยำในการเคลื่อนที่ด้วยก็ต้องเลือกใช้แบบ ผสม (HYBRID) เป็นต้น

2.4 การควบคุม Bipolar Stepper Motor ด้วย L298

Driver Motor ซึ่งสามารถต่องานร่วมกับบอร์ดไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์ต่างๆ ได้อย่างง่ายดาย โดยที่ตัวบอร์ดเองต้องการสัญญาณในการควบคุมการทำงานทั้งหมด 6 เส้นสัญญาณ โดยสามารถรับสัญญาณลอจิกแบบ TTL มาตราฐานได้โดยตรงซึ่งบอร์ดสามารถขับกระแสได้กับ Stepper Motor แบบ Bipolar Stepper Motor ได้ทันทีโดยไม่ต้องดัดแปลงใดๆ ซึ่งเราได้ทราบถึงเทคนิควิธีการควบคุม Stepper Motor แบบ ต่างๆ แล้วสำหรับในส่วนนี้จะเป็นการแนะนำการสร้างสัญญาณเพื่อควบคุมการทำงานของมอเตอร์ให้ได้ตามต้องการด้วยโปรแกรม

โดยในส่วนของวงจรขับกระแสให้กับมอเตอร์นั้นจะใช้ IC เบอร์ L298N ของ SGS-THOMSON ซึ่งสามรถแบ่งสัญญาณการควบคุมการทำงานออกเป็น 2 ชุด คือ

- EN1 ,INPUT-A, INPUT-B สำหรับขับกระแสให้กับขดลวดขดที่ 1 ของมอเตอร์
- EN2, INPUT-C, INPUT-D สำหรับขับกระแสให้กับขดลวดขดที่ 2 ของมอเตอร์

สัญญาณ EN1 เป็นสัญญาณควบคุมหลักทำงานที่ลอจิก "1" เพื่อใช้สำหรับเปิด-ปิดการไหลของกระแส ของ OUTPUT A และ B

สัญญาณ EN2 เป็นสัญญาณควบคุมหลักทำงานที่ลอจิก "1" เพื่อใช้สำหรับเปิด-ปิดการไหลของกระแส ของ OUTPUT C และ D และสามารถแสดงให้เห็นความสัมพันธ์ของสัญญาณของสัญญาณต่างๆ ได้ดังตาราง

EN1	INPUT-B	INPUT-A	การทำงานของมอเตอร์
"1"	"0"	"0"	ไม่มีกระแสไหล มอเตอร์หยุดหมุน
	"0"	"1"	กระแสไหลจาก A ไป B
	"1"	"0"	กระแสไหลจาก B ไป A
	"1"	"1"	ไม่มีกระแสไหล มอเตอร์หยุดหมุน
"0"	X	X	ไม่มีกระแสไหล มอเตอร์หยุดหมุน

ตารางที่ 2-1 แสดงถึงการควบคุมมอเตอร์ช่องการไหลที่ 1 โดยใช้ L298N

EN2	INPUT-C	INPUT-D	การทำงานของมอเตอร์
"1"	"0"	"0"	ไม่มีกระแสไหล มอเตอร์หยุดหมุน
	"0"	"1"	กระแสไหลจาก C ไป D
	"1"	"0"	กระแสไหลจาก D ไป C
	"1"	"1"	ไม่มีกระแสไหล มอเตอร์หยุดหมุน
"0"	X	X	ไม่มีกระแสไหล มอเตอร์หยุดหมุน

ตารางที่2 -2 แสดงถึงการควบคุมมอเตอร์ช่องการไหลที่ 2 โดยใช้ L298N

บทที่ 3

การประยุกต์ใช้ไมโครคอนโทรลเลอร์ MCS-51 ด้วยภาษาซี

การเขียนโปรแกรมสำหรับไมโครคอนโทรลเลอร์ด้วยภาษาแอสเซมบลีนั้นมีโครงสร้างของโปรแกรมจะดูยาก การควบคุมทิศทางของโปรแกรมก็เข้าใจได้ยาก แม้แต่ผู้ที่เขียนโปรแกรมเองเมื่อเวลาผ่านไปแล้วกลับมาดูโปรแกรมใหม่ก็อาจดูงานที่ตัวเองเขียนได้ไม่เข้าใจ แม้ว่าจะเขียนคำอธิบายไว้ทุกบรรทัดก็ตาม ในอดีตการเขียนโปรแกรมด้วยภาษรซีสำหรับไมโครคอนโทรลเลอร์นั้นเมื่อแปลออมาเป็นรหัสของภาษาเครื่องจะได้รหัสที่มีขนาดใหญ่กว่าการเขียนด้วยภาษาแอสเซมบลีมาก แต่ปัจจุบันคอมไพเลอร์ส่วนใหญ่จะแปลภาษาออกมาได้รหัสที่มีขนาดเล็กเกือบใกล้เคียงกับภาษาแอสเซมบลี การเขียนโปรแกรมด้วยภาษาซีสามารถนำไปใช้กับงานที่ซับซ้อนได้ เช่น งานควบคุมแบบป้อนกลับ (close loop control) การใช้ไมโครคอนโทรลเลอร์ควบคุมเครื่องมือวัดต่างๆ (microcontroller based instrumentation) และการใช้ไมโครคอนโทรลเลอร์ในการควบคุมหุ่นยนต์ เป็นต้น

3.1 โครงสร้างภาษาซี

ด้วยภาษาซีเป็นภาษาที่สามารถเขียนโปรแกรมเป็นแบบโครงสร้างได้ โดยโปรแกรมจะแบ่งการทำงานต่างๆ ออกเป็นกลุ่ม ๆ หรือ ฟังก์ชัน โดยฟังก์ชันเหล่านั้นสามารถเรียกขึ้นมาใช้ใหม่ได้ ในการเขียนโปรแกรมจะต้องระบุไว้ว่าในโปรแกรมนั้นมีฟังก์ชันใดให้ใช้บ้าง แต่ทุกโปรแกรมจะต้องมีฟังก์ชันหลักที่ชื่อว่า main() เสมอ พิจารณาตัวอย่างโครงสร้างของโปรแกรมต่อไปนี้

```
#include<reg51.h>              /*Preprocessor*/
void func1 (void);             /*Por to type*/
int func2 (int x);
void main ()                   /* ฟังก์ชันหลัก*/
{
        int a;                 /*ประกาศตัวแปร*/
        P1 = 0x0FF;
        func1();               /*เรียกใช้ฟังก์ชัน*/
        a = func2 (4);         /*เรียกใช้ฟังก์ชันที่มีการส่งค่า*/
        P1 = a;
}
void func1(void)               /*ฟังก์ชันที่ไม่มีการส่งค่า*/
{
```

```
        **********
        **********
}
int func2(intx)                 /*ฟังก์ชันที่มีการส่งค่า*/

{
        return(x*2);
}
```

จากโปรแกรมพบว่า ส่วนประกาศโพรโตไทป์จะบอกว่าโปรแกรมนี้มีฟังก์ชันชื่อ func1 ให้ใช้งานและฟังก์ชันนี้จะทำงานเป็นโปรแกรมย่อยเพราะมี void นำหน้า และมีฟังก์ชันชื่อ func2(intx) ซึ่งรับค่าเข้าไปผ่านทางตัวแปร x และคืนค่าออกมาเป็นจำนวนเต็ม

3.2 ตัวแปรและค่าคงที่

การใช้งานตัวแปรและค่าคงที่ต่างๆ จะต้องมีการประกาศชื่อตัวแปรขึ้นมาเสียก่อนเมื่อมีการคอมไพล์โปรแกรม ตัวคอมไพเลอร์จะเตรียมพื้นที่ในหน่วยความจำแรมเอาไว้สำหรับเก็บตัวแปรและค่าคงที่เหล่านั้นในการประกาศตัวแปรสามารถทำได้ดังนี้

ประเภทของข้อมูล ชื่อตัวแปร[,.....,];

การประกาศตัวแปรจะต้องเริ่มด้วยชื่อประเภทของข้อมูล และตามด้วยชื่อตัวแปรโดยจะประกาศครั้งละกี่ตัวก็ได้ ส่วนชื่อของตัวแปรนั้นจะซ้ำกันไม่ได้และต้องไม่ซ้ำกับชื่อของคำสงวน (keywords) ของคอมไพเลอร์ตัวนั้นๆ สำหรับประเภทของข้อมูลในการประกาศตัวแปรดังนี้

ประเภทของข้อมูล	ขนาด (บิต)	ค่าที่เก็บได้
bit	1	0 ถึง 1
char	8	-128 ถึง 127
unsigned char	8	0 ถึง 255

ตารางที่ 3-1 แสดงการประกาศตัวแปลต่างๆ

ประเภทของข้อมูล	ขนาด (บิต)	ค่าที่เก็บได้
long	32	-2147483648 ถึง -2147483647
unsigned long	32	0 ถึง 4294967295
float	32	-1.17549e-38ถึง 3.402823 e+38
unsigned int	16	0 ถึง 65535
int	16	-32768 ถึง 32767

ตารางที่ 3-1 แสดงการประกาศตัวแปลต่างๆ (ต่อ)

สำหรับคำสงวนเป็นคำที่คอมไพเลอร์รู้จักและจะถูกใช้งานเฉพาะ เราไม่สามารถนำมาตั้งเป็นชื่อตัวแปรและฟังก์ชันได้ คำสงวนของโปรแกรม C51 เป็นดังต่อไปนี้

_at_	idata	sfr
alien	interrupt	sfr16
bdata	large	small
bit	pdata	_task_
code	_priority_	using
compact	reentrant	Xdata
data	sbit	

3.3 ตัวดำเนินการในภาษาซี

ตัวดำเนินการจะเป็นตัวที่ใช้กระทำกับตัวแปร ค่าคงที่ต่างๆ ให้รวมเป็นค่าเดียวกันโดยอาจกระทำทางคณิตศาสตร์หรือกระทำทางลอจิกก็ได้ ในการเขียนโปรแกรมด้วยภาษีนั้น ตัวดำเนินการจะแบ่งออกเป็นสองกลุ่มใหญ่ๆ คือตัวดำเนินการที่กระทำกับตัวถูกกระทำตัวเดียว (single operand operators) และตัวดำเนินการที่กระทำกับตัวถูกกระทำสองตัว (two operands operators)

3.3.1 ตัวดำเนินการที่กระทำกับตัวถูกกระทำตัวเดียว

ตัวดำเนินการประเภทนี้จะกระทำกับตัวถูกกระทำเพียงตัวเดียว ประกอบด้วยตัวดำเนินการต่างๆ ดังนี้

- `-` ลบ (negate)
- `~` กลับค่าลอจิกของบิตข้อมูล (bit wise complement)
- `!` กลับค่าทางลอจิก (logical complement)
- `++` เพิ่มค่าขึ้นหนึ่งค่า (increment)

-- ลดค่าลงหนึ่งค่า (decrement)

* ตัวดำเนินการทางพอยน์เตอร์

& ตำแหน่งหน่วยความจำของตัวแปร

3.3.2 ตัวดำเนินการที่กระทำกับตัวถูกกระทำสองตัว

ตัวดำเนินการแระเภทนี้จะกระทำกับตัวถูกกระทำสองตัว ถ้าหากมีตัวถูกกระทำหลายๆ ตัวสามารถนำมาเขียนรวมกันเป็นประโยคได้ ซึ่งประกอบไปด้วยตัวดำเนินการที่ใช้กำหนดค่า ตัวดำเนินการทดสอบค่าซึ่งจะให้ผลลัพธ์เป็นค่าทางลอจิก (จริง, เท็จ) ตัวดำเนินการทางคณิตศาสตร์ และตัวดำเนินการทางลอจิกดังต่อไปนี้

= กำหนดค่าในประโยค (assignment)

+ บวก

- ลบ

* คูณ

/ หาร (division)

% หารแบบบวก (modulo)

&& การแอนด์ (logical AND)

|| การออร์ (logical OR)

& การแอนด์แบบบิตต่อบิต (bit wise AND)

| การออร์แบบบิตต่อบิต (bit wise OR)

^ การเอ็กคลูซีฟออร์แบบบิตต่อบิต (bit wise exclusive OR)

<< เลื่อนบิตไปทางซ้าย

>> เลื่อนบิตไปทางขวา

= = ทดสอบว่าเท่ากันหรือไม่

!= ทดสอบว่าไม่เท่ากันหรือไม่

> ทดสอบว่ามากกว่าหรือไม่

< ทดสอบว่าน้อยกว่าหรือไม่

>= ทดสอบว่ามากกว่าหรือเท่ากันหรือไม่

<= ทดสอบว่าน้อยกว่าหรือเท่ากันหรือไม่

3.4 ประโยคควบคุมในภาษาซี

การทำงานของโปรแกรมนั้นจะทำคำสั่งแต่ละคำสั่งเรียงลำดับกันไป และเราสามารถให้โปรแกรมตัดสินใจในการเลือกทำได้ หรือให้ทำงานใดงานหนึ่งซ้ำๆ ตามเงื่อนไขที่กำหนดได้โดย ใช้คำสั่งควบคุม ในภาษาซีนั้น จะมี ประโยคคำสั่งควบคุมที่ใช้ในการเลือกทำและทำงานซ้ำๆ ดังนี้

3.4.1 ประโยค IF/ELSE

```
if(expression)
        statement;
```

ถ้าหากเป็นการทำงานเลือกทำแบบมีสองทางเลือก และต้องการทำงานเพียงอย่างใดอย่างหนึ่งจะใช้ประโยค if-else ซึ่งมีรูปแบบดังนี้

```
if (expression)
        statement 1;
else
        Statement 2;
```

3.4.2 ประโยค switch

การเลือกทำที่มีทางเลือกหลายๆ ทางเลือกนั้นเราสามารถนำประโยค if-else มาซ้อนกันได้ แต่ทำให้มองดูเข้าใจยาก ในภาษาซีจะมีประโยค switch ที่ใช้ในการเลือกทำอย่างใดอย่างหนึ่งจากหลายๆทางเลือกโดยมีรูปแบบของประโยคดังนี้

```
        Switch(k)
        {
              case 1:   statement 1;
                        break;
             case2:     statement 2;
                        Break;
             case3:     statement 3;
                        break;
                  :     :
                  :     :
                  :     :
            default:    statement n;
}
```

3.5 การทำซ้ำ

คำสั่งให้โปรแกรมทำงานซ้ำถือว่าเป็นประโยคคำสั่งควบคุมอย่างหนึ่ง การทำซ้ำหรือที่เรียกว่าการทำลูปนั้นจะมีประโยคคำสั่งอยู่ 3 ประเภทคือ for, while และ do-while ซึ่งแต่ละแบบจะต่างกันตรงเงื่อนไขของการทำซ้ำ

3.5.1 ประโยค for

ประโยคคำสั่งนี้จะใช้ในกรณีที่มีคำจำนวนรอบของการทำซ้ำที่แน่นอน โดยมีรูปแบบดังนี้

```
for (initialization : condition : increment)
statement ;
```

โดยที่ initialization เป็นค่ากำหนดเริ่มต้นให้กับตัวแปรของการทำลูป condition เป็นเงื่อนไขที่ใช้ทดสอบการทำซ้ำครั้งต่อไป ซึ่งจะเป็นการกระทำลอจิก increment เป็นการเพิ่มค่าให้ตัวแปรในการทำซ้ำแต่ละครั้ง สำหรับ statement จะกเป็นสเตตเมนต์ของคำสั่งที่จะทำซ้ำ ซึ่งอาจเป็นสเตตเมนต์รวมก็ได้

3.5.2 ประโยค while

การทำซ้ำแบบนี้จะตรวจสอบเงื่อนไขก่อนการทำซ้ำ ถ้าเงื่อนไขเป็นจริงจะทำสเตตเมนต์ทีกำหนด และทดสอบเงื่อนไขใหม่ ถ้าเงื่อนไขเป็นเท็จจะออกจากการทำซ้ำทันที โดยมีรูปแบบดังนี้

```
while (expression)
        statement 1;
```

3.5.3 ประโยค do – while

การทำซ้ำประเภทนี้จะตรวจสอบเงื่อนไขภายหลังการทำสเตตเมนต์แต่ละครั้ง ถ้าหากเงื่อนเป็นเท็จจะออกจากการทำซ้ำทันทีโดยมีรูปแบบดังนี้

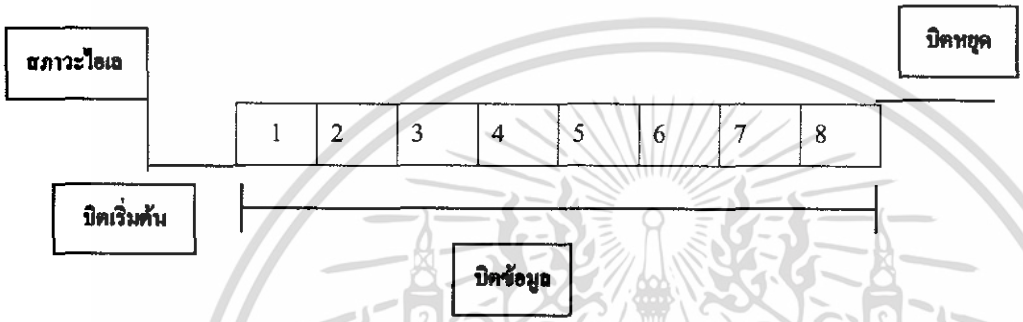
```
do {
        statement ;
    }
while (condition);
```

3.6 Microcontroller เขียนโปรแกรมควบคุม 8051 ผ่าน Serial Port

โดยปกติแล้ว MCS - 51 จะเป็นไมโครคอนโทรเลอร์ที่มีความสามารถในการ รับข้อมูลจากภายนอกและนำมาประมวลผล พร้อมทั้งสามารถส่งสัญญาณ เพื่อทำการควบคุมการทำงานของอุปกรณ์ต่างๆ ได้อย่างดี และในส่วนของการติดต่อสื่อสารข้อมูล (Data Communication) กับระบบภายนอกอื่นๆ ก็สามารถกระทำ โดยผ่านทางพอร์ทอนุกรม (Serial Port) ซึ่งพอร์ทอนุกรมนี้ จะเป็นส่วนที่เหมาะสมในการรับ หรือส่งข้อมูลในระยะทางไกลได้ดีกว่าพอร์ทขนาน

3.6.1 การส่งข้อมูลแบบอนุกรมในโหมด 1 (Standard UART)

การส่งข้อมูลในโหมดนี้จะเป็นการส่งข้อมูลแบบอะซิงโครนัส 8 บิตข้อมูล 1 บิตเริ่มต้นและ 1 บิตหยุดดังรูปที่ 5.1 การทำงานในโหมดนี้จะทำโดยการกำหนดข้อมูลในรีจีสเตอร์ SCON บิต SM0 และบิต SM1 ให้มีค่าเป็น "01" ซึ่งเป็นการกำหนดให้รีจีสเตอร์ SBUF กลายเป็นตัวรับส่งข้อมูลขนาด 10 บิตแบบฟูลดูเพลกซ์ (Full Duplex) ซึ่งสามารถรับและส่งข้อมูลได้ภายในเวลาเดียวกัน โดยใช้ขา RxD ทำหน้าที่รับสัญญาณ อนุกรมที่เข้ามา และขา TxD ทำการส่งข้อมูล แบบอนุกรมไปภายนอก



รูปที่ 3-1 แสดงการส่งข้อมูลของพอร์ตอนุกรม

การส่งข้อมูลจะเริ่มต้นด้วยการส่งบิตเริ่มต้น (Start bit) ออกไปแล้วตามด้วยบิตข้อมูล (โดยส่งบิต 0 ออกไปก่อน) จากนั้นจึงเป็นการส่งบิตหยุด (Stop bit) แฟล็ก TI จะเซ็ตเมื่อส่งข้อมูลครบทั้ง 10 บิต

การรับข้อมูลจะเริ่มจากลอจิกในสายสัญญาณเปลี่ยนสภาวะจาก 1 เป็น 0 (ขอบบวกของบิตเริ่มต้น) บิตเริ่มต้นจะถูกข้ามไปไม่สนใจ จะสุ่มเอาข้อมูลอีก 8 บิตที่เหลือเข้ารีจีสเตอร์เลื่อนบิตภายในพอร์ทอนุกรมเมื่อครบทั้ง 8 บิตแล้ว สิ่งต่อไปนี้จะเกิดขึ้น

1. บิต 9 (บิตหยุด) จะถูกเก็บเข้าไปในบิต RB8 ของ SCON
2. SBUF จะทำการโหลดข้อมูลทั้ง 8 บิตเข้าตัวเอง
3. แฟล็ก RI เซ็ต

สิ่งที่กล่าวมาทั้งหมดจะเป็นจริงต่อเมื่อ

1. RI = 0
2. SM2 = 1 และบิตหยุดที่รับเข้ามาเป็น 1 หรืออีกกรณีหนึ่งคือ SM2 = 0

3.6.2 การคำนวณอัตราบอดของโหมด 1

ไทม์เมอร์ 1 จะถูกใช้เป็นตัวสร้างอัตราบอด เมื่อกำหนดการรับส่งข้อมูลแบบอนุกรมในโหมด 1 โดยไมม์เมอร์ 1 จะถูกใช้ทำงานในโหมด 2 (โหลดค่าเข้าอัตโนมัติ) การคำนวณจะเป็นดังนี้

$$f_{buad} = \frac{2^{SMOD} \times Oscillator\_frequency}{32d \times 12d \times [256d - (TH1)]} \quad \text{.....[3.1]}$$

เมื่อ SMOD เป็นบิทควบคุมอยู่ในรีจีสเตอร์ PCON ซึ่งเป็น 0 หรือ 1 ก็ได้ ถ้าเป็น 0 จะเป็นความถี่ปกติ ถ้าเป็น 1 ความถี่จะเพิ่มขึ้นเป็น 2 เท่าได้ดังกล่าวมาแล้วในรีจีสเตอร์ PCON

ถ้าไทเมอร์ 1 ไม่ถูกใช้งานในโหมด 2 ค่าอัตราบอดจะเป็น

$$f_{buad} = \frac{2^{SMOD} \times (timer1\_overflow\_frequency)}{32_D} \quad \text{.....[3.2]}$$

ถ้าต้องการสร้างอัตราบอดมาตรฐาน ที่ 9600 ความถี่ของคริสตอลที่ใช้คำนวณเป็นมาตรฐานคือ 11.0592MHz ค่าที่ต้องกำหนดลงไปใน TH1 คือ

$$TH1 = 256_D - \frac{2^0 \times 11.0592 \times 10^6)}{(32_D \times 12_D \times 9600_D)} = 253.00_D \quad \text{.....[3.3]}$$

บทที่ 4
โปรแกรมวิชวลเบสิค

วิชวลเบสิก (Visual Basic) เป็นโปรแกรมที่ใช้สร้างโปรแกรมประยุกต์ สำหรับระบบปฏิบัติการวินโดว์ (Windows) เนื่องจากความง่ายของภาษาที่ใช้ในการเขียนและวิชวลเบสิกเป็นโปรแกรมที่เน้นในการสร้างแอพพลิเคชั่นเพื่อติดต่อกับผู้ใช้งานเป็นหลักทำให้ได้ผลงานออกมารวดเร็วสวยงาม และนอกจากนี้วิชวลเบสิกยังมีเครื่องมือที่ใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมซึ่งนำไปใช้ในการควบคุมเครื่องอีกด้วย

4.1 ขั้นตอนการสร้างโปรแกรมประยุกต์

การสร้างโปรแกรมประยุกต์ วิชวลเบสิก ประกอบด้วยขั้นตอนหลัก 3 ขั้นตอน คือ

1. การสร้างอินเตอร์เฟส (Interface) โดยมีฟอร์ม (Form) เป็นอ๊อบเจค (Object) พื้นฐานและเป็นที่วางตัวคอนโทรล (Control) สำหรับการติดต่อกับผู้ใช้
2. ตั้งค่าคุณสมบัติ เป็นการกำหนดพฤติกรรมและการทำงานให้กับอ๊อบเจคต่างๆ
3. การเขียนคำสั่ง เป็นการควบคุมการประมวลผลผ่านโพรซีเจอร์ (Procedure) ที่กำหนด

4.2 คอนโทรลพื้นฐาน

กลุ่มคอนโทรลพื้นฐานของวิชวลเบสิกประกอบด้วย Text Box, Label, Command Button, Picture Box, Image, Check Box, Frame เป็นต้น

ไอคอน	ชื่อตัว Control	ชื่อ Class	คำอธิบาย
	Check box	CheckBox	ใช้กับการเลือกแบบ ถูก/ผิด)True/False, Yes/No)
	Combo box	ComboBox	เป็นตัว Control เป็นการผสมระหว่าง Text box กับ List box ซึ่งจะปรากฏรายการ เมื่อมีการคลิกลูกศร และ Combo box ไม่สนับสนุนการเลือกแบบหลายค่า
	Command button	CommandButton	ปุ่มคำสั่งเป็นตัว Control ที่ใช้ในทุกฟอร์ม ตามปกติจะเขียนคำสั่งใน Click Event Procedure ของตัว Control นี้

ตารางที่ 4-1 แสดงถึงคอนโทรลพื้นฐานของวิชวลเบสิกที่ใช้ในการสร้างโปรแกรม

ไอคอน	ชื่อตัว Control	ชื่อ Class	คำอธิบาย
	Data	Data	เป็นตัว Control ที่สามารถรวมข้อมูลกับฐานข้อมูลได้ และเป็นส่วนที่ Visual Basic ให้ผู้ใช้สามารถติดต่อระหว่างตัว Control บนฟอร์มกับฟิลด์ใน Table ของฐานข้อมูล โดย Data จะทำงานกับ Database Jet ของฐานข้อมูล แต่ไม่สามารถทำงานกับ ActiveX Data Object (ADO) ได้
	Directory List box	DirListBox	เป็น List box แบบหนึ่ง ที่แสดงไดเรคทอรีและพาร์ทที่เลือก
	Drive List box	DriveListBox	คล้ายกับ Combo box ที่ใช้เลือกชื่อของไดร์ฟในระบบ
	File list box	FileListBox	เป็น List box ชนิดพิเศษที่ใช้แสดงชื่อ ไฟล์ในไดเรคทอรี
	Frame	Frame	สามารถใช้เป็น Container สำหรับตัว control อื่น
 	Horizontal และ Vertical Scroll Bar	HScrollBar และ VScrollBar	ใช้เป็นแถบเลื่อนแบบ Stand-alone แต่มักจะไม่ค่อยมีการใช้ เพราะตัว Control อื่น ๆ ส่วนใหญ่ จะมีแถบเลื่อนของตัวเอง แถบเลื่อนแบบ Stand-alone อาจจะใช้ในลักษณะ Slider ได้
	Image	Image	เป็นตัว Control ใช้เก็บภาพคล้ายกับ Picture Box แต่ไม่สามารถทำงานแบบ Container ได้ Image มีข้อดีที่ใช้ทรัพยากรของระบบน้อยกว่า Picture Box
	Label	Label	เป็นตัว Control ที่ใช้แสดงข้อความ หรือป้ายชื่อ
	Line	Line	เป็นตัว Control ใช้สำหรับการตกแต่งด้านกราฟฟิก
	List box	ListBox	เป็นตัว Control ที่เก็บรายการของค่า และให้ผู้ใช้เลือก ซึ่งสามารถเป็นการเลือกค่าเดียวหรือหลายค่า ขึ้นกับการกำหนดคุณสมบัติ MultiSelect

ตารางที่ 4-1 แสดงถึงคอนโทรลพื้นฐานของวิชวลเบสิกที่ใช้ในการสร้างโปรแกรม (ต่อ)

ไอคอน	ชื่อตัว Control	ชื่อ Class	คำอธิบาย
	OLE container	OLE	เป็นตัว Control ที่สามารถเป็น Host Window ให้กับโปรแกรมภายนอก เช่น Microsoft Excel หรืออาจจะกล่าวว่าเป็นการสร้าง Window ให้กับโปรแกรมอื่นบนโปรแกรมประยุกต์ Visual Basic
	Option button	OptionButton	เป็นตัว Control ใช้กับกลุ่มตัว Control โดยให้เลือกได้เพียงตัว Control เดียวต่อครั้งหนึ่ง เมื่อมีการเลือกตัว Control ในกลุ่มแล้ว ตัว Control อื่นในกลุ่มจะเปลี่ยนจากการเลือกโดยอัตโนมัติ ถ้ามีการใช้ Option Button มากกว่า 2 กลุ่ม ต้องวางแต่ละกลุ่มใน Container เช่น Frame
	Picture box	PictureBox	ใช้แสดงภาพในฟอร์แมต BMP, DIB (Bitmap), ไอคอน)Icon), WMF (Metafile), GIF และ JPEG เป็นต้น และสามารถใช้เป็น container สำหรับตัว Control อื่น
	Shape	Shape	เป็นตัว Control ใช้สำหรับการตกแต่งด้านกราฟฟิก
	Text box	TextBox	เป็นตัว Control ที่เป็นฟิลด์ ใช้เก็บตัวอักษรที่สามารถแก้ไขโดยผู้ใช้ได้ และได้รับการใช้งานมาก
	Timer	Timer	เป็นตัว Control พิเศษที่ไม่เห็นเมื่อเวลาเรียกใช้ วัตถุประสงค์การใช้คือการสร้าง Event ในฟอร์มแม่ โดยการเขียนคำสั่งใน Procedure ที่เจาะจงสำหรับการทำงานเบื้องหลัง เช่น การตรวจสถานะของอุปกรณ์ต่อพ่วง

ตารางที่ 4-1 แสดงถึงคอนโทรลพื้นฐานของวิชวลเบสิกที่ใช้ในการสร้างโปรแกรม (ต่อ)

4.3 คุณสมบัติร่วม

1. Left, Top, Width, Height ใช้จัดตำแหน่งของอ๊อบเจค เช่น การวางฟอร์มที่มุมซ้ายบน มีความกว้าง 5000 Twips และสูง 3000 Twips

```
Form1.Left = 0
Form1.Top = 0
Form1.Width = 5000
Form1.Height = 3000
```

2. ForeColor และ BackColor ใช้กำหนดสีของข้อความและสีพื้นหลังของอ๊อบเจค ซึ่งสามารถกำหนดสีเป็นแบบ Palette และ system เช่น

กำหนดสีแบบ Palette Label1.ForeColor = vbHighlightText

กำหนดสีแบบ System Label1.BackColor = &H800000D

รวมถึงการกำหนดด้วยค่าคงที่, เลขฐานสิบ และเลขฐานสิบหก เช่นตัวอย่างเป็นสีเดียวกัน

ค่าคงที่ Text1.BackColor = vbCyan

เลขฐานสิบ Text1.BackColor = 16776960

เลขฐานสิบหก Text1.BackColor = &HFFFF00

3. Font ใช้กำหนดลักษณะการแสดงตัวอักษร การตั้งค่าคุณสมบัติ Size, Bold, Italic, Underline และ Strikethrough เช่น

```
Text1.Font.Name = "Arial"
Text1.Font.Size = 12
Text1.Font.Bold = True
```

4. Caption, Text ใช้ในการแสดงข้อความ โดยคุณสมบัติ Caption เป็นข้อความที่ปรากฏภายในตัวคอนโทรล และไม่สามารถแก้ไขได้โดยผู้ใช้เมื่อเรียกใช้โปรแกรม เช่น

```
Label1.Caption = "Title"
Text1.Text = "Hello Word"
```

5. Enabled, Locked และ Visible โดยคุณสมบัติ Visible ใช้กำหนดการมองเห็น คุณสมบัติ Enabled และ Locked กำหนดการเข้าถึงตัว Control แต่ต่างกันที่ คุณสมบัติ Enabled ถ้ากำหนดไม่ให้เข้าถึง)Enabled = False) แล้ว ตัว Control ไม่สามารถรับโฟกัสได้ ส่วนคุณสมบัติ Locked ถ้ากำหนดไม่ให้เข้าถึง)Locked = True) แล้ว ตัว Control ยังสามารถรับโฟกัสได้

6. MousePointer และ MouseIcon ใช้กำหนดไอคอนของเมาส์เมื่อเคลื่อนที่ผ่านตัวคอนโทรล เช่น

```
Text1.MousePointer = vbCrosshair
```

7. คุณสมบัติอื่นๆ ได้แก่

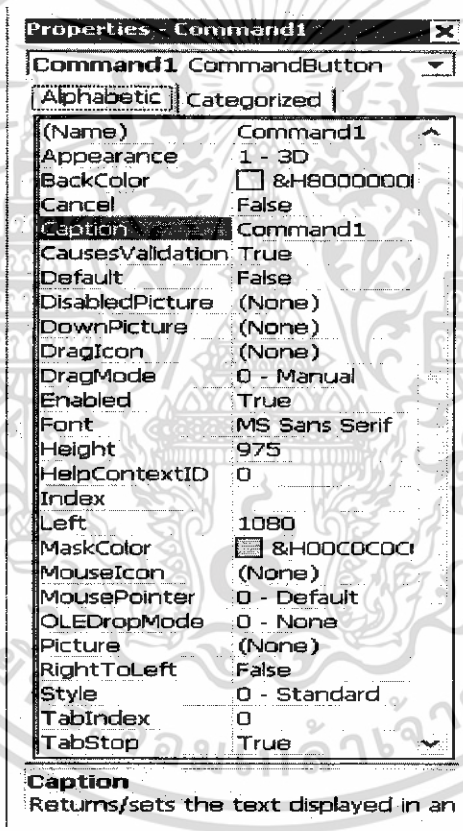
- คุณสมบัติ Value มีค่าตามชนิดของตัวคอนโทรล

- คุณสมบัติ Index ใช้ในการสร้าง array ของตัวคอนโทรลและมีคุณสมบัติแบบอ่านอย่างเดียว

- คุณสมบัติ Appearance เป็นลักษณะหน้าตาของคอนโทรล ตามปกติจะกำหนดค่าคุณสมบัติเริ่มต้นเป็น 3 มิติ การกำหนดค่าทำได้ในเวลาออกแบบ และเป็นแบบอ่านอย่างเดียว เมื่อมีการเรียกใช้

- คุณสมบัติ Border Style คือคุณสมบัติเส้นขอบ มีคอนโทรลสนับสนุน ได้แก่ Text box, Label, Frame, Picture box, Image และ OLE โดยถ้าตั้งค่าเป็น 0-none จะไม่มีเส้นขอบ ถ้าตั้งค่าเป็น 1-Fixed จะมีเส้นขอบ

- คุณสมบัติ ToolTip เป็นหน้าต่างสีเหลืองขนาดเล็กแสดงคำอธิบายสั้นๆ ซึ่งปรากฏขึ้นเมื่อเมาส์เคลื่อนที่ไปอยู่เหนือตัว control หรือไอคอน



รูปที่4 -1 แสดงถึงการกำหนดคุณสมของคอนโทรลที่ใช้ในการสร้างโปรแกรม

4.4 เมธอดร่วม

1. Move ใช้ในการย้ายอ๊อบเจค หรือตัวคอนโทรล พร้อมทั้งปรับขนาดในเวลาเดียวกัน มีไวยากรณ์ คือ

[object.]Move Left [, Top, Width, Height]

2. Refresh ใช้ในการวาดตัวคอนโทรลใหม่ ตามปกติไม่ใช้เนื่องจาก Visual Basic มีการปรับค่าโดยอัตโนมัติ แต่สามารถใช้ได้ในกรณีที่ต้องการให้มีการปรับคุณสมบัติทันที เช่น

```
Dim i As Integer
For i = 1000 to 1 Step - 1
Label1.Caption = (str(i))
Label1.Refresh
Next
```

3. SetFocus ใช้ย้ายโฟกัสไปที่คอนโทรลตัวที่ระบุ เช่น

```
Text1.SetFocus
```

4.5 อีเวนต์ร่วม

1. Click และ DblClick Event โดย Click Event จะเกิดเมื่อผู้ใช้กดปุ่มเมาส์ด้านซ้ายบนตัวคอนโทรล และ DblClick Event เกิดเมื่อเป็นการปุ่มเมาส์ด้านซ้ายแบบดับเบิลคลิก

2. Change Event เป็นอีเวนต์พื้นฐานที่เกิดเมื่อข้อมูลของตัวคอนโทรล มีการเปลี่ยนแปลง

3. GotFocus และ LostFocus Event ซึ่ง GotFocus Event เกิดขึ้นเมื่อตัวคอนโทรล ได้รับการโฟกัส เช่นเมื่อเมาส์เลื่อนมาถึง และ LostFocus Event จะเกิดเมื่อการโฟกัสออกไปจากตัวคอนโทรล

4. KeyPress, KeyDown และ KeyUp Event เป็นอีเวนต์ ที่เกิดเพื่อกดปุ่มแป้นพิมพ์ ขณะที่ตัวคอนโทรลได้รับการโฟกัส โดยมีลำดับดังนี้ KeyDown (เมื่อผู้ใช้กดปุ่มแป้นพิมพ์(KeyPress) Visual Basic แปลปุ่มนั้นเป็นรหัส ANSI และ KeyUp เมื่อปล่อยปุ่มแป้นจากพิมพ์

5. MouseDown, MouseUp และ MouseMove Event เป็นอีเวนต์ ที่เกิดขึ้นเมื่อมีการคลิก, ปล่อย หรือย้าย ของเมาส์บนตัว Control โดยลำดับการเกิด Click Event มีลำดับ คือ MouseDown -> MouseUp -> Click -> MouseMove ลำดับการเกิด DblClick Event มีลำดับ คือ MouseDown -> MouseUp -> Click -> MouseMove -> DblClick -> MouseUp -> MouseUp

4.6 อีเวนต์ของฟอร์ม

อีเวนต์ของฟอร์มเป็นอ๊อบเจคพื้นที่ใช้ในการติดต่อกับผู้ใช้งาน ดังนี้

1. Load Event เป็นอีเวนต์ที่เกิดขึ้นต่อจาก Initialize Event ใช้ในการกำหนดค่าเริ่มต้นให้กับฟอร์มและอ๊อปเจคต์

2. Resize Event เกิดขึ้นก่อนที่จะเห็นฟอร์ม โดยอีเวนต์สามารถใช้ในการปรับตัว คอนโทรลให้พอดีกับฟอร์ม นอกจาก Resize event เกิดขึ้นผู้ใช้ปรับขนาดฟอร์มเอง

3. Unload Event เป็นช่วงสุดท้ายก่อนที่โปรแกรมจะถูกปิด อีเวนต์นี้เกิดเมื่อผู้ใช้ปิดโปรแกรม ก่อนที่โปรแกรมจะถูกปิด เราสามารถแจ้งข่าวสารให้ผู้ใช้ได้ เช่น ถามถึงการบันทึกข้อมูลก่อนออกจากโปรแกรม ถ้าไม่มีการยกเลิกการ Unload ขั้นต่อไป วิชวลเบสิกจะทำการยกเลิกตัว คอนโทรล ปิดฟอร์ม และยกเลิกทรัพยากรต่าง ๆ ของวินโดว์

4.7 โพรซีเจอร์และฟังก์ชั่น

โพรซีเจอร์หรือฟังก์ชั่นเป็นส่วนที่ใช้เขียนโปรแกรมลงไป ซึ่งสามารถช่วยลดความซ้ำซ้อนของโปรแกรมแต่ละส่วน สำหรับโพรซีเจอร์สามารถแบ่งได้เป็น 2 ประเภท คือ

1. โพรซีเจอร์เหตุการณ์ จะทำงานเมื่อเกิดเหตุการณ์ต่างๆขึ้นกับอ็อปเจ็ค เช่น เมื่อผู้ใช้คลิกปุ่ม "Start" ถ้าหากภายในปุ่มมีโพรซีเจอร์อยู่ ก็จะเกิดการทำงานตามคำสั่งในโพรซีเจอร์นั้นทันที

2. โพรซีเจอร์ที่กำหนดเอง เป็นโพรซีเจอร์แบบทั่วไป หลังจากสร้างขึ้นแล้วเราสามารถเรียกใช้ได้ทันที เพียงแค่อ้างชื่อของโพรซีเจอร์นั้นเท่านั้น

4.8 ฟังก์ชั่น

ลักษณะของฟังก์ชั่นจะคล้ายกับโพรซีเจอร์ เพียงแต่ฟังก์ชั่นจะมีการส่งค่ากลับไปยังโปรแกรมที่เรียกฟังก์ชั่นนั้นมาใช้

4.9 การประกาศตัวแปร

การประกาศตัวแปรมีคำสั่งแตกต่างกันไปตามชนิดของการใช้งานดังนี้

Static : ตัวแปรนี้จะใช้ได้เฉพาะภายในโพรซีเจอร์เท่านั้นและค่าในตัวแปรจะถูกเก็บไว้ตลอดเวลา แม้ว่าจะออกโพรซีเจอร์ไปแล้วก็ตาม

Dim : มีลักษณะเป็น Static แต่สามารถประกาศตัวแปรได้ 2ที่ คือ ในโพรซีเจอร์ และ General Declarations (ส่วนแรกของโปรแกรม ใช้ในการประกาศตัวแปร(หากประกาศในโพรซีเจอร์แล้ว เมื่อออกจากโพรซีเจอร์แล้ว ค่าในตัวแปรจะถูกรีเซ็ทใหม่อัตโนมัติ ดังนั้นในแต่ละโพรซีเจอร์อาจมีตัวแปรชื่อซ้ำกันได้ โดยไม่มีผลต่อกัน แต่การประกาศตัวแปรบน General Declarations จะทำให้โพรซีเจอร์ทุกตัวบนฟอร์มนั้นสามารถเรียกใช้ได้ และหากมีการเปลี่ยนแปลงค่าในโพรซีเจอร์ใดๆแล้ว ก็จะมีผลไปกับโพรซีเจอร์อื่นๆด้วย

Private : มีรูปแบบเช่นเดียวกันกับ Dim แต่ประกาศได้ที่ General Declarations เท่านั้น และจะใช้ได้เฉพาะบนฟอร์มนั้นเท่านั้น

Public : มีรูปแบบเช่นเดียวกันกับ Private แต่สามารถใช้ได้ในทุกๆฟอร์มบนโปรแกรมนั้น

ประเภทข้อมูล	ประเภท	ขนาด	การเก็บข้อมูล หรือ ช่วงข้อมูล
Integer	จำนวนเต็ม	2 ไบต์	–32,768 ถึง 32,767
Long	จำนวนเต็ม	4 ไบต์	–2,147,483,648 ถึง 2,147,483,647
Boolean	จำนวนเต็ม	2 ไบต์	เก็บค่า 0 และ –1 ซึ่งแทน False หรือ True
Byte	จำนวนเต็ม	1 ไบต์	เก็บค่าในช่วง 0 ถึง 255
Single	จำนวนทศนิยม	4 ไบต์	ค่าลบ –3.402823E38 ถึง –1.401298E-45 ค่าบวก 1.401298E-45 ถึง 3.402823E38
Double	จำนวนทศนิยม	8 ไบต์	ค่าลบ –1.79769313486232E308 ถึง -4.94065645841247E-324 ค่าบวก 4.94065645841247E-324 ถึง 1.79769313486232E308
Currency	จำนวนทศนิยม (4 ตำแหน่ง)	8 ไบต์	–922,337,203,477.5808 ถึง –922,337,203,477.5807
Decimal	จำนวนทศนิยม	8 ไบต์	ค่าที่ไม่มีทศนิยม +/-79,228,162,514,264,337,593,543,950,335 ค่าที่มีทศนิยม +/- 7.92281625142643 37593543950335 และมีทศนิยม 28 ตำแหน่ง
String	ข้อความ	-	-
Date	วันที่/เวลา	8 ไบต์	เก็บค่าระหว่าง 1 มกราคม ค.ศ. 100 ถึง 31 ธันวาคม ค.ศ. 9999 และเวลาใดๆ

ตารางที่ 4-2 แสดงถึงประเภทของข้อมูลพื้นฐานที่ใช้ในการสร้างตัวแปร

4.10 คำสั่งพื้นฐาน

คำสั่งพื้นฐานที่ใช้ในการสร้างโปรแกรม ได้แก่

คำสั่งเงื่อนไข If… Then… Else… End If มีรูปแบบดังนี้

IF

< เงื่อนไขที่ทำการเปรียบเทียบ >

Then

< ถ้าเงื่อนไขเป็นจริง เขียนโค้ดที่นี่ >

Else

< ถ้าเงื่อนไขไม่เป็นจริง เขียนโค้ดที่นี่ >

End If

คำสั่งเงื่อนไข Select Case มีรูปแบบดังนี้

Select Case

< ชื่อตัวแปรที่ต้องหารนำไปเปรียบเทียบ >

Case

< ค่าที่กำหนดสำหรับเปรียบเทียบ >

< กรณีค่าตัวแปรเปรียบเทียบตรงกับค่าที่กำหนดไว้ จะทำคำสั่งในส่วนนี้ >

Case

< ค่าอื่นที่ต้องการเปรียบเทียบ >

< กรณีค่าตัวแปรเปรียบเทียบตรงกับค่าที่กำหนดไว้ จะทำคำสั่งในส่วนนี้ >

Case Else

< กรณีนอกเหนือจากข้างต้น จะทำคำสั่งในส่วนนี้ >

End Select

คำสั่งวนรอบแบบ For...Next มีรูปแบบดังนี้ คือ

For

< ชื่อตัวแปรประเภทตัวเลข > = < ค่าเริ่มต้น > To < ค่าสิ้นสุด >

< โปรแกรมที่ต้องการให้ทำงานวนรอบ >

Next

< ชื่อตัวแปร >

คำสั่งวนรอบแบบ Do Until หรือ While...Loop มีรูปแบบดังนี้ คือ

Do While/Until

< เงื่อนไขการเปรียบเทียบ >

< คำสั่งที่ต้องการให้ทำงานวนรอบ >

Loop

คำสั่งวนรอบแบบ Do...Until หรือ While...Loop มีรูปแบบดังนี้

Do

< คำสั่งที่ต้องการให้ทำงานวนรอบ >

Loop While/Until

< เงื่อนไขการเปรียบเทียบ >

4.11 คำสั่งเกี่ยวกับการจัดการรูปภาพ

คำสั่งเกี่ยวกับการจัดการรูปภาพ ได้แก่

LoadPicture : เพื่อทำการโหลดรูปภาพเข้ามาใส่ใน Picture Box หรือ Image มีรูปแบบ คือ

[ชื่อ Picture Box].Picture = LoadPicture(ที่อยู่ของรูปภาพ)

Point : ใช้อ่านค่าสีจากจุดที่กำหนดบนรูปภาพ ซึ่งจะให้ค่าสีในระบบ Long Color มีรูปแบบ คือ
[ชื่อ Picture Box].Point (พิกัด X, พิกัด Y)

PSet : ใช้เพื่อวาดจุดลงไปบนรูปภาพตามจุดที่กำหนด รูปแบบ คือ
[ชื่อ Picture Box Name].PSet (พิกัด X, พิกัด Y), รหัสสีที่ต้องการ

PaintPicture : ใช้เพื่อคัดลอกรูปภาพจากที่หนึ่งไปอีกที่ โดยสามารถปรับขนาดได้
ชื่อกล่องรูปภาพปลายทาง.PaintPicture(ชื่อกล่องรูปภาพต้นทาง, [ตำแหน่งเริ่มต้นของรูปต้นทางแกนX], [แกนY], [ขนาดความกว้างรูป], [ความสูงรูป], [ตำแหน่งเริ่มต้นของรูปต้นทางแกนX], [แกนY], [ขนาดความกว้างรูป], [ความสูงรูป], [รูปแบบการคัดลอกภาพ])

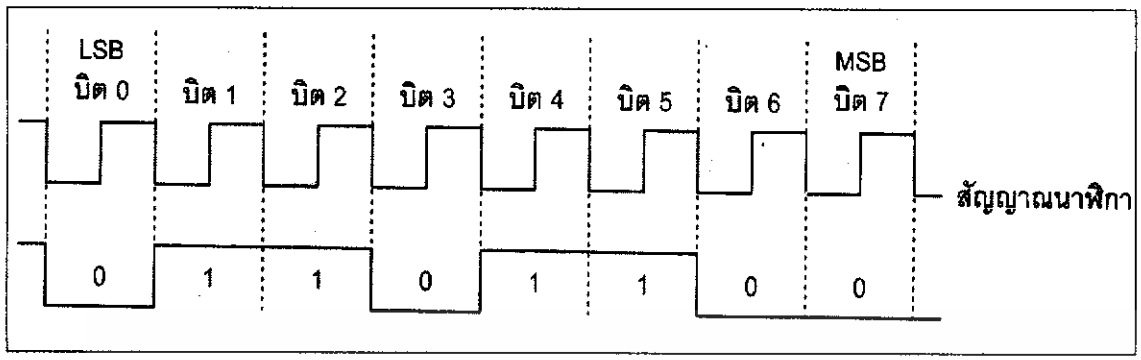
4.12 คำสั่งเกี่ยวกับการติดต่อพอร์ตอนุกรม

4.12.1 ทำความรู้จักกับพอร์ตอนุกรม

การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพวงอื่น ๆ หรือคอมพิวเตอร์ด้วยกันแบบอนุกรมเป็นการรับข้อมูลครั้งละ 1 บิต แต่ก็สามารถรับส่งข้อมูลได้คราวละหลาย ๆ บิตได้ หากแต่จะต้องมีการตกลงกันระหว่างตัวส่งและตัวรับว่า จะรับส่งข้อมูลคราวละกี่บิต ตัวรับจะต้องรอข้อมูลมาให้ครบทุกบิตเสียก่อนจึงทำการประมวลผล ส่งผลให้การสื่อสารข้อมูลอนุกรมอาจมีความเร็วต่ำกว่าแบบขนาน ในด้านจำสวนสายสัญญาณการรับส่งข้อมูลแบบอนุกรมจะใช้จำนวนสายที่น้อยกว่ามาก อย่างน้อยที่สุดใช้เพียง 2 – 3 เส้นเท่านั้น แต่อัตราเร็วในการรับส่งข้อมูลอาจต่ำกว่าแบบขนาน อย่างไรก็ตามการรับส่งข้อมูลแบบอนุกรมสามารถใช้สายสัญญาณที่มีความยาวมากกว่าแบบขนาน ทำให้ระยะทางในการสื่อสารข้อมูลแบบอนุกรมสามารถทำได้มากกว่า

4.12.2 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมนั้นแบ่งออกได้เป็น 2 แบบคือการสื่อสารอนุกรมแบบซิงโครนัสและกาสสื่อสารอนุกรมแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสคือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้นคือ สัญญาณนาฬิกา, ข้อมูลและกราวด์ รูปที่ 4-2 แสดงให้เห็นถึงไทมิ่งไดอะแกรมของการส่งข้อมูลแบบซิงโครนัส



รูปที่ 4-2 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรม

4.12.3 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยเหมือนกับการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูล หรือ บอดเรต (baudrate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

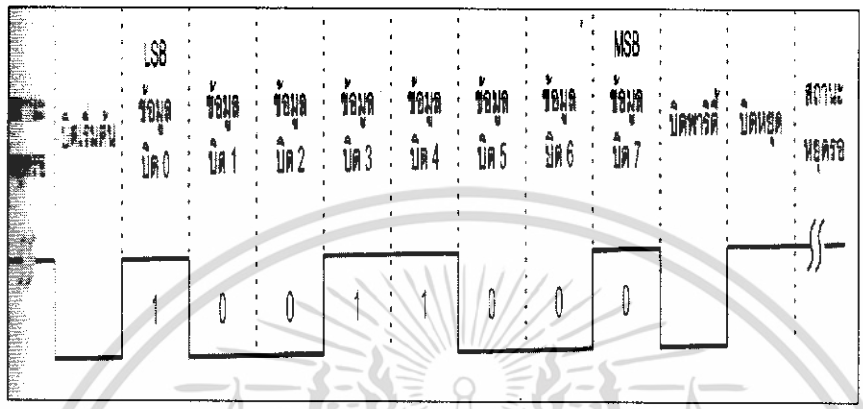
รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมจะมีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) จะมีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้าย (Stop Bit) จะมีขนาด 1,1.5 หรือ 2 บิต

รูปที่ 4- 3 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีข้อมูลที่จะส่ง ขา DATA จะมีสถานะลอจิก "1" ซึ่งจะเรียกสถานะนี้ว่าสถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด (LSB) ก่อน ซึ่งข้อมูลในไบต์ที่จะส่งอาจจะมีจำนวนบิต 5,6,7 หรือ 8 บิตก็ได้ จากนั้นตามด้วยบิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่ส่งคือ บิตปิดท้าย ซึ่งจะให้ขาดาต้ามีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต, 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและส่งข้อมูลแบบอะซิงโครนัสเรียกว่า Universal Asynchronous Receiver/Transmitter หรือ UART อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสคือ ค่าบอดเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับและส่งข้อมูล บอดเรตมาตรฐานที่ใช้สำหรับพอร์ตอนุกรม RS-232 ได้แก่ 110, 150, 300, 600, 1200, 2400, 4800, 9600, และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มอาจจะสามารถกำหนดค่าบอดเรตได้สูงถึง 11520 บิตต่อวินาที เนื่องจากบอดเรตคือจำนวนบิตของข้อมูลที่สามารถถ่ายทอดได้ภายใน 1 วินาที ยกตัวอย่าง

ข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาริตี้ความเร็วในการรับส่งข้อมูลจะเหลือเป็น 872 ไบต์ต่อวินาที



รูปที่ 4-3 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรมแบบอะซิงโครนัส

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd), แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบพาริตี้เป็นการตรวจสอบจำนวนรวมของบิตที่เป็นลอจิก "1" ภายในข้อมูลที่ส่งไป 1 ไบต์ว่ามีจำนวนรวมเป็นเลขคู่หรือเลขคี่โดยต้องรวมบิตพาริตี้เข้าไปด้วย ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิตและมีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสองจะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ค่าในบิตพาริตี้ จะต้องมีลอจิกเป็น "0" แต่ถ้าพาริตี้เป็นคี่ ค่าที่บิตพาริรี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้มีจำนวนบิตที่เป็นลอจิก "1" รวมกันเป็นเลขคี่ ในตารางที่ 1-1 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART โดยภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ให้ตรงกันว่า จะตรวจสอบพาริตี้คี่หรือคู่ จากนั้นภาครับของ UART จะตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วยถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแจ้งข้อผิดพลาดให้ผู้ใช้ทราบ นับเป็นการตรวจสอบความผิดพลาดของการถ่ายทอดข้อมูลที่ง่ายที่สุด แต่จะเชื่อถือได้เมื่อมีบิตข้อมูลที่ทำการส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	1	0
11111111	0	1

ตารางที่ 4-3 แสดงลักษณะการทำงานของบิตพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ UART เบอร์ 8250 UART ชิปเหล่านี้มีระดับแรงดันเป็นแบบทีทีแอล (0 และ +5V) แต่เพื่อให้มีแรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้นโดยลอจิก "0" มีระดับแรงดัน +3V ถึง +12V ในขณะที่ลอจิก "1" มีระดับแรงดัน -3V ถึง -12V

4.12.4 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดอยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 ถึง -12V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12V แสดงว่าเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) ไว้ว่าอุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์ หรือ ไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ในโมเด็มจะเป็นแบบ DCE

สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

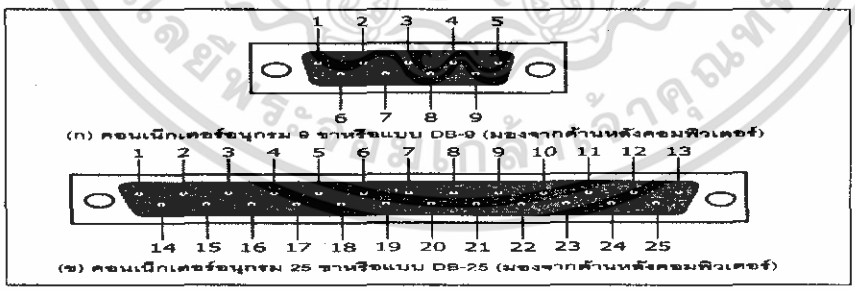
4.12.5 คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่น ๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 1-3

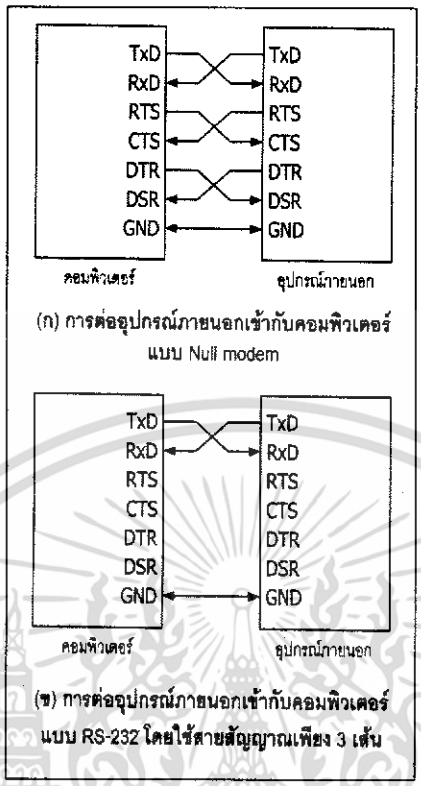
สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกดังในรูปที่ 1-4 ลูกศรในรูปแสดงถึงทิศทางของข้อมูล ในรูปที่ 1-4 (ก) เป็นการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยตรงโดยไม่ต่องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็กเต็มรูปแบบ ส่วนในรูปที่ 1-4(ข) เป็นการเชื่อมต่อแบบ Null modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้

- Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขา นี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูลเช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก

- Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์ บัฟเฟอร์



รูปที่ 4-4 การจัดขาของคอนเน็กเตอร์อนุกรมตามมาตรฐาน RS-232 ทั้งแบบ DB-9 และ DB-25



รูปที่ 4-5 การต่ออุปกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่าง ๆ

-Transmitted Data : TD หรือ TxD ใช้ส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

-Data Terminal Ready : DTR เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วยโดยขา DTR นี้ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และ ขา DTR ของอุปกรณ์ปลายทางต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบ Null Modem ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

-Signal Ground : GND กราวด์ระบบ

-Data Set Ready : DSR ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

-Request To Send : RTS เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null Modem 3 จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

-Clear To Send : CTS ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่

-Ring Indicator : RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

4.12.6 UART

UART มาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

หน้าที่หลักของ UART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัส แล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังแจ้งข้อมูลอื่น ๆ ให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล (บอดเรต), รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น

ภายใน UART จะมีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ (programmable buadrate generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้มีขนาด 16 บิต ดังนั้นจึงสามารถกำหนดตัวหารอยู่ในช่วง 1 – 65,535 UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (half duplex) และฟูลดูเพล็กซ์ (full duplex) โดยการส่งแบบฮาล์ฟดูเพล็กซ์เป็นการส่งแบบทิศทางเดียว ส่วนการส่งแบบฟูลดูเพล็กซ์นั้นสามารถรับและส่งข้อมูลได้ในคราวเดียวกัน

4.12.7 วงจรภายในและรีจิสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์โดยทั่วไปสามารถต่อพอร์ตอนุกรมสูงสุดได้ 4 พอร์ต มีชื่อเรียกเป็น com1, com2, com3 และ com4 ซึ่งพอร์ตอนุกรมแต่ละตัวต่างก็ใช้งาน UART ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอกเช่นเดียวกัน ยกตัวอย่าง พอร์ตอนุกรม com1 มีแอดเดรสอยู่ที่ 3F8H ตำแหน่งของรีจีสเตอร์ต่าง ๆ จะเป็นตำแหน่งที่บวกเข้าไปกับค่า 3F8H โดยรีจีสเตอร์ที่ใช้งานกับพอร์ตอนุกรมมีดังนี้

00H เป็นรีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาหรือเตรียมข้อมูลก่อนที่จะส่งออกไป

01H รีจิสเตอร์เอ็นเอเบิลการอินเตอร์รัปต์ ใช้เซตโหมดการอินเตอร์รัปต์ของพอร์ตอนุกรม

02H รีจิสเตอร์แสดงโหมดการอินเตอร์รัปต์ ใช้เพื่อตรวจสอบโหมดของการอินเตอร์รัปต์

03H รีจิสเตอร์กำหนดรูปแบบของข้อมูล

04H รีจิสเตอร์ควบคุมโมเด็ม ใช้ตรวจสอบบิตสำหรับติดต่อกับโมเด็ม เช่น RTS หรือ DTR

05H รีจิสเตอร์แสดงสถานการณ์รับและการส่งข้อมูลแบบอนุกรม

06H รีจิสเตอร์แสดงสถานะของโมเด็ม ซึ่งจะแสดงสถานะของขา DCD, RI, DSR และ CTS

07H รีจิสเตอร์สำหรับการเก็บข้อมูลชั่วคราว

4.12.8 ลักษณะสัญญาณอินพุตและเอาท์พุตของพอร์ต RS-232

สัญญาณเอาต์พุตที่ใช้ควบคุม (RTS และ DTR) และสัญญาณแสดงสถานะอินพุต (CTS, DSR และ DCD) ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและรับจะไม่ถูกกลับสถานะ UART จะให้ระดับสัญญาณเอาต์พุตออกมาเป็นแบบทีทีแอลเท่านั้น ดังนั้นเมื่อสัญญาณถูกส่งออกมาจาก UART จึงต้องส่งเข้าสู่วงจรขับเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรขับที่ใช้ทั้งหมดภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะ

แอดเดรสของพอร์ตอนุกรม

แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแหน่งดังนี้คือ

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H

บิต 3	บิต 2	บิต 1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์ตอนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 2 พอร์ต
0	1	1	มีพอร์ตอนุกรม 3 พอร์ต
1	0	0	มีพอร์ตอนุกรม 4 พอร์ต

ตารางที่ 4-4 แสดงข้อมูลในแอดเดรส 0000:0411H ที่ใช้แจ้งจำนวนพอร์ตอนุกรม

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหมด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรม ไบออสจะนำแอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับพอร์ตอนุกรม COM1 จะเก็บไว้ที่แอดเดรส 0000:0400H และ 0000:0401H ส่วนตำแหน่งอื่น ๆ มีรายละเอียดดังนี้

COM2 = 0000:0402H – 0000:0403H

COM3 = 0000:0404H – 0000:0405H

COM4 = 0000:0406H – 0000:0407H

นอกจากนี้ที่หน่วยความจำแอดเดรส 0000:0411H ยังใช้สำหรับแสดงจำนวนของพอร์ตอนุกรมที่มีอยู่ในคอมพิวเตอร์อีกด้วย โดยมีรายละเอียดดังแสดงในตารางที่ 4-4

4.12.9 การเขียนโปรแกรมเพื่อการใช้งานพอร์ตอนุกรม

ก่อนการใช้งานพอร์ตอนุกรมนั้นจะต้องมีการกำหนดค่าเริ่มต้นให้กับตัวมันก่อน ซึ่งก็คือการกำหนดจำนวนบิตข้อมูลที่ต้องการส่ง,จำนวนบิตปิดท้าย,ชนิดของพาริตี้ที่ใช้ และบอดเรต การกำหนดสามารถทำได้หลายวิธี วิธีแรกเป็นการกำหนดจากดอสพร้อมพ์ โดยใช้คำสั่ง MODE ซึ่งมีวิธีการใช้งานดังนี้

แต่เมื่อใช้คำสั่งนี้ในขณะที่โปรแกรมทำงานผ่านระบบปฏิบัติการวินโดวส์ จะไม่สามารถใช้งานได้เนื่องจากระบบปฏิบัติการวินโดวส์ ได้เข้าผังตัวพอร์ตอนุกรมเข้าเป็นส่วนหนึ่งของระบบปฏิบัติการแล้ว ดังนั้นการเรียกใช้งานจึงจำเป็นต้องเรียกผ่านเครื่องมือที่ติดต่อผ่านระบบปฏิบัติการ เช่นการใช้คอนโทรล MSCOMM32.OCX ของโปรแกรม Visual BASIC

4.12.10 คอนโทรล MSComm

สำหรับการใช้งาน Visual BASIC ตั้งแต่เวอร์ชัน 2 เป็นต้นมา ใน Visual BASIC จะมีคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์มาให้ โดยใน Visual BASIC เวอร์ชัน 2 และเวอร์ชัน 3 จะใช้ชื่อว่า MSCOMM.VBX ส่วนเวอร์ชัน 4 ใช้ชื่อว่า MSCOMM16.OCX สำหรับการทำงานกับระบบปฏิบัติการ 16 บิต และ MSCOMM32.OCX สำหรับการทำงานกับระบบปฏิบัติการ 32 บิต สำหรับใน Visual BASIC เวอร์ชัน 5 จะมีเพียง MSCOMM 32.OCX เท่านั้นเพราะถูกออกแบบมาให้ใช้งานกับระบบปฏิบัติการ 32 บิต

MSComm จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (event-driven communications) เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier Detect (DCD) หรือขา Request To Send (RTS) เหตุการณ์ Oncomm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในฟังก์ชั่นต่าง ๆ ไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSComm 1 ตัวสามารถควบคุมการทำงานพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบคุมพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel

CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่ (COM1, COM2, COM3, COM4)

รูปแบบการใช้งาน

Object.Settings [=value]

ค่า Value มีชนิดข้อมูลเป็นแบบ String มีรูปแบบเป็น "BBBB,P,D,S" โดย BBBB เป็นค่าอัตราบอด, P เป็นค่าพาริตี้, D เป็นจำนวนของบิตข้อมูล และ S เป็นจำนวนของบิตปิดท้าย ปกติแล้วค่านี้ถูกกำหนดไว้เป็น "9600, N, 8, 1"

ตัวอย่างการใช้งานคำสั่ง Settings โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และ บิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

MSComm1.Settings = "9600, N, 8, 1"

PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

Object.PortOpen [= value]

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีน คือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรมและ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ถ้าคุณสมบัติ DTREnable หรือ RTSEnable ถูกกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบัติของ DTREnable หรือ RTSEnable จะถูกเซตเป็น False หลังจากปิดพอร์ต แต่ถ้าเซตเป็น False หลังจากปิดโปรแกรมแล้ว ค่าที่กำหนดไว้จะเป็นค่าเดิม

ตัวอย่างการใช้คำสั่งเปิดพอร์ต เพื่อติดต่อสื่อสารกับพอร์ตอนุกรม COM1 และมีบอดเรต 9600 บิตต่อวินาที ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต มีดังนี้

```
MSComm.Settings = "9600, n, 8, 1"
MSComm.CommPort = 1
MSComm.PortOpen = True
```

Input

อ่านค่าและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

Object.Input

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดให้คุณสมบัติ Input ทำการอ่านค่าข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้าInputMode ถูกกำหนดเป็น comInputModeText คุณสมบัติ Input จะส่งค่าข้อมูลกลับมาในรูปของข้อความชนิดข้อมูลเป็นแบบ Variant ถ้า InputMode กำหนดเป็น comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

ตัวอย่างโปรแกรมแสดงให้เห็นถึงวิธีในการรับข้อมูลจากบัฟเฟอร์รับข้อมูลทั้งหมด

```
Private Sub Commandl_Click ( )
Dim InString as String
    MSComm1.InputLen = 0          ' Retrieve all available data.
    If MSComm1.InBufferCount Then   ' Check for data.
    InString = MSComm.Input          ' Read data.
    End If
End Sub
```

InBufferCount

ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.InBufferCount[= value]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งานคำสั่ง

Object.InBufferSize [= value]

คำสั่ง InBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ ค่าเริ่มต้นกำหนดไว้ที่ 1024 ไบต์

InputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.InputLen [= value]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ "0" การกำหนดค่าเท่ากับ "0" จะทำให้ คำสั่ง Input ของ MSComm อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาครับมากเท่ากับจำนวน InputLen คำสั่ง Input จะส่งค่าว่าง (" ") กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InBufferCount โดยกำหนดให้มีข้อมูลอยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับการอ่านค่าข้อมูลจากเครื่องมือหรือเครื่องจักรที่มีการกำหนดค่าขนาดความยาวของข้อมูลเอาไว้แล้ว

InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งานคำสั่ง

Object.InputMode [= value]

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภทคือ

comInputModeText สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น "0" และค่าเริ่มต้นของการรับค่าข้อมูลก็จะเป็นค่านี้

comInputModeBinary สำหรับข้อมูลอื่น ๆ ซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูลและเก็บข้อมูลไว้ในตัวแปรแบบอาเรย์ ชนิดข้อมูลเป็นแบบไบต์

Output

ใช้ในการส่งขบวนการของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

Object.Output [= value]

ค่า value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดของข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ Byte

OutBufferCount

คืนค่าจำนวนของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่ง และสามารถใช้คำสั่งนี้เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย

รูปแบบการใช้งานคำสั่ง

Object.OutBufferCount [= value]

ผู้ใช้งานสามารถเคลียร์บัฟเฟอร์ภาคส่งได้โดยการกำหนดค่า OutBufferCount เท่ากับ "0"

OutBufferSize

กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดตัวแปรเป็นแบบไบต์

รูปแบบการใช้งานคำสั่ง

Object.OutBufferSize [= object]

คุณสมบัติ OutBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่ง โดยค่าปกติที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

ParityReplace

กำหนดและคืนค่าตัวอักษรที่ไปวางแทนในตำแหน่งที่เกิดข้อผิดพลาดจากพาริตี้

รูปแบบการใช้งานคำสั่ง

Object.ParityReplace [= value]

บิตพาริตี้ เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมกับข้อมูล เพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MSCOM จะทำการบวกบิตทุกบิตที่มีค่าลอจิก "1" ในแต่ละไบต์ และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก "1" เป็นเลขคู่หรือคี่ และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าค่าที่นำมาบวกแล้วมีพาริตี้ไม่ตรงแสดงว่าการรับส่งข้อมูลผิดพลาด

การกำหนดค่า เริ่มต้นให้กับ ParityReplace นั้นกำหนดให้ใช้เครื่องหมาย (?) ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ผิดพลาด ถ้ากำหนดค่า ParityReplace ให้เป็นค่าว่าง ("") จะเป็นการยกเลิกการใช้งาน ParityReplace ใช้ชนิดข้อมูลเป็นแบบสตริง แต่จะทำการกำหนด จะกำหนดได้เพียงไบต์เดียวเท่านั้น ซึ่งจะสามารถใช้ค่าใด ๆ ก็ได้ที่เป็นโค้ด ANSI มีค่าอยู่ระหว่าง 0-255

DTREnable

ใช้ในการกำหนดสถานะลอจิกของขา Data Terminal Ready (DTR) โดยสัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบบูลีน

รูปแบบการใช้งาน

Object.DTREnable [= value]

ค่า Value เป็นสถานะ True หรือ False เพื่อกำหนดลอจิกของขา DTR ให้เป็น “0” หรือ “1” โดย

True หมายถึง ให้ขา DTR มีลอจิก “1”

False หมายถึง ให้ขา DTR มีลอจิก “1”

สำหรับการใช้งานกับโมเด็ม การทำให้ขา DTR เป็นลอจิก “0” จะเป็นการวางหูโทรศัพท์หรือยกเลิกการติดต่อ

RTSEnable

ใช้เพื่อกำหนดสถานะลอจิกให้ขา Request To Send (RTS) โดยขา RTS จะเป็นสัญญาณที่ส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอส่งข้อมูล ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

Object.RTSEnable [= value]

ค่า Value เป็นสถานะ True หรือ False เพื่อกำหนดลอจิก “0” หรือ “1” ให้ขา RTS โดย

True หมายถึง ให้ขา RTS มีลอจิก “1”

False หมายถึง ให้ขา RTS มีลอจิก “0” (เป็นค่าปกติ)

EOFEnable

เป็นการกำหนดให้ MSComm รอสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์ (End of file : EOF) ระหว่างการรับอินพุตเข้ามา ถ้าพบสัญลักษณ์ EOF ภาคอินพุตจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ ComEvEOF

รูปแบบการใช้งาน

Object.EOFEnable [= value]

โดย value เป็นค่าสถานะ True หรือ False เพื่อเอ็นเอเบิลหรือดิสเอเบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF (เป็นค่าปกติ)เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

CTSHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Clear To Send (CTS) ได้ว่ามีสถานะลอจิก "0" หรือ "1" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CTSHolding เป็น True ขา CTS จะมีสถานะลอจิกเป็น "1" ถ้าค่า CTSHolding เป็น False ขา CTS จะมีสถานะลอจิกเป็น "0"

รูปแบบการใช้งาน

Object.CTSHolding

เมื่อขา CTS เป็นลอจิก "0" (CTSHolding = False) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCTSTO (Clear To Send Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

CDHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Data Carrier Detect (DCD) ได้ว่ามีสถานะลอจิกเป็น "1" หรือ "0" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CDHolding เป็น True ขา DCD จะมีสถานะลอจิก "1" หรือ "0" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า DSRHolding เป็น True ขา DSR จะมีสถานะลอจิก "1" ถ้าค่า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิก "0"

รูปแบบการใช้งาน

Object.DSRHolding

เมื่อขา DSR เป็นลอจิก "1" (DSRHolding = True) และเกิดไทม์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventDSRTO (Data Set Ready Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

รูปแบบการใช้งานคำสั่ง

Object.Handshaking [= value]

ค่าตัวแปร Value ที่ใช้กำหนดค่ากำหนดได้ 4 รูปแบบด้วยกันคือ

1. comNone ค่าที่กำหนดคือ 0 เป็นการกำหนดให้ไม่มีการแฮนด์เช็ก (เป็นค่าเริ่มต้น)
2. comXOnXOff ค่าที่กำหนดคือ 1 เป็นการกำหนดให้ใช้แฮนด์เช็กแบบ XON/XOFF
3. commRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS (Request To Send/Clear To Send)
4. comRTSXOnXOff ค่าที่กำหนดคือ 3 เป็นการกำหนดให้ใช้ทั้งแบบ Request To Send และ XON/XOFF

คุณสมบัติ Handshaking ใช้เพื่อกำหนดรูปแบบการสื่อสารภายใน ระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลตัวอักษรถูกส่งมาถึงพอร์ตอนุกรม อุปการณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลไปยังบัฟเฟอร์ภาครับ เพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ ถ้าไม่มีบัฟเฟอร์ภาครับ โปรแกรมที่ใช้งานจะต้องทำการอ่านค่าข้อมูลโดยตรงจากฮาร์ดแวร์ของพอร์ตอนุกรม ซึ่งผู้ใช้งานจะเกิดปัญหาข้อมูลสูญหายได้ เนื่องจากว่าการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติ handshaking ช่วยให้ผู้ใช้งานแน่ใจได้ว่าข้อมูลที่ได้รับมานั้นไม่มีการสูญหายเมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือโอเวอร์โฟลว (overflow) โดยใช้วิธีการตรวจสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งข้อมูลมาให้

รูปแบบการใช้งาน

Object.Break [= value]

เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าของคุณสมบัติ CommEvent มีการเปลี่ยนแปลงเพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใดหรือ แสดงข้อผิดพลาดที่เกิดขึ้น ตัวอย่างโปรแกรมย่อย OnComm เพื่อนำเหตุการณ์ CommEvent มาแสดงมีดังนี้

การใช้ MSComm เพื่อการติดต่อฮาร์ดแวร์

จากรายละเอียดของ MSComm ที่กล่าวไปในตอนต้นนั้น จะเห็นได้ว่าวิธีการที่จะอ่านค่าหรือเขียนค่าไปยังขาสถานะและขาควบคุมของพอร์ตอนุกรมสามารถทำได้ง่ายดายมาก โดยใช้คำสั่งเหล่านี้

DTREnable สำหรับสั่งให้ขา DTR มีลอจิก “0” หรือ “1”

RTSEnble สำหรับสั่งให้ขา RTS มีลอจิก “0” หรือ “1”

CTSHolding สำหรับอ่านค่าสถานะจากขา CTS ว่ามีลอจิก “0” หรือ “1”

CDHolding สำหรับอ่านค่าสถานะจากขา DCD ว่ามีลอจิก “0” หรือ “1”

DSRHolding สำหรับอ่านค่าสถานะจากขา DSR ว่ามีลอจิก “0” หรือ “1”

Break สำหรับการสั่งให้ขา TxD มีลอจิก “0” หรือ “1”

4.13 พอร์ต USB กับวิชวลเบสิก

USB เป็นการส่งข้อมูลที่มีรูปแบบการเชื่อมต่อในระบบบัสคือ อุปกรณ์ทุกๆ ตัวจะต้องส่งสัญญาณรวมกันไปในสายส่งสัญญาณเพียงคู่เดียว ดังนั้นดอุปกรณ์ทุกๆ ตัวที่เชื่อมต่อกับบัสจะต้องส่งข้อมูลเรียงลำดับกันไปเพื่อไม่ให้เกิดการชนกันของข้อมูล และเนื่องจาก USB เป็นระบบบัสที่ใสช้สายส่งสัญญาณเพียงคู่เดียว (2 เส้น) ทำให้ในช่วงเวลาหนึ่งๆ จะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียวเท่านั้น ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันได้หรือที่เรียกกันว่า การส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ (half duplex)

เนื่องจากการถ่ายอดข้อมูลในบัส USB เกิดขึ้นด้วยความเร็วสูง ทำให้ต้องนำเทคนิคการถ่ายทอดสัญญาณรูปแบบต่างๆ มาใช้งานเพื่อให้ข้อมูลส่งไปยังปลายทางได้ถูกต้อง การถ่ายทอดสัญญาณแบบผลต่างเป็นเทคนิคที่นำมาใช้งานเพื่อให้ข้อมูลส่งไปยังปลายทางได้ถูกต้อง การถ่ายทอดสัญญาณแบบผลต่างเป็นเทคนิคที่นำมาใช้เพื่อลดคลื่นรบกวนที่จะแพร่ออกมาจากสายดังที่กล่าวไปแล้ว นอกจากนั้นยังต้องอาศัยการเข้ารหัสแบบ NRZI เพื่อให้ด้าน

รับทราบถึงจังหวะข้อมูลในแต่ละบิตเพื่อไม่ให้เกิดความผิดพลาด และสามรถตรวจสอบการเชื่อมต่อหรือปลดออกของอุปกรณ์ในบัส จึงต้องมีการกำหนดรูปแบบของสัญญาณต่างๆ เพื่อใช้งานขึ้น

4.13.1 การส่งข้อมูลในสายสัญญาณ

หลังจากข้อมูลถูกเข้ารหัสเรียบร้อยแล้ว ข้อมูลเหล่านั้นก็จะถูกส่งออกไปยังอุปกรณ์ต่างๆที่ต่ออยู่กับฮับ แต่เนื่องจาก USB ส่งข้อมูลด้วยความเร็วที่สูงมาก โดยใช้สัญญาณที่ยาว (เมื่อเทียบกับสายสัญญาณข้อมูลภายในคอมพิวเตอร์) ทำให้ไม่สามารถใช้การส่งสัญญาณแบบขั้วเดียว (single end) ได้เพราะจะทำให้เกิดการแพร่กระจายของคลื่นแม่เหล็กไฟฟ้าไปกวนอุปกรณ์อื่นๆ และยังทำให้สัญญาณรบกวนจากภายนอกเข้าไปกวนข้อมูลที่ต้องการส่งในสายสัญญาณได้ง่าย

การส่งสัญญาณที่แก้ไขปัญหา 2 ข้อที่กล่าวมาก็คือการส่งสัญญาณผลต่าง (differential pair signaling) เนื่องจากการส่งสัญญาณด้วยวิธีนี้จะทำให้สนามแม่เหล็กไฟฟ้าที่เกิดขึ้นจากสายสัญญาณทั้งสองเส้นหักล้างกัน เนื่องจากมีศักย์ตรงข้ามกัน จึงไม่เกิดการแพร่กระจายไปรบกวนอุปกรณ์อื่นๆ สัญญาณ 2 เส้นอยู่คู่กัน เมื่อสัญญาณรบกวนเหนี่ยวนำเข้าสู่สายเส้นหนึ่งก็จะเหนี่ยวนำเข้าสู่สายอีกเส้นในขนาดที่เท่ากัน เมื่อถึงปลายทางในวงจรภาครับจะใช้วงจรขยายผลต่าง (differential amplifier) เป็นตัวรับสัญญาณ ซึ่งวงจรนี้จะมีคุณสมบัติขยายสัญญาณที่มีขนาดต่างกันและลดทอนสัญญาณที่มีขนาดเท่ากัน ดังนั้นสัญญาณข้อมูลซึ่งมีศักย์ตรงข้ามกันจะถูกขยายเพื่อส่งต่อไปยังส่วนถัดไป แต่สัญญาณรบกวนซึ่งมีขนนาดเท่ากันในสายทั้ง 2 เส้นจะถูกลดทอนหรือตัดทิ้ง

จะเห็นได้ว่าทั้งสองด้านมีวงจรที่เหมือนกันเนื่องจาก USB เป็นบัสที่รับส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ นั่นคือทั้งสองด้านสามารถรับส่งข้อมูลได้ทั้งคู่ แต่คนละช่วงเวลากัน ทำให้ทั้งสองด้านมีชุดวงจรที่เหมือนกัน และเนื่องจากการรับส่งข้อมูลแบบนี้จะมีอุปกรณ์เพียงตัวเดียวที่สามารถรับหรือส่งข้อมูลได้ในช่วงเวลาหนึ่งๆ อุปกรณ์ที่เหลือที่ไม่ได้ใช้งาน บัสจำเป็นจะต้องปล่อยบัสสัญญาณให้มีสภาพอิมพีแดนซ์สูง (high impedance) ดังนั้นวงจรภาครับส่งของอุปกรณ์แต่ละตัวจะต้องสามารถกำหนดสถานะของเอาต์พุตให้เป็นสภาพอิมพีแดนซ์สูงได้

4.13.2 HID มาตรฐานของอุปกรณ์ USB ในระดับเชื่อมต่อกันผู้ใช้งาน

HID Class (Human Interface Device Class) เป็นลักษณะการเชื่อมต่อของอุปกรณ์ USB ที่เน้นไปที่การติดต่อกับมนุษย์ (human) หรือผู้ใช้งานนั่นเอง ซึ่งอุปกรณ์ USB ที่ใช้ระดับการเชื่อมต่อแบบนี้ซึ่งรู้จักกันดีคือ คีย์บอด และเมาส์ ซึ่งจะอธิบายถึงรูปแบบ มาตรฐาน ลักษณะการเชื่อมต่อและถ่ายทอดข้อมูล เนื่องจากในส่วนของการทดลองจะใช้วงจรเชื่อมต่อพอล์ต USB ที่ใช้มาตรฐานการเชื่อมต่อในระดับ HID นี้ ดังนั้นการทำความรู้จักกับ HID จึงเป็นสิ่งที่ผู้สนใจเชนื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ต USB ควรทราบ กอปรกับระดับการเชื่อมต่อแบบนี้เป็นระดับการเชื่อมต่อที่สามรถทำความเข้าใจได้ง่ายและเป็นที่นิยมนำมาใช้สร้างอุปกรณ์ USB มากที่สุด

1) สิ่งที่ควรรู้เกี่ยวกับ HID คลาส

1.1) การเปลี่ยนแปลงของข้อมูลที่เกิดขึ้นในโครงสร้างการทำงานจะเรียกว่า รายงานหรือรีพอร์ตตามข้อตกลงกำหนดของ HID รีพอร์ต โดยโฮสต์จะทำการรับส่งข้อมูลด้วยการส่งและร้องขอรีพอร์ตในการควบคุมหรือมีลักษณะเป็นการถ่ายทอดสัญญาณแบบอินเตอร์รัปต์ อย่างไรก็ตามรูปแบบของรีพอร์ตนี้ยังไม่มีข้อมูลสรุปชัดเจน แต่ก็มีความอ่อนตัวสูง ทำให้สามารถรองรับกับข้อมูลได้ทุกรูปแบบ

1.2) อุปกรณ์ USB สามารถส่งข้อมูลกลับไปยังโฮสต์(ซึ่งส่วนใหญ่คือ คอมพิวเตอร์) ได้ตลอดเวลา โดยไม่สามารถคาดหมายล่วงหน้า ดังนั้นจึงเป็นหน้าที่ของโฮสต์เองที่ต้องคอยวนตรวจสอบอยู่ตลอดเวลา เพื่อให้สามารถรองรับกับการทำงานของอุปกรณ์ได้ทันท่วงที

1.3) ในการถ่ายทอดสัญญาณหรือข้อมูลบนบัสของ USB ไม่สามารถทราบถึงอัตราการส่งผ่านข้อมูลได้อย่างแน่นอน สิ่งที่กำหนดได้มีเพียงคาบเวลาในแต่ละทรานแซกชั่น และค่าเวลาระหว่างทราบแซกชั่นนั้นอาจน้อยกว่าก็ได้ อย่างไรก็ตาม ระบบจะพยายามทำให้เกิดอัตราการส่งผ่านข้อมูลเร็วที่สุดเท่าที่จะเป็นไปได้

1.4) อัตราการถ่ายทอดสัญญาณหรือข้อมูลสูงสุดจะถูกจำกัดไว้ตามประเภทของอุปกรณ์ ซึ่งมีด้วยกัน 3 แบบคือ ความเร็วต่ำ, ความเร็วเต็มที่ และความเร็วสูง โดยโฮสต์สามารถกำหนดคาบเวลาในการทำงานต่อทรานแซคชั่นสูงสุดแยกกันไปตามประเภทของอุปกรณ์ โดย

1.4.1 สำหรับอุปกรณ์ความเร็วต่ำ โฮสต์กำหนดคาบเวลาในการทำงาน 1 ทรานแซคชั่นสูงสุดไม่เกิน 10 มิลลิวินาที ทำให้อัตราการถ่ายทอดสัญญาณเกิดขึ้นได้สูงสุดไม่เกิน 800 ไบต์ต่อวินาที

1.4.2 สำหรับอุปกรณ์ความเร็วต่ำ โฮสต์กำหนดคาบเวลาในการทำงาน 1 ทรานแซคชั่นสูงสุดไม่เกิน 1 มิลลิวินาที ทำให้อัตราการถ่ายทอดสัญญาณเกิดขึ้นได้สูงสุดไม่เกิน 8,000 ไบต์ต่อวินาที

1.4.3 สำหรับอุปกรณ์ความเร็วต่ำ โฮสต์กำหนดคาบเวลาในการทำงาน 3 ทรานแซคชั่นสูงสุดไม่เกิน 125 ไมโครวินาที ทำให้อัตราการถ่ายทอดสัญญาณเกิดขึ้นได้สูงสุดไม่เกิน 24.56 เมกะไบต์ต่อวินาที

5.)ในระบบปฏิบัติการวินโดวส์ที่ต่ำกว่า 98SE จะไม่รองรับการถ่ายทอดสัญญาณแบบอินเตอร์รัปต์ ดังนั้นการติดต่อจากโฮสต์ไปยังอุปกรณ์จะต้องใช้การถ่ายทอดสัญญาณควบคุมเข้ามาช่วยแทน

2) โครงสร้างดิสคริปเตอร์

ไม่ว่าจะเป็นอุปกรณ์ USB ในคลาสหรือระดับใด ล้วนแล้วแต่ต้องมีดิสคริปเตอร์ในการทำงานทั้งสิ้น ใน HID คลาสเองก็เช่นกัน มีดิสคลิปเตอร์เป็นของตัวเองชื่อ HID ดิสคริปเตอร์ ซึ่งภายในดิสคริปเตอร์นี้จะแบ่งออกเป็น 2 ส่วน คือ ดิสคริปเตอร์รายงานผลการทำงาน หรือ รีพอร์ตดิสคริปเตอร์ (report descriptor) และกลุ่มดิสคลิปเตอร์ทางกายภาพ หรือ ฟิสิกัลดิสคริปเตอร์ (physical descriptor) รูปด้านล่างแสดงโครงสร้างดิสคริปเตอร์ของอุปกรณ์ USB ในระดับ HID จะเห็นได้ว่า HID ดิสคริปเตอร์จะบรรจุอยู่ในอินเตอร์เฟสดิสคริปเตอร์ร่วมกันเอ็นด์พอยต์ดิสคริปเตอร์

HID ดิสคริปเตอร์

Length มีขนาด 1 ไบต์ ใช้ระบุขนาดรวมของดิสคริปเตอร์ในหน่วยไบต์ ใน HID คลาสจะกำหนดให้มีค่าเท่ากับ 09H

Descriptor type มีขนาด 1 ไบต์ ใช้ระบุชนิดของดิสคริปเตอร์ ในกรณีเป็น HID ดิสคริปเตอร์จะมีค่าเท่ากับ 21H

HID vertion มีขนาด 2 ไบต์ กำหนดในลักษณะรหัส BCD ใช้ระบุหมายเลขคุณสมบัติของ HID หรืออาจกล่าวได้ว่าเป็นการระบุเวอร์ชั่นก็ได้ โดยค่านี้จะใช้ตัวเลขฐานสิบ 4 ตัว 2 ตัวแรกใช้กำหนดตัวเลขเวอร์ชั่นหลัก ส่วน 2 ตัวหลังใช้กำหนดตัวเลขเวอร์ชั่นย่อย คั่นด้วยจุดทศนิยม ยกตัวอย่าง 0100 หมายถึง เวอร์ชั่น 1.0 ถ้าเป็นเวอร์ชั่น 1.1 ข้อมูลจะเป็น 0110 เป็นต้น

CountryCode มีขนาด 1 ไบต์ใช้กำหนดรหัสของประเทศที่เกี่ยวข้องกับอุปกรณ์ USB ตัวนั้นๆ ขึ้น ตัวอย่างที่เห็นได้นชัดคือ คีย์บอร์ด เนื่องจากในแต่ละประเทศอาจมีการใช้คีร์บอร์ดแตกต่างกัน เช่น ในประเทศไทย, จีน ,อังกฤษ เป็นต้น

NumDescriptors มีขนาด 1 ไบต์ ใช้ระบุจำนวนดิสคลิปเตอร์ว่าเป็น รีพอร์ต หรือ ฟิสิกัลดิสคริปเตอร์ ถ้าเป็นรีพอร์ตดิสคริปเตอร์จะมีค่าเท่ากับ 22H

DescriptorLength มีขนาด 2 ไบต์ ใช้แจ้งขนาดหรือความยาวของรีพอร์ตดิสคริปเตอร์

3) รีพอร์ตดิสคริปเตอร์ (Report descriptor)

เป็นส่วนประกอบหลักที่อาจกล่าวได้ว่าสำคัญมากที่สุด เนื่องจากข้อมูลที่เก็บอยู่ในดิสคริปเตอร์ตัวนี้เป็นสิ่งที่อธิบายถึงรูปแบบและวิธีการใช้ข้อมูลเพื่อให้บรรจุวัตถุประสงค์ตามที่อุปกรณ์ USB ตัวนั้นกำหนดไว้ ยกตัวอย่าง หากอุปกรณ์ USB นี้เป็นเมาส์ ข้อมูลในรีพอร์ตดิสคริปเตอร์จะรายงานให้ทราบถึงการเคลื่อนที่ของเมาส์เพื่อระบุตำแหน่งและสถานะการกำปุ่มของเมาส์ เป็นต้น

รีพอร์ตดิสคริปเตอร์จะมีความยาวเท่าใดก็ได้ แต่ต้องแจ้งความยาวลงใน DescriptorLength ภายใน HID ดิสคริปเตอร์ เพื่อให้โฮสต์ทราบด้วย หลักและเป้าหมายเบื่องต้นของการใช้งานรีพอร์ตดิสคริปเตอร์มีดังนี้

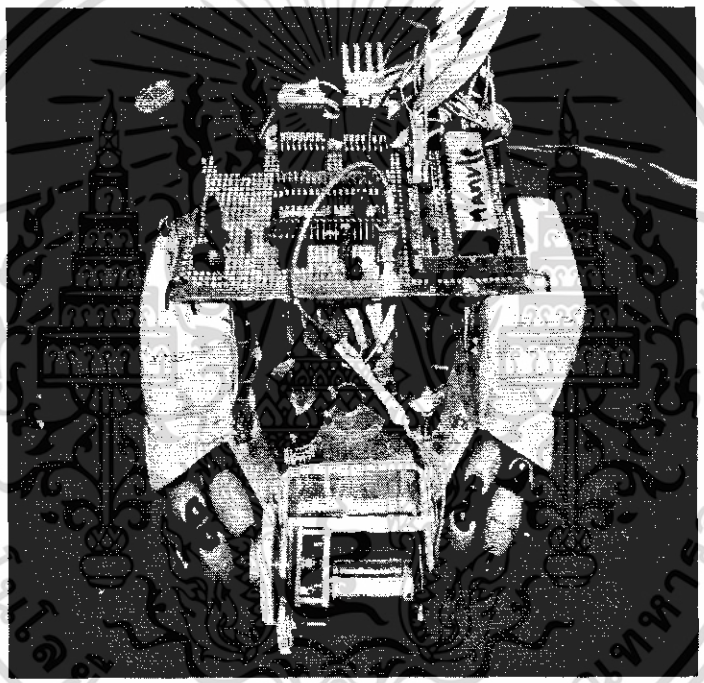
1. ใช้เก็บรายละเอียดข้อมูลของอุปกรณ์ให้ครบถ้วนที่สุดภายใต้พื้นที่ที่เล็กที่สุดเท่าที่เป็นไปได้
2. ยอมให้ซอฟต์แวร์ประยุกต์ที่เกี่ยวข้องสามารถข้ามข้อมูลที่ไม่ชัดเจนไปได้ เพื่อให้ยังคงสามารถดำเนินการต่อไปได้
3. สามารถขยายได้
4. รองรับการจัดเก็บและรวบรวมข้อมูล
5. สามารถอธิบายรายละเอียดด้วยข้อมูลภายในดิสคริปเตอร์เอง เพื่อประโยชน์ในการพัฒนาซอฟต์แวร์ประยุกต์สำหรับใช้งานร่วมกันได้

บทที่ 5

การออกแบบหุ่นยนต์เตะฟุตบอล

5.1 การติดตั้งโครงสร้างของหุ่นยนต์

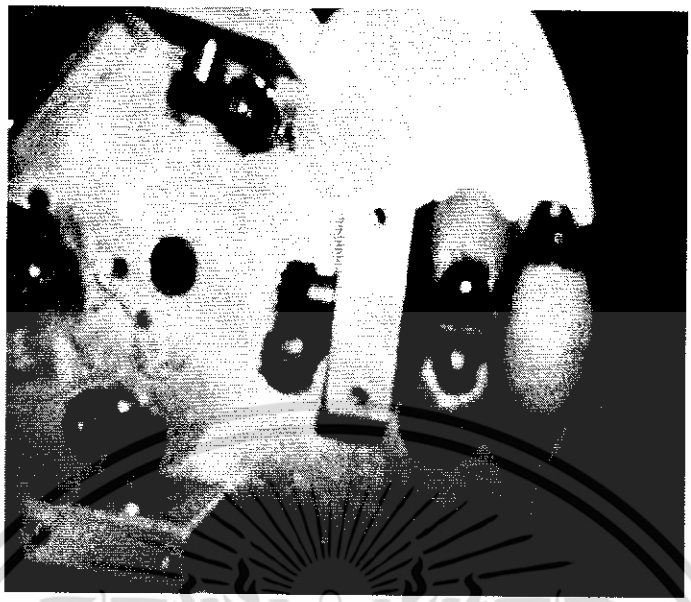
ส่วนประกอบของหุ่นยนต์จะใช้ล้อ OMNI DIRECTIONAL เพื่อให้รถเคลื่อนที่ได้อิสระรอบทิศทางด้วยมอเตอร์ทดรอบ 12 Volt สี่ตัวติดตั้งในแนวตั้งเพื่อลดพื้นที่ในการติดตั้วแบะหมุนล้อด้วยเฟืองเฉียง พร้อมด้วยชุดปั่นลูกบอลให้ติดกับตัวหุ่นด้วย DC MOTOR และชุดยิงลูกบอลด้วย SOLINOID 12 Volt และลักษณะการติดตั้งดังภาพ



รูปที่ 5-1 แสดงการโครงสร้างของตัวหุ่นยนต์

5.2 การติดตั้งของล้อของตัวหุ่นยนต์

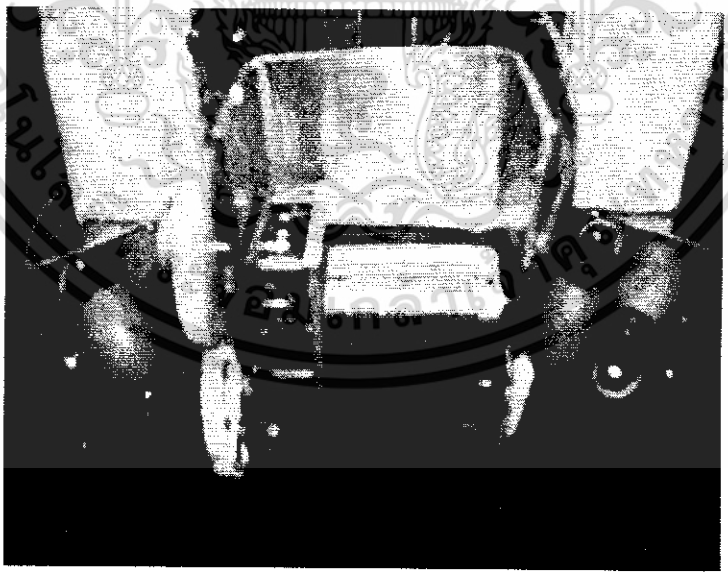
ล้อที่ใช้ในการติดตั้งจะเป็นล้อแบบ OMNI DIRECTIONAL ซึ่งจะเป็นแบบสำเร็จรูปมาแล้วจะใช้ล้อละ 2 ตัวประกอบกันเพื่อให้การเคลื่อนที่ได้อิสระและจะใช้ตัวมอเตอร์ขับล้อ ใช้เฟืองส่งกำลังเพื่อส่งถ่ายกำลังไปยังล้อ และทำการติดตั้งในทิศทางตั้งฉากกันระหว่างมอเตอร์กับล้อ



รูปที่ 5-2 แสดงการติดตั้งตัวล้อขับของตัวหุ่นยนต์

5.3 การติดตั้งตัวเลี้ยงลูกบอลของตัวหุ่นยนต์

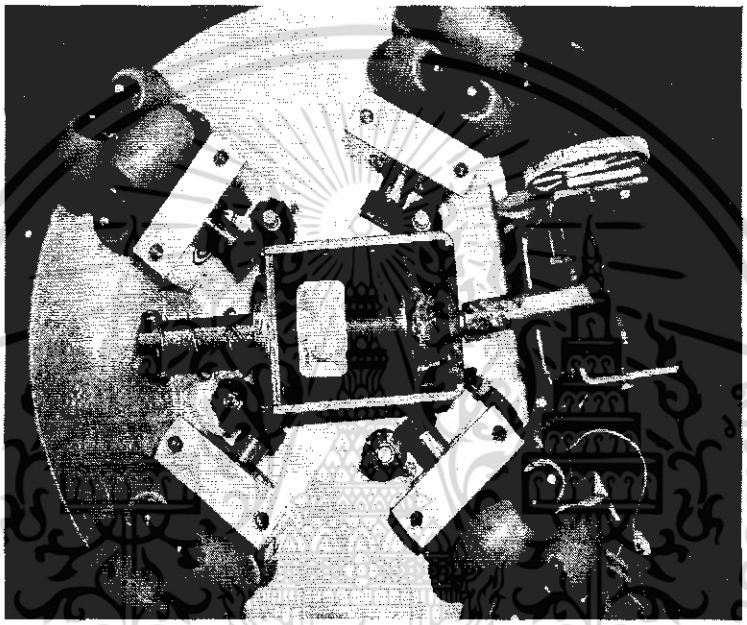
การติดตั้งตัวเลี้ยงบอลจะเป็นยางเพื่อให้เกิดความหนืดซึ่งเป็นสมบัติของยางในการปั่นบอลเข้ากับตัวหุ่นยนต์ เพื่อให้บอลติดกับตัวหุ่นยนต์ตลอดเวลาโดยปลายด้านหนึ่งของตัวชุดปั่นนี้จะต่อเฟืองส่งกำลังเข้ากับมอเตอร์ดังรูปที่ 5.3



รูปที่ 5-3 แสดงการติดตั้งตัวเลี้ยงของตัวหุ่นยนต์

5.4 การติดตั้งตัวยิงลูกบอลของตัวหุ่นยนต์

การติดตั้งตัวยิงลูกบอลถือเป็นอุปกรณ์อีกอย่างที่มีความสำคัญในการทำประตู ในส่วนอุปกรณ์หลักที่ใช้โซลินอยด์ 12 โวลต์ โดยตำแหน่งที่ติดตั้งนั้นจะอยู่ใต้ตัวเลี้ยงลูกบอลลงไป และตำแหน่งที่กระทบกับลูกบอลนั้นจะต้องโดนกึ่งกลางของลูกบอลพอดี เพื่อที่การยิงนั้นไม่ให้มีการหมุนย้อนกลับ ในส่วนตัวของโซลินอยด์จะต้องมีการติดตัวสปริงให้มีการดึงกลับหลังการยิงแล้วดังรูปที่ 5.4

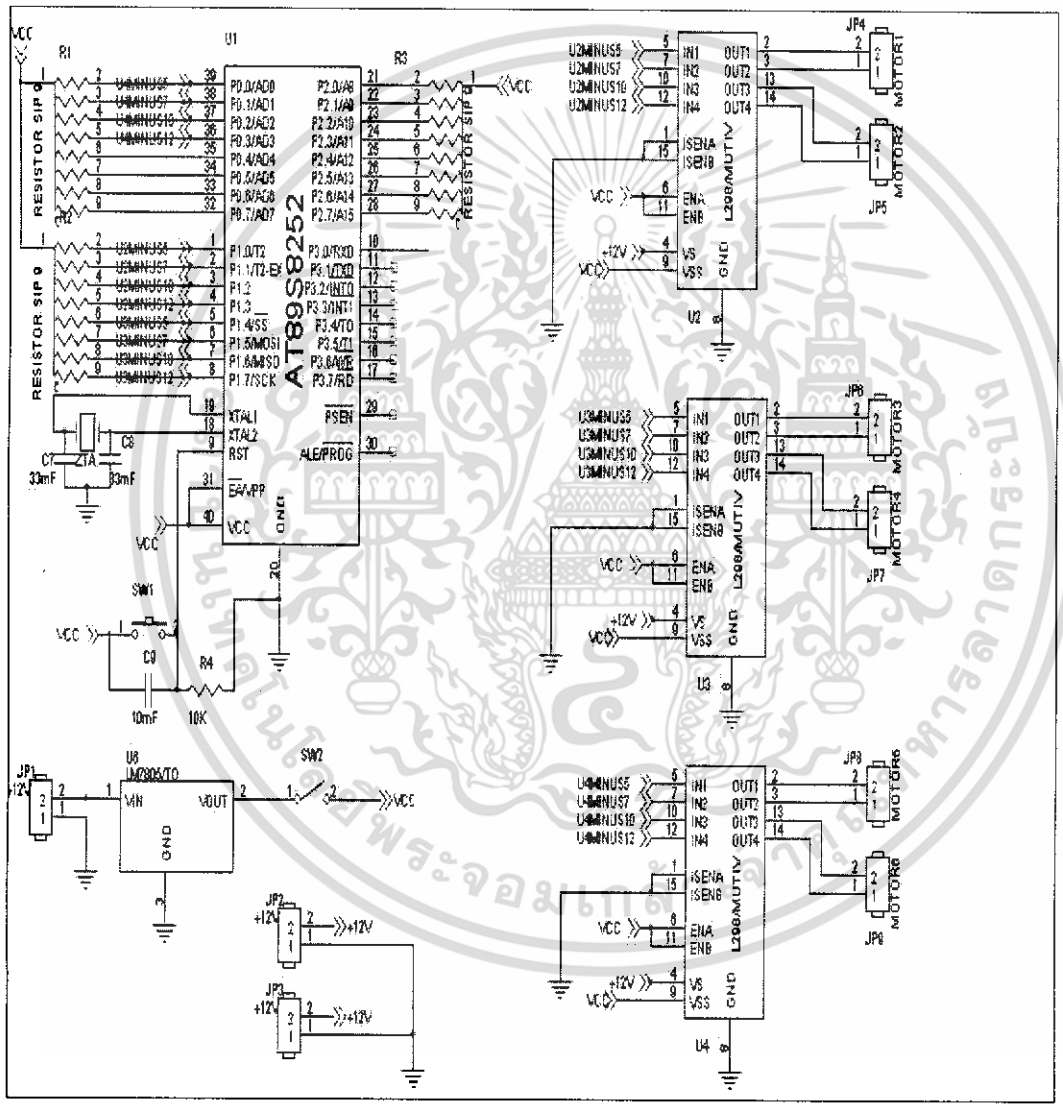


รูปที่ 5-4 แสดงการติดตั้งตัวยิงลูกบอลของตัวหุ่นยนต์

บทที่ 6
ระบบควบคุมของหุ่นยนต์เตะฟุตบอล

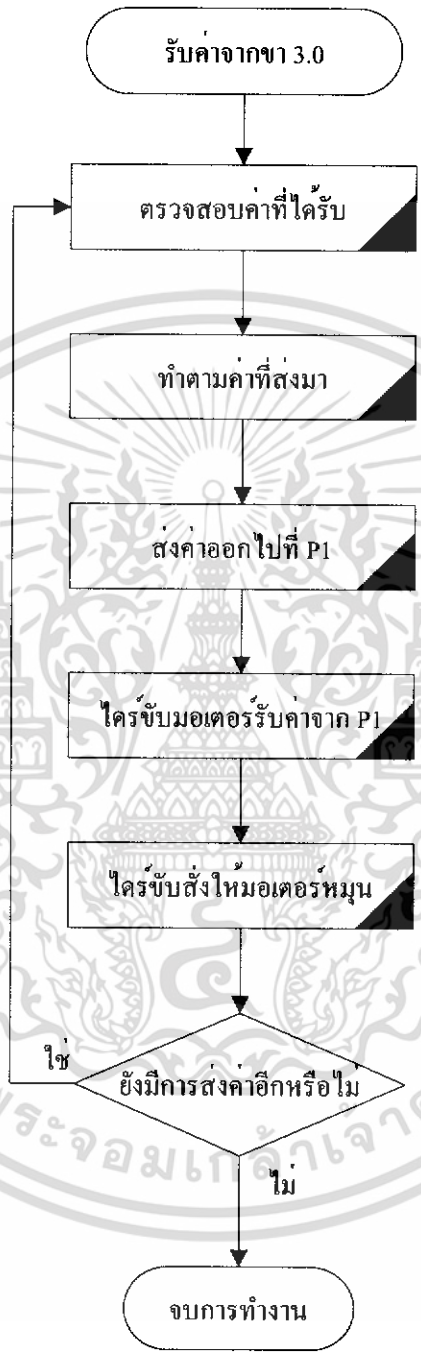
6.1 การออกแบบวงจรไฟฟ้า

ในการออกแบบวงจรไฟฟ้าจะใช้ โปรแกรม ORCAD ในการออกแบบวงจร และการออกแบบ แผ่นปริ้นท์ โดยจะได้รูปแบบของวงจรในแต่ละส่วน ดังนี้



รูปที่ 6-1 วงจรไมโครคอนโทรลเลอร์ MCS-51 ที่ออกแบบโดยโปรแกรม ORCAD

6.2 การออกแบบโปรแกรมการสั่งงาน



รูปที่ 6-2 แผนผังลำดับการทำงานของโปรแกรมภาษาซีในไมโครคอนโทรลเลอร์

บทที่ 7

การออกแบบโปรแกรมคำนวณพิกัดของตัวหุ่นยนต์เตะฟุตบอล

7.1 สาเหตุที่เลือกใช้โปรแกรมวิชวลเบสิกในการเขียนโปรแกรม

7.1.1 เป็นภาษาระดับกลาง มีลักษณะคล้ายภาษาอังกฤษ จึงทำให้ง่ายต่อการทำความเข้าใจและพัฒนาโปรแกรม

7.1.2 มีคำสั่งที่ใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมซึ่งสามารถนำไปควบคุมเครื่องได้

7.1. 3 มีความสามารถในงานประเภทกราฟฟิคอยู่ในระดับที่น่าพอใจ ทำให้สามารถพัฒนาโปรแกรมได้อย่างสะดวก

7.1.4 วิชวลเบสิกใช้พัฒนาโปรแกรมที่ใช้สื่อสารกับผู้ใช้เป็นหลัก ดังนั้นจึงสามารถพัฒนาหน้าตา(Interface)ของโปรแกรมออกมาได้สวยงามน่าใช้

โปรแกรมคำนวณพิกัดในการเคลื่อนที่

โปรแกรมจึงได้ถูกพัฒนาออกให้ให้ผู้ใช้สามารถใช้งานได้อย่างง่ายดาย โดยโปรแกรมจะแบ่งขั้นตอนต่างๆ ออกเป็น 3ส่วน คือส่วนการรับค่าจากการคลิกเมาส์ ,ส่วนของการคำนวลค่ามุมของตัวหุ่นและระยะทางที่ทำกับลูกบอล ผู้ใช้สามารถคลิกเมาส์ที่รูปสนามฟุตบอลจำลองได้เลย, และส่วนที่ทำการสร้างค่าของตัวอักษรที่ต้องการจะส่ง หลังจากนั้นก็ส่งออกไปที่พอร์ตอนุกรมได้เลยโดยมีการส่งออกมาที่ระตัวอักษรในระยะเวลาที่สามารถกำหนดได้ส่วนทางด้านไมโครคอนโทรลเลอร์ก็จะรับค่าที่ได้มาแปรเป็นลักษณะการทำงานที่เราต้องการเพื่อให้ตัวหุ่นเคลื่อนที่ ในส่วนของการใช้งานพอร์ตยูเอสบีก็จะเขียนโปรแกรมติดต่อเหมือนกันแต่ว่าจะส่งค่าตัวแปรโดยตรงโดยไม่ต้องคำนวนหาค่าพิกัดที่ได้

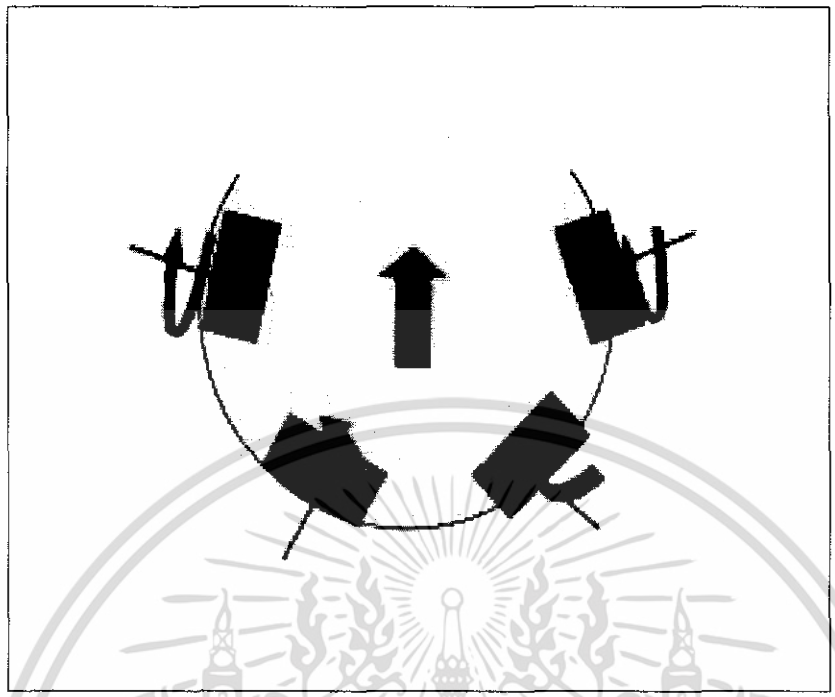
ในการเคลื่อนที่ของหุ่นเองยังคงเป็นการเคลื่อนที่แบบแยกส่วนของการเลี้ยวและการเดินตรงออกจากกันจึงทำให้การเคลื่อนที่เข้าหาลูกบอลเป็นไปด้วยความไม่รวดเร็วซึ่งสามารถพัฒนาให้ทำงานคู่กันได้ต่อไป

7.2 หลักการทำงานของโปรแกรม

หลังจากที่เราได้ทำการเปิดโปรแกรมจะพบกับหน้าต่างของรูปภาพสนามฟุตบอลซึ่งเราจะเริ่มต้นตั้งตัวหุ่นให้หันหน้าออกไปยังฝั่งตรงข้ามในตำแหน่งประตูของตัวเอง แล้วทำการคลิกเส้นทางการเคลื่อนที่ที่เราต้องการ เมื่อเราเริ่มคลิกจะทำการเก็บค่าพิกัดของตำแหน่งเมาส์ต่างๆ เป็นตัวแปรอะเรย์เมื่อเราคลิกเสร็จแล้วโปรแกรมจะทำการคำนวนหามุมของตัวหุ่นที่กระทำกับลูกฟุตบอลแล้วเปลี่ยนค่าของมุมเป็นค่าของตัวอักษรซึ่งจะแปรผันตามค่าของมุมนั้นเมื่อทำการเช็คมุมเรียบร้อยแล้วก็จะทำการหาระยะทางของตัวหุ่นกับลูกบอล ซึ่งก็จะเปรียบเทียบค่าของตัวเลขระยะทางที่ได้มาเป็นตัวอักษรเหมือนกันกับการหามุม ซึ่งการกระทำเช่นนี้จะเป็นลักษณะของกระทำแบบเวกเตอร์ คือจะเป็นแบบหางต่อหัวโดยเลื่อนตำแหน่งของการคำนวนไปเลื่อยๆ จนถึงค่าสุดท้ายค่าของตัวอักษรที่ต้องการส่งไปมีดังนี้คือ

ตัวอักษรที่ต้องการจะส่ง	สถานะของการเคลื่อนที่มอเตอร์	ค่า % PWM
1	เลี้ยวซ้าย	20%
2	เลี้ยวซ้าย	40%
3	เลี้ยวซ้าย	60%
4	เลี้ยวซ้าย	80%
5	เลี้ยวซ้าย	100%
6	เลี้ยวขวา	20%
7	เลี้ยวขวา	40%
8	เลี้ยวขวา	60%
9	เลี้ยวขวา	80%
0	เลี้ยวขวา	100%
A	หยุดนิ่ง	0%
B	เดินหน้า	20%
C	เดินหน้า	40%
D	เดินหน้า	60%
E	เดินหน้า	80%
F	เดินหน้า	100%

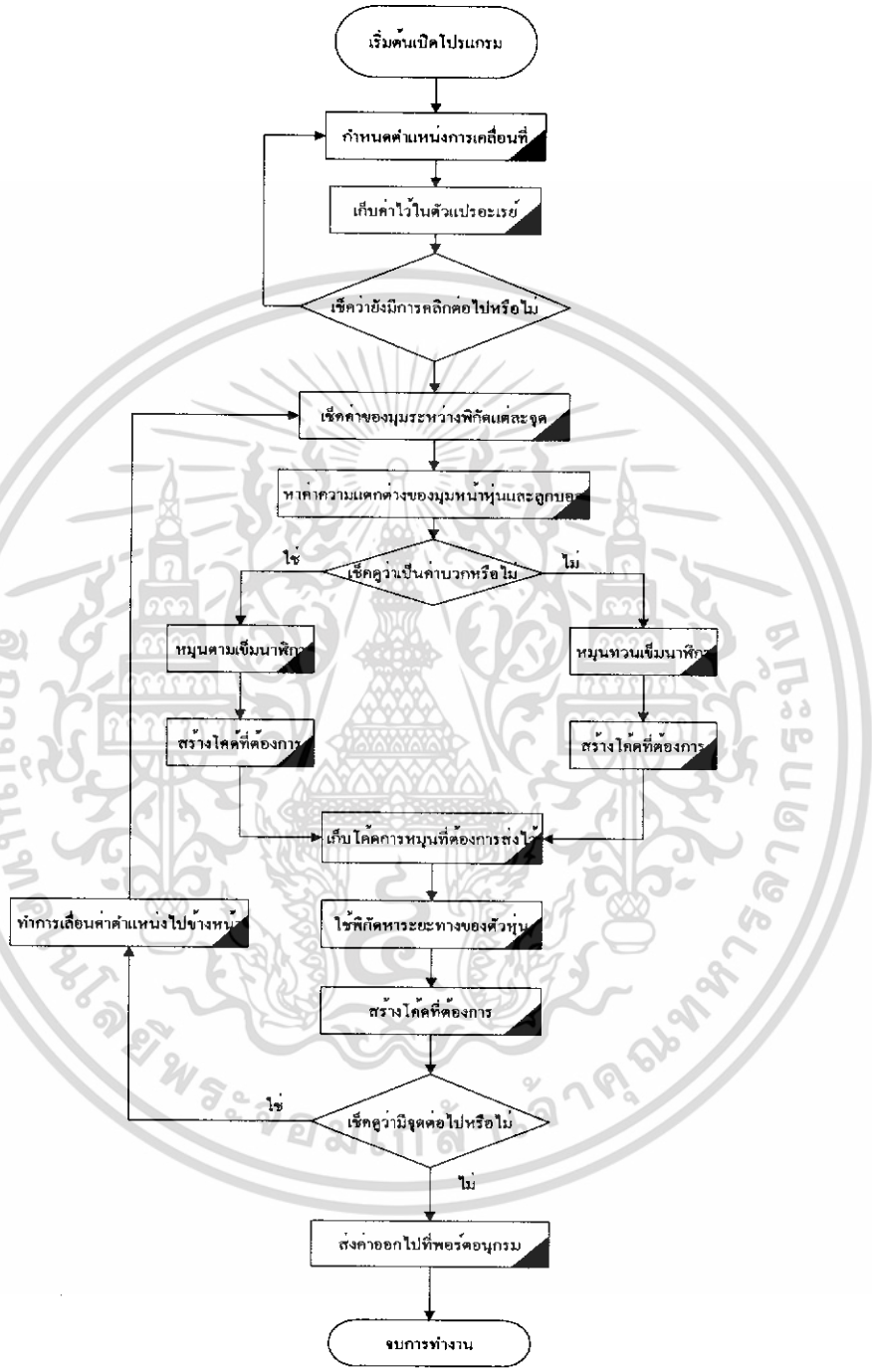
ตารางที่ 7.1 แสดงตัวอักษรและสถานะของการเคลื่อนที่ของตัวหุ่นเมื่อเราส่งค่าออกไป



รูปที่ 7.1 แสดงลักษณะการเดินหน้าของหุ่นยนต์

รูปที่ 7.2 แสดงลักษณะการเลี้ยวของหุ่นยนต์

7.3 แผนผังลำดับการทำงานของโปรแกรม

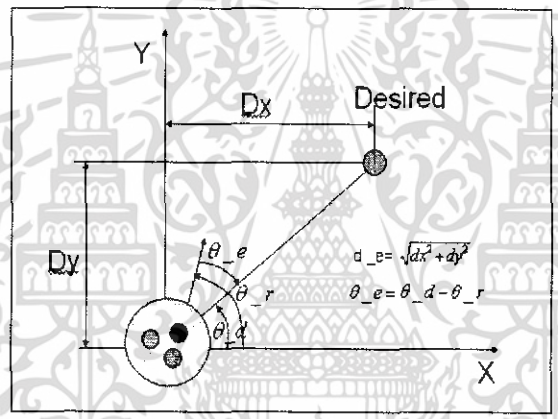


รูปที่ 7.3 แผนผังลำดับการทำงานของโปรแกรมวิชวลเบสิกในคอมพิวเตอร์

7.4 โปรแกรมการสร้างโค้ด (Generat Code Program)

จากการที่รูปภาพในคอมพิวเตอร์เกิดจากการนำจุดสีขนาดเล็ก)หรือที่เรียกกันว่า Pixels (หลายๆจุดมาเรียงต่อกัน ประกอบกันเป็นภาพ เมื่อเราใช้ฟังก์ชันเมาส์คลิกจะมีการแสดงค่าของสีซึ่งค่าสีในแต่ละจุดจะถูกจัดเก็บในระบบตัวเลข (Digital) และค่าของพิกัดในแกน x และ y ของตัวโปรแกรมเอง ซึ่งเราจะทำการเก็บค่าเฉพาะพิกัดมาคำนวนค่าของมุมและระยะทางที่เราต้องการได้

โปรแกรมคำนวณตำแหน่งจุดที่จะเคลื่อนที่จะอาศัยกฎของปีทาก้อรัสเปรียบเทียบค่าที่เราต้องการและนำเอาค่าความแตกต่างของมุมมาสร้างโค้ดที่ต้องการซึ่งจะทำให้หุ่นเคลื่อนที่ไปในตำแหน่งที่เราต้องการได้



รูปที่ 7.4 แสดงวิธีการคำนวณหามุมเพื่อสร้างโค้ดต่างๆ

ตัวอย่างโค้ด 33321CDEEEEEDCA

ความหมาย

33 : หมุนตัวหุ่นไปทางซ้ายโดยใช้ PWM 60%

321 : ลดความเร็วของการหมุนของตัวหุ่นลง

CDE : หุ่นยนต์เคลื่อนที่ไปข้างหน้าในลักษณะการเพิ่มความเร็วจนได้ค่า PWM 80%

EEE : หุ่นยนต์เคลื่อนที่ไปข้างหน้าด้วยความเร็วคงที่ที่ค่า PWM 80%

EDC : ลดความเร็วของตัวหุ่นลง

A : หยุดการเคลื่อนที่ของตัวหุ่น

บทที่ 8

รูปแบบและผลการทดลอง

8.1รูปแบบการทดลอง

รูปแบบของการทดลองเพื่อหาประสิทธิภาพรวมของเครื่องสร้างแบบสำหรับหุ่นยนต์เตะฟุตบอลนี้ จะแบ่งการทดลองออกเป็น 2 ส่วนด้วยกัน คือ

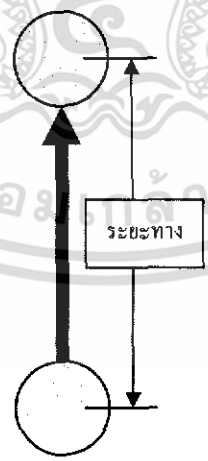
1. ส่วนโปรแกรมที่ใช้ในการกำหนดตำแหน่งการเคลื่อนที่ของตัวหุ่นยนต์
2. ส่วนของการเคลื่อนที่ของตัวหุ่นยนต์ที่เคลื่อนที่เป็นเส้นตรง
3. ส่วนของการเคลื่อนที่ของตัวหุ่นยนต์ที่เคลื่อนที่เป็นมุม

การทดลองนี้จะเป็นการตรวจเช็คว่าการทำงานของตัวหุ่นยนต์ยังมีข้อบกพร่องที่จะต้องแก้ไขบ้างหรือไม่ และจะทำการแก้ไขอย่างไรจึงจะทำให้การทำงานของตัวหุ่นยนต์ได้มีประสิทธิภาพมากที่สุด

8.2 วิธีการทดลอง

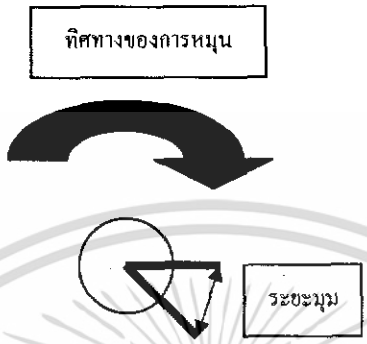
1. ส่วนโปรแกรมที่ใช้ในการกำหนดตำแหน่งการเคลื่อนที่ของตัวหุ่นยนต์มีขั้นตอนดังนี้ คือจะต้องทำการคลิกที่หน้าต่างรูปสนามฟุตบอลโดยตำแหน่งแต่ละค่าที่จะเก็บไว้จะอยู่ที่ปลายเมาส์ จากนั้นโปรแกรมจะทำการคำนวนค่าให้เป็นองศาเพื่อตรวจเช็คค่าที่เราคำนวนว่าจะต้องมีค่าประมาณกับความเป็นจริงของการเคลื่อนที่ของตัวหุ่นยนต์

2. ส่วนของการเคลื่อนที่ที่เป็นเส้นตรงของตัวหุ่นยนต์ มีขั้นตอนดังนี้คือการสั่งให้ตัวหุ่นยนต์เคลื่อนที่เป็นเส้นตรง การทดลองนี้จะทำการวัดความผิดพลาดของแต่ละจุดแล้วมาทำการเฉลี่ยกันโดยคำนวณจะทำการคิดเป็นเปอร์เซ็นที่เคลื่อนที่ออกจากตำแหน่งที่ถูกต้อง



รูปที่ 8-1 แสดงถึงลักษณะการเคลื่อนที่แบบเส้นตรงที่ใช้ในการทดลอง

3. ส่วนของการเคลื่อนที่แบบหมุนของตัวหุ่นยนต์จะขั้นตอนของการทดลองคล้ายกันกับการทดลองของข้อที่แล้ว คือ การวัดเป็นองศาของการเคลื่อนที่ของแต่ละมุม แล้วนำค่ามาเฉลี่ยจะได้เปอร์เซ็นของความผิดพลาด



รูปที่ 8-2 แสดงถึงลักษณะการเคลื่อนที่แบบหมุนที่ใช้ในการทดลอง

8.3 ผลการทดลอง

ตารางผลการทดลองของการเคลื่อนที่เป็นเส้นตรงแบบไม่เลี้ยงลูกบอล

ระยะทาง (ซม.)	ระยะคลาดเคลื่อน	ค่าความผิดพลาด (%)	หมายเหตุ
30	4.1	13.64	-
60	6.4	10.67	-
90	11.7	13.0	-
120	14.9	12.4	-
150	13.4	8.9	-
180	10.1	5.16	-
210	28.3	13.47	-

ตารางที่ 8-1 แสดงผลการทดลองที่ได้แบบเลี้ยงลูกบอล

ตารางผลการทดลองของการเคลื่อนที่เป็นเส้นตรงแบบเฉี้ยงลูกบอล

ระยะทาง (ซม.)	ระยะคลาดเคลื่อน	ค่าความผิดพลาด (%)	หมายเหตุ
30	7.5	25.0	-
60	13.9	23.2	-
90	18.7	20.78	-
120	20.5	17.12	-
150	12.8	8.53	-
180	10.8	6.0	-
210	32.4	13.42	-

ตารางที่ 8-2 แสดงผลการทดลองที่ได้จากการวิ่งแบบมีลูกบอล

ตารางผลการทดลองของการหมุนแบบไม่เฉี้ยงลูกบอล

มุม (องศา)	ระยะมุมคลาดเคลื่อน (องศา)	ค่าความผิดพลาด (%)	หมายเหตุ
30	2.3	7.67	-
60	4.4	7.3	-
90	6.87	7.53	-
120	8.95	7.46	-
150	11.3	7.53	-
180	12.78	7.12	-

ตารางที่ 8-3 แสดงผลการทดลองที่ได้จากการหมุนแบบไม่เฉี้ยงมีลูกบอล

บทที่ 9
วิเคราะห์และสรุปผลการทดลอง

9.1 การวิเคราะห์ผลการทดลองและสรุปผลการทดลอง

การวิเคราะห์และสรุปผลการทดลองแบ่งออกเป็น 3 ส่วนคือ

9.1.1 ส่วนโปรแกรมควบคุมการเคลื่อนที่ของตัวหุ่น

จากการทดลองการทำงานของโปรแกรมและระบบต่างๆ ยังมีค่าความค่าผิดพลาด จึงทำให้ตำแหน่งของการเคลื่อนที่ในแนวเส้นตรงของตัวหุ่นยนต์มีค่าของความผิดพลาดมากที่สุดเป็น 13.64% สำหรับการเคลื่อนที่ที่ไม่มีลูกบอล ค่าความผิดพลาด 25% สำหรับการเคลื่อนที่ที่มีลูกบอล ค่าความผิดพลาด 7.67% สำหรับการหมุนที่ไม่มีลูกบอล

เนื่องจากการทำงานเป็นแบบทำงานที่ละส่วนคือเดินหน้าแล้วหมุนไปเลื่อยๆ จึงทำให้ความเร็วในการเคลื่อนที่ของตัวหุ่นไปยังลูกบอลยังไม่เร็วพอ

คำสั่งที่ใช้ในการเขียนโปรแกรมบางส่วนมีประสิทธิภาพไม่สูงนัก เนื่องด้วยเวลาที่จำกัดผู้จัดเลือกจึงที่จะใช้คำสั่งพื้นฐานเป็นส่วนใหญ่ แม้ว่าจะให้ผลออกมาใกล้เคียงกันแต่คำสั่งที่เขียนขึ้นนั้นมีสภาพการทำงานที่ไม่เร็วนักหรือมีปัญหาได้ในบางกรณี

จากโปรแกรมคำนวณตำแหน่งบางครั้งถ้าเราเลือกใช้ค่าในระดับเท่าสนามจริงจะทำให้ค่าของมุมหาค่าไม่ได้เพราะ arctan = 0 หาค่าไม่ได้จึงควรเขียนโปรแกรมป้องกันไว้ก่อน

9.1.2 ส่วนของแผงควบคุม

เนื่องจากวงจรควบคุมยังไม่ได้ใช้คลื่นวิทยุในการควบคุม จึงมีสายไฟโยงจากคอมพิวเตอร์ไปยังชุดควบคุมของตัวหุ่น นอกจากนี้ยังมีการเชื่อมโยงสายสัญญาณต่างๆระหว่างแผงอีกด้วย ทำให้เกิดการสับสนวุ่นวายเป็นอย่างมาก

แผงวงจรบางส่วนถูกออกแบบตามทฤษฎีโดยที่ไม่ได้ทดลองก่อน ทำให้วงจรบางส่วนไม่สามารถทำงานได้ตามที่ออกแบบไว้ เนื่องจากปัจจัยอื่นๆที่ไม่ได้คาดคิดไว้ก่อน

9.1.3 ตัวโครงสร้างของหุ่นยนต์เตะฟุตบอล

เนื่องจากอุปกรณ์ไม่ค่อยจะมีมากนักและเวลาที่มีจำกัด ทำให้ชิ้นส่วนแต่ยังมีประสิทธิภาพน้อยกว่าที่ต้องการ จึงทำให้ตัวหุ่นเองไม่ค่อยมีประสิทธิภาพเท่าที่ควร

แม้ว่าชิ้นส่วนแต่ละชิ้นจะถูกออกแบบมาให้ง่ายต่อการผลิต แต่ว่าเนื่องจากประสบการณ์และเครื่องมือที่ใช้ยังไม่ค่อยมีประสิทธิภาพเท่าที่ควรจึงทำให้เกิดอุปสรรคบ้าง

9.2 แนวทางในการแก้ไข

แนวทางในการแก้ไขจะแบ่งการวิเคราะห์ออกเป็น 3ส่วนคือ

1) ส่วนโปรแกรมควบคุมการเคลื่อนที่

เนื่องจากการทำงานเป็นแบบไล่อ่านสีทีละพิกเซล ดังนั้นควรมีปรับปรุงให้การเคลื่อนที่เป็นการวิ่งตามเส้นแทน อาจจะใช้การคำนวนเชิงตัวเลขเข้ามาช่วยเพื่อลดค่าความผิดพลาดของการเคลื่อนที่ของตัวหุ่นด้วยซึ่งจะส่งผลกับค่าของตัวโค้ดที่ส่งไปแม่นยำมากขึ้น

คำสั่งที่ใช้ในการเขียนโปรแกรมบางส่วนอาจมีประสิทธิภาพไม่สูงนัก เนื่องด้วยเวลาที่จำกัดผู้จัดทำจึงเลือกที่จะใช้คำสั่งพื้นฐานเป็นส่วนใหญ่ แม้ว่าจะให้ผลออกมาใกล้เคียงกันแต่คำสั่งที่เขียนขึ้นนั้นอาจทำงานช้าหรือมีปัญหาได้ในบางกรณี ดังนั้นในการพัฒนาเครื่อง

จากโปรแกรมคำนวณทางด้านฝั่งของไมโครคอนโทรลเลอร์(MCS-51) จะไม่เป็นการสะดวกนักถ้าเราจะนับเอนโค้ดเดอร์เพราะสามารถนับได้แค่ 2 ตัว จึงควรหาตัวไมโครคอนโทรลเลอร์ตระกูลใหม่ที่เหมาะสมกว่าตัวนี้เพื่อที่จะให้การทำวงจรสะดวกขึ้นกว่านี้

2) ส่วนของแผงควบคุม

เนื่องจากวงจรควบคุมต้องควบคุมอุปกรณ์จำนวนมากแต่ต้องอยู่ในพื้นที่ที่จำกัดจึงทำให้การวางตำแหน่งที่ต้องการไม่ได้จึงต้องออกแบบวงจรให้เหมาะสมกับตำแหน่งที่สามารถจะวางได้

แผงวงจรบางส่วนถูกออกแบบตามทฤษฎีโดยที่ไม่ได้ทดลองก่อน ทำให้วงจรบางส่วนไม่สามารถทำงานได้ตามที่ออกแบบไว้ เนื่องจากปัจจัยอื่นๆที่ไม่ได้คาดคิดไว้ก่อน ดังนั้นแผงวงจรจึงควรได้รับการออกแบบหรือคำแนะนำจากผู้ที่มีประสบการณ์

3) ส่วนของโครงสร้าง

เนื่องจากงบประมาณและเวลาที่มีจำกัด ทำให้ชิ้นส่วนแต่ละชิ้นถูกออกแบบมาให้สามารถสร้างได้โดยง่าย วัสดุหรืออุปกรณ์ที่เหมาะสมกับตัวหุ่นไม่ค่อยจะมีตามท้องตลาด และอุปกรณ์บางตัวเป็นชิ้นส่วนที่หาได้รอบตัว ดังนั้นเครื่องอาจดูไม่สวยงามมากนัก แม้ว่าชิ้นส่วนแต่ละชิ้นจะถูกออกแบบมาให้ง่ายต่อการผลิต แต่ในความเป็นจริงแล้วการผลิตด้วยมือจะทำให้เกิดความคลาดเคลื่อนขึ้นทำให้ประสิทธิภาพบางส่วนลดลง ดังนั้นหากต้องการสร้างเพื่อการแข่งขันแล้วอาจจะต้องออกแบบบางส่วนใหม่ให้สามารถทำงานได้ดีขึ้น และชิ้นส่วนควรสร้างด้วยเครื่องจักรอัตโนมัติเพื่อลดความคลาดเคลื่อนที่เกิดขึ้น

บรรณานุกรม

[1] อภิชาติ กู่พลับ, "เริ่มต้นเขียนโปรแกรมติดต่อกันและควบคุมฮาร์ดแวร์ด้วย Visual Basic 6", Infopress Develoop Book, 2546

[2] ธนพล ฉันจรัสวิชัย, "ปฏิบัติการ Visual Basic สำหรับ Common Windows", ซีเอ็ดยูเคชั่น2546

[3] โชติพันธุ์ หล่อเลิศสุนทรและฐิตะพันธุ์ หล่อเลิศสุนทร, "สอนเขียน Visual Basic 6.0 ให้เป็นProject", Soft Express&Publishing, 2543

[4] สมศักดิ์ ศรีขจรเกียรติ, "Visual Basic 6. Teach Yourself", บิบลิโอไฟล์, 2542

[5] "Visual Basic Graphic Programming", John Wiley & sons, Inc, 1997

[6] A.F. Bakker, "Design of a high speed low friction XY-table", Phillips CentreFor industrial Technology (CFT)

[7] Hassian Z. Tameem, "Design and development of x-y positioning table using stepper motor and belt drive", Department of Mechanical and Production Engineering Yeshwantrao Chaven college of engineering Wanadongri,Nagpur

[8] วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล, "เรียรู้และปฏิบัติการ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชฉบับ AT89C5X ของ Atmel," Innovative Experiment Co., Ltd.

[9] Joseph E. Shigley, Charles R. Mischke, Richard G. Budynas, "Mechanical Engineering Design",Seventh Edition,McGrawHill.

[10] www.thaiio.com

[11] www.pantip.com

[12] www.howstuffwork.com

[13] www.vbcode.com

[14] www.download.com

[15] www.google.co.th

ภาคผนวก ก

วิชวลเบสิกซอร์สโค้ด (Visual Basic Source Code)

'**

ส่วนของหน้าต่างแสดงสนามฟุตบอล

'**

```
Const Pi = (22 / 7)
Const Deg = (180 / Pi)
Public Num1 As Integer, Num2 As Integer, Long1 As Integer
Public Sent As String, DeltaDeg As String, STR As String
Dim DataX(100) As Integer, DataY(100) As Integer
Private Sub Form_Load()
Num1 = 1
Num2 = 1
End Sub
Private Sub GenCode_Click()
Static TText As String
Text1.Text = Sent + "A"
TText = Text1.Text
End Sub
Private Sub Image1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Num2 = 1 Then
DataX(Num1) = X
DataY(Num1) = Y
End If
If Num2 >= 2 Then
Num1 = Num1 + 1
DataX(Num1) = X
DataY(Num1) = Y
Call Rotation
Call DDistant
```

```
DataX(Num1 + 1) = DataX(Num1)
DataY(Num1 + 1) = DataY(Num1)
Num1 = Num1 + 1
End If
Num2 = Num2 + 1
End Sub
Private Sub mnuExit_Click()
End
End Sub
Private Sub mnuJoystick_Click()
Joystick.Show
End Sub
Private Sub mnuSettingPort_Click()
FormConnet.Show
End Sub
Private Sub Rest_Click()
Num1 = 1
Num2 = 1
Sent = ""
Text1.Text = ""
Label1.Caption = ""
End Sub
Function DDistant()
Dim DelX As Integer, DelY As Integer
    DelX = (DataX(Num1) - DataX(Num1 - 1))
    DelY = (DataY(Num1) - DataY(Num1 - 1))
    Long1 = Sqr(((DelX / 9) * (DelX / 9)) + ((DelY / 9) * (DelY / 9)))
   Call DisSentCode
End Function
Function Rotation()
Dim DelX As Integer, DelY As Integer
```

```
Static DDeg1 As String
If Num1 < 3 Then
   DeltaDeg = 0
   Else
    DeltaDeg = DDeg1
End If
    DelX = (DataX(Num1) - DataX(Num1 - 1))
    DelY = (DataY(Num1) - DataY(Num1 - 1))
    If DelX = 0 Then
     DelX = 1
   End If
    DDeg1 = Deg * Atn(DelY / DelX)
     If DelX > 0 Then
       If DelY > 0 Then
          DDeg1 = DDeg1 * -1
          DeltaDeg = DDeg1 - DeltaDeg
          Label1.Caption = DeltaDeg
       If DeltaDeg > 0 Then
        Call RotTurnLiftSentCode
        Else
         Call RotTurnRightSentCode
        End If
        End If
     End If
     If DelX > 0 Then
       If DelY < 0 Then
          DDeg1 = Abs(DDeg1)
          DeltaDeg = DDeg1 - DeltaDeg
          Label1.Caption = DeltaDeg
       If DeltaDeg > 0 Then
        Call RotTurnLiftSentCode
```

```
    Else
     Call RotTurnRightSentCode
    End If
    End If
  End If
  If DelX < 0 Then
    If DelY > 0 Then
      DDeg1 = -180 + Abs(DDeg1)
      DeltaDeg = DDeg1 - DeltaDeg
      If DeltaDeg < -180 Then
        DeltaDeg = DeltaDeg + 360
      End If
      If DeltaDeg > 180 Then
        DeltaDeg = DeltaDeg - 360
      End If
      Label1.Caption = DeltaDeg
    If DeltaDeg > 0 Then
     Call RotTurnLiftSentCode
    Else
     Call RotTurnRightSentCode
    End If
  End If
 End If
 If DelX < 0 Then
    If DelY < 0 Then
      DDeg1 = 180 - Abs(DDeg1)
      DeltaDeg = DDeg1 - DeltaDeg
      If DeltaDeg < -180 Then
        DeltaDeg = DeltaDeg + 360
      End If
      If DeltaDeg > 180 Then
```

```
            DeltaDeg = DeltaDeg - 360
          End If
          Label1.Caption = DeltaDeg
        If DeltaDeg > 0 Then
         Call RotTurnLiftSentCode
        Else
         Call RotTurnRightSentCode
        End If
        End If
      End If
End Function
Function RotTurnRightSentCode()
Dim num As Integer
 Dim STR As String
  For num = 1 To Abs(DeltaDeg) Step 8
   STR = STR + "9"
  Next num
  'STR = "1234" + STR + "4321"
  Sent = Sent + STR
End Function
Function RotTurnLiftSentCode()
Dim num As Integer
 Dim STR As String
  For num = 1 To Abs(DeltaDeg) Step 8
   STR = STR + "4"
  Next num
  'STR = "6789" + STR + "9786"
  Sent = Sent + STR
End Function
Function DisSentCode()
  Dim num As Integer
```

```
  Dim STR As String
  For num = 1 To Long1 Step 4
     STR = STR + "E"
  Next num
  'STR = "ABCD" + STR + "DCBA"
   Sent = Sent + STR
End Function
```

****

ส่วนของฟังก์ชันติดต่อพอร์ตอนุกรม

****

```
Option Explicit
Public Buffer As String
Public num As String
Public M As Integer, N As Integer
Dim j As String
'»Ô´Ë'éÒµèÒ§
Private Sub Command3_Click()
End
End Sub
Private Sub Command1_Click()
Tx.Text = ""
Rx.Text = ""
End Sub
Private Sub connect_click()
If Option4.Value = True Then
 MSComm1.Settings = "110,n,8,1"
 Option4.Enabled = False
 End If
If Option5.Value = True Then
 MSComm1.Settings = "300,n,8,1"
 Option5.Enabled = False
```

```
 End If
If Option6.Value = True Then
 MSComm1.Settings = "600,n,8,1"
 Option6.Enabled = False
 End If
If Option7.Value = True Then
 MSComm1.Settings = "1200,n,8,1"
 Option7.Enabled = False
 End If
If Option8.Value = True Then
 MSComm1.Settings = "2400,n,8,1"
 Option8.Enabled = False
 End If
If Option9.Value = True Then
 MSComm1.Settings = "4800,n,8,1"
 Option9.Enabled = False
 End If
If Option10.Value = True Then
 MSComm1.Settings = "9600,n,8,1"
 Option10.Enabled = False
 End If
If Option11.Value = True Then
 Option11.Enabled = False
 MSComm1.Settings = "14400,n,8,1"
 End If
If Option12.Value = True Then
 MSComm1.Settings = "19200,n,8,1"
 Option12.Enabled = False
 End If
If com1.Value = True Then
 MSComm1.CommPort = 1
```

```
com1.Enabled = False
End If
If com2.Value = True Then
MSComm1.CommPort = 2
com2.Enabled = False
End If
If com3.Value = True Then
MSComm1.CommPort = 3
com3.Enabled = False
End If
MSComm1.PortOpen = True
Connect.Enabled = False
Timer1.Enabled = True
Call Timer1_Timer
End Sub
Private Sub Disconnect_Click()
If Not MSComm1.PortOpen Then
    MSComm1.PortOpen = True
  Else
    MsgBox ("Port already Close"), , "ComPort Error"
    MSComm1.PortOpen = False
    Timer1.Enabled = False
    Connect.Enabled = True
    com1.Enabled = True
    com2.Enabled = True
    com3.Enabled = True
    Option4.Enabled = True
    Option5.Enabled = True
    Option6.Enabled = True
    Option7.Enabled = True
    Option8.Enabled = True
```

```
        Option9.Enabled = True
        Option10.Enabled = True
        Option11.Enabled = True
        Option12.Enabled = True
'*************************************************************
        com1.Value = False
        com2.Value = False
        com3.Value = False
        Option4.Value = False
        Option5.Value = False
        Option6.Value = False
        Option7.Value = False
        Option8.Value = False
        Option9.Value = False
        Option10.Value = False
        Option11.Value = False
        Option12.Value = False
      num = 0
        M = 0
        N = 0
    End If
End Sub
Private Sub Quit_Click()
FormConnet.Hide
End Sub
Function Rev()
Dim InString As String
MSComm1.InputLen = 1
If MSComm1.InBufferCount Then
Rx.Text = Rx.Text + InString
 End If
```

```
End Function
Private Sub Timer1_Timer()
Dim STR() As String
num = Len(FormDisplay.Text1.Text)
ReDim STR(num)
  For M = 1 To num
    If M <= num Then
      STR(M) = Mid(FormDisplay.Text1.Text, M, 1)
    End If
  Next M
    N = N + 1
    If N <= num Then
      Buffer = STR(N)
      MSComm1.Output = Buffer
      Tx.Text = Tx.Text + Buffer
    End If
  Call Rev
End Sub
```

****

ส่วนของการติดต่อพอร์ต USB

****

```
Option Explicit
Public SentCode As String, RevCode As String
Public RevCodeX As String, RevCodeY As String
Private Sub cmdConnect_Click()
HIDComm.Browse
HIDComm.Connect
Timer1.Enabled = True
If Not MSComm1.PortOpen Then
    MSComm1.PortOpen = True
  Else
```

```
    MsgBox ("Port already Close"), , "ComPort Error"
End If
End Sub
Private Sub cmdDisconnect_Click()
  HIDComm.Disconnect
  Timer1.Enabled = False
  MSComm1.PortOpen = False
End Sub
Private Sub Command1_Click()
'Text1.Text = ""
End Sub
Private Sub Exit_Click()
Joystick.Hide
End Sub
Private Sub Form_Load()
MSComm1.Settings = "9600,n,8,1"
'MSComm1.PortOpen = True
End Sub
Private Sub HIDComm_ConnectFailure(ByVal Status As Long)
  MsgBox "HIDComm_ConnectFailure"
End Sub
Private Sub HIDComm_ConnectSuccess(ByVal Status As Long)
  MsgBox "HIDComm_ConnectSuccess"
End Sub
Private Sub HIDComm_Disconnected(ByVal Status As Long)
  MsgBox "HIDComm_Disconnected"
End Sub
Private Sub Timer1_Timer()
Dim Buffer() As Byte
  Buffer = HIDComm.ReadFrom(1)
 RevCodeX = Hex(Buffer(0))
```

```
    Label1.Caption = Hex(Buffer(0))
    Buffer = HIDComm.ReadFrom(2)
    RevCodeY = Hex(Buffer(1))
    Label3.Caption = Hex(Buffer(1))
  Buffer = HIDComm.ReadFrom(6)
    Label4.Caption = Hex(Buffer(5))
    Buffer = HIDComm.ReadFrom(7)
    Label5.Caption = Hex(Buffer(6))
     If Hex(Buffer(5)) = "F" Then
        LEDG1.LEDstate = 0
        LEDG2.LEDstate = 0
        LEDG3.LEDstate = 0
        LEDG4.LEDstate = 0
     End If
     If Hex(Buffer(5)) = "1F" Then
        LEDG1.LEDstate = 1
     End If
        If Hex(Buffer(5)) = "2F" Then
        LEDG2.LEDstate = 1
     End If
        If Hex(Buffer(5)) = "4F" Then
        LEDG3.LEDstate = 1
     End If
        If Hex(Buffer(5)) = "8F" Then
        LEDG4.LEDstate = 1
     End If
      If Hex(Buffer(6)) = "4" Then
         LEDR5.LEDstate = 1
      End If
     If Hex(Buffer(6)) = "1" Then
         LEDR6.LEDstate = 1
```

```
End If
  If Hex(Buffer(6)) = "8" Then
     LEDR7.LEDstate = 1
  End If
 If Hex(Buffer(6)) = "2" Then
     LEDR8.LEDstate = 1
  End If
  If Hex(Buffer(6)) = "0" Then
    LEDR5.LEDstate = 0
    LEDR6.LEDstate = 0
    LEDR7.LEDstate = 0
     LEDR8.LEDstate = 0
  End If
If Hex(Buffer(0)) = "FF" Then
   If Hex(Buffer(1)) = "7F" Then
    LEDR1.LEDstate = 1
    LEDR3.LEDstate = 0
    LEDR2.LEDstate = 0
    LEDR4.LEDstate = 0
    SentCode = "0"
   MSComm1.Output = SentCode
     End If
End If
 If Hex(Buffer(0)) = "0" Then
   If Hex(Buffer(1)) = "7F" Then
      LEDR2.LEDstate = 1
      LEDR1.LEDstate = 0
      LEDR3.LEDstate = 0
      LEDR4.LEDstate = 0
       SentCode = "4"
       MSComm1.Output = SentCode
```

```
      Label1.Caption = Hex(Buffer(0))
    End If
End If
 If Hex(Buffer(0)) = "FF" Then
   If Hex(Buffer(1)) = "0" Then
      LEDR2.LEDstate = 0
      LEDR1.LEDstate = 1
      LEDR3.LEDstate = 1
      LEDR4.LEDstate = 0
   End If
End If
 If Hex(Buffer(0)) = "7F" Then
  If Hex(Buffer(1)) = "0" Then
      LEDR2.LEDstate = 0
      LEDR1.LEDstate = 0
      LEDR3.LEDstate = 1
      LEDR4.LEDstate = 0
       SentCode = "E"
       Label1.Caption = Hex(Buffer(0))
        MSComm1.Output = SentCode
  End If
End If
    If Hex(Buffer(0)) = "7F" Then
  If Hex(Buffer(1)) = "FF" Then
      LEDR2.LEDstate = 0
      LEDR1.LEDstate = 0
      LEDR3.LEDstate = 0
      LEDR4.LEDstate = 1
       SentCode = "I"
       Label1.Caption = Hex(Buffer(0))
        MSComm1.Output = SentCode
```

```
  End If
End If
    If Hex(Buffer(0)) = "FF" Then
   If Hex(Buffer(1)) = "FF" Then
      LEDR2.LEDstate = 0
      LEDR1.LEDstate = 1
      LEDR3.LEDstate = 0
      LEDR4.LEDstate = 1
   End If
End If
    If Hex(Buffer(0)) = "0" Then
   If Hex(Buffer(1)) = "FF" Then
      LEDR2.LEDstate = 1
      LEDR1.LEDstate = 0
      LEDR3.LEDstate = 0
      LEDR4.LEDstate = 1
   End If
End If
    If Hex(Buffer(0)) = "0" Then
   If Hex(Buffer(1)) = "0" Then
      LEDR2.LEDstate = 1
      LEDR1.LEDstate = 0
      LEDR3.LEDstate = 1
      LEDR4.LEDstate = 0
   End If
End If
If Hex(Buffer(0)) = "7F" Then
 If Hex(Buffer(1)) = "7F" Then
      LEDR1.LEDstate = 0
      LEDR3.LEDstate = 0
      LEDR2.LEDstate = 0
```

```
        LEDR4.LEDstate = 0
        SentCode = "A"
        Label1.Caption = Hex(Buffer(0))
        MSComm1.Output = SentCode
        'Text1.Text = Text1.Text + SentCode
    End If
  End If
End Sub
```

ภาคผนวก ข

ซีซอร์สโค้ด (C Source Code)

```
/*********************************************************/
//Program PWM   Recive Data by Serial Port
/*********************************************************/
#include<reg51.h>
#include<stdio.h>
/*************** Function delay ON-OFF ******************/
void delay_on(int count);
void delay_off(int count);
/************************ Main Program **********************/
void main(void)
{
        char command = 0;
        SCON = 0x50;
        TMOD = 0x21;
        TH1  = 0xFD;
        TL1  = 0xFD;
        TI   = 1;
        TR1  = 1;
        while(1)
        {
        // RECIVE DATA BY SERIAL PORT
                command = SBUF;
                RI = 0;
                switch(command)
                {
        // CALL FUNCTION TURE LIFF
                        case '1' :
                                P1 = 0xAA;
                                delay_on(10);
```

```
                break;
        case '2' :
                P1 = 0xAA;
                delay_on(20);
                break;
        case '3' :
                P1 = 0xAA;
                delay_on(30);
                break;
        case '4' :
          P1 = 0xAA;
           delay_on(40);
                break;
        case '5' :
                P1 = 0xAA;
                delay_on(50);
                break;
//CALL FUNCTION TURE RIGHT
        case '6' :
                P1 = 0x55;
                delay_on(10);
                break;
        case '7' :
                P1 = 0x55;
                delay_on(20);
                break;
        case '8' :
                P1 = 0x55;
                delay_on(30);
                break;
        case '9' :
```

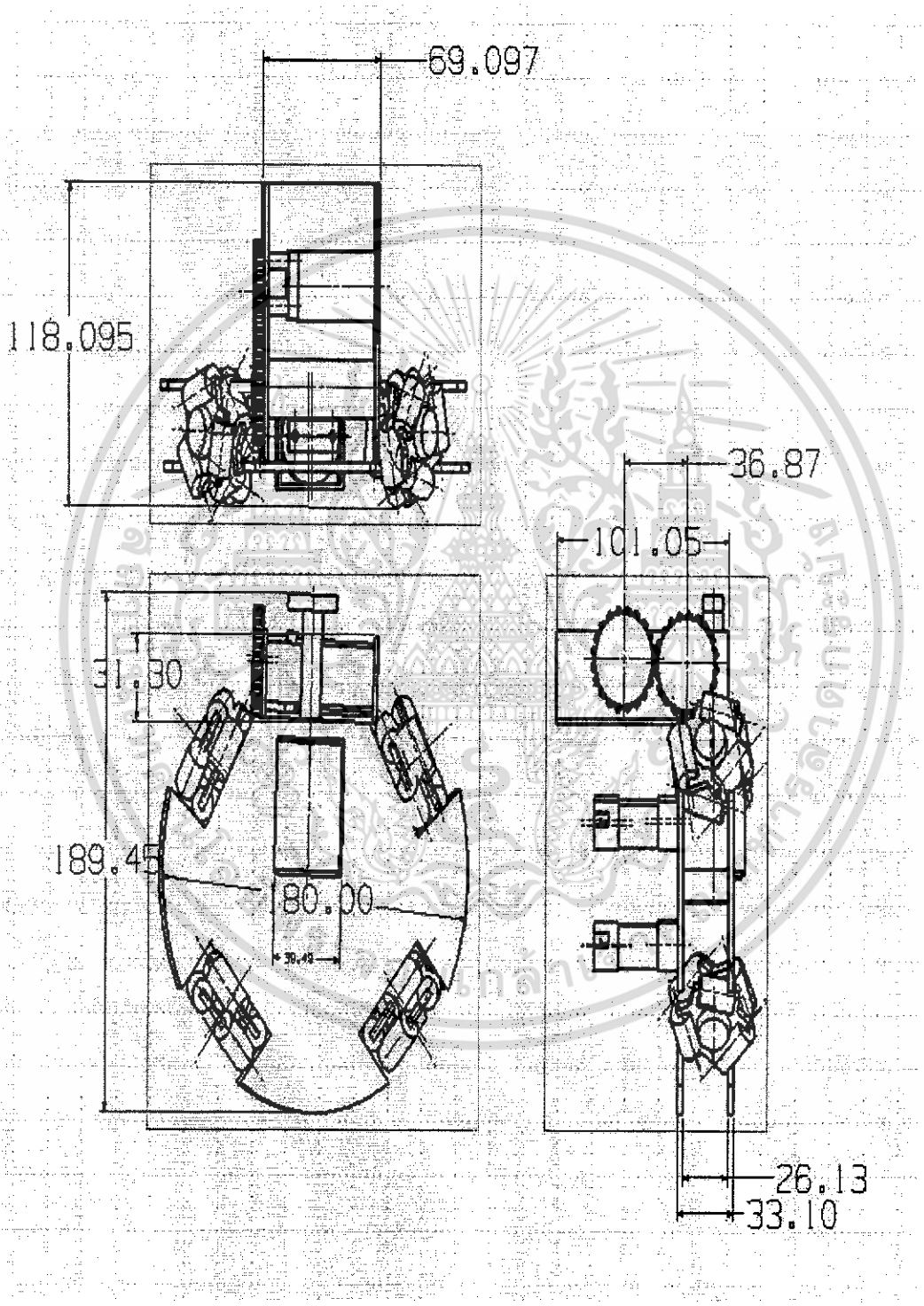
```
                P1 = 0x55;
                delay_on(40);
                break;
        case '0' :
                P1 = 0x55;
                delay_on(50);
                break;
//CALL FUNCTON GO ON
        case 'A' :
                P1 = 0xA5;
                delay_on(10);
                break;
        case 'B' :
                P1 = 0xA5;
                delay_on(20);
                break;
        case 'C' :
                P1 = 0xA5;
                delay_on(30);
                break;
        case 'D' :
                P1 = 0xA5;
                delay_on(40);
                break;
        case 'E' :
                P1 = 0xA5;
                delay_on(50);
                break;
//CALL FUNCTION BACK ON
 case 'F' ;
                P1 = 0x5A;
```

```
                    delay_on(10);
                    break;
            case 'G'  ;
                P1 = 0x5A;
                delay_on(20);
                break;
            case 'H' ;
                    P1 = 0x5A;
                    delay_on(30);
                    break;
            case 'I'  ;
                P1 = 0x5A;
                delay_on(40);
                break;
            case 'J' ;
                P1 = 0x5A;
                delay_on(50);
                break;
            }
        }
}
/*************** Delay TURELIFT ON  ****************************/
void delay_on(int count)
{
        int i,j;
        for(i=0;i<count;i++);
        delay_off(50-count);
}
/*************** Delay TURELiff OFF ****************************/
void delay_off(int count)
{
```

```
    int i,j;
    P1 = 0x00;
    for(j=0;j<count;j++);
}
```

ภาคผนวก ค

แบบชิ้นส่วนของหุ่นยนต์เตะฟุตบอล



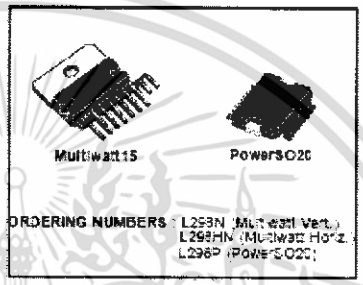
ภาคผนวก ง

Data Sheet L298N

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)

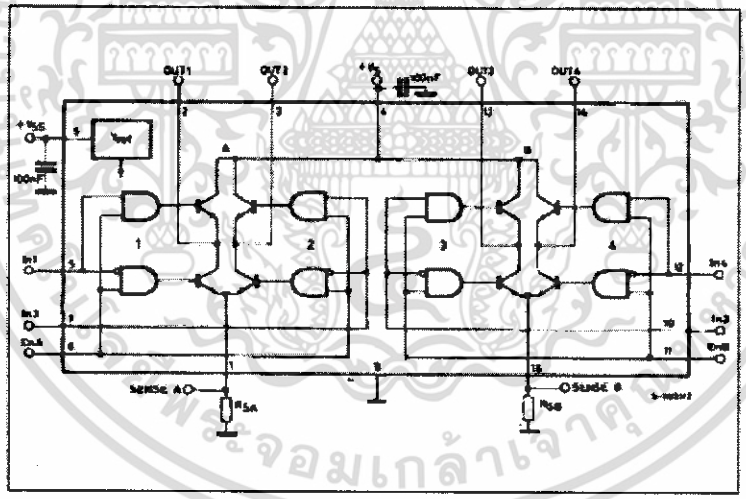


ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

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.

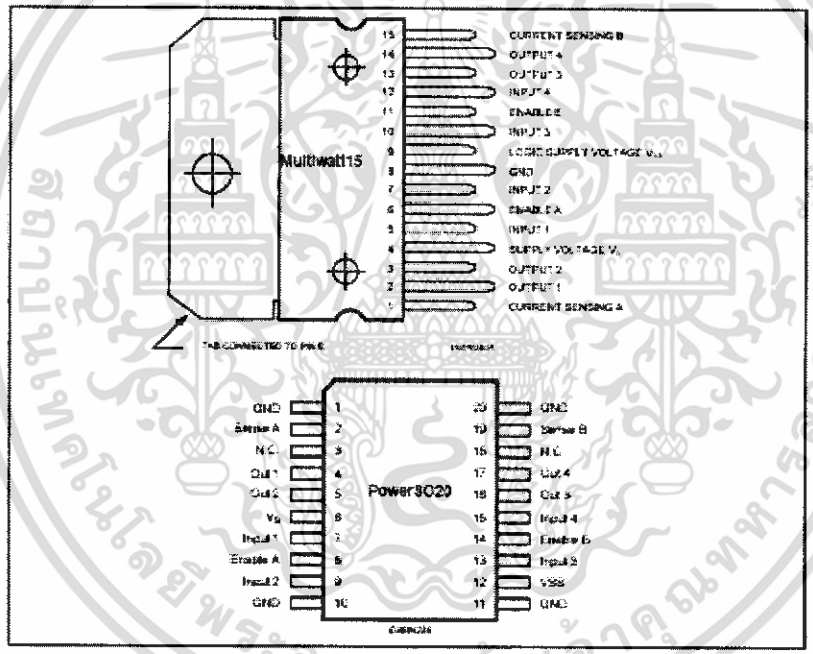
BLOCK DIAGRAM



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_{op}	Junction Operating Temperature	−25 to 130	°C
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, Tj = 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		V_{SS}	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)	V_i = 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
$V_{CE(sat)H}$	Source Saturation Voltage	I_L = 1A I_L = 2A		0.95	1.35 2	1.7 2.7	V V
$V_{CE(sat)L}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)		0.85	1.2 1.7	1.6 2.3	V V
$V_{CE(sat)}$	Total Drop	I_L = 1A (5) I_L = 2A (5)		1.80		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

1) Sensing voltage can be −1 V for t ≤ 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.

Figure 1 : Typical Saturation Voltage vs. Output Current.

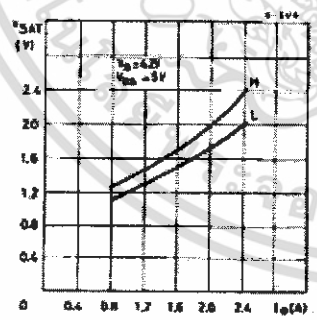
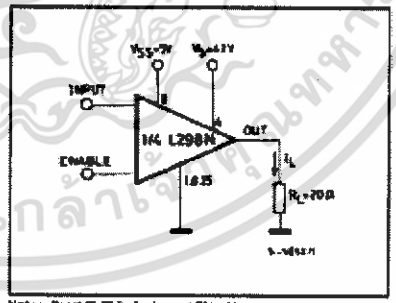


Figure 2 : Switching Times Test Circuits.



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

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

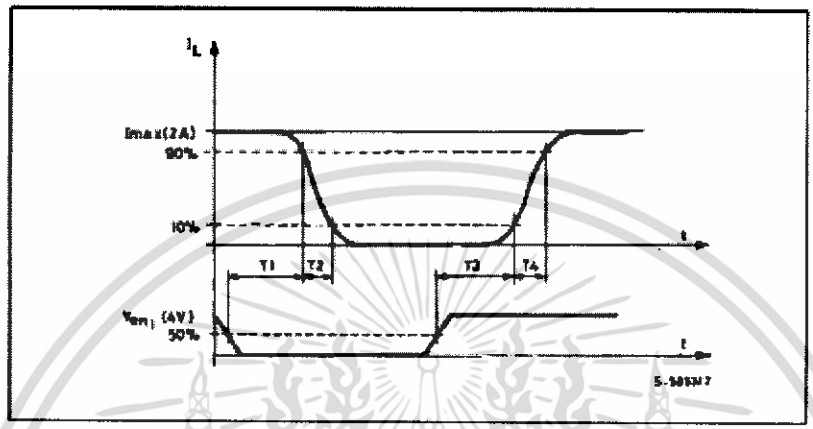
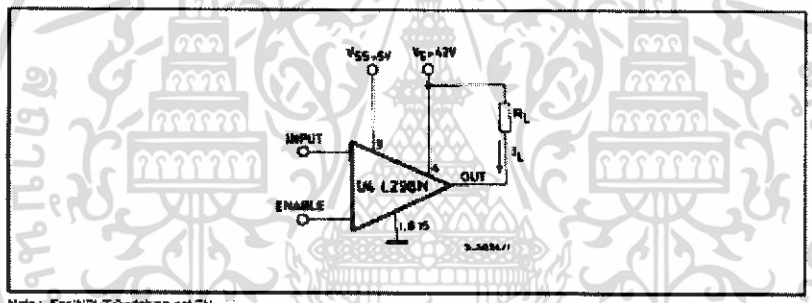
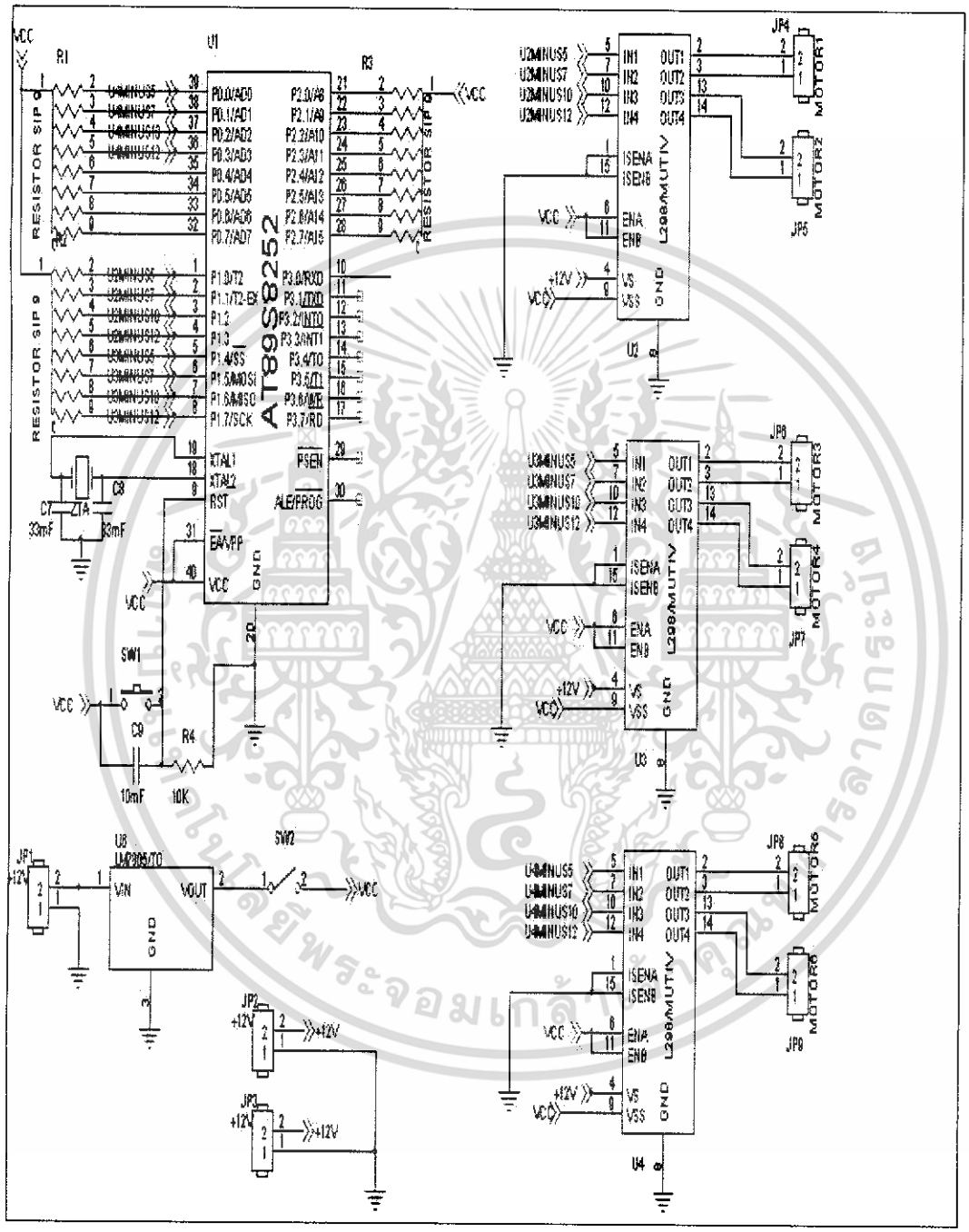


Figure 4 : Switching Times Test Circuits.



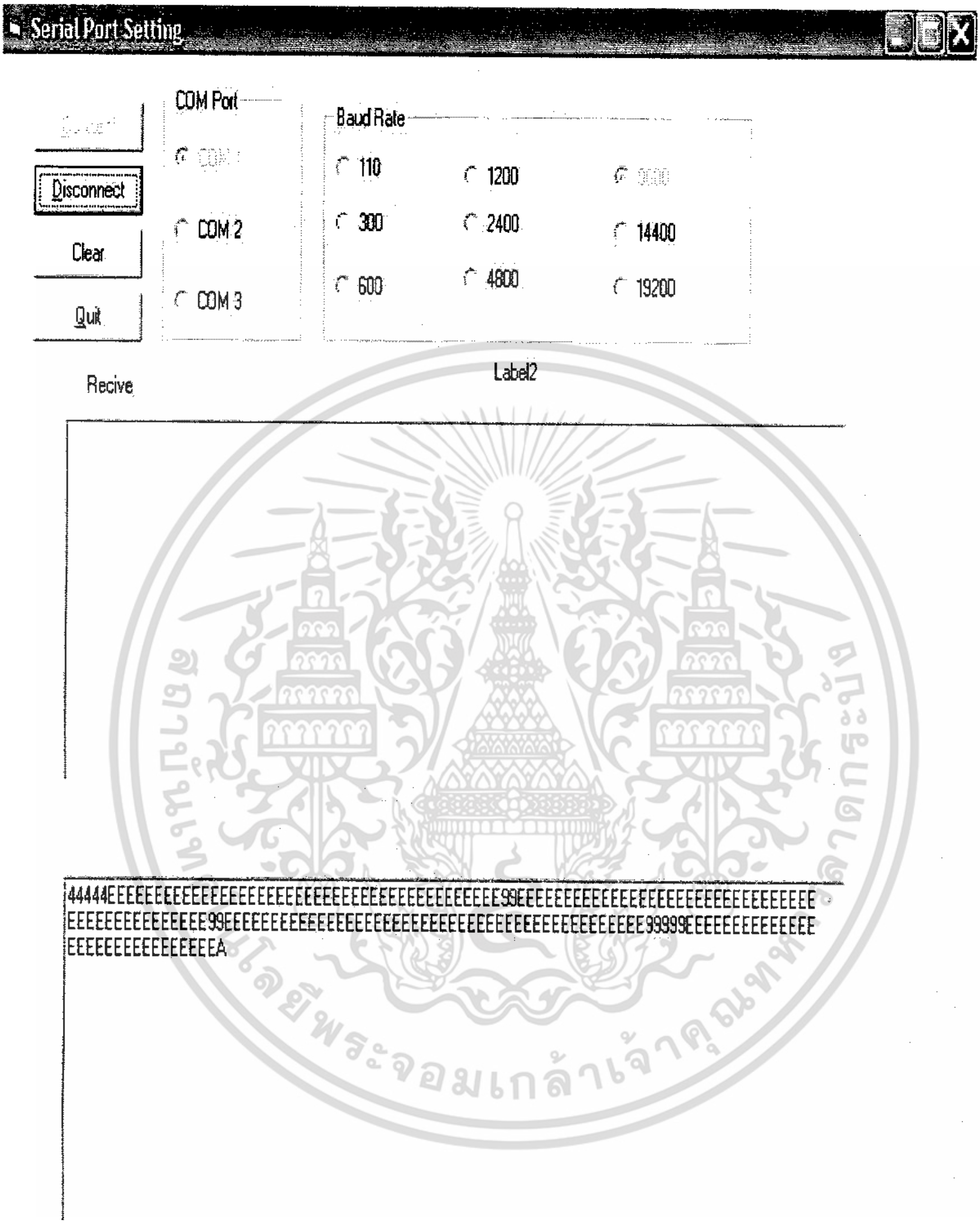
Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

ภาคผนวก จ
แสดงวงจรไฟฟ้า

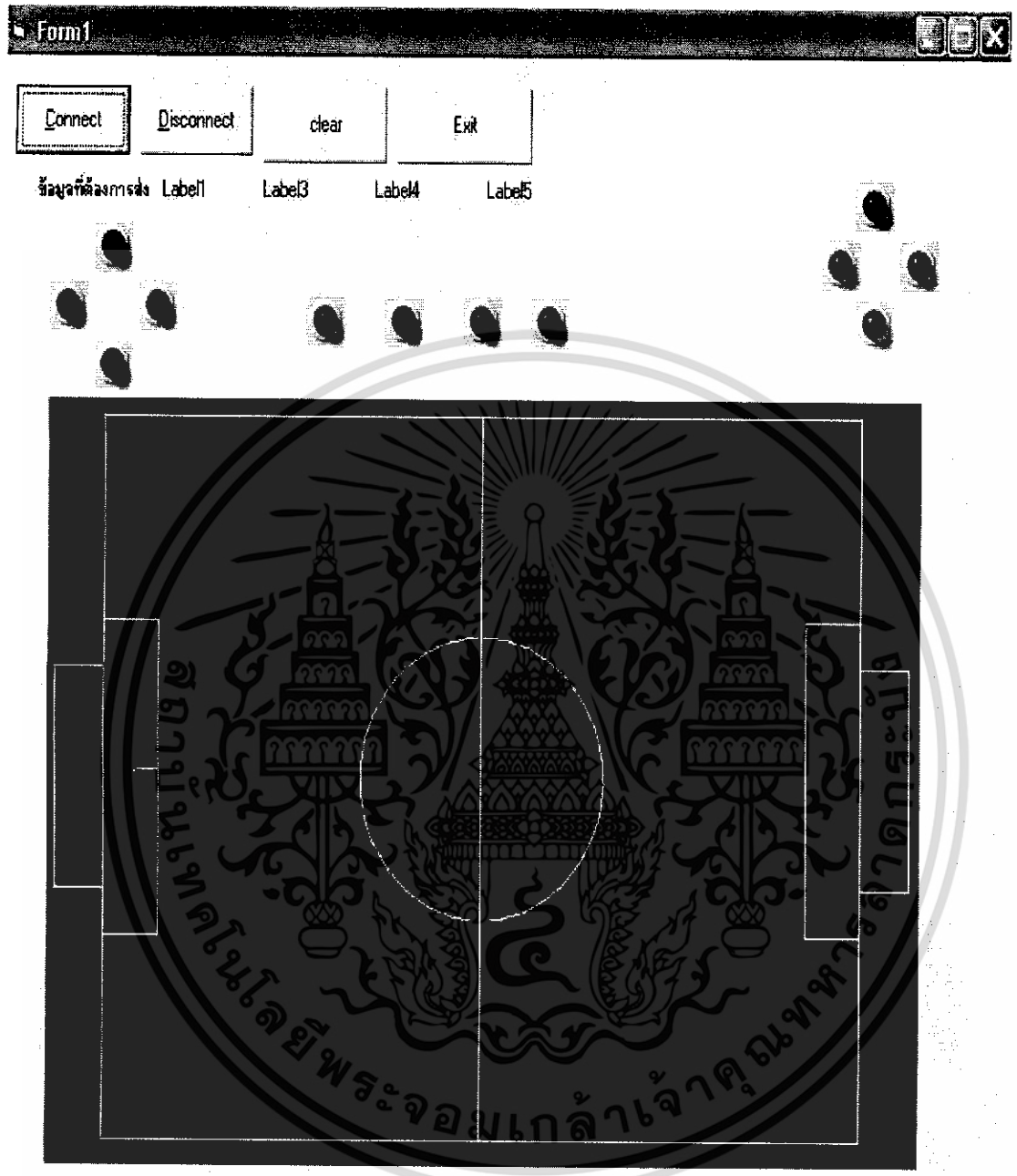


ภาคผนวก ฉ
แสดงหน้าต่างของโปรแกรมที่ใช้งาน

หน้าต่างโปรแกรมหลัก



หน้าต่างของโปรแกรมส่งข้อมูลผ่านพอร์ตอนุกรม



หน้าต่างของการติดต่อยูเอสบี (USB)