การควบคุมหุ่นยนต์เพื่อการขนส่ง

MOBILE ROBOT'S CONTROLLER FOR TRANSPORT

นายชัยวัฒน์ แก้วสาย

นายสันติ ยะยอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์ ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

MOBILE ROBOT'S CONTROLLER FOR TRANSPORT

Mr.CHAIWAT KEAWSAI

Mr.SANTI YAYONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF TER TECHNOLOGY ELECTRONICS

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2000

หัวข้อปริญญานิพนธ์	การควบคุมหุ่นยนต์เพื่อการขนส่ง
	MOBILE ROBOT 'S CONTROLLER FOR TRANSPORT
นักศึกษา	นายชัยวัฒน์ แก้วสาย เลขประจำตัว 41013433
	นายสันติ ยะยอง เลขประจำตัว 41013472
อาจารย์ที่ปรึกษา	ผศ.ดร.อรรถสิทธิ์ หล่าสกุล
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2543

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังอนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

..... อ.ที่ปรึกษา
(ผศ.ดร.อรรถสิทธิ์ หล่าสกุล)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปริญญานิพนธ์	การควบคุมหุ่นยนต์เพื่อการขนส่ง MOBILE ROBOT'S CONTROLLER FOR TRANSPORT
นักศึกษา	นายชัยวัฒน์ แก้วสาย เลขประจำตัว 41013443 นายสันติ ยะยอง เลขประจำตัว 41013472
อาจารย์ที่ปรึกษา	ผศ.ดร.อรรถสิทธิ์ หล่าสกุล
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2543

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการสร้างและจำลองหุ่นยนต์สำหรับการขนส่งในโรงานอุตสาหกรรม ซึ่งการออกแบบเน้นที่ความง่ายต่อการควบคุมและราคาที่ไม่แพงสามารถจัดสร้างโดยไม่ยาก เนื่องจากใช้โปรแกรมควบคุมที่ออกแบบมาให้ง่ายต่อความเข้าใจและกำหนดงานในลักษณะต่างๆ สามารถกำหนดจุดรับส่งสิ่งของได้อย่างไม่จำกัด มีขนาดเล็กที่เหมาะสมต่อการเรียนรู้เบื้องต้น นอกจากนี้ระบบยังสามารถขยายออกไปสู่การใช้งานจริงได้ เนื่องจากระบบมีการปรับสเกลที่เหมาะสม ทำให้สามารถนำไปประยุกต์ใช้งานอย่างจริงจังได้

Thesis Title MOBILE ROBOT'S CONTROLLER FOR TRANSPORT

Student Mr.CHAIWAT KEAWSAI ID 41013443
Mr.SANTI YAYONG ID 41013472

Advisor Asst.Prof.Dr.ATTASIT LASAKUL

Academic Year 2000

ABSTRACT

In this project the construction of Mobile robot for transportation is presented. Due to the wide range of mobile robotics application has become the most important task, especially in the area of factory operation this project, the small robot can be controlled by GUI software is constructed. System consists of many functions that are very useful for transportation such as waiting time at each station. Furthermore, the system can be adjusted scaling freely

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี โดยได้รับคำแนะนำและความช่วยเหลือต่างๆ เป็นอย่างดีจากอาจารย์ที่ปรึกษา ผศ.ดร.อรรถสิทธิ์ หล่าสกุล และอาจารย์อีกหลายท่านในภาคเทคนิคอุตสาหกรรม ที่ให้คำแนะนำในด้านต่างๆ

ขอขอบคุณ อาจารย์ทุกท่านที่ช่วยประสาทวิชาให้ความรู้ต่าง ๆ จนสามารถทำโครงงานนี้ได้ และประโยชน์อันที่พึงได้รับจากปริญญานิพนธ์ฉบับนี้ขอมอบให้แด่ผู้มีพระคุณทุกท่าน

นายชัยวัฒน์ แก้วสาย
นายสันติ ยะยอง

สารบัญ

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

ปัจจุบันนี้หุ่นยนต์ได้มีการออกแบบสร้างและพัฒนาขึ้นมาโดยวัตถุประสงค์ของการสร้างและพัฒนาหุ่นยนต์ คือสร้างความสะดวกสบายให้กับมนุษย์ โดยหุ่นยนต์ที่สร้างขึ้นนั้นจะทำงานแทนหรือทำการรับใช้มนุษย์ โดยเฉพาะอย่างยิ่งในทางระบบอุตสาหกรรมได้นำหุ่นยนต์มาทำงานแทนแรงงานมนุษย์มากขึ้นเรื่อยๆ เนื่องจากหุ่นยนต์มีการทำงานได้แม่นยำและแน่นอนมากกว่ามนุษย์ และหุ่นยนต์สามารถทำงานได้ตลอดเวลา ทำให้ต้นทุนการผลิตของระบบอุตสาหกรรมลดลง โดยการทำงานของหุ่นยนต์นั้นจะทำตามโปรแกรมที่กำหนด อุปกรณ์ที่นิยมใช้ควบคุมการทำงานหุ่นยนต์ได้แก่ ไมโครคอนโทรลเลอร์และคอมพิวเตอร์

โดยทั่วไปแล้วหุ่นยนต์แบ่งออกเป็น

1.Fixed Robotic เป็นหุ่นยนต์อยู่กับที่

2.Mobile Robot เป็นหุ่นยนต์สามารถเคลื่อนที่ได้โดยอิสระโดยมีฐานเป็นล้อ

ในโครงานนี้จะกล่าวถึง Mobile Robot ที่บางทีเรียกว่า Autonomous Guide Vehicle System (AGVS) ซึ่งเป็นระบบจัดการหรือขนส่งพัสดุ ที่ใช้ยานพาหนะขับเคลื่อนด้วยตัวเองไปตามพื้นที่ที่กำหนดโดยรับพลังงานจากแบตเตอร์รี่ภายในตัวเอง

ประเภทของ AGVSแบ่งออกได้เป็น 3 ประเภท คือ

1.Driveless Trains

เปรียบเสมือนหัวรถจักรที่สามารถจูงรถพ่วงได้อีกหลายคัน

2.AGVS PALLET

เปรียบเสมือนตู้โดยสารของรถไฟ

3. AGVS Unit Load Carries

เป็นเครื่องจักรที่ทำหน้าที่ที่รับสำภาระจากพนักงานแล้วส่งขึ้นไป PalleTruck โดยอัตโนมัติ

การเคลื่อนที่ของ AGVS

1.การเคลื่อนที่ตามสายนำสัญญาณ(Signal Carrying Wire) โดย AGVS เคลื่อนที่ตามสายนำสัญญาณที่พื้น

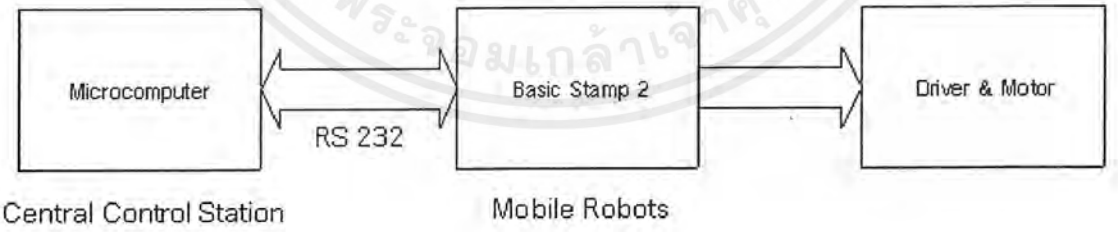
2. การเคลื่อนที่ตามเส้นแถบสีที่พื้น(Tracking line on the floor) โดย AGVS เคลื่อนที่ตามแถบสีที่พื้น

การเคลื่อนที่ทั้งสองวิธีนั้น AGVS จำเป็นต้องมีความสามารถในการตรวจจับสภาพแวดล้อมหลายๆจุด เพื่อถ้าเจอปัญหาและอุปสรรคสามารถหยุดการเคลื่อนที่ได้ทันที การเคลื่อนที่โดยใช้สายนำสัญญาณ การติดตั้งสายทำได้ลำบากและราคาของสายแพง การเปลี่ยนแปลงโปรแกรมทำได้ยาก ส่วนการเคลื่อนที่ตามเส้นต้องมีการบำรุงรักษาเส้นแถบสีให้สะอาดเสมอ การเปลี่ยนแปลงโปรแกรมทำได้ยากเช่นกัน จากข้อเสียของการเคลื่อนที่ของ AGVS ดังกล่าวมานำไปสู่การคิดและพัฒนาโครงงานการควบคุมหุ่นยนต์เพื่อการขนส่งฉบับนี้

ในโครงงานนี้จะกล่าวถึงการควบคุมเส้นทางการเคลื่อนที่ของ Mobile Robot(AGVS) ให้ไปตามระยะทางและทิศทางที่กำหนด โดยคำสั่งควบคุมมาจากสถานีควบคุม(Control Station) จะเห็นว่าวิธีนี้ เป็นวิธีที่จะประหยัดและคุ้มค่าเมื่อมีการเปลี่ยนแปลงเส้นทางการเดินหุ่นยนต์บ่อยๆ และไม่ต้องสิ้นเปลืองค่าสายนำสัญญาณหรือเส้นแถบสีบนพื้น แต่การออกแบบทำยากกว่า เพราะต้องใช้ระบบสื่อสารที่ดีติดต่อระหว่างหุ่นยนต์และสถานีควบคุม และหุ่นยนต์ต้องมีข้อผิดพลาดในเคลื่อนที่น้อยมาก

ลักษณะและขอบเขต

ลักษณะการโปรแกรมการเคลื่อนที่ของ Mobile Robot ในโครงงาน จะใช้ Mouse วาดเส้นทางการเดินของ Mobile Robot ผ่านหน้าจอคอมพิวเตอร์ทำสามารถเปลี่ยนแปลงโปรแกรมการทำงานของ Mobile Robot ได้สะดวกและรวดเร็ว รวมทั้งสามารถบันทึกเส้นทางการเคลื่อนไว้เพื่อเปิดมาใช้งานในครั้งต่อไป ตรวจสอบข้อผิดพลาดการทำงานของหุ่นยนต์สามารถดูได้จากการแสดงตำแหน่งและการทำงานจากจอมอนิเตอร์ของคอมพิวเตอร์ เราสามารถเขียนบล็อกไดอะแกรมการทำงานของโครงงานได้ดังรูป



รูปที่ 1.1 แสดงบล็อกไดอะแกรมควบคุมการทำงานของ Mobile Robot

โดยบล็อกไดอะแกรมประกอบด้วย

1. สถานีควบคุม ใช้ไมโครคอมพิวเตอร์

2. ตัวหุ่นยนต์ เป็น Mobile Robot ขนาดเล็ก ใช้ไมโครคอนโทรลเลอร์เบสิกสแตมป์ 2 เป็นตัวควบคุมการทำงาน ระบบการขับเคลื่อนใช้สเต็ปเปอร์มอเตอร์ การติดต่อระหว่างหุ่นยนต์และสถานีควบคุมใช้สายนำสัญญาณสื่อสารข้อมูลตามมาตรฐาน RS- 232

วัตถุประสงค์และจุดมุ่งหมาย

1.เพื่อศึกษาการทำงานของ Mobile Robot

2.เพื่อออกแบบโปรแกรมการควบคุมการเคลื่อนที่ของ Mobile Robot

3.เพื่อออกแบบโปรแกรมแสดงผล ตำแหน่งและการทำงานของหุ่นยนต์

ส่วนประกอบของโครงงาน

โครงงานนี้ประกอบด้วย 2 ส่วนคือ

1. ซอร์ฟแวร์

ส่วนที่เป็นซอร์ฟแวร์ คือส่วนที่เป็นโปรแกรมใช้ในการควบคุมเส้นทางการเคลื่อนที่และแสดงตำแหน่งของ Mobile Robot

2.ส่วนฮาร์ดแวร์

ส่วนของฮาร์ดแวร์ คือ Mobile Robot เป็นแบบจำลองการทำงาน

ประโยชน์ที่คาดว่าจะได้รับ

1.สามารถนำไปใช้ในการขนส่งในโรงงานอุตสาหกรรม สนามบิน ท่าเรือ สถานีรถไฟหรือที่อื่นๆได้จริง

2.สามารถนำ Mobile Robot ไปประยุกต์ใช้งานเป็นหุ่นยนต์ทำความสะอาดตามสถานที่ต่างๆ เช่น โรงยิม ได้

3. สามารถนำ Mobile Robot ไปประยุกต์ใช้งานเป็นหุ่นยนต์ ไกด์นำชมสถานที่ต่างๆเช่น พิพิธภัณฑ์ งานแสดงสินค้า

บทที่ 2
ทฤษฎี

2.1 ไมโครคอนโทรเลอร์เบสิกสแตมป์ 2(BASIC STAMP 2)

เบสิกสแตมป์คือแผงวงจรไมโครคอนโทรเลอร์สำเร็จรูปที่บรรจุตัวแปลภาษาเบสิกหรือเบสิกอินเตอร์พรีตเตอร์(BASIC INTERPRETER)รวมไว้ด้วยกัน สามารถใช้การเขียนโปรแกรมด้วยภาษาเบสิกควบคุมการทำงานได้ ในการพัฒนาระบบด้วยเบสิกแสตมป์ ไม่จำเป็นต้องใช้เครื่องโปรแกรม(Programmer)หรือเครื่องเลียนแบบ(Emulator)ไมโครคอนโทรเลอร์และหน่วยความจำแต่อย่างใด เบสิกแสตมป์ผลิตโดย Parallax Inc และเหตุผลที่มีคำว่าแสตมป์ต่อท้ายก็เพื่อให้ต้องการทราบว่าบอร์ดไมโครคอนโทรเลอร์ตัวนี้มีขนาดเล็กเท่ากับแสตมป์ การเขียนโปรแกรมด้วยชุดคำสั่งภาษาเบสิกเรียกว่า พีเบสิก(PBASIC)มีด้วยกัน 36 คำสั่งแต่ละคำสั่งสามารถนำไปใช้ได้ทันทีไม่ต้องเขียนโปรแกรมย่อยมากมายความเร็วในการทำคำสั่งเบสิกแสตมป์2ซึ่งสูงถึง 4,000 คำสั่งภาษาเบสิกต่อวินาทีทำให้เบสิกแสตมป์ทำงานได้รวดเร็ว

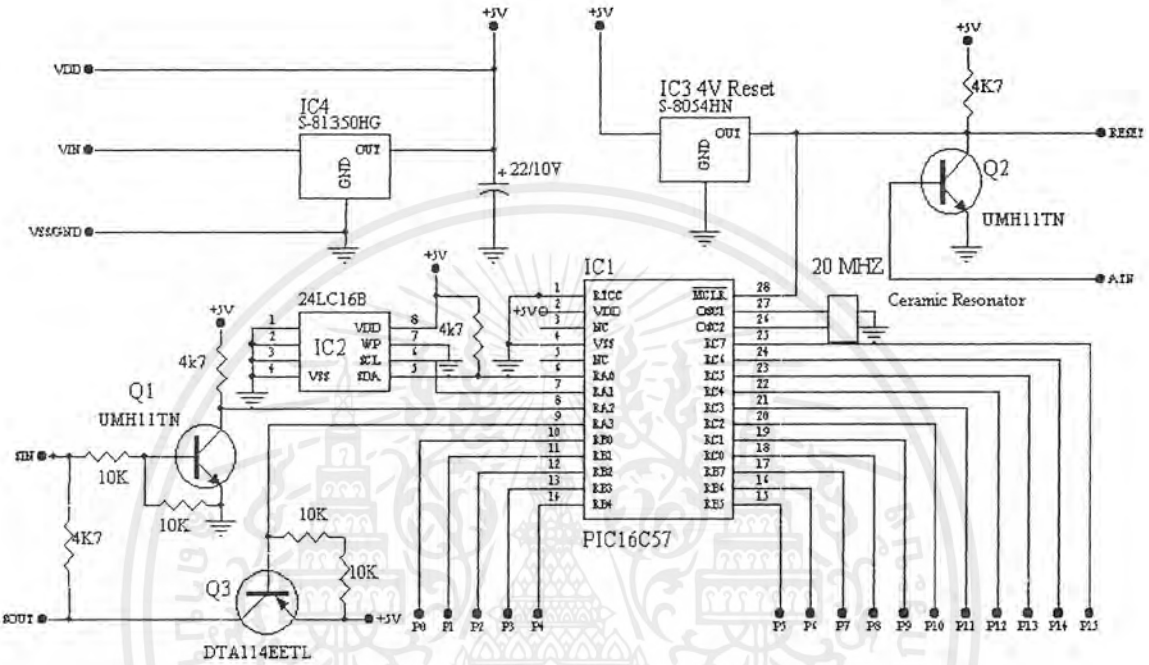
2.2 วงจรเบสิกแสตมป์ 2

วงจรของเบสิกแสตมป์ 2 แบ่งออกเป็น 4 ส่วนเริ่มจากตัวแปลภาษา พีเบสิก สำหรับ เบสิกแสตมป์ ใช้ไมโครคอนโทรเลอร์ PIC16C57 ทางด้านหน่วยความจำโปรแกรม IC ใช้หน่วยความจำอีอีพรอมอนุกรมเบอร์ 24LC16 มีความจุ 2 กิโลไบต์สามารถบรรจุคำสั่งของพีเบสิก ที่ใช้ในการรันโปรแกรมของ วงจรเบสิกแสตมป์ 2 ได้ 500 คำสั่ง

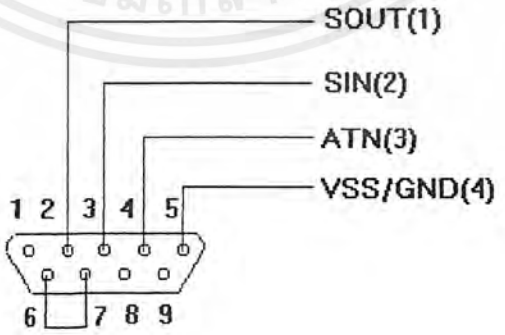
ส่วนการโปรแกรมข้อมูล ยังคงใช้การโปรแกรมในลักษณะอนุกรม แต่จะติดต่อกับพอร์ตอนุกรมคอมพิวเตอร์แทน โดยต่อสายจากขา TxD,RxD,DTR และกราวด์ ของพอร์ตอนุกรมเข้าสู่ขา Sin,Sout,ATN และกราวด์ของBS2นอกจากนั้นที่ขา 6 และ 7 ของพอร์ตอนุกรมต้องต่อถึงกันด้วยเพื่อให้สามารถโปรแกรมข้อมูลบนเบสิกแสตมป์ 2 ได้ตลอดเวลาอย่างอัตโนมัติดังแสดงในรูป

สัญญาณจากพอร์ตอนุกรมของคอมพิวเตอร์โดยทั่วไปมีระดับแรงดันไฟสูงกว่าระดับทีทีแอล กล่าวคือมีแรงดันประมาณ $\pm$12 V สำหรับคอมพิวเตอร์ตั้งโต๊ะ และ $\pm$6 V ถึง $\pm$ 8 V สำหรับคอมพิวเตอร์โน้ตบุ๊ก ดังนั้นจึงต้องมีวงจรแปลงระดับสัญญาณจากคอมพิวเตอร์ให้เป็นระดับทีทีแอล(0-5 V) เพื่อให้สามารถติดต่อกับ PIC16C57 ได้

ขา SIN เป็นขาสำหรับรับข้อมูลจากคอมพิวเตอร์เมื่อขาคอมพิวเตอร์ส่งข้อมูล “1” มาจะทำให้ SIN เป็นลบ ทรานซิสเตอร์ Q1 จะ ไม่ทำงาน ทำให้ที่ขา RA 2 ของ PIC16C57(หรือ U1)มีแรงดัน +5 Vเกิดเป็นลอจิก “1” ในทางตรงข้ามเมื่อคอมพิวเตอร์ส่งขอมูล “0” ขา SIN จะเป็นบวก ทรานซิสเตอร์ Q1 ทำงานส่งผลให้ขา RA2 เสมือนต่อลงกราวด์ มีลอจิกเป็น “0”



รูปที่ 2.1 แสดงวงจรเบสิกแสตมป์ 2



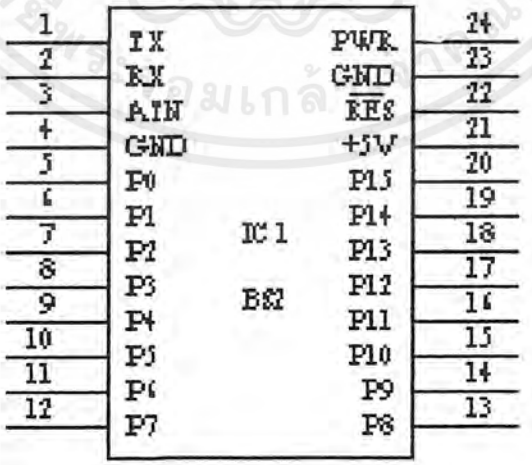
รูปที่ 2.2 การเชื่อมต่อเบสิกแสตมป์ 2 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์

ขา SOUT ใช้สำหรับส่งข้อมูลจาก เบสิกแสตมป์ 2 ไปยังคอมพิวเตอร์ เมื่อ เบสิกแสตมป์ 2 ส่งข้อมูล "1" ทรานซิสเตอร์ Q3 ไม่ทำงาน ที่ขา SOUT จะได้รับแรงดันลบจากขา SIN ผ่านตัวต้านทาน 4.7K ทำให้ที่ขา SOUT มีระดับแรงดันเป็นลบ ซึ่งคอมพิวเตอร์จะอ่านเป็นลอจิก "1"

ทั้งนี้เนื่องจากในมาตราฐาน RS-232 แล้วลอจิก "1" คือระดับแรงดันตั้งแต่ –3 ถึง –12 V ในขณะที่ลอจิก "0" คือระดับแรงดัน +3 ถึง +12 V เมื่อ เบสิกแสตมป์ 2 ส่งข้อมูล "0" ทรานซิสเตอร์ Q3 ทำงาน ที่ขา SOUT จึงเกิดแรงดัน +5V ทำให้คอมพิวเตอร์อ่านข้อมูลได้เป็น "0"

เมื่อเป็นเช่นนี้ในการติดต่อระหว่าง เบสิกแสตมป์ 2 กับคอมพิวเตอร์จะต้องสลับกันรับและส่งข้อมูลกล่าวคือเมื่อคอมพิวเตอร์ส่งข้อมูลมา เบสิกแสตมป์ 2 ต้องทำหน้าที่รับข้อมูลอย่างเดียว ไม่สามารถที่จะส่งข้อมูลกลับไปยังคอมพิวเตอร์ในเวลาเดียวกันได้

ขา ATN ซึ่งต่อเข้ากับขา DTR (Data Terminal Ready)ใช้ในการแฮนด์เชก หรือตรวจสอบความพร้อมในการรับส่งข้อมูลของพอร์ตอนุกรม จะมีลักษณะการทำงานคล้ายกับขา SIN ถ้าหากคอมพิวเตอร์ส่งข้อมูล "0" หรือทำให้ขา DTR มีแรงดันเป็น +12V ทรานซิสเตอร์ Q2 ทำงาน ทำให้คอลเล็กเตอร์ของ Q2 เสมือนต่อลงกราวด์เท่ากับว่าขา MCLR ของ PIC16C57 ถูกต่อลงกราวด์ด้วย อันเป็นการสร้างสัญญาณรีเซตให้แก่ PIC16C57 ดังนั้นในขณะที่ทำการโปรแกรมข้อมูลลงบน เบสิกแสตมป์ 2 ซอฟต์แวร์ที่ใช้ในการทำโปรแกรม ซึ่งเรียกว่า BASIC Stamp Editor ที่จะส่งพัลส์มายังขา ANT เพื่อรีเซต IC แล้วตามด้วยการส่งข้อมูลมายังขา SIN เพื่อแจ้งให้ทราบว่า ต้องการเขียนโปรแกรมใหม่ลงบน เบสิกแสตมป์ 2 แต่ถ้าหาก BASIC Stamp Edittor ทำให้ขา ATN เป็น –12 V หรือเป็นลอจิก "1" ก็จะหมายความว่าขณะนี้ เบสิกแสตมป์ 2 อยู่ในโหมดรัน นั่นคืออยู่ในโหมดทำงานตามปกติ



รูปที่ 2.3 แสดงการจัดขาของเบสิกแสตมป์ 2

2.3 รายละเอียดชุดคำสั่งของเบสิกแสตมป์ 2

คำสั่งหลักในโปรแกรมพีเบสิกแสตมป์ 2 มีทั้งสิ้น 36 คำสั่ง สามารถแบ่งออกเป็นกลุ่มตามลักษณะการทำงานได้ 11 กลุ่ม ดังนี้

1. กลุ่มคำสั่งการกระโดด มีทั้งสิ้น 5 คำสั่งคือ
   - IF...THEN เปรียบเทียบเงื่อนไขก่อนกระโดด
   - BRANCH กระโดดไปยังตำแหน่งที่กำหนดตามค่าของตัวแปร
   - GOTO กระโดดไปยังแอดเดรสใดๆ
   - GOSUB กระโดดไปยังโปรแกรมย่อย
   - RETURN กระโดดไปยังโปรแกรมย่อย
2. กลุ่มคำสั่งเกี่ยวกับลูป มี 1 คำสั่งคือ
   - FOR...NEXT กำหนดจำนวนรอบที่ต้องการวนลูปหรือวนทำงานซ้ำ
3. กลุ่มคำสั่งเลือกข้อมูล มี 3 คำสั่งคือ
   - LOOKUP เปิดตารางข้อมูล
   - LOOKDOWN ค้นหาตัวเลขที่เหมือนกัน แล้วเก็บค่าไว้ในตัวแปร
   - RANDOM สุ่มตัวเลข
4. กลุ่มคำสั่งควบคุมการทำงานของขาพอร์ต มี 13 คำสั่งคือ
   - INPUT กำหนดให้ทำงานเป็นอินพุต
   - OUTPUT กำหนดให้ทำงานเป็นเอาต์พุต
   - REVERSE เปลี่ยนจากขาอินพุตเป็นเอาต์พุตหรือจากเอาต์พุตเป็นอินพุต
   - LOW ทำให้ขาเอาต์พุตเป็นลอจิก "0"
   - HIGH ให้ขาเอาต์พุตเป็นลอจิก "1"
   - TOGGLE ทำให้ขาเอาต์พุตกลับสถานะลอจิก
   - PULSIN วัดสัญญาณพัลส์อินพุต (ความละเอียด 2 ไมโครวินาที)
   - PULSOUT ส่งสัญญาณพัลส์ออก (ความละเอียด 2 ไมโครวินาที)
   - BUTTON แก้การเบาซ์ของสวิตช์
   - SHIFTIN เลื่อนข้อมูลเข้าจากแบบขนานเป็นแบบอนุกรม
   - SHIFTOUT ส่งข้อมูลออกจากแบบอนุกรม
   - COUNT นับจำนวนไซเกิลของสัญญาณอินพุต (มีค่า 0-125kHz)
   - XOUT กำเนิดรหัสควบคุมสำหรับอุปกรณ์ต่อพ่วงอนุกรม X-10
5. กลุ่มคำสั่งติดต่อพอร์ตอนุกรม มี 2 คำสั่งคือ
   - SERIN รับข้อมูลอนุกรมเข้า มีรูปแบบข้อมูลแบบ N81 หรือ E71

SEROUT ส่งข้อมูลอนุกรมในรูปแบบ N81 หรือ E71 ออกไปทางขา SOUT ของเบสิกแสตมป์

6. กลุ่มคำสั่งควบคุมพอร์ตอินพุตเอาต์พุตแบบอนาลอก มี 2 คำสั่งคือ

PWM สร้างสัญญาณ PWM ขนาด 0-5V ออกไปทางขาพอร์ต

RCTIME วัดค่าเวลาการประจุและคายประจุของวงจร RC สามารถนำไปใช้ในการวัดค่าค่าของตัวต้านทานปรับค่าได้

7. กลุ่มคำสั่งจัดการด้านเสียง มี 2 คำสั่งคือ

FREQOUT กำเนิดสัญญาณไซน์หนึ่งหรือสองความถี่ ตั้งแต่ 0-32.67 kHz

DTMFOUT กำเนิดสัญญาณ DTMF ของระบบโทรศัพท์

8. กลุ่มคำสั่งเข้าถึงหน่วยความจำอีอีพรอมเบสิกแสตมป์ มี 3 คำสั่งคือ

DATA เก็บข้อมูลลงในหน่วยความจำอีอีพรอมก่อนการทำการโหลดโปรแกรมพีเบสิก

READ อ่านข้อมูลระดับไบต์จากหน่อยความจำอีอีพรอมมาเก็บไว้ในตัวแปร

WRITE เขียนข้อมูลระดับไบต์ไปเก็บไว้ในหน่วยความจำอีอีพรอม

9. กลุ่มคำสั่งจัดการด้านเวลา มี 1 คำสั่งคือ

PAUSE หน่วงเวลา 0-65,536 มิลิวินาที

10. กลุ่มคำสั่งควบคุมพลังงาน มี 3 คำสั่งคือ

NAP หยุดทำงานในช่วงเวลาสั้นๆ การกินพลังงานจะลดลง

SLEEP กำหนดให้หยุดทำงานได้นาน 1-65,536 วินาที เมื่อกำหนดให้ทำงานในโหมดสลีปนี้เบสิกแสตมป์ 2 กินกระแสไฟฟ้าลดลงเหลือเพียง 50uA

END ทำงานในโหมดสลีปจนกว่าจะมีการจ่ายไฟหรือต่อเบสิกแสตมป์ 2 เข้ากับคอมพิวเตอร์ เมื่อกระทำคำสั่งนี้เบสิกแสตมป์ 2 จะกินกระแสไฟฟ้าลดลง

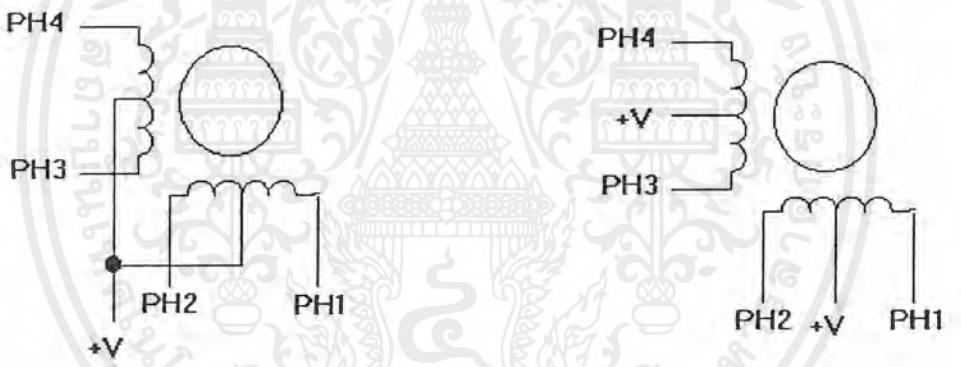
11. กลุ่มคำสั่งแก้ไขโปรแกรม มี 1 คำสั่ง

DEBUG แสดงค่าของตัวแปรผ่านทางคอมพิวเตอร์

2.4 สเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์(StepperMotor) เป็นมอเตอร์ที่มีการหมุนเป็นจังหวะหรือเรียกว่าเป็นสเต็ป กล่าวคือ เมื่อจ่ายสัญญาณให้แก่มอเตอร์อย่างถูกต้อง มอเตอร์จะหมุนไปเป็นจังหวะไม่เหมือนมอเตอร์อื่นทั่วไป ที่เมื่อทำการจ่ายไฟแล้วก็จะหมุนไปทันที ส่วนจะหมุนซ้ายหรือขวานั้นขึ้นอยู่กับการจ่ายไฟให้แก่ขดลวดภายในมอเตอร์ ความละเอียดของการหมุนของสเต็ปเปอร์จะขึ้นอยู่กับองค์ประกอบสำคัญ 2 ประการคือ โครงสร้างของสเต็ปเปอร์มอเตอร์และวงจรขับ โดยวงจรขับสามารถที่จะช่วยให้การหมุนของสเต็ปเปอร์มอเตอร์มีความละเอียดมากขึ้นได้

เหตุผลที่สเต็ปเปอร์มอเตอร์ได้รับความนิยมมากก็คือสามารถกำหนดตำแหน่งของการหมุนได้อย่างแม่นยำ โดยไม่ต้องใช้วงจรขับที่มีความซับซ้อนมากนักไม่ต้องสร้างวงจรป้อนกลับเพื่อตรวจสอบตำแหน่งต้นทางหรือปลายทางและเมื่อมอเตอร์ไม่หมุนก็จะไม่มีการใช้พลังงานจึงไม่เกิดความสูญเสียมากมายดังเช่นมอเตอร์ธรรมดา



รูปที่2.4 โครงสร้างพื้นฐานของสเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์

2.5 สเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์

สเต็ปเปอร์มอเตอร์ที่มีใช้มีด้วยกัน 2 แบบคือ แบบยูนิโพล่าร์และไบโพล่าร์ ในปัจจุบันสเต็ปเปอร์มอเตอร์แบบไบโพล่าร์แทบจะไม่มีการใช้งาน เนื่องจากวงจรที่ขับมีคววามยุ่งยากและใช้อุปกรณ์มากโครงสร้างของสเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์แสดงดังรูป สเต็ปเปอร์มอเตอร์แบบนี้จะมีการพันมอเตอร์สองขดบนแต่ละขั้วแม่เหล็กสเตเตอร์ แต่ละขดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวมีขดลวดทั้งสิ้น 4 เฟส คือเฟส 1,2,3 และ 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้มอเตอร์แบบนี้จึงมีทั้งแบบ 5และ 6 สาย โดยสายที่ 5 คือสายจ่ายไฟเลี้ยง ส่วนกรณี 6 สายจะต้องนำสายไฟเลี้ยงของขดลวดทั้ง 2 ของมอเตอร์มาต่อรวมกันแล้วจ่ายไฟจึงจะทำให้สเต็ปเปอร์มอเตอร์ทำงานได้

2.6 การขับสเต็ปเปอร์มอเตอร์

การขับสเต็ปเปอร์มอเตอร์หมุนจะต้องป้อนสัญญาณไฟฟ้าไปยังแต่ละเฟสของขดลวดอย่างเหมาะสมและมีรูปแบบที่ถูกต้อง สเต็ปเปอร์มอเตอร์จึงหมุนได้ โดยมีรูปแบบการขับอย่างง่าย 3 รูปแบบคือ แบบ1 เฟส,แบบ 2 เฟสหรือฟูลสเต็ป(full step) และแบบฮาล์ฟสเต็ป(half step)

แบบ 1 เฟสเป็นการขับที่ง่ายที่สุด โดยทำการป้อนสัญญาณกระตุ้นขดลวดครั้งละเฟสในช่วงเวลาหนึ่งไล่เรียงกันไป เช่น เริ่มจากเฟสที่ 1 ต่อด้วยเฟสที่ 2,3 และ 4 แล้ววนกลับมาเฟส 1 ใหม่ หรือจะให้เริ่มเฟสที่ 1 ไปยังเฟสที่ 4,3 และ 2 แล้ววนกลับมาเฟสที่ 1 อีกครั้ง ด้วยลำดับการป้อนสัญญาณกระตุ้นที่ต่างกัน ทำให้ทิศทางการหมุนของสเต็ปเปอร์มอเตอร์สวนทางกันด้วย การขับสเต็ปเปอร์มอเตอร์แบบนี้ จะมีเพียงเฟสเดียวที่ได้รับสัญญาณกระตุ้น ในตารางที่ 2.1 แสดงลำดับการป้อนสัญญาณเพื่อขับสเต็ปเปอร์มอเตอร์แบบ 1 เฟส

ตารางที่ 2.1 ลำดับการป้อนสัญญาณกระตุ้นของวงจรขับสเต็ปเปอร์มอเตอร์แบบ 1 เฟส

สเต็ปที่	PHASE1	PHASE2	PHASE3	PHASE4
1	ทำงาน			
2		ทำงาน		
3			ทำงาน	
4				ทำงาน

ตารางที่ 2.2 ลำดับการป้อนสัญญาณกระตุ้นของวงจรขับสเต็ปเปอร์มอเตอร์แบบ 2 เฟส

สเต็ปที่	PHASE1	PHASE2	PHASE3	PHASE4
1	ทำงาน	ทำงาน		
2		ทำงาน	ทำงาน	ทำงาน
3			ทำงาน	
4	ทำงาน			ทำงาน

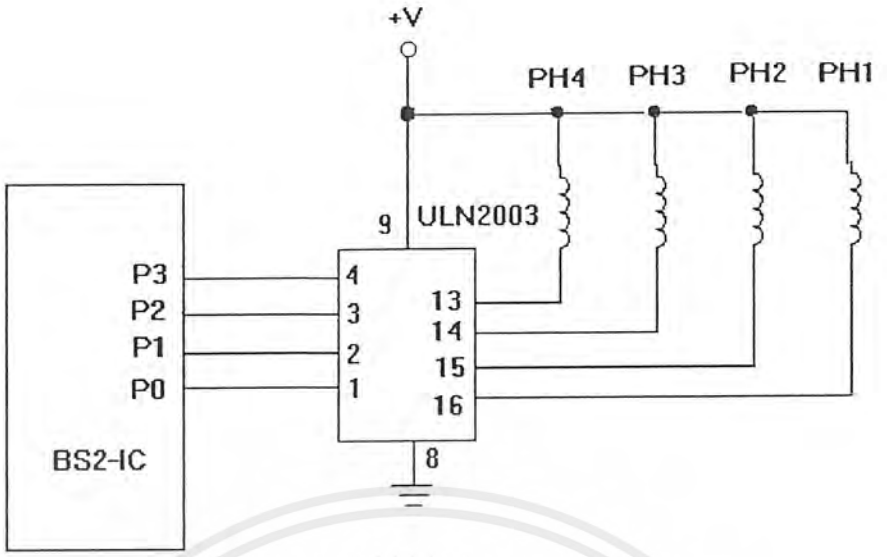
แบบ 2 เฟสจะมีลักษณะคล้ายกับแบบ 1 เฟส บางทีเรียกการขับแบบนี้ว่า แบบฟูลสเต็ป (Full Step) แต่แทนที่จะส่งสัญญาณกระตุ้นเพียงเฟสเดียว ในการขับแบบนี้จะป้อนสัญญาณกระตุ้นไปยังเฟสของมอเตอร์ที่อยู่ใกล้กันในเวลาเดียวกันและเรียงลำดับกันไปเช่นเดียวกับแบบ 1 เฟส ดังแสดงในตารางที่ 2.2 คือเริ่มด้วยการป้อนสัญญาณกระตุ้นไปยังเฟส 1 และ 2 พร้อมกัน

ในสเต็ปที่ 1 ในสเต็ปที่ 2 จะป้อนสัญญาณไปยังเฟสที่ 2 และ 3 ถัดมาในเฟสที่ 3 จะทำการป้อนสัญญาณกระตุ้นไปที่เฟส 3 และ 4 ในสเต็ปที่ 4 จะป้อนสัญญาณไปยังเฟสที่ 4 และ 1แล้ววนกลับไปที่เฟส 1 และ 2 อีกครั้ง ด้วย การขับแบบนี้ทำให้แรงบิดหรือทอค(toque) มากกว่าแบบหนึ่งเฟส แต่ข้อเสียคือ ใช้พลังงานในการขับเพิ่มมากขึ้น

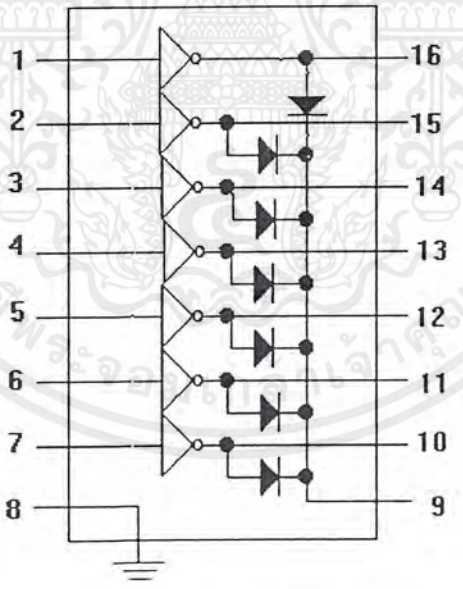
ตารางที่ 2.3 ลำดับการป้อนสัญญาณกระตุ้นของวงจรขับสเต็ปเปอร์มอเตอร์แบบฮาล์ฟสเต็ป

สเต็ปที่	PHASE1	PHASE2	PHASE3	PHASE4
1	ทำงาน			
2	ทำงาน	ทำงาน		
3		ทำงาน		
4		ทำงาน	ทำงาน	
5			ทำงาน	
6			ทำงาน	ทำงาน
7				ทำงาน
8	ทำงาน			ทำงาน

แบบฮาล์ฟสเต็ป การขับแบบนี้ได้รับความนิยมมากที่สุด เนื่องจากสามารถช่วยให้สเต็ปเปอร์มอเตอร์สามารถหมุนได้อย่างละเอียดมากขึ้นเป็นสองเท่าของความละเอียดปกติของสเต็ปเปอร์มอเตอร์ โดยมีรูปแบบการขับให้หมุนแสดงในตารางที่ 2.3 จะเห็นได้ว่า การขับสเต็ปเปอร์มอเตอร์แบบนี้เป็นการผสมผสานระหว่างแบบ 1เฟสและ 2 เฟส กล่าวคือ มีทั้งการป้อนสัญญาณกระตุ้นไปยังขดลวดเพียงเฟสเดียวและพร้อมกับแบบ 2 เฟสในช่วงเวลาหนึ่ง ด้วยการขับแบบนี้ส่งผลให้แรงบิดที่ได้จาการหมุนเพิ่มขึ้น เพราะระยะทางในการหมุนสั้นลง ความถูกต้องของตำแหน่งที่หมุนมีเพิ่มมากขึ้น เพียงแต่ว่าในการขับแต่ละสเต็ปจะให้ผลเพียงครึ่งสเต็ปของการขับปกติดังนั้นหากต้องการให้เคลื่อนที่เป็นไปแบบเต็มสเต็ปจะต้องกำหนดให้ทำการหมุนไป 2 สเต็ปต่อเนื่องกัน



รูปที่ 2.5 วงจรขับสเต็ปเปอร์มอเตอร์ของเบสิกแสตมป์ 2



IC-7Channel CMOS/TTL Input Driver

รูปที่ 2.6 แสดงวงจรภายในของ IC เบอร์ ULN2003A

2.7 การขับสเต็ปเปอร์มอเตอร์โดยใช้เบสิกแสตมป์ 2

เนื่องจากขาพอร์ตของเบสิกแสตมป์ 2 ทำงานเป็นขาเอาต์พุต มีความสามารถในการจ่ายกระแสประมาณ 10-20 mA ดังนั้นจึงไม่สามารถนำไปขับสเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์ได้โดยตรง จะต้องส่งสัญญาณที่ใช้ในการขับสเต็ปเปอร์มอเตอร์ผ่านไอซีหรือวงจรขับกระแสสูง ไอซีขับโหลดกระแสและแรงดันสูงเบอร์ ULN2003มีวงจรแสดงดังรูป

ภายในไอซี ULN2003 เป็นวงจรอินเวอร์เตอร์หรือวงจรกลับสถานะลอจิก ดังนั้นในการป้อนสัญญาณในการขับสเต็ปเปอร์มอเตอร์จะต้องป้อนด้วยลอจิก "1" และการต่อเฟสของมอเตอร์เข้ากับ ULN2003 จะต้องต่อในลักษณะคอลเลคเตอร์เปิดกล่าวคือขดลวดด้านหนึ่งต้องต่อกับไฟเลี้ยงมอเตอร์ส่วนอีกด้วนหนึ่งต่อขับกับไอซีขับเบอร์ ULN2003 เมื่อ ULN2003 ได้รับสัญญาณลอจิก "1" ก็จะกลับเป็นลอจิก "0" ทำให้เสมือนว่าขดลวดถูกต่อลงกราวด์ทำให้เกิดการครบวงจรมีกระแสไหลผ่านขดลวดนั้นๆอย่างสมบูรณ์ มอเตอร์ก็จะเคลื่อนไปหนึ่งสเต็ป

การเขียนโปรแกรมควบคุมต้องกำหนดเสียก่อนว่าต้องการขับสเต็ปมอเตอร์ในลักษณะใด 1 เฟส 2 เฟส หรือแบบฮาล์ฟสเต็ป จากนั้นทำการเขียนข้อมูลส่งออกไปยังพอร์ตที่ต่อกับไอซีขับ ULN2003 ทุกครั้งที่ส่งข้อมูลออกไปในแต่ละสเต็ปต้องทำการหน่วงเวลาเล็กน้อยก่อนส่งข้อมูลของการหมุนในสเต็ปต่อไป ทั้งนี้เพื่อให้ไอซีขับและตัวสเต็ปเปอร์มอเตอร์สามารถตอบสนองข้อมูลก่อนหน้านี้ได้ทัน ก่อนที่จะได้รับข้อมูลใหม่ต่อไป

ข้อมูลสำหรับการขับสเต็ปเปอร์มอเตอร์

ข้อมูลที่เบสิกแสตมป์ 2 จะต้องส่งออกไปยัง ULN2003 เพื่อขับสเต็ปเปอร์มอเตอร์จะขึ้นอยู่กับรูปแบบการขับ ในกรณีขับแบบหนึ่งเฟส ข้อมูลสำหรับขับสเต็ปเปอร์มอเตอร์จะเป็นดังนี้

สเต็ปที่	ข้อมูล		
	ฐานสอง	ฐานสิบ	ฐานสิบหก
1	%0001	1	$1
2	%0010	2	$2
3	%0100	4	$4
4	%1000	8	$8
5	%0001	1	$1(เริ่มต้นวนรอบใหม่)

ตัวอย่างโปรแกรมพีเบสิกเพื่อส่งข้อมูลขับสเต็บมอเตอร์แบบหนึ่งเฟสมีดังนี้

```
'one-phase stepper motor driver example program
DIRA=%1111
Loop:
        OUTA=%0001
        PAUSE 100
        OUTA=%0010
        PAUSE 100
        OUTA=%0100
        PAUSE 100
        OUTA=%1000
        PAUSE 100

        GOTO loop
```

ส่วนข้อมูลของการขับสเต็ปเปอร์มอเตอร์แบบสองเฟสมีดังนี้

สเต็ปที่	ข้อมูล		
	ฐานสอง	ฐานสิบ	ฐานสิบหก
1	%0011	3	$3
2	%0110	6	$6
3	%1100	12	$C
4	%1001	9	$9
5	%0011	3	$3(เริ่มต้นวนรอบใหม่)

ตัวอย่างโปรแกรมพีเบสิกเพื่อส่งข้อมูลขับสเต็บมอเตอร์แบบสองเฟสมีดังนี้

```
'two-phase stepper motor driver example program
DIRA=%1111
Loop:
        OUTA=%0011
        PAUSE 100
        OUTA=%0110
        PAUSE 100
```

```
OUTA=%1100
PAUSE 100
OUTA=%1001
PAUSE 100
GOTO loop
```

โปรแกรมพีเบสิกเพื่อส่งข้อมูลขับสเต็บมอเตอร์แบบฮาล์ฟสเต็ป มีดังนี้

สเต็ปที่	ข้อมูล		
	ฐานสอง	ฐานสิบ	ฐานสิบหก
1	%0001	1	$1
2	%0011	3	$3
3	%0010	2	$2
4	%0110	6	$6
5	%0100	4	$4
6	%1100	12	$C
7	%1000	8	$8
8	%1001	9	$9
9	%0001	1	$1(เริ่มต้นวนรอบใหม่)

ตัวอย่างโปรแกรมพีเบสิกเพื่อส่งข้อมูลขับสเต็บมอเตอร์แบบฮาล์ฟสเต็ปมีดังนี้

```
'Half-step stepper motor driver example program
DIRA=%1111
Loop:
        OUTA=%0001
        PAUSE 100
        OUTA=%0011
        PAUSE 100
        OUTA=%0010
        PAUSE 100
        OUTA=%0110
        PAUSE 100
```

```
OUTA=%1100
PAUSE 100
OUTA=%1000
PAUSE 100
OUTA=%1001
PAUSE 100
GOTO loop
```

ถ้าหากต้องการขับสเต็ปเปอร์มอเตอร์ให้หมุนไปในทิศทางตรงข้าม สามารถทำได้โดยการส่งข้อมูลย้อนกลับจากที่กำหนดไว้ในโปรแกรมตัวอย่าง ยกตัวอย่าง ในกรณี 1 เฟส ให้ทำการส่งข้อมูล %1000 หรือ $8 ออกไปก่อน ตามด้วย %0100, %0010 และ %0001 ในที่สุดจากนั้นวนกลับมาที่ %1000 อีกครั้ง

การขับสเต็ปเปอร์มอเตอร์หมุนครบหนึ่งรอบ จะใช้จำนวนพัลส์ที่ต่างกันออกไปขึ้นอยู่กับโครงสร้างของมอเตอร์และรูปแบบในการขับ กล่าวคือ หากมอเตอร์มีความละเอียด 7.5 องศาต่อสเต็ป ดังนั้นต้องการให้หมุนครบรอบต้องส่งพัลส์ 48 ลูก คำนวณได้จากการเคลื่อนที่เป็นวงวกลมหนึ่งรอบจะมีมุมรวมทั้งสิ้น 360 องศา นำความละเอียดของมอเตอร์มาหาร 360 องศา ก็จะได้ค่าเท่ากับ 48 ในกรณีที่ใช้การขับแบบฮาล์ฟสเต็ป จำนวนพัลส์ที่ใช้ก็เพิ่มเป็นสองเท่า ถ้าหากใช้มอเตอร์ 7.5 องศาต่อสเต็ป ต้องส่งพัลส์ทั้งสิ้น 96 ลูก สำหรับการขับให้มอเตอร์หมุนครบ 1 รอบ

ความเร็วของการหมุนจะขึ้นอยู่กับอัตราการส่งข้อมูลไปยังวงจรขับและการตอบสนองของมอเตอร์เอง ถ้าหากส่งข้อมูลได้เร็วเท่าใดมอเตอร์ก็จะหมุนเร็วขึ้นเท่านั้น อย่างไรก็ตามตัวสเต็ปมอเตอร์ก็มีข้อจำกัดในการสัญญาณพัลส์ที่ส่งมากระตุ้นเช่นกัน หากส่งสัญญาณมาเร็วเกินไป สเต็ปมอเตอร์จะหยุดหมุน ทั้นนี้เนื่องจากตัวสเต็ปมอเตอร์ไม่สามารถตอบสนองพัลส์กระตุ้นที่ส่งมาได้ทันที

2.8 วิชวลเบสิก(Visual basic)

วิชวลเบสิก(visual basic) เป็นโปรแกรมภาษาที่ใช้ในการเขียนโปรแกรมบนระบบปฏิบัติการวินโดวส์(Windows) ที่พัฒนาโดยบริษัทไมโครซอฟท์ ซึ่งที่รากฐานจากภาษาเบสิก (basic) วิชวลเบสิก เป็นโปรแกรมภาษาที่มีวิธีการโปรแกรมในแบบ Graphic User Interface (GUI) ซึ่งแตกต่างจากโปรแกรมคำสั่งแบบบรรทัดคำสั่ง ซึ่งจะต้องเขียนคำสั่งเพื่อกำหนดภาพลักษณ์ของสิ่งต่างๆ ที่ปรากฏบนจอภาพ แต่การโปรแกรมด้วย วิชวลเบสิกนั้นสามารถกำหนดภาพลักษณ์ต่างๆ ด้วยการจับมาวางไว้บนจอภาพได้เลย

Event and Massage

ระบบปฏิบัติการวินโดวส์จะประกอบด้วยส่วนต่างๆ เช่น ปุ่มกด (command bottom) ช่องใส่ข้อความ (text box) แถบเลือก (tab control) เมนูบาร์ (menu bar) แถบเลื่อน (scroll bar) ซึ่งส่วนต่างๆ นี้ก็เป็นวินโดวส์ทั้งสิ้น ระบบปฏิบัติการวินโดวส์มีวิธีการจัดการและควบคุมวินโดวส์ต่างๆ เหล่านี้ด้วยการกำหนดหมายเลขไม่ซ้ำกันให้กับวินโดวล์แต่ละตัวเรียกว่า Windows handle (HAND) ระบบจะคอยเฝ้าดูอาการ (sign soft Activity) และเหตุการณ์ (event) ที่เกิดขึ้นกับแต่ละวินโดวส์ เหตุการณ์ต่างๆ ที่เกิดขึ้นจากผู้ใช้ (user) เลือกหรือกดปุ่มใดๆ จากเม้าส์ (mouse) หรือแป้นพิมพ์ (keyboard) ซึ่งก่อให้เกิดข่าวสาร (Massage) ส่งไปที่ระบบปฏิบัติการ (OS) เมื่อระบบปฏิบัติการได้รับข่าวสารก็จะประมวลผล (process) และกระจายข่าวสารให้กับวินโดวส์อื่นๆ แต่ละวินโดวส์ก็จะมีวิธีการที่เหมาะสมคอยรับข่าวสารเฉพาะสำหรับตัวเองเพื่อนำมาประมวลผลต่อไป

Event-Driver

การเขียนโปรแกรมในยุคแรกๆนั้น การทำงานของโปรแกรมนั้นจะมีการทำงานตามโค้ดของโปรแกรมไปเรื่อยๆ หากมีการเรียนโปรแกรมย่อย (procedure) ใดก็จะกระโดดไปยัง Procedure นั้นจนจบโปรแกรม

ในส่วนโปรแกรมแบบ Event-Driver นั้นการทำงานกับโค้ดโปรแกรมจะขึ้นกับเหตุการณ์ (event) ที่มากระทำกับวัตถุ (object) หนึ่งๆ คือ หากมีการกระทำใดๆกับ Object ก็จะมีเหตุการณ์ต่างๆเช่น Click, Double Click Mouse เป็นต้น จะเขียนโค้ดภายในเหตุการณ์ (event) นั้นๆ เพื่อให้โปรแกรมทำงานตามที่ต้องการเมื่อมีเหตุการณ์นั้นๆเกิดขึ้นกับวัตถุ (object) ซึ่งเหตุการณ์จะเกิดขึ้นโดยการกระทำของผู้ใช้

ตารางที่ 2.4 แสดง Event ที่เกิดขึ้นจาก Mouse

Event	ลักษณะการควบคุมจากผู้ใช้
Mouse Down	กดปุ่ม Mouse ค้างไว้
Mouse Up	ปล่อยปุ่ม
Click	กดและปล่อยปุ่ม Mouse ในเวลาเดียวกัน
DbClick	กดและปล่อยปุ่ม Mouse 2 ครั้งในเวลาเดียวกัน
MouseMove	ขยับ Mouse
Drag	กดปุ่ม Mouse ค้างพร้อมกับเลื่อน Mouse
Drop	กดปุ่ม Mouse ค้างเลื่อน Mouse ปล่อยปุ่ม Mouse

ตารางที่ 2.5 แสดง Event ที่เกิดจาก Keyboard

Event	ลักษณะการควบคุมจากผู้ใช้
Key Down	กดปุ่มค้างไว้
Key Up	ปล่อยปุ่ม
Key Press	กดปุ่มและปล่อยปุ่มในเวลาเดียวกัน

2.9 การสื่อสารแบบอนุกรม

การสื่อสารแบบแบบอนุกรมนั้นจะแบ่งออกได้เป็น 2 แบบ คือการสื่อสารอนุกรมแบบซิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัสการสื่อสารอนุกรมแบบซิงโครนัสนั้นจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนอีกสายหนึ่งจะเป็นสายข้อมูลดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 สาย คือ สัญญาณนาฬิกา สัญญาณข้อมูล และกราวด์รูปที่ 2.7 แสดงให้เห็นถึงไทย์มิ่งไดอะแกรมของการส่งข้อมูลแบบซิงโครนัส

รูปที่ 2.7 รูปแบบของข้อมูลอนุกรม

การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัลคือการรับและส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยเหมือนกับการส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทางภาครับและภาคส่งมีค่าเท่ากัน ซึ่งเรียกสัญญาณที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูลหรือบอดเรต(baudrate)มีหน่วยเป็นบิตต่อวินาที(bit per secone:bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1.บิตเริ่มต้น(start bit)ซึ่งมีขนาด 1 บิต

2.บิตข้อมูลแบบอนุกรมจะมีขนาด 5,6,7 หรือ8 บิต

3.บิตตรวจสอบพาริตี้(parity bit)จะมีขนาด 1 บิตหรือไม่มี

4.บิตปิดท้าย(stop bit)จะมีขนาด1,1.5หรือ 2 บิต

รูปที่ 2.8 รูปแบบของข้อมูลแบบอะซิงโครนัส

รูปที่ 2.8 แสดงรูปแบบของข้อมูลแบบอะซิงโครนัสซึ่งเมื่อไม่มีข้อมูลที่จะส่งขา DATA จะมีสถานะลอจิก 1 ซึ่งจะเรียกสถานะนี้ว่าสถานะหยุดรอ (waiting stage)การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก 0 ด้วยช่วงระยะเวลา 1 บิต ซึ่งจะเรียกบิตนี้ว่าบิตเริ่มต้นจากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยต่ำสุด(LSB) ก่อน ซึ่งข้อมูลในไบต์ที่จะส่งอาจจะมีจำนวนบิต 567 หรือ 8 บิตก็ได้ จากนั้นจะตามด้วยพาริตี้ ซึ่งใช้เพื่อการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้าย ซึ่งจะให้ขาดาต้ามีสถานะลอจิก 1 อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต 1.5 บิตหรือ2 บิตเพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

2.10 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทางโดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียวโดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็กทรอนิกส์(Electronic Industries Association:EIA)ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3 ถึง –12v แสดงว่ามีข้อมูล(Mark)และ +3ถึง +12 แสดงว่าเป็นช่องว่าง(Space)

คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-25 เนื่องจากขาอื่นๆที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป

สำหรับรายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232

Data Carrier Detect :DCD หรืออาจจะเรียกว่า Carrier Detect :CD ขานี้จะมีการแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก

Receive Data:RD หรือ RxD ขาใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บได้เก็บไว้ในรีจิสเตอร์บัฟเฟอร์

Transmitted Data:TD หรือ TxD ขาที่ใช้ส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลออกไป

Data Terminal Ready:DTR เป็นขาที่สัญญาที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางได้รับรู้ว่าต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทางและขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์

Signal Ground:GND ขากราวด์ของระบบ

Data Set Ready:DSR ขานี้จะถูกต่อกับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

Request To Sent:RTS เป็นขาสำหรับส่งสัญญาณร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์โดยที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null modem 3 สาย จะต้องเชื่อมต่อขา RST และ CTS ของตัวของมันเองเข้าด้วยกันเพื่อจะให้การรับและส่งข้อมูลข่าวสารสามารถเกิดขึ้นได้ตลอดเวลา

Clear To Send:CTS ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นจึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่

Ring Indicator:RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยปกติสายนี้จะไม่ถูกใช้งาน จะใช้งานต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

2.11 การรับและส่งข้อมูลแบบอนุกรมด้วย Visual Basic

เนื่องจากระบบปฏิบัติการบนวินโดว์ ได้ฝังตัวพอร์ตอนุกรมเข้าเป็นส่วนหนึ่งของระบบปฏิบัติการแล้ว ดังนั้นการเรียกใช้งานจึงจำเป็นต้องเรียกผ่านเครื่องมือที่ติดต่อกับระบบปฏิบัติการ เช่นการใช้คอนโทรล MSCOMM32.OCX ของโปรแกรม Visual Basic

คอนโทรล MSComm

สำหรับการใช้งาน Viual Basic ตั้งแต่เวอร์ชัน 2 เป็นต้นมา ใน Visual Basic จะมีคอนโทรลสำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมคอมพิวเตอร์มาให้ Mscomm จัดเตรียมเอาไว้สองทางเพื่อความสะดวกในการสื่อสารข้อมูลทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์(event driven communication)เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่นเมื่อตัวอักษรส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier (DCD) หรือขา Requast To Send(RTS)เหตุการณ์ OnComm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นทันที ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อคุณสมบัติ commevent ต่อไป ส่วนทางเลือกที่สองเป็นการตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูการที่เปลี่ยนแปลงภายในคุณสมบัติ commevent หลังจากที่ให้โปรแกรมทำงานในฟังชันต่างๆไปเรียบร้อยแล้ว ซึ่งวิธีใช้งานได้ดีในโปรแกรมที่มีขนาดเล็ก

คุณสมบัติของ MSComm

CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่(com1,com2,com3,com4)

รูปแบบการใช้งาน

object.CommPort[= *value*]

โดย value เป็นค่าของพอร์ตอนุกรม ชนิดของข้อมูลเป็น Integer ค่า Value สามารถกำหนดในช่วง 1-6(ค่าเริ่มต้นกำหนดไว้ที่ 1)เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด Error 68 ขึ้นมา ซึ่งหมายถึงอุปกรณ์ตัวนี้ไม่อยู่ในระบบดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คำสั่ง OpenPort

Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอดพาริตี้จำนวนบิตข้อมูลจำนวนของบิตปิดท้าย

รูปแบบการใช้งาน

object.**Settings** [= *value*]

ค่าของ Value มีชนิดข้อมูลแบบ String มีรูปแบบเป็น “BBBB,P,D,S”BBBB เป็นค่าอัตราบอด,P เป็นพาริตี้,D เป็นจำนวนบิตข้อมูล และ S เป็นจำนวนของบิตปิดท้าย ปกติค่านี้จะถูกกำหนดไว้เป็น “9600,N,8,1”

ค่าของบอดเรตมาตรฐานที่ใช้กับ MSComm มีดังนี้

110 บิตต่อวินาที

300 บิตต่อวินาที

600 บิตต่อวินาที

1,200 บิตต่อวินาที

2,400 บิตต่อวินาที

9,600 บิตต่อวินาที(ค่าปกติ)

14,400 บิตต่อวินาที

19,200 บิตต่อวินาที

28,800 บิตต่อวินาที

38,400 บิตต่อวินาที(สงวน)

56,000 บิตต่อวินาที(สงวน)

128,000 บิตต่อวินาที(สงวน)

256,000 บิตต่อวินาที(สงวน)

สำหรับมาตรฐานในการกำหนดค่าพาริตี้บิตมีดังนี้

สัญลักษณ์	รายละเอียด
E	พาริตี้คู่(Even)
M	ลอจิก "1"(MARK)
N	ไม่ใช่(ค่าปกติ)
O	พาริตี้คี่(Odd)
S	ลอจิก "0"(Space)

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 ค่าคือ 4,5,6,7 และ8(เป็นค่าปกติ)

ค่าที่ระบุจำนวนบิตปิดท้ายมี 3 ค่าคือ 1(เป็นค่าปกติ),1.5 และ2

ตัวอย่างการใช้งานคำสั่ง setings โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต สามารถเขียนโปรแกรมได้ดังนี้

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

หมายเหตุ สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายในเครื่องหมายคำพูด " " เนื่องจาก ค่าที่กำหนดนี้อยู่ในรูปตัวแปร String

PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

object.PortOpen [= *value*]

ค่า Value มีชนิดข้อมูลแบบบูลีน คือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรม สำหรับการเปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะเปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen
ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งให้ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ถ้าคุณสมบัติ DTREable ถูกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ตค่าคุณสมบัติของ DTREable จะถูกเซตเป็น False หลังจากการเปิดพอร์ต แต่ถ้าเซตเป็น False ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิตและบิตปิดท้าย 1 บิต มีดังนี้

ตัวอย่างการใช้คำสั่งเปิดพอร์ตเพื่อติดต่อสื่อสารกับพอร์ตอนุกรม COM 1 และมีบอดเรต 9,600 บิตต่อวินาที ไม่มีพาริตี้จำนวนบิตข้อมูล 8 บิตและบิตปิดท้าย 1 บิตมีดังนี้

```
MSComm1.Settings = "9600,N,8,1"
MSComm1.Commport=1
MSComm1.PortOpen=True
```

Input

อ่านค่าและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

object.Input

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดให้ค่า Input Len เท่ากับ 0 เป็นกรกำหนดให้คุณสมบัติ Input ทำการอ่านข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้า InputMode ถูกกำหนดเป็น ComInputModeText คุณสมบัติ Input จะส่งค่าข้อมูลกลับมาในรูปแบบของข้อความชนิดข้อมูลเป็นแบบ Variant ถ้า InputMode กำหนดเป็น ComInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

ตัวอย่างโปรแกรมแสดงให้เห็นถึงวิธีการรับข้อมูลจากบัฟเฟอร์รับข้อมูลทั้งหมด

```
Private Sub Command1_Click()
Dim InString as String
' Retrieve all available data.
MSComm1.InputLen = 0

' Check for data.
If MSComm1.InBufferCount Then
   ' Read data.
   InString = MSComm1.Input
End If
End Sub
```

InBufferCount

ส่งค่าจำนวนของตัวอังษรที่อยูรในบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

object.InBufferCount[= *value*]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

หมายเหตุ อย่าสับสนระหว่างคำสั่ง InputBufferSize และ InputBufferCount คำสั่ง InputBufferSize นั้นเพื่อใช้กำหนดบัฟเฟอร์ภาครับ

InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งานคำสั่ง

object.InBufferSize[= *value*]

คำสั่ง InBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ โดยค่าเริ่มต้นถูกกำหนดไว้ที่ 1,024 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาครับขนาดใหญ่จะทำให้หน่วยความจำที่เหลือสำหรับการใช้งานส่วนอื่นๆจะเหลือน้อย อย่างไรก็ตามการกำหนดค่าบัฟเฟอรีภาครับที่น้อยเกินไปจะทำให้เกิดโอเวอร์โฟลวหรือข้อมูลล้นบัฟเฟอร์ เว้นแต่จะมีการแฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมคือค่า1,024 ซึ่งเป็นค่าเริ่มต้นนั่นเอง แต่ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลวแล้วจึงค่อยปรับค่าขนาดของบัฟเฟอร์ให้มีค่ามากขึ้น

InputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

object.InputLen [= *value*]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ "0" จะทำให้คำสั่ง Input ของ MSComm อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ภายในบัฟเฟอร์ภาครับมากเท่ากับจำนวน UnputLen คำสั่ง Input จะส่งค่าว่าง (" ")กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InBufferCount โดยกำหนดให้มีข้อมูลอยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงอ่านค่าข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับการอ่านข้อมูลจากเครื่องมือหรือเครื่องจักรที่มีการกำหนดค่าขนาดความยาวของข้อมูลเอาไว้แล้ว

ตัวอย่างโปรแกรมการอ่านค่าตัวอักษรออกมา 10 อักษร

```
Private Command1_Click()
Dim CommData as String' Specify a 10 character block of data.
MSComm1.InputLen = 10' Read data.
CommData = MSComm1.Input
End Sub
```

InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งานคำสั่ง

object.InputMode [= *value*]

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภทคือ

comInputModeText สำหรับข้อมูลที่อยู่ในรูปของข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น 0 และค่าเริ่มต้นของการรับค่าข้อมูลก็จะเป็นค่านี้

comInputModeBinary สำหรับข้อมูลอื่นๆซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูล

ตัวอย่างการใช้งาน InputMode ต่อไปนี้จะทำการอ่านค่าข้อมูล 10 ไบต์จากพอร์ตอนุกรมและเก็บข้อมูลไว้ในตัวแปรแบบอาเรย์ชนิดข้อมูลเป็นแบบไบต์

```
Private Sub Command1_Click()
Dim Buffer as Variant
Dim Arr() as Byte

' Set and open port
MSComm1.CommPort = 1
MSComm1.PortOpen = True
```

```
' Set InputMode to read binary data
MSComm1.InputMode = comInputModeBinary

' Wait until 10 bytes are in the input buffer
Do Until MSComm1.InBufferCount < 10
  DoEvents
Loop

' Store binary data in buffer
Buffer = MSComm1.Input

' Assign to byte array for processing
Arr = Buffer

End Sub
```

Output

ใช้ในการส่งขบวนข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

object.Output [= *value*]

ค่า value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอังษรจะต้องกำหนดข้อมูลเป็นแบบ Varint และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดของข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ Byte

ตัวอย่างโปรแกรมการส่งค่าที่ป้อนจากคีย์บอร์ดไปยังพอร์ตอนุกรม โดยใช้คุณสมบัติ Output

```
Private Sub Form_KeyPress (KeyAscii As Integer)
   Dim Buffer as Variant

   ' Set and open port
   MSComm1.CommPort = 1
   MSComm1.PortOpen = True

   Buffer = Chr$(KeyAscii)
   MSComm1.Output = Buffer
End Sub
```

OutbufferCount

คืนค่าจำนวนของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่ง และสามารถใช้คำสั่งนี้เคลียร์บัฟเฟอร์ภาคส่งได้ด้วย

รูปแบบการใช้งานคำสั่ง

object.OutBufferCount [= *value*]

ผู้ใช้งานสามารถเคลียร์บัฟเฟอรีภาคส่งได้โดยการกำหนดค่า OutBufferCount เท่ากับ "0"

หมายเหตุ ระวังการสับสนระหว่างคุณสมบัติ OutBufferCount กับ OutBufferSize ซึ่ง OutBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาคส่ง

OutBufferSize

กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดของตัวแปรเป็นแบบไบต์

รูปแบบการใช้งานคำสั่ง

object.OutBufferSize [= *object*]

คุณสมบัติ OutputBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่งโดยปกติค่าที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาคส่งที่มากเกินไปจะทำให้ มีหน่วยความจำเหลือให้ใช้งานน้อย แต่อย่างไรก็ตามถ้ากำหนดค่าน้อยเกินไป จะทำให้ข้อมูลล้นบัฟเฟอร์ขึ้นได้ยกเว้นจะมีการใช้แฮนด์เช็ค วิธีการที่ถูกต้องในการกำหนดค่าคือ ทดลองใช้ค่าเริ่มต้นคือค่า 512 ไบต์ดูก่อนถ้าโปรแกรมทำงานแล้วเกิดการล้นของข้อมูลค่อยเพิ่มค่าของ OutBufferSize ให้มากขึ้น

ParityReplace

กำหนดค่าและคืนค่าตัวอักษรที่ไปวางแทนในตำแหน่งที่เกิดข้อผิดพลาดจากพาริตี้

รูปแบบและการใช้งานคำสั่ง

object.ParityReplace [= *value*]

บิตพาริตี้ เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมกับข้อมูล เพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MSCOM จะทำการบวกบิตทุกบิตที่มีค่าลอจิก "1" ในแต่ละไบต์ และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก "1" เป็นเลขคู่หรือเลขคี่ และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าค่าที่นำมาบวกแล้วมีพาริตี้ที่ไม่ตรงแสดงว่าการรับส่งข้อมูลผิดพลาด

การกำหนดค่าเริ่มต้นให้กับ ParityReplace นั้นกำหนดให้ใช้เครื่องหมาย(?)ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ผิดพลาด ถ้ากำหนดค่า ParityReplace ให้เป็นค่าว่าง (" ") จะเป็นการยกเลิกการใช้งาน ParityReplace และไม่มีการป้อนข้อมูลแทนเมื่อตรวจพบข้อผิดพลาด

ParityReplace ใช้ข้อมูลเป็นแบบสตริงแต่การกำหนด จะกำหนดได้เพียงไบต์เดียวเท่านั้น ซึ่งสามารถใช้ค่าใดๆก็ได้ที่เป็นโค้ด ANSI มีอยู่ระหว่าง 0-255

DTREnable

ใช้ในการกำหนดสถานะลอจิกขา Data Terminal Ready(DTR) โดยสัญญาณของขา DTR จะส่งออกจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดข้อมูลเป็นแบบบูลีน

รูปแบบการใช้งาน

object.DTREnable[= *value*]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิกของขา DTR ให้เป็น "0" หรือ "1" โดย

True หมายถึง ให้ขา DTR มีลอจิก "1"

False หมายถึง ให้ขา DTR มีลอจิก "0"(เป็นค่าปกติ)

หมายเหตุ เมื่อขา TDR ถูกกำหนดสถานะให้เป็น Ture ที่ขา DTR จะมีสถานะเป็นลอจิก "1" เมื่อทำการเปิดพอร์ตและจะมรสถานะเป็น "0" เมื่อมีการปิดพอร์ต เมื่อขา DTR ถูกกำหนดสถานะเป็น False ที่ขา DTR จะมีสถานะลอจิก "0"ตลอดเวลาไม่ว่าจะใช้คำสั่งเปิดพอร์ตหรือปิดพอร์ต

สำหรับการใช้งานกับโมเด็ม การทำให้ขา DTR เป็นลอจิก "0"จะเป็นการวางหูโทรศัพท์หรือยกเลิกการติดต่อ

RTSEnable

เพื่อใช้กำหนดสถานะลอจิกให้ขา Request To Send(RTS) โดยที่ขา RTS เป็นสัญญาณที่ส่งออกจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอส่งข้อมูล ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

object.RTSEnable[= *value*]

ค่า value เป็นค่าสถานะ Ture หรือ False เพื่อกำหนดลอจิก "0" หรือ "1" ให้ขา RTS โดย

True หมายถึง ให้ขา RTS มีลอจิก "1"

False หมายถึง ให้ขา RTS มีลอจิก "0"(เป็นค่าปกติ)

หมายเหตุ เมื่อขา RTSEnable ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก "1" เมื่เปิดพอร์ตและมีสถานะลอจิก "0" เมื่อเปิดพอร์ต

EOFEnable

เป็นการกำหนดให้ MSComm รอสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์(End Of File:EOF) ระหว่างรับอินพุตเข้ามา ถาพบสัญลักษณ์ EOF ภาคอินพุตจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ ConEvEOF

รูปแบบการใช้งาน

object.EOFEnable [= *value*]

โดย value เป็นค่าสถานะ Treu หรือ False เพื่ออีนาเบิลหรือดิสเอเบิล การทำงานของเหตุการณ์OnComm เมื่อตรวยพบสัญลักษณ์ EFO โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะไม่ถูกกระตุ้นให้ทำงานด้วย EOF(เป็นค่าปกติ)

เมื่อEOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

CTSHolding

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

รูปแบบการใช้งาน

object.CTSHolding

เมื่อขา CTS เป็นลอจิก "0"(CTSHolding=False)และเกิดไทย์เอาต์ 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 มีสถานะลอจิก "1" ถ้าค่า CDHolding เป็น False ขา DCD มีสถานะลอจิก "0"

รูปแบบการใช้งาน

object.CDHolding

เมื่อขา DCD มีลอจิก "1"(CDHolding=True)และเกิดไทย์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCDTO (Clear Detect Timeout Error)และกระตุ้นให้เกิดเหตุการณ์ OnComm

DSRHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา DSR(DSR)ได้ว่ามีสถานะลอจิก "1"หรือ "0" โดยค่าที่อ่านได้จะเป็นบูลีน True หรือ False ถ้าค่า DSRHolding เป็น True ขา DSR จะมีสถานะลอจิก "1" ถ้าค่า DSRHolding เป็น False ขา DSR มีสถานะลอจิก "0"

รูปแบบการใช้งาน

object.DSRHolding

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

Handshaking

กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์

รูปแบบกรใช้งานคำสั่ง

object.Handshaking [= *value*]

ค่าตัวแปร Value ที่ใช้กำหนดค่ากำหนดได้ 4 รูปแบบด้วยกันคือ

1.comNone ค่าที่กำหนดคือ 0 เป็นการกำหนดให้ไม่มีการแฮนด์เช็ก(เป็นค่าเริ่มต้น)

2.comXOnXOff ค่าที่กำหนดคือ 1 เป็นการกำหนดให้ใช้แฮนด์เช็ก XONXOFF

3.comRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS(Request To Send/Clear TO Send)

4.commRTSXOnXOff ค่าที่กำหนดคือ 3 เป็นการกำหนดให้ใช้ทั้งแบบ Request To Send และ XONXOFF

คุณสมบัติ Handshaing ใช้เพื่อกำหนดการสื่อสารภายใน ระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลตัวอักษรถูกส่งมายังพอร์ตอนุกรม อุปกรณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลบัฟเฟอร์ภาครับ เพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ถ้าไม่มีบัฟเฟอร์ภาครับโปรแกรมที่ใช้งานจะต้องทำการอ่านค่าข้อมูลโดยตรงจากฮาร์ดแวร์ของพอร์ตอนุกรม ซึ่งผู้ใช้งานจะเกิดปัญหาข้อมูลสูญหาย เนื่องจากว่าการเปลี่ยนแปลงข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติ handshaking ช่วยให้ผู้ใช้งานแน่ใจว่าข้อมูลที่ได้รับมานั้นไม่มีการสูญหายเมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือโอเวอร์โฟลว โดยใช้วิธีการตรวยสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งข้อมูลมาให้

Break

ใช้ในการเซตและเคลียร์ค่าสัญญาณ Break ชนิดข้อมูลเป็นแบบ บูลีน

รูปแบบการใช้งาน

object.Break [= *value*]

โดย value เป็นค่าบูลีน ถ้า value=True หมายถึง การส่งสัญญาณ Break ออกไป ถ้า value=False หมายถึงการเคลียร์สัญญาณ Break

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดการส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น False

ตังอย่างเป็นวิธีการส่งัญญาณ Break ออกไปเป็นช่วงสั้นๆที่ 1/10 ของวินาที

```
' Set the Break condition.
MSComm1.Break = True
' Set duration to 1/10 second.
Duration! = Timer + .1
' Wait for the duration to pass.
Do Until Timer > Duration!
   Dummy = DoEvents()
Loop
' Clear the Break condition.
MSComm1.Break = False
```

เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าคุณสมบัติ CommEvent มีการเปลี่ยนแปลงเพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นทันทีทันใด หรือแสดงข้อผิดพลาดที่เกิดขึ้น

ตัวอย่างโปรกแรมย่อย OnComm เพื่อนำเหตุการณ์ CommEvent มาแสดง

```
Private Sub MSComm_OnComm ()
   Select Case MSComm1.CommEvent
   ' Handle each event or error by placing
   ' code below each case statement

   ' Errors
      Case comEventBreak   ' A Break was received.
      Case comEventFrame   ' Framing Error
      Case comEventOverrun   ' Data Lost.
      Case comEventRxOver   ' Receive buffer overflow.
      Case comEventRxParity   ' Parity Error.
      Case comEventTxFull   ' Transmit buffer full.
      Case comEventDCB   ' Unexpected error retrieving DCB]

   ' Events
      Case comEvCD   ' Change in the CD line.
      Case comEvCTS   ' Change in the CTS line.
      Case comEvDSR   ' Change in the DSR line.
      Case comEvRing   ' Change in the Ring Indicator.
      Case comEvReceive   ' Received RThreshold # of
                          ' chars.
      Case comEvSend   ' There are SThreshold number of
```

```
                    ' characters in the transmit
                    ' buffer.
      Case comEvEof   ' An EOF charater was found in
                    ' the input stream
   End Select
End Sub
```

ค่าคงที่สำหรับคุณสมบัติ Hnadshake

ค่าคงที่	ค่า	รายละเอียด
ComNone	0	ไม่ใช้การตรวจสอบแฮนด์เชก
ComXONXOff	1	ใช้การตรวจสอบแฮนด์เชกแบบ XOnXOff
ComRTS	2	ใช้การตรวจสอบแฮนด์เชกผ่านทางเข้า RTS และ CTS
ComRTSXONXOff	3	กำหนอการตรวจสอบแฮนด์เช็กทั้งแบบ RTS,CTS และ Xon/XOff

ค่าคงที่สำหรับคุณสมบัติ OnComm

ค่าคงที่	ค่า	รายละเอียด
CommEvsend	1	ส่งค่าเหตุการณ์(Send Event)
CommEvReceive	2	รับค่าเหตุการณ์(Receive Event)
CommEvCTS	3	มีการเปลี่ยนแปลงที่ขา CTS
CommEvDSR	4	มีการเปลี่ยนแปลงที่ขา DSR
CommEvCD	5	มีการเปลี่ยนแปลงที่ขา DCD
CommEvRing	6	ตรวจจับสัญญาณกระดิ่งโทรศัพท์
CommEvEOF	7	ตรวจพบตำแหน่งท้ายสุดของไฟล์(End Of File)

ค่าคงที่สำหรับคุณสมบัติ Error

ค่าคงที่	ค่า	รายละเอียด
CommEventBreak	1001	ได้รับสัญญาณ Break
CommEventCTSTO	1002	ขา CTS เกิดไทย์เอาต์
CommEventDSRTO	1003	ขา DSR เกิดไทย์เอาต์
CommEventFrame	1004	เกิดข้อผิดพลาดที่เฟรมข้อมูล(Framimg Error)
CommEventOverrun	1006	พอร์ตอนุกรมเกิดโอเวอร์โหลด(Port Overrun)
CommEventCDTO	1007	ขา DCD เกิดไทย์เอาต์
CommEventRxOver	1008	บัฟเฟอร์รับข้อมูลเกิดเวอร์โฟลว
CommEventRxParity	1009	เกิดข้อผิดพลาดที่พาริตี้(Parity Error)
CommEventTxFull	1010	บัฟเฟอร์ส่งข้อมูลเต็ม

บทที่ 3
การออกแบบและหลักการทำงาน

3.1 การออกแบบหุ่นยนต์

หุ่นยนต์ที่ใช้ในการทดลองนี้จะออกแบบให้เป็นแบบจำลอง Mobile Robot ขนาดเล็กเป็น โดยใช้ไมโครคอนโทรลเลอร์เบสิกสแตมป์ 2 เป็นตัวควบคุมการทำงานของหุ่นยนต์ทั้งหมด ระบบการขับเคลื่อนจะใช้สเต็ปเปอร์มอเตอร์ การติดต่อระหว่างหุ่นยนต์และคอมพิวเตอร์จะใช้การสื่อสารข้อมูลตามมาตรฐาน RS-232 ซึ่งลักษณะการส่งข้อมูลเป็นแบบอนุกรมแบบอะซิงโครนัสแบบฮาล์ฟดูเพล็กซ์(Half Duplex)คือผลัดกันส่งและรับ จะส่งและรับพร้อมกันไม่ได้

การออกแบบหุ่นยนต์แบ่งเป็น การออกแบบทางแมคคานิกส์ และระบบอิเล็กทรอนิกส์ โครงการนี้จะเน้นทางระบบอิเล็กทรอนิกส์เป็นหลักส่วนทางแมคคานิกส์จะใช้สเต็ปเปอร์มอเตอร์ต่อเข้ากับล้อโดยตรงเพื่อไม่ให้เกิดความยุ่งยาก ซึ่งจะทำให้บรรทุกของได้ไม่มากนัก

ลักษณะของหุ่นยนต์ที่ใช้ในการทดลองดังนี้

1.สามารถเลี้ยวรอบตัวสามารถเลี้ยวได้ครั้งละ 1 องศา โดยเลี้ยวซ้ายได้ 180 องศาและเลี้ยวขวาได้ 180 องศา

2.มีแบตเตอร์รี่ภายในตัวของหุ่นยนต์เอง และมีลำโพงเป็น Monitor เพื่อแสดงการทำงานของโปรแกรมควบคุมหุ่นยนต์

3.มีวงจรตรวจจับความต่างสี(Tracking line on the floor) ไว้ตรวจสอบความผิดพลาดจากการเคลื่อนที่ของหุ่นยนต์ ทำให้การเคลื่อนที่ของหุ่นยนต์เป็นไปอย่างถูกต้อง

การสร้างหุ่นยนต์ต้นแบบให้มีมุมเลี้ยวได้ 1 องศา ทำได้โดยกำหนดให้การเลี้ยวสเต็ปเปอร์มอเตอร์สองตัวจะหมุนพร้อมกันซึ่งจะทำให้ตำแหน่งของหุ่นยนต์ไม่เปลี่ยนแปลง ระยะทางที่ได้จากการเลี้ยวรอบตัวเท่ากับเส้นรอบวงที่มีระยะห่างระหว่างล้อทั้งสองเป็นเส้นผ่านศูนย์กลาง การเลี้ยวรอบตัวต้องสั่งให้มอเตอร์หมุน 360 สเต็ป ระยะทางที่ได้จากการเลี้ยวรอบตัวเท่ากับ360 คูณระยะที่มอเตอร์หมุน 1 สเต็ป และระยะที่มอเตอร์หมุน 1 สเต็ปเท่ากับเส้นรอบวงล้อของหุ่นยนต์หารจำนวนสเต็ปของมอเตอร์ ดังนั้น

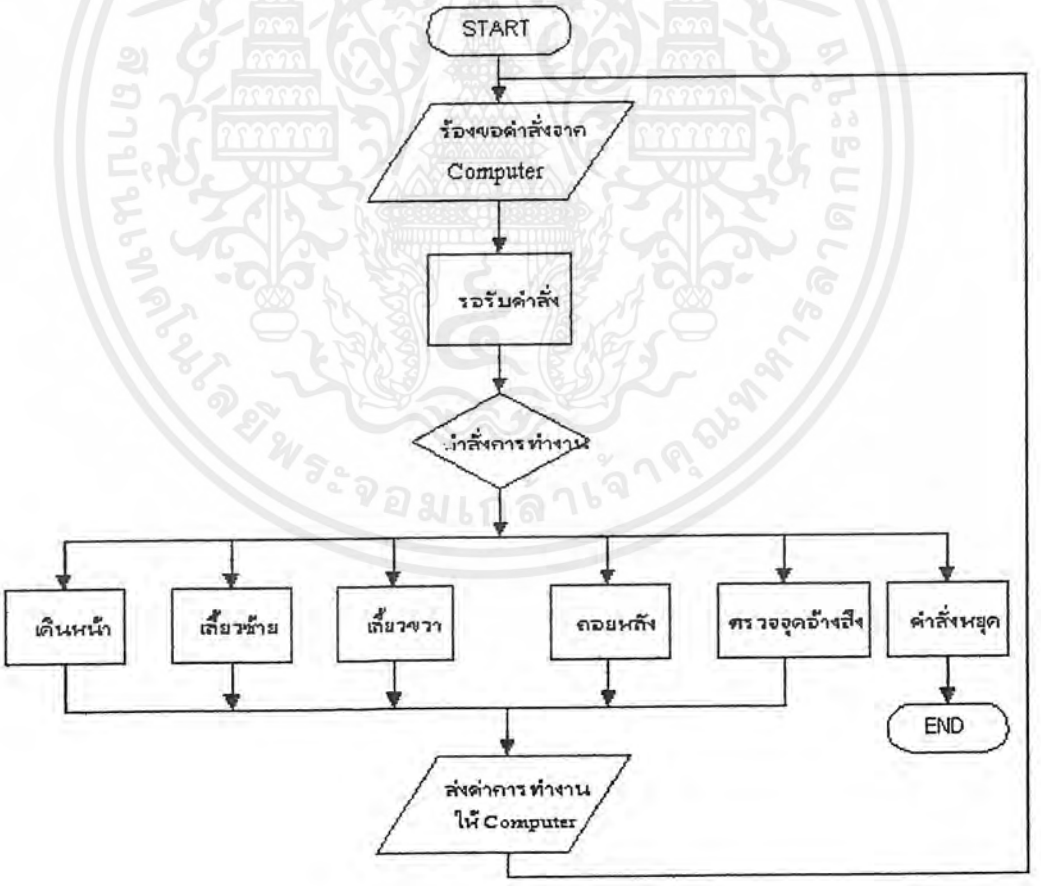
ระยะห่างระหว่างล้อทั้งสอง = ระยะทางที่ได้จากการเลี้ยวรอบตัว / π

=(360*เส้นรอบวงล้อ/จำนวนสเต็ปของมอเตอร์)/ π

=360*เส้นผ่าศูนย์กลางล้อ/จำนวนสเต็ปของมอเตอร์

หุ่นยนต์ในการทดลองใช้สเต็ปเปอร์มอเตอร์ที่ใช้มีจำนวนสเต็ป 200 สเต็ป/รอบ และใช้ล้อที่มีเส้นผ่านศูนย์กลาง 7.5 เซนติเมตร ระยะทางที่ได้จากการหมุน 1 สเต็ป $(7.5x\pi)/200$ ซึ่งเท่ากับ 0.1778 เซนติเมตร ระยะทางที่ได้จากการเลี้ยวรอบตัวเท่ากับ 42.141 เซนติเมตร ดังนั้นระยะห่างระหว่างล้อทั้งสองเท่ากับ 13.5 เซนติเมตร

หุ่นยนต์จะทำงานตามคำสั่งที่สั่งมาจากสถานีควบคุม (Central Control Station) คือ คอมพิวเตอร์ หุ่นยนต์ตัวนี้ขอเรียกกว่าหุ่นยนต์ a เพราะว่าการควบคุมหุ่นยนต์ตัวนี้ การรับและส่งข้อมูลในการควบคุมจะใช้รหัส ASCII 97 หรือ "a" นำหน้าเพื่อไม่ให้ปะปนกับสัญญาณตัวอื่นๆ เวลาพัฒนาให้ใช้งานแบบร่วมกันหลายๆตัว โดยเริ่มแรกหุ่นยนต์จะส่งรหัสเพื่อร้องขอคำสั่งการทำงานจากสถานีควบคุม เสร็จแล้วหุ่นยนต์จะรอจนมีคำสั่งการทำงานจากสถานีควบคุม เมื่อสถานีควบคุมรับทราบรหัสการร้องขอก็จะส่งคำสั่งการทำงานมาให้หุ่นยนต์ เมื่อหุ่นยนต์ได้รับแล้วก็จะทำตามคำสั่งนั้น แล้วก็ส่งรหัสข้อมูลว่าหุ่นยนต์ได้ทำงานอย่างไหนเสร็จแล้วเพื่อให้สถานีควบคุมนำข้อมูลของการทำงานหุ่นยนต์ไปแสดงผลที่สถานีควบคุมต่อไป ดังโฟลว์ชาร์ตแสดงการทำงานในรูปที่ 3.1



รูปที่ 3.1 แสดงโฟลว์ชาร์ตการทำงานของหุ่นยนต์

ส่วนประกอบทางอิเล็กทรอนิกส์ของหุ่นยนต์แบ่งเป็น

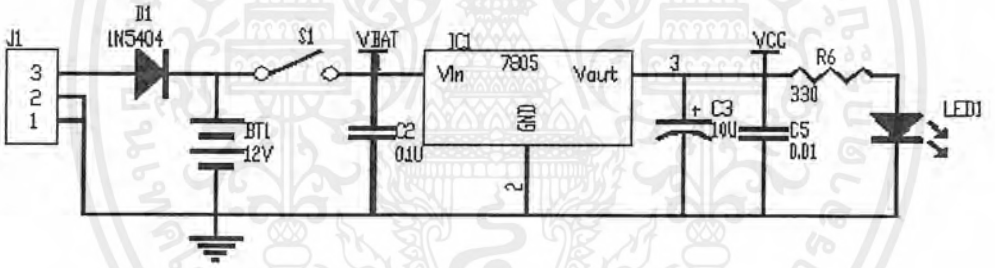
1.วงจรควบคุม (Control circuit)

2. วงจรขับเคลื่อน (Driver circuit)

3.วงจรตรวจจับความต่างสี (Tracking line on the floor circuit)

วงจรควบคุม (Control circuit)

เป็นวงจรที่ควบคุมการทำงานของหุ่นยนต์ทั้งหมดโดยวงจรควบคุมโดยอาศัยไมโครคอนโทรเลอร์เป็นตัวบังคับสั่งงาน ไมโครคอนโทรเลอร์ที่ใช้ในหุ่นยนต์นี้คือ เบสิกสแตมป์ 2 ทำหน้าที่รับข้อมูลจากคอมพิวเตอร์แล้วนำคำสั่งที่ได้มาทำงาน ส่งข้อมูลการทำงานของหุ่นยนต์ให้คอมพิวเตอร์ เพื่อนำไปแสดงผลการทำงานและแสดงตำแหน่งของหุ่นยนต์พร้อมกับตรวจสอบข้อผิดพลาดตำแหน่งของหุ่นยนต์จาการทำงานโดยการใช้วงจรตรวจจับตามสีพื้นและส่งข้อมูลให้คอมพิวเตอร์ การแสดงผลข้อมูลต่างๆของหุ่นยนต์จะส่งผ่าน LED Monitor และแสดงผลทางเสียงผ่านทางลำโพงเปียร์โซ ดังรูปที่ 3.2และรูปที่ 3.3



รูปที่ 3.2 แสดงแหล่งจ่ายไฟของวงจรควบคุม

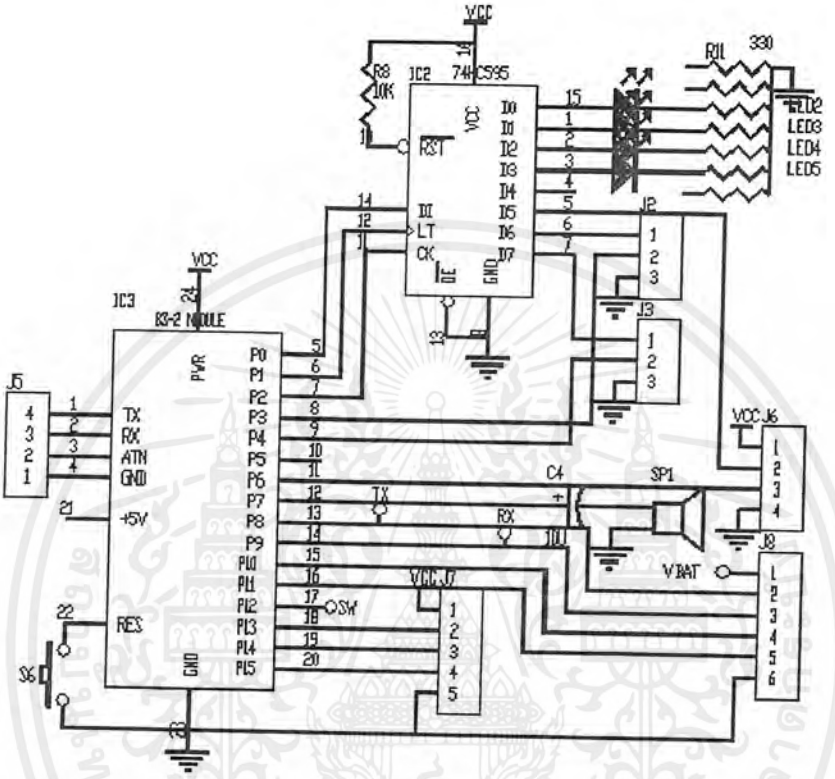
วงจรประกอบไปด้วย IC 1 เป็นไอซีรักษาระดับแรงดัน 5V เพื่อจ่ายให้วงจรควบคุมทั้งหมด โดยคอนเน็คเตอร์ J1 เป็นตัวรับแหล่งจ่ายไฟจากภายนอกเพื่อชาร์ทแบตเตอร์รี่และจ่ายไฟให้วงจร (กรณีชาร์จแบตเตอรี่เพียงอย่างเดียวให้ off สวิตช์ S1) IC # 74HC579 ซึ่งเป็นไอซีชิฟต์รีจิสเตอร์แบบ SIPO(Serial In Parallel Out) เป็นตัวรับข้อมูลอนุกรมแบบ synchronous จาก เบสิกสแตมป์ 2 แล้วเปลี่ยนข้อมูลขนานเพื่อไปขับ LED แสดงผลทั้ง 4 ตัวและไปควบคุมการทำงานของวงจรตรวจจับความต่างสี ส่วน ส่วน J5 เมื่อต่อเข้ากับคอมพิวเตอร์เพื่อโปรแกรมการทำงานของเบสิกสแตมป์ 2 ผ่านทางคอมพิวเตอร์อนุกรมคอมพิวเตอร์

J2 และ J3 ต่อเข้ากับส่วนจับความต่างสี

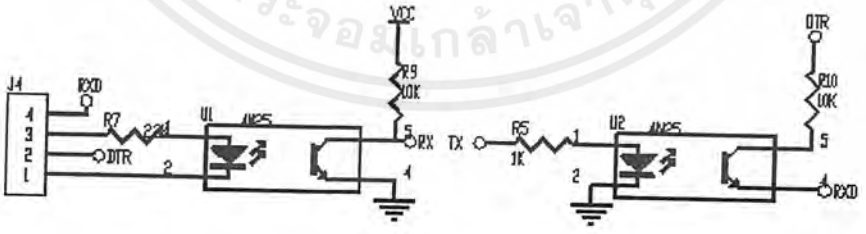
J4 เป็นตัวสื่อสารระหว่างหุ่นยนต์กับคอมพิวเตอร์โดยใช้สายผ่านพอร์ต RS-232 Serial port

J7 ต่อเข้ากับภาคขับเคลื่อนมอเตอร์ โดยการส่งข้อมูลอนุกรมแบบ synchronous โดยมีออปโต้คัปเปอร์ (Opto couple) ป้องกันไม่ให้สัญญาณรบกวนจากคอมพิวเตอร์เข้าไปยังหุ่นยนต์ได้

P12เป็นตัวตรวจสอบสวิตช์อินพุตทั้ง 4 ตัว เพื่อทดสอบและปรับแต่งการทำงานของโปรแกรมและส่วนของวงจรตรวจจับต่างๆที่ P7 เป็นภาคแสดงผลทางเสียงโดยใช้ลำโพงเปียร์โซ



รูปที่ 3.3 แสดงวงจรควบคุม

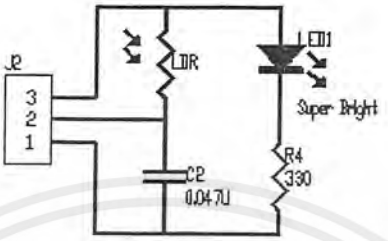


รูปที่ 3.4 แสดงวงจรเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

วงจรตรวจจับความต่างสี(Tracking line on the floor circuit)

วงจรประกอบด้วย LED 1 ซึ่งเป็นแบบ ซูปเปอร์ไบต์ LED แบบนี้จะให้ระดับแสงสว่างกว่า LED ปกติ ทำหน้าที่กำเนิดลำแสงความเข้มสูงไปตกกระทบพื้น โดยใช้ LDR เป็นตัวตรวจจับการสะท้อนแสงของ LED โดยพื้นสีขาวจะสะท้อนแสงได้ดีกว่าพื้นสีดำ

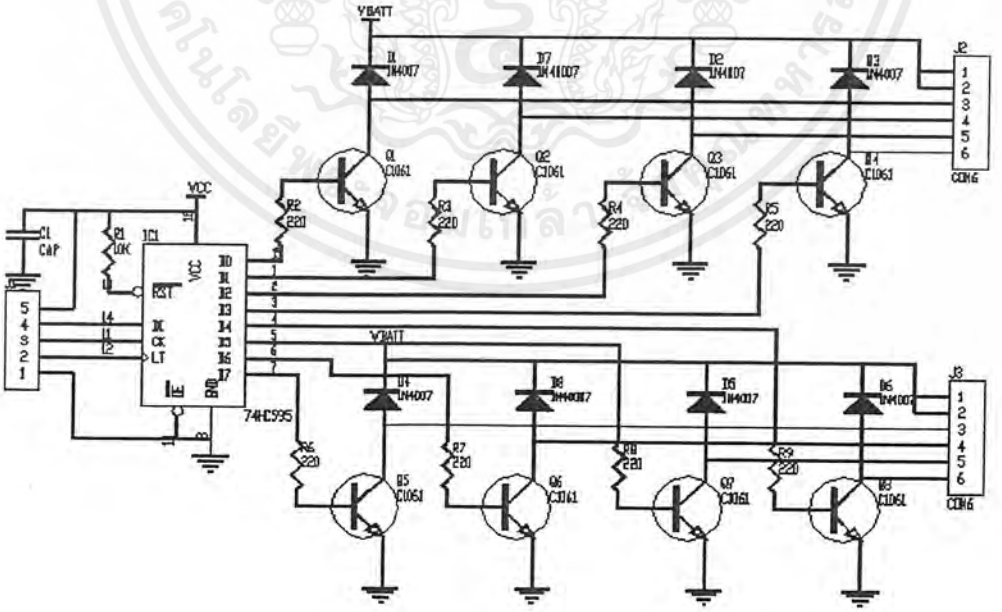
J2 ขา 3 จะต่อกับ +5V ขา 2 จะต่อกับเบสิกสแตมป์ และขา 1 ต่อกับกราวด์



รูปที่ 3.5 แสดงวงจรวงจรตรวจจับความต่างสี

วงจรขับสเต็ปเปอร์มอเตอร์

IC 1 เบอร์ 74HC595 จะทำหน้าที่รับข้อมูลอนุกรมแบบ Synchronous จากวงจรควบคุมทาง J1 เปลี่ยนเป็นข้อมูลแบบขนานเพื่อไปขับให้ทราซิสเตอร์ทำงานเพื่อขับจากสเต็ปเปอร์มอเตอร์ทาง J2 และ J3 เนื่องจากสเต็ปเปอร์มอเตอร์ใช้กระแสในการทำงานสูงการออกแบบได้ใช้ทรานซิสเตอร์เบอร์ 2SC1061 ซึ่งสามารถขับกระแสได้ถึง 7 A



รูปที่ 3.6 แสดงวงจรวงจรขับสเต็ปเปอร์มอเตอร์

3.2 การออกแบบโปรแกรมควบคุมหุ่นยนต์

การสื่อสารข้อมูลระหว่างหุ่นยนต์กับสถานีควบคุมจะส่งข้อมูลที่ละ 2 ไบต์ โดยไบต์แรกเป็นรหัสเริ่มต้นเพื่อป้องกันสัญญาณอื่นๆที่ไม่ต้องการเข้ามาปะปน ไบต์ 2 เป็นรหัสข้อมูลคำสั่งโดยโปรแกรมควบคุมจะใช้รหัสคำสั่งดังนี้

รหัสจากคอมพิวเตอร์ไปสู่หุ่นยนต์

1. "af" คำสั่งเดินหน้า 1 สเต็ป
2. "al" คำสั่งถอยหลัง 1 สเต็ป
3. "ar" คำสั่งเลี้ยวขวา 1 สเต็ป
4. "ab" คำสั่งเลี้ยวซ้าย 1 สเต็ป

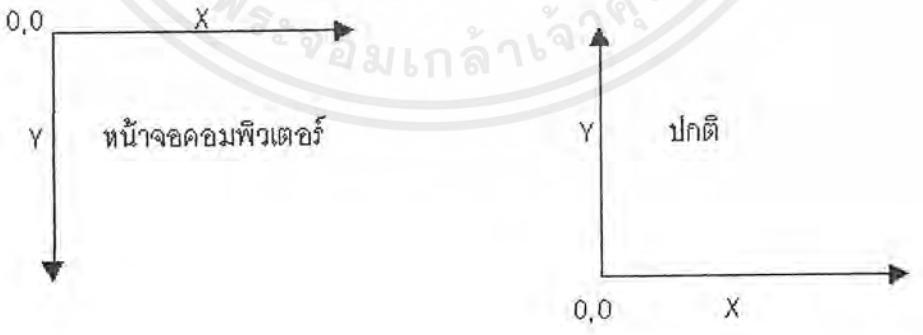
รหัสจากหุ่นยนต์ไปสู่คอมพิวเตอร์

1. "af" คำสั่งแสดงว่าหุ่นยนต์เดินหน้า 1 สเต็ป
2. "ab" คำสั่งแสดงว่าหุ่นยนต์ถอยหลัง 1 สเต็ป
3. "ar" คำสั่งแสดงว่าหุ่นยนต์เลี้ยวขวา 1 สเต็ป
4. "al" คำสั่งแสดงว่าหุ่นยนต์เลี้ยวซ้าย 1 สเต็ป

นอกจากนี้จะสามารถเพิ่มคำสั่งอื่นๆได้ถึง 256 คำสั่ง

หลักการโปรแกรมเส้นทางของหุ่นยนต์โดยการวาด

เมื่อมีการเคลื่อนเมาส์(MoveMouse)ในหน้าจอคอมพิวเตอร์ณ.ตำแหน่งใดๆจะได้ค่าตำแหน่งอยู่ 2 ค่าคือ ค่า Xและค่า Y โดยค่า X และ Y =0 จะเริ่มจากมุมซ้ายบนของหน้าจอ ซึ่งมีลักษณะดังรูป จะเห็นว่าค่าจะตรงกันข้ามกับพิกัดปกติ



รูปที่ 3.7 แสดงพิกัดบนหน้าจอคอมพิวเตอร์และพิกัดปกติ

เมื่อมีการคลิกเมาส์ด้านซ้ายจะได้ค่า X และ Y หนึ่งค่าซึ่งกำหนดให้เป็นจุดเริ่มต้น และเมื่อเคลื่อนเมาส์ไปคลิกจุดต่อไปก็จะได้ค่า X และ Y เป็นจุดที่ 2 การหาค่าระยะทาง(Distance)จากทั้ง 2 จุด ทำได้โดยใช้สูตร

$$Dis\tan ce = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

โดย Distance เป็นค่าระยะทาง

X_1, Y_1 เป็นตำแหน่งจุดที่ 1

X_2, Y_2 เป็นตำแหน่งจุดที่ 2

การหาทิศทางระหว่างจุดทั้งสอง(Direction)

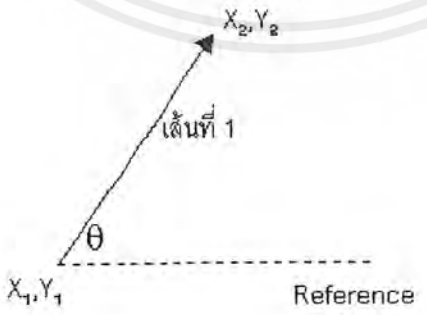
เราจะกำหนดให้ทิศทางเส้นที่พุ่งขนานแกน X จากซ้ายไปขวาเป็นทิศทางอ้างอิง(Direction Reference)เป็น 0 องศาและมุมจะหมุนทวนเข็มนาฬิกามุมขนานแกน Y ไปด้านบนเป็น 90 องศา ไปด้านล่างเป็น 270 องศา ค่าของทิศทางระหว่างจุดทั้งสองหาได้ดังสูตร

$$Direction(\theta) = \tan^{-1}\frac{Y_2 - Y_1}{X_2 - X_1}$$

โดย Direction เป็นค่าทิศทางระหว่างจุดทั้งสอง

X_1, Y_1 เป็นตำแหน่งจุดที่ 1

X_2, Y_2 เป็นตำแหน่งจุดที่ 2



รูปที่ 3.8 แสดงการทิศทางระหว่างจุดทั้งสอง

การหามุมเลี้ยว(Turn)ระหว่างสองเส้น

โดยการนำทิศทางที่ 2 ลบด้วยเส้นทางที่ 1 โดยกำหนดให้การเลี้ยวซ้ายเป็นค่าบวก และการเลี้ยวขวาเป็นค่าลบ เนื่องจากการทำงานเป็นวงรอบมุมเลี้ยวจะมีค่าไม่เกิน 180 องศา การหามุมเลี้ยวหาได้ดังสูตร

$$Turn = Direction\,2 - Direction\,1$$

โดย Turn เป็นมุมเลี้ยว

Direction1 เป็นทิศทางของเส้นที่ 1

Direction2 เป็นทิศทางของเส้นที่ 2

รูปที่ 3.9 แสดงทิศทางระหว่างเส้นทางเดินของหุ่นยนต์

ลักษณะโปรมแกรมที่ใช้ควบคุม

โปรแกรมที่ใช้ควบคุมหุ่นยนต์ แสดงการทำงานของหุ่นยนต์เขียนขึ้นด้วยโปรแกรมวิชวลเบสิก 6 ของไมโครซอฟท์ซึ่งเป็นโปรมแกรมที่ทำงานบนระบบปฏิบัติการวินโดส์ การทำงานของโปรแกรมนี้จะประกอบด้วยฟอร์มแม่และฟอร์มลูกหลายๆฟอร์มที่เรียกว่า Multiple Document Interface โดยฟอร์มคือสิ่งที่ปรากฏบนจอคอมพิวเตอร์เพื่อติดต่อระหว่างผู้ใช้งานกับคอมพิวเตอร์ ซึ่ง ในโปรแกรมนี้จะประกอบไปด้วยฟอร์ม(Form)จำนวน 5 ฟอร์ม คือ

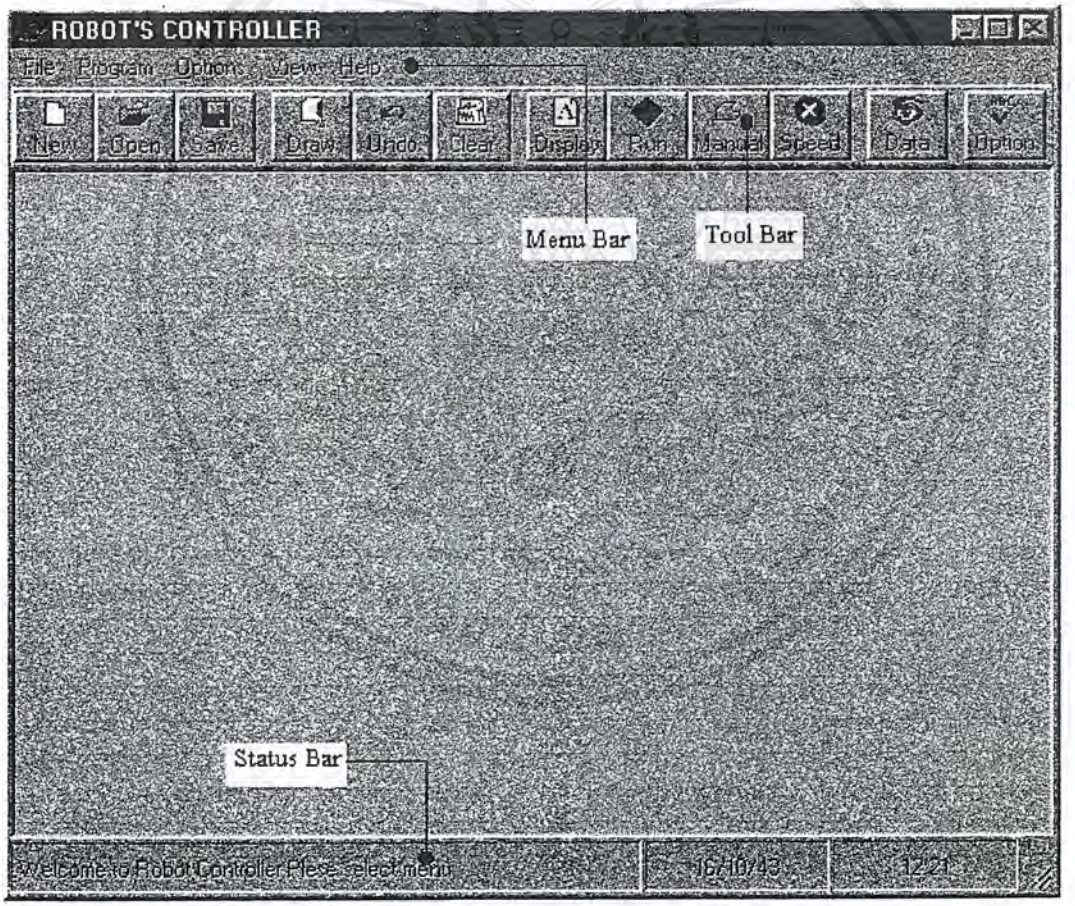
1.ฟอร์มหลัก(FormMain)เป็นฟอร์มแม่ โดยเมื่อโปรแกรมทำงานฟอร์มนี้จะแสดงที่หน้าจอคอมพิวเตอร์ตลอดเมื่อปิดฟอร์มนี้ฟอร์มลูกก็จะถูกปิดไปด้วย

2.ฟอร์มวาด(Formdraw)เป็นฟอร์มลูกที่ใช้ในการโปรแกรมการเคลื่อนที่ของหุ่นยนต์โดยการใช้ Mouse คลิกเลือกเส้นทางการเคลื่อนที่

3.ฟอร์มแสดงผล(Formdisplay)เป็นฟอร์มลูกที่ใช้ควบคุมการเคลื่อนที่ของหุ่นยนต์และแสดงผลการทำงานและตำแหน่งของหุ่นยนต์

4.ฟอร์มข้อมูล(Formdata)เป็นฟอร์มลูกที่ใช้แสดงข้อมูลโปรแกรมลักษณะการเคลื่อนที่ของหุ่นยนต์ เช่น ระยะทาง มุมเลี้ยว เป็นต้น

5.ฟอร์มออฟชั่น(FormOption)เป็นฟอร์มที่ใช้ปรับแต่งฟังก์ชั่นของโปรแกรม เช่น ขนาดของฟอร์มแสดงผล เลือกพอร์ตสื่อสาร เป็นต้น



รูปที่ 3.10 แสดงส่วนประกอบต่างๆของฟอร์มหลัก

ส่วนประกอบของฟอร์มหลัก(MainForm)

จะประกอบไปด้วยส่วนต่างๆดังต่อไปนี้

1.Menu bar จะรวมสิ่งต่างๆที่ใช้งาน เช่น การเปิดฟอร์มวาด การเปิดฟอร์มแสดงผล

2.Tool bar เป็นปุ่มต่างๆทำหน้าที่เหมือนMenu bar จะให้ความสะดวกในคำสั่งที่ใช้งานบ่อย เช่น การสั่งให้หุ่นยนต์ทำงาน การหยุดการทำงานของหุ่นยนต์ชั่วขณะ การปรับความเร็วในการเคลื่อนที่ของหุ่นยนต์

3.Status bar เป็นบรรทัดแสดงการทำงานของหุ่นยนต์และตำแหน่งของหุ่นยนต์ เช่น เมื่อใช้ฟอร์มในการโปรแกรมการเคลื่อนที่ของหุ่นยนต์จะแสดงตำแหน่ง X,Y ระยะทางและมุมเลี้ยว

การทำงานของฟอร์มวาด

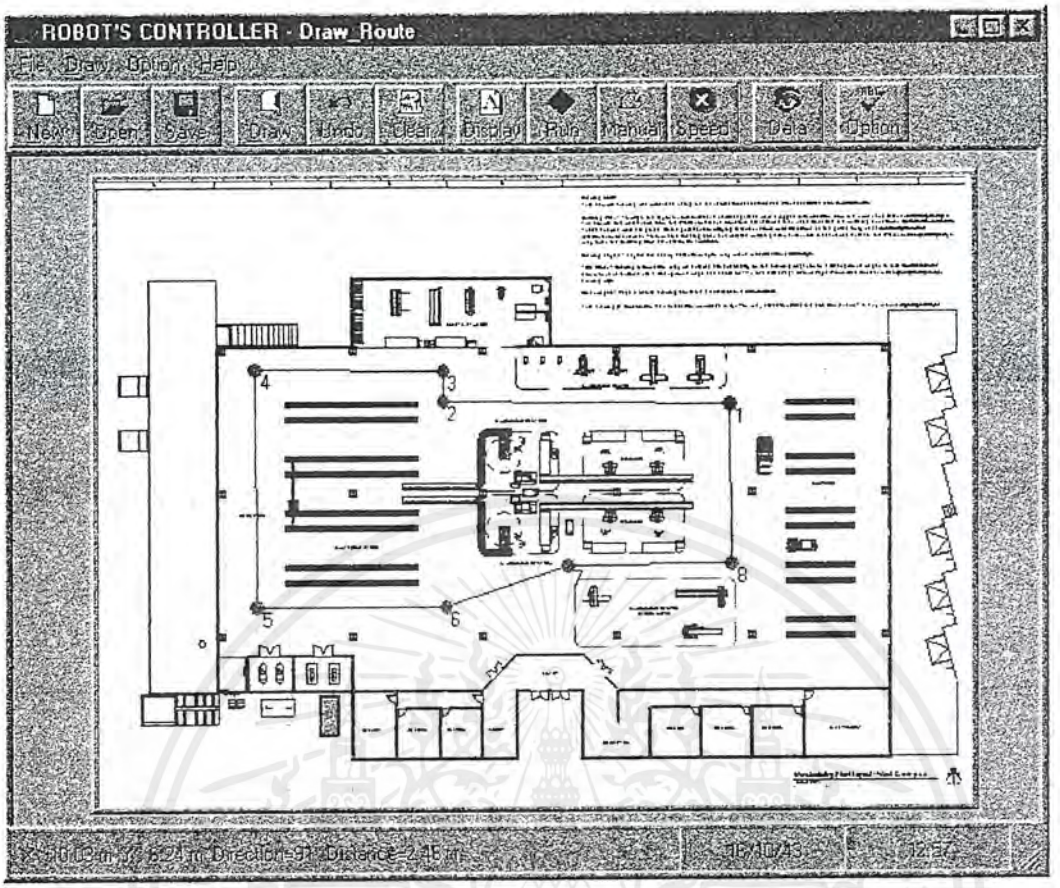
ฟอร์มวาดเป็นฟอร์มที่ใช้โปรแกมการเคลื่อนที่ของหุ่นยนต์ โดยการใช้เมาส์คลิกวาดเส้นบนฟอร์มโดยหุ่นยนต์จะเคลื่อนที่ตามเส้นทางที่เรากำหนดบนฟอร์ม โดยนำค่าระยะทางของฟอร์มวาดนำไปคูณกับมาตราส่วนที่เราตั้งไว้ ในฟอร์มออฟชั่น จะได้ระยะทางที่หุ่นยนต์เคลื่อนที่ได้จริง เมื่อได้ระยะทางการเคลื่อนที่ที่มีหน่วยจริงแล้วจึงนำมาหาค่าสเต็ปในการหมุนของสเต็ปเปอร์มอเตอร์โดยนำระยะทางที่ได้มาหารด้วยระยะทางที่มอเตอร์เคลื่อนที่ 1 สเต็ป ซึ่งเกิดจากเส้นผ่าศูนย์กลางของล้อหารด้วยจำนวนสเต็ปการหมุนของมอเตอร์ต่อ 1 รอบ เช่น ในระยะทางการเคลื่อนที่จริงเป็น 1 เมตร สเต็ปเปอร์มอเตอร์มีขนาด 1.8 องศา ล้อของหุนยนต์มีเส้นผ่านศูนย์กลาง 7.5 เซนติเมตร สเต็ปมอเตอร์จะต้องหมุนเท่าไร

ก่อนอื่นเราต้องหาจำนวนสเต็ปต่อการหมุนต่อ 1 รอบของมอเตอร์ในตัวอย่างนี้สเต็ปมอเตอร์มีขนาด 1.8 องศาแสดงว่ามอเตอร์มีจำนวนสเต็ปต่อรอบเท่ากับ การหมุน 1 รอบเท่ากับ 360 องศา จำนวนสเต็ปต่อรอบเป็น 360หารด้วย 1.8 เท่ากับ 200 จากนั้นระยะทางที่เคลื่อนที่ได้ต่อ 1 สเต็ปจากระยะทางที่ล้อเคลื่อนที่ได้ 1 สเต็ป เท่ากับเส้นผ่านศูนย์กลางของล้อคูณ π หารด้วยจำนวนสเต็ปต่อรอบซึ่งเท่ากับ 7.5 คูณกับ3.14 หารด้วย 200 เท่ากับ 0.1778 เซนติเมตร

ดังนั้นเราจะได้จำนวนสเต็ปทั้งหมดคือ 100หารด้วย 0.1778 เท่ากับ 562 สเต็ป

การเลี้ยวของหุ่นยนต์

เมื่อได้ค่ามุมเลี้ยวก็สามารถสั่งให้ สเต็ปเปอร์มอเตอร์หมุนเท่ากับมุมเลี้ยวได้เลย เพราะหุ่นยนต์ออกแบบให้สามารถมีมุมในการเลี้ยวได้รอบตัวเท่ากับ 360 องศา เนื่องจากมุมเลี้ยวที่ได้จะมีค่าไม่เกิน 180 องศา เพราะฉนั้นเครื่องหมายหน้ามุมเลี้ยวจะเป็นการกำหนดว่าหุ่นยนต์จะทำการเลี้ยวซ้ายหรือเลี้ยวขวา โดยกำหนดให้ค่าเป็นบวกให้ทำการเลี้ยวซ้ายและถ้าค่าเป็นลบให้ทำการเลี้ยวขวา



รูปที่ 3.11 แสดงการวาดเส้นทางเดินของหุ่นยนต์บนจอกับแผนที่จริง

การควบคุมการทำงานและแสดงตำแหน่งของหุ่นยนต์

การส่งสัญญาณระหว่างหุ่นยนต์และโปรแกรมควบคุมเป็นการส่งสัญญาณแบบตอบรับ HandShaking คือ หุ่นยนต์จะทำงานเมื่อมีคำสั่งจากโปรแกรมและส่งข้อมูลการทำงานของหุ่นยนต์ออกไปโปรแกรมจะตอบรับข้อมูลการทำงานของหุ่นยนต์ก่อนส่งให้หุ่นยนต์ทำงาน

การแสดงผลการทำงานบนจอคอมพิวเตอร์โปรแกรมจะคอยรับข้อมูลจากหุ่นยนต์ที่ส่งมา

เมื่อหุ่นยนต์เดินหน้า 1 Step โปรแกรมก็จะคำนวณตำแหน่งของหุ่นยนต์ ดังนี้

X = X+cos (Direction)*Step/Scale

Y = Y-sin(Direction)*Step/Scale

เมื่อหุ่นยนต์ถอยหลัง 1 Step โปรแกรมก็จะคำนวณตำแหน่งของหุ่นยนต์ ดังนี้

X = X-cos (Direction)*Step/Scale

Y = Y+sin(Direction)*Step/Scale

เมื่อหุ่นยนต์เลี้ยวซ้ายหรือเลี้ยวขวา 1 สเต็ปโปรแกรมก็จะคำนวณทิศทางการเคลื่อนที่ของหุ่นยนต์ทันทีเช่นกัน คือ

ถ้าหุ่นยนต์เลี้ยวซ้าย

Direction = Direction+1

ถ้าหุ่นยนต์เลี้ยวขวา

Direction = Direction-1

โดย X,Y คือ ตำแหน่งของหุ่นยนต์บนฟอร์มแสดงผล

Direction คือ ทิศทางของหุ่นยนต์ขณะนั้น

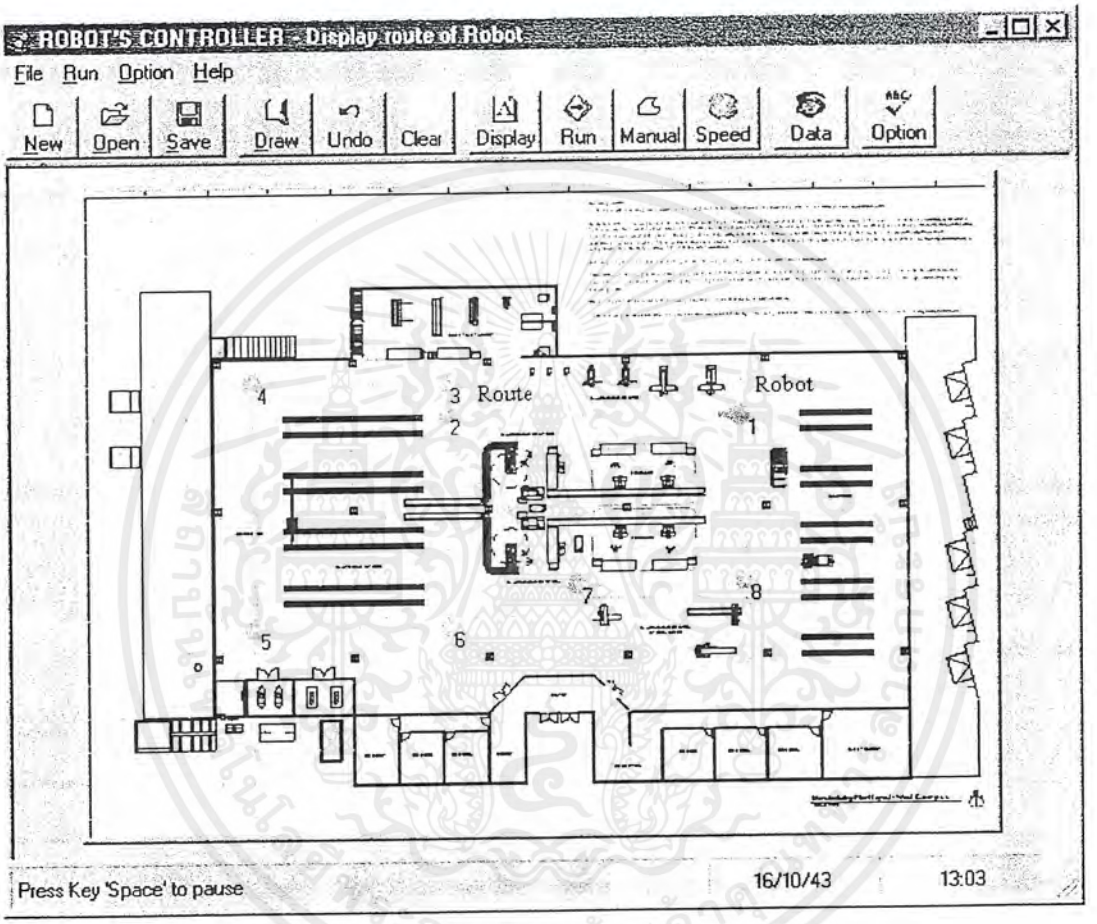
Step คือ ระยะทางที่จริงที่หุ่นยนต์เคลื่อนที่ได้ 1 สเต็ป

Scale คือ ค่ามาตราส่วนของแผนที่

รูปที่ 3.12 แสดงลักษณะการทำงานของหุ่นยนต์

การทำงานฟอร์มแสดงผล

ฟอร์มแสดงผลที่ใช้ควบคุมการเคลื่อนที่ของหุ่นยนต์และแสดงผลการทำงานและตำแหน่งของหุ่นยนต์ โดยฟอร์มแสดงผลรับข้อมูลจากฟอร์มวาด แล้วนำข้อมูลที่ได้สั่งให้หุ่นยนต์ทำงานโดยจะส่งข้อมูลทางพอร์ตอนุกรม RS-232โดยใช้ฟังก์ชั่น MSComm ของวิชวลเบสิก และรับข้อมูลจากหุ่นยนต์โดยแสดงผลที่หน้าจอ



รูปที่ 3.13 แสดงการเดินของหุ่นยนต์บนจอกับแผนที่จริง

บทที่ 4
สรุปผลและวิจารณ์

สรุปผลการทำงานของหุ่นยนต์และโปรแกรมควบคุมการทำงาน

1.การเลี้ยวของหุ่นยนต์ซึ่งออกแบบให้มีความละเอียดในการเลี้ยวรอบตัว 360 องศา ยังมีข้อผิดพลาด(error)เล็กน้อย ซึ่งสาเหตุเกิดจากระยะห่างระหว่างล้อไม่ถูกต้องเนื่องจากขาดเครื่องมือวัดที่มีความเที่ยงตรงและแน่นอนในการสร้าง

2.ระยะทางที่หุ่นยนต์เคลื่อนที่ยังเกิด error ขึ้นเล็กน้อย เนื่องจากโปรแกรมซึ่งคำนวนระยะการเคลื่อนที่ของหุ่นยนต์ซึ่งนำค่าระยะทางจริงมาหารกับระยะทางที่หุ่นยนต์เคลื่อนที่ได้ 1 สเต็ป มีการปัดทศนิยมเป็นจำนวนเต็มและการคำนวนหามุมเลี้ยวก็มีการปัดขึ้นเป็นจำนวนเต็มเช่นเดียวกัน ถึงแม้ว่าเราจะใช้สเต็ปเปอร์มอเตอร์ที่มีจำนวนสเต็ปมาก ซึ่งจะทำให้ระยะทางการเคลื่อนที่ต่อ1 สเต็ปน้อยลง แต่เมื่อหุ่นยนต์เคลื่อนที่ซ้ำๆกันหลายรอบ จึงมี error เกิดขึ้น

การแก้ไขข้อผิดพลาดนี้จะทำโดยนำวงจรตรวจจับความต่างสี(Tracking line on the floor) มาเช็คจุดสีที่พื้นที่จุดเริ่มต้นการเคลื่อนที่ของหุ่นยนต์ นำมาเป็นจุดอ้างอิง (reference) ถ้าไม่ตรงจุดจะทำให้ทราบว่ามีการเกิด error ขึ้น

3.การส่งข้อมูลเพื่อแสดงพิกัดตำแหน่ง ซึ่งทำการส่งทุก 1 สเต็ปการเคลื่อนที่ของหุ่นยนต์จะทำให้การเกิด error ของการเคลื่อนที่ลดลงแต่การรับส่งข้อมูลทุกๆ 1 สเต็ป จะทำไม่สามารถเพิ่มความเร็วในการเคลื่อนที่เพราะต้องเสียเวลาในการรับส่งข้อมูล ดังนั้นถ้าต้องการความเร็วในการเคลื่อนที่สูงต้องเพิ่มอัตราการรับส่งข้อมูล(baud rate)ให้สูงขึ้น

4.เมื่อหุ่นยนต์หยุดทำงานจำเป็นต้องจ่ายกระแสให้สเต็ปเปอร์มอเตอร์ตลอดเพื่อรักษาตำแหน่งของหุ่นยนต์ให้คงที่ เพราะว่าถ้าไม่มีกระแสให้สเต็ปเปอร์มอเตอร์เมื่อหุ่นยนต์เริ่มเคลื่อนที่ทิศทางของหุ่นยนต์จะเปลี่ยนไป ซึ่งการจ่ายกระแสให้สเต็ปเปอร์มอเตอร์ตอนหุ่นยนต์หยุดทำงานจะกินกระแสมากกว่าตอนทำงาน

5.สเต็ปเปอร์มอเตอร์สามารถกำหนดตำแหน่งการหมุนได้อย่างแม่นยำโดยไม่ต้องใช้วงจรป้อนกลับ แต่ข้อเสียของมอเตอร์นี้คือ มีขนานใหญ่ น้ำหนักมาก แรงบิดน้อย จะใช้พลังงานมากกรณีต้องการแรงบิดสูง

6. ค่า Turn error เกิดจากการสร้างหุ่นยนต์ทางแมคคานิกส์ไม่ได้ตามที่ออกแบบไว้ ซึ่งทำให้มุมเลี้ยวผิดพลาดไม่ได้ตามที่กำหนดซึ่งเป็นค่าที่เราไม่ต้องการ ค่า Turn error หาได้จากจำนวนสเต็ปในการหมุนของหุ่นยนต์จริงหารด้วยจำนวนสเต็มที่โปรแกรมคำนวนออกมา ซึ่งเราต้องการให้ค่าทั้งสองนี้เท่ากัน คือ ค่า Turn error เท่ากับ 1 จะทำให้การเลี้ยวไม่มีข้อผิดพลาด

แนวทางการพัฒนาโครงงาน

แนวทางการพัฒนาโปรแกรมควบคุม

1.ควรพัฒนาให้โปรแกรมสามารถควบคุมให้หุ่นยนต์ได้อย่างน้อย 2 ตัวขึ้นไป เพราะในระบบการขนส่งจริงจะใช้หุ่นยนต์หลายตัวทำงานร่วมกันซึ่งจะทำให้ระบบมีประสิทธิภาพและทำให้เกิดความรวดเร็ว

2. ควรพัฒนาโปรแกรมให้หาระยะทางการเคลื่อนที่ของหุ่นยนต์จากแผนที่ของระบบขนส่งจริง โดยมีการกำหนดมาตราส่วนและสัญลักษณ์ เช่น รูปร่าง ลักษณะ สี เป็นต้น ทางแผนที่

แนวทางการพัฒนาหุ่นยนต์

1.ควรพัฒนาให้หุ่นยนต์หาค่าพิกัดตำแหน่งและทิศทางด้วยตนเองเพื่อให้สถานีควบคุมสามารถทำงานได้รวดเร็วขึ้น และแก้ไขพิกัดตำแหน่งตัวเองกรณีเดินออกนอกเส้นทาง(เกิด error)

2. ควรพัฒนาหุ่นยนต์ให้สามารถเคลื่อนที่ในทางขรุขระได้

3. ควรพัฒนาให้หุ่นยนต์รับน้ำหนักของสิ่งของได้

4.ควรพัฒนาให้หุ่นยนต์หาจุดเริ่มต้นซึ่งใช้เป็นจุด Reference โดยอาศัยวงจรตรวจจับสีที่พื้น หรือกล้องมอนิเตอร์

ปัญหาและอุปสรรค์

1.วัสดุ อุปกรณ์ที่ใช้สร้างหุ่นยนต์ เช่น ล้อของหุ่นยนต์ หาซื้อยากและราคาแพง

2.ขาดเครื่องมือที่มีประสิทธิภาพสำหรับการสร้างหุ่นยนต์ เช่น เครื่องกลึง เครื่องมือวัดต่างๆ

3.การปรับแต่งระยะห่างระหว่างล้อของหุ่นยนต์เพื่อให้ได้มุมเลี้ยวรอบตัว 360 องศา ทำได้ยากและต้องทำขั้นตอนนี้หลายครั้ง

บรรณานุกรม

กฤษดา ใจเย็น และคณะ.2542. เรียนรู้ไมโครคอนโทรเลอร์เบสิกแสตมป์ 2. กรุงเทพ: บริษัท อินโนเวตีฟเอ็กเพอริเมนต์ จำกัด

กฤษดา ใจเย็น และคณะ.2543. เรียนรู้และปฏิบัติ การเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม. กรุงเทพ: บริบัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

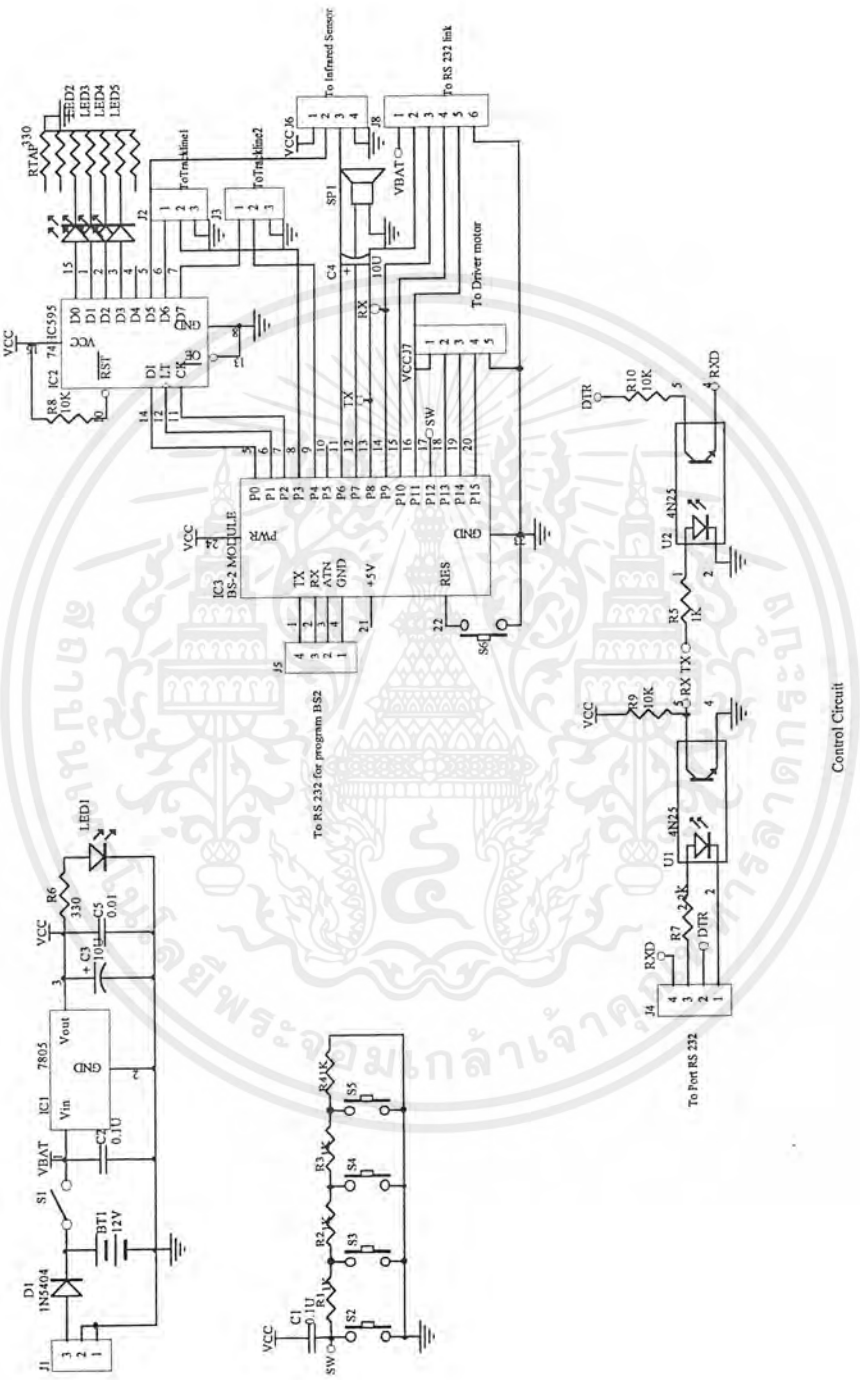
ธาริน สิทธิธรรมชารี.2542. คู่มือการเขียนโปรแกรม Visual Basic Vesion 6.0. กรุงเทพ:บรัษัท ซัคเซส มีเดีย จำกัด

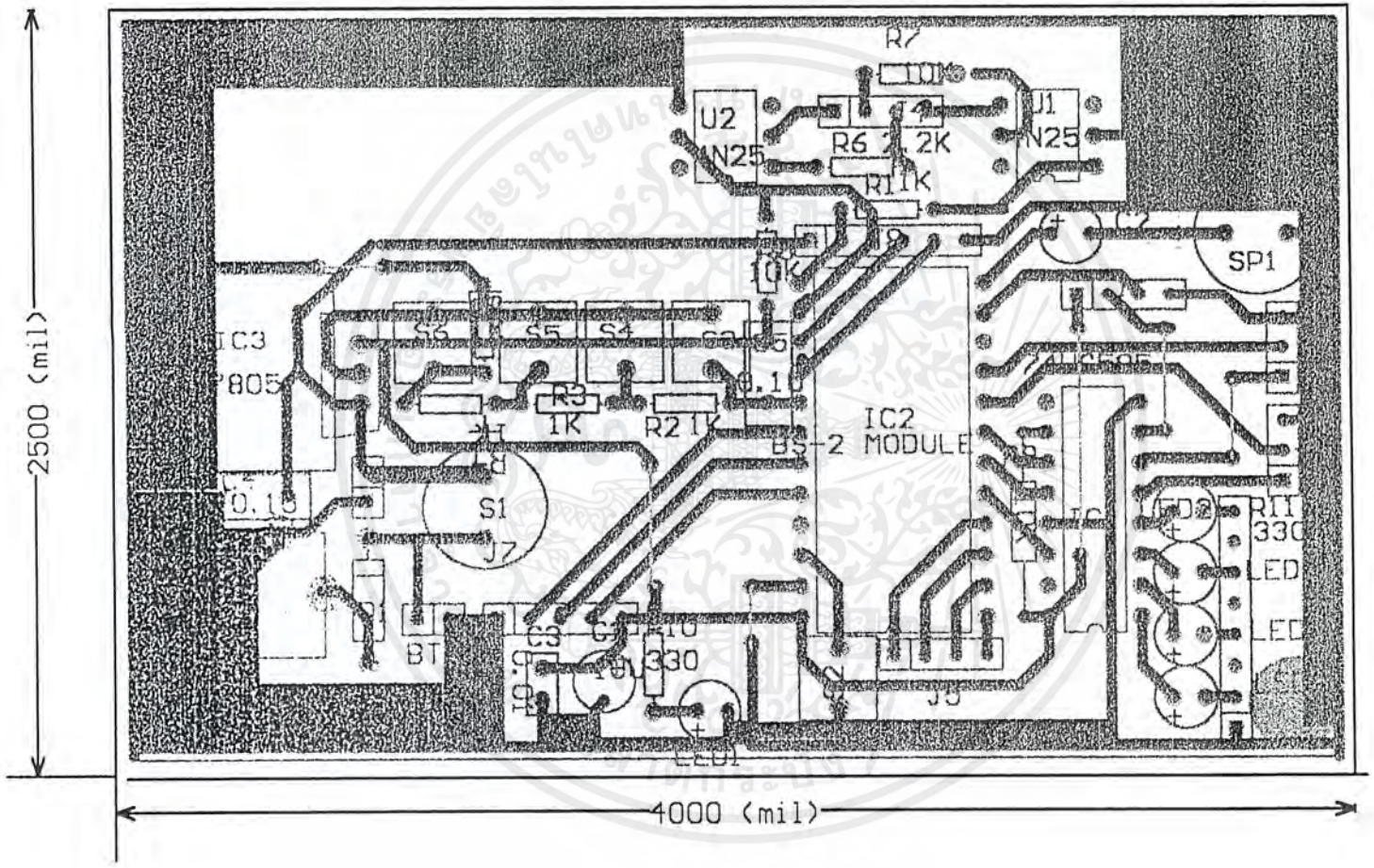
David Kortenkamp et.al .1998. Artificial Intelligence and Mobile Robots Massachusetts. Institute Of Tecnology USA

ภาคผนวก

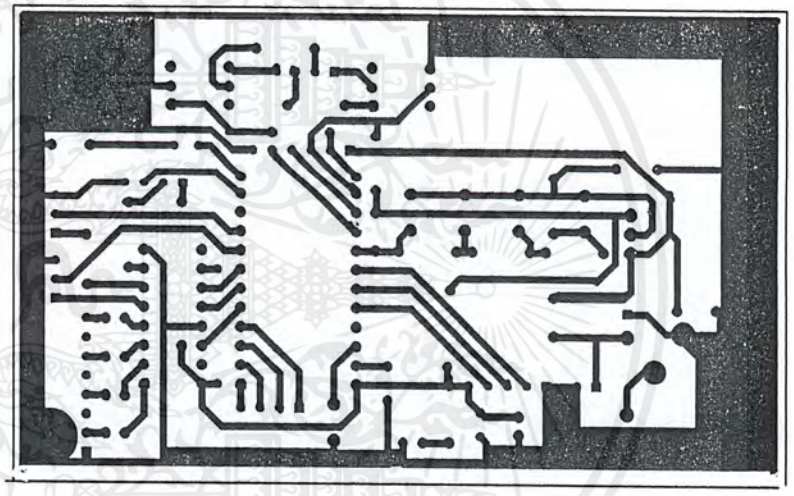
ภาคผนวก ก

รายละเอียดวงจร

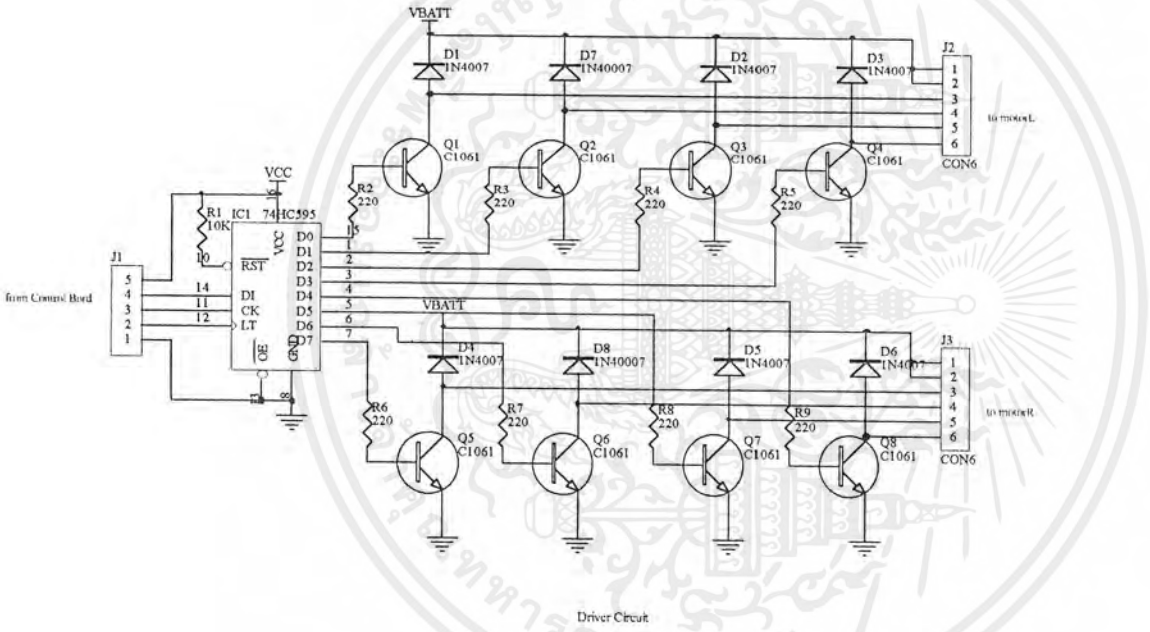




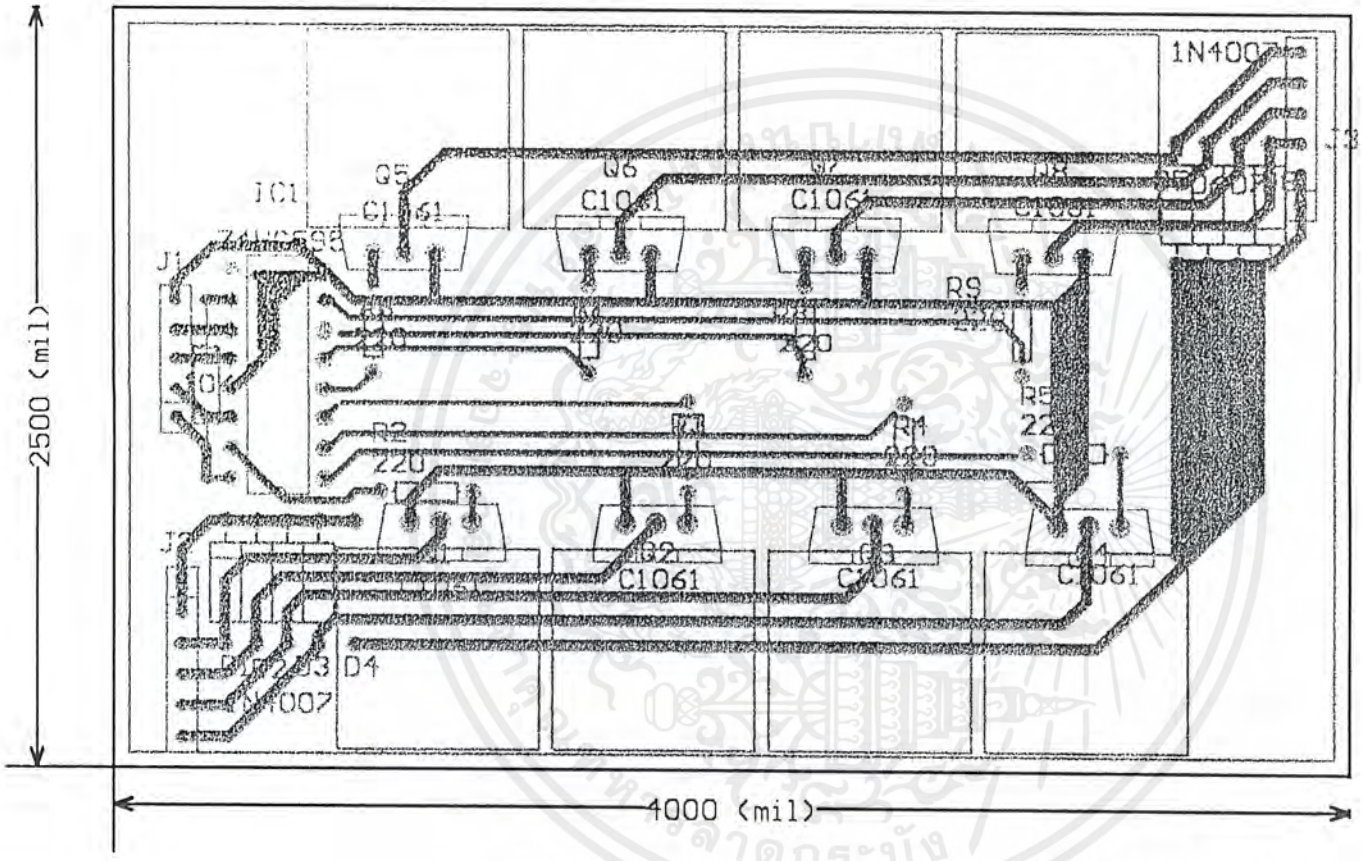
รูปแสดงการจัดวางอุปกรณ์บนแผ่นวงจรพิมพ์ของวงจรควบคุมหุ่นยนต์



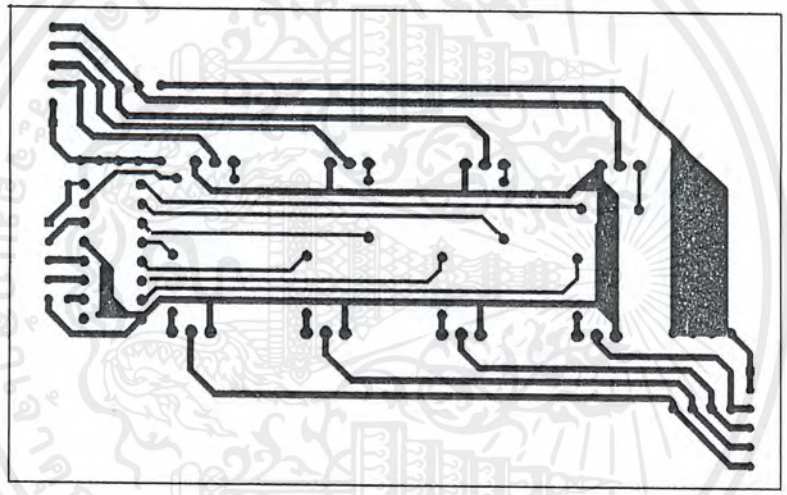
รูปแสดงลายวงจรพิมพ์เท่าของจริงของวงจรควบคุมหุ่นยนต์



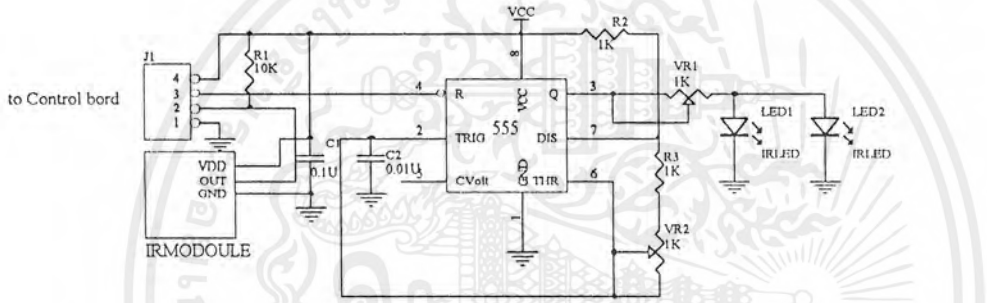
Driver Circuit



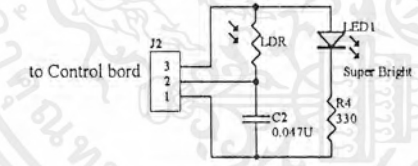
รูปแสดงการจัดวางอุปกรณ์บนแผ่นวงจรพิมพ์ของวงจร ขับสเต็ปเปอร์มอเตอร์



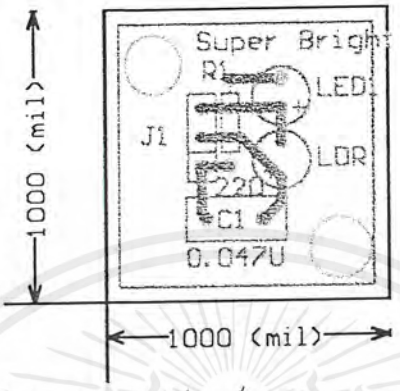
รูปแสดงลายวงจรพิมพ์เท่าของจริงของวงจรขับสเต็ปเปอร์มอเตอร์



Infrared Senser Circuit



Track Line on the floor Sensor



รูปแสดงการวางอุปกรณ์วงจร Tracking line

รูปแสดงลายวงจรพิมพ์ของวงจร Tracking line

ภาคผนวก ข

โปรแกรม

FormMain

```
Option Explicit
Dim fso As New FileSystemObject
Dim tbalign As Long

Private Sub MDIForm_DragDrop(Source As Control, X As Single, Y As Single) 'Move toolbar
If X < 1500 Then
tbToolbar.Align = 3
ElseIf X > Me.Width - 1500 Then
tbToolbar.Align = 4
ElseIf Y < 1500 Then
tbToolbar.Align = 1
ElseIf Y > Me.Height - 2000 Then
tbToolbar.Align = 2
End If
End Sub

Private Sub MDIForm_Load()
If fso.FileExists("c:\windows\blackuprct.ini") = False Then 'Case No Blackup file
Open "c:\windows\blackuprct.ini" For Random As #11
'Open file to saveparameter of program
frmWidth = 1
frmHeight = 1
ScaleMap = 10
Gridrang = 5
Nostep = 200
Wheel = 72
Portno = 1
turnError = 1
Speed = 10
'save parameter
Put #11, 1, frmWidth
Put #11, 2, frmHeight
Put #11, 3, Portno
```

```
Put #11, 4, ScaleMap
Put #11, 5, tbalign
Put #11, 6, Gridrang
Put #11, 7, Gridvisible
Put #11, 8, Wheel
Put #11, 9, Nostep
Put #11, 10, cgridline
Put #11, 11, turnError
Put #11, 12, Loopcheck
Put #11, 13, Speed
Put #11, 14, Refcheck
Close #11
Else 'Case Black up file
Open "c:\windows\blackuprct.ini" For Random As #10
'Read parameter
Get #10, 1, frmWidth
Get #10, 2, frmHeight
Get #10, 3, Portno
Get #10, 4, ScaleMap
Get #10, 5, tbalign
Get #10, 6, Gridrang
Get #10, 7, Gridvisible
Get #10, 8, Wheel
Get #10, 9, Nostep
Get #10, 10, cgridline
Get #10, 11, turnError
Get #10, 12, Loopcheck
Get #10, 13, Speed
Get #10, 14, Refcheck
Close #10
End If
tbToolbar.Align = tbalign 'Align position of toolbar
Vgridcolor = &H8000000C
sb.Panels(1) = "Welcome to Robot Controller Plese select menu"
End Sub
```

```
Private Sub MDIForm_Unload(Cancel As Integer)
tbalign = tbToolbar.Align
Open "c:\windows\blackuprct.ini" For Random As #12
'Save parameter of Program
Put #12, 1, frmWidth
Put #12, 2, frmHeight
Put #12, 3, Portno
Put #12, 4, ScaleMap
Put #12, 5, tbalign
Put #12, 6, Gridrang
Put #12, 7, Gridvisible
Put #12, 8, Wheel
Put #12, 9, Nostep
Put #12, 10, cgridline
Put #12, 11, turnError
Put #12, 12, Loopcheck
Put #12, 13, Speed
Put #12, 14, Refcheck
Close #12
End Sub

Public Sub mhelp_Click() ' Program help
With frmMain.dlg
.DialogTitle = "Help"
.HelpFile = App.Path & "\Help.HLP"
.HelpCommand = cdlHelpContents
.ShowHelp
End With
End Sub

Public Sub mnudata_Click() 'Show Data Route of Robot
If Programok = True Then
mnudata.Checked = Not mnudata.Checked
If mnudata.Checked = True Then
```

```
frmdata.Show
Else
frmdata.Hide
End If
End If
End Sub

Private Sub Mnudisplay_Click()
frmdisplay.Show
End Sub

Private Sub mnudroute_Click() 'Program Route of Robot
If Programok = False Then
i = 0
frmdraw.Show
Else
Res = MsgBox("You want to Program Route of Robot ? ", vbQuestion + vbYesNo, "Plese answer")
If Res = vbYes Then
frmdraw.Show
Call frmdraw.mnew_Click 'Clear form
End If
End If
End Sub

Private Sub mnuFileNew_Click() 'Click to Start Program
If Runautocheck = False Then 'Robot not run
i = 0
ofilePicture = ""
Programok = False
Unload frmdraw
Unload frmdisplay
Unload frmdata
'Close all from
End If
End Sub
```

```
Public Sub mnuoption_Click() '
frmOptions.Show vbModal, Me
End Sub

Private Sub tbToolBar_ButtonClick(ByVal Button As MSComctlLib.Button)
On Error Resume Next
Select Case Button.Key
Case "help"
mhelp_Click
Case "New"
mnuFileNew_Click
Case "draw"
mnudroute_Click
Case "dis"
frmdisplay.Show
Case "data"
mnudata_Click
Case "about"
mnuoption_Click
End Select
If frmdisplay.Visible = True Then 'Form display and control active
On Error Resume Next
Select Case Button.Key
Case "Save"
frmdraw.msave_Click
Case "Open"
frmdisplay.mdorf_Click
Case "palse"
frmdisplay.mspeed_Click
Case "stop"
frmdisplay.mdstop_Click
Case "speed"
frmdisplay.mspeed_Click
Case "run"
```

```
frmdisplay.mdra_Click
Case "man"
frmdisplay.mdma_Click
End Select
Elself frmdraw.Visible = True Then 'Form draw active
On Error Resume Next
Select Case Button.Key
Case "map"
frmdraw.minsert_Click
Case "undo"
frmdraw.mundo_Click
Case "Save"
frmdraw.msave_Click
End Select
End If
End Sub

Private Sub mnuHelpAbout_Click()
MsgBox "ROBOT'S CONTROLLER FOR TRANSPORTATIO VERTION 1 COPY  2000"
End Sub

Private Sub mnuViewStatusBar_Click()
mnuViewStatusBar.Checked = Not mnuViewStatusBar.Checked
sb.Visible = mnuViewStatusBar.Checked
End Sub

Private Sub mnuViewToolbar_Click()
mnuViewToolbar.Checked = Not mnuViewToolbar.Checked
tbToolbar.Visible = mnuViewToolbar.Checked
End Sub

Private Sub mnuFileExit_Click()
Unload Me
End Sub
```

```
Private Sub tbToolbar_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbLeftButton Then
tbToolbar.Drag vbBeginDrag
End If
End Sub
```

FormDraw

```
Option Explicit
Dim frmX As Single
Dim frmY As Single
Dim delx As Single
Dim dely As Single

Private Sub data_Click()
frmMain.mnudata_Click
End Sub

Private Sub Form_GotFocus() 'form not active
frmMain.Caption = "ROBOT'S CONTROLLER - Draw_Route"
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeyDelete Then
mundo_Click
End If
End Sub

Private Sub Form_Load()
Me.Width = (frmWidth * 1000 / ScaleMap) * MM 'Set scale "mm" to twip
Me.Height = (frmHeight * 1000 / ScaleMap) * MM 'Set scale "mm" to twip
Me.Move (frmMain.ScaleWidth - Me.Width) / 2, (frmMain.ScaleHeight - Me.Height) / 2
'Move form to ceter screen
frmMain.sb.Panels(1) = "Select Station by click LeftButton "
If ofilePicture <> "" Then 'Case Insert Map
Call subpicture 'Function picture
End If
If i > 2 Then
Call newdraw
End If
End Sub
```

```
Private Sub Form_LostFocus()
frmWidth = (Me.Width / MM) * ScaleMap / 1000
frmHeight = (Me.Height / MM) * ScaleMap / 1000
End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbLeftButton Then
If Programok = True Then
Exit Sub
End If
i = i + 1
Call seti
XX(i) = X: YY(i) = Y
If i = 1 Then
FillColor = QBColor(9)
FillStyle = 0
Circle (XX(i), YY(i)), 1, QBColor(9)
Print i
ElseIf i > 1 Then
delx = Abs(XX(i) - XX(1))
dely = Abs(YY(i) - YY(1))
If i > 2 And delx < 1 And dely < 1 Then
Call drawfinish
Exit Sub
End If
Line (XX(i), YY(i))-(XX(i - 1), YY(i - 1)), (QBColor(11))
FillColor = QBColor(13)
Circle (XX(i), YY(i)), 1, QBColor(13)
Print i
End If
ElseIf Button = vbRightButton Then
Me.MousePointer = 99
frmX = X: frmY = Y
End If
```

```
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Me.Move Left + ((X - frmX) * MM), Top + ((Y - frmY) * MM)
Exit Sub
End If
If Programok = False Then
Me.MousePointer = 2
Rx = X * ScaleMap
Ry = (ScaleHeight - Y) * ScaleMap
Unitx = FUNIT(Rx)
Unity = FUNIT(Ry)
Rx = Fnumber(Rx)
Ry = Fnumber(Ry)
frmMain.sb.Panels(1) = " X : " & Rx & Unitx & "  Y : " & Ry & Unity
If i > 0 Then
Line1.Visible = True
Line1.x1 = XX(i)
Line1.y1 = YY(i)
Line1.x2 = X
Line1.y2 = Y
delx = (X - XX(i)): dely = (Y - YY(i))
Rd = fdistance(delx, dely)
Unitdis = FUNIT(Rd)
Rd = Fnumber(Rd)
Direction(0) = fdirection(XX(i), YY(i), X, Y)
frmMain.sb.Panels(1) = " X : " & Rx & Unitx & "  Y : " & Ry & Unity & "  Direction=" & Direction(0) &
"  Distance=" & Rd & Unitdis
delx = Abs(X - XX(1)): dely = Abs(Y - YY(1))
If i > 1 And delx < 1 And dely < 1 Then
Me.MousePointer = 12
Else
Me.MousePointer = 2
End If
```

```
End If
End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Me.MousePointer = 0
End If
End Sub

Private Sub mexit_Click()
Unload Me
End Sub

Public Function fdistance(a As Single, b As Single) As Single
Dim c As Single
c = Sqr((a ^ 2) + (b ^ 2))
c = c * ScaleMap
fdistance = Round(c, 2)
End Function

Function fdirection(a1 As Single, b1 As Single, a2 As Single, b2 As Single) As Single
Dim deg As Single, deg1 As Single, delx As Single, dely As Single
delx = a2 - a1: dely = b2 - b1
If dely = 0 Then
If a1 > a2 Then
fdirection = 180
Else
fdirection = 0
End If
Else
deg1 = (Atn(delx / dely)) * (180 / Pi)
deg = Round(deg1, 0)
If b1 > b2 Then
fdirection = deg + 90
```

```
Else
fdirection = deg + 270
End If
End If
End Function

Function fturn(a As Single) As Single
If a > 180 Then
a = a - 360
Else
If a < -180 Then
a = a + 360
End If
End If
fturn = Round(a, 0)
End Function

Private Sub drawfinish()
Dim delx As Single, dely As Single
XX(i) = XX(1): YY(i) = YY(1)
Line1.Visible = False
Me.Line (XX(i), YY(i))-(XX(i - 1), YY(i - 1)), (QBColor(11))
For no = 1 To i - 1
delx = XX(no + 1) - XX(no)
dely = YY(no + 1) - YY(no)
Distance(no) = fdistance(delx, dely)
Direction(no) = fdirection(XX(no), YY(no), XX(no + 1), YY(no + 1))
Turn(no) = Direction(no) - Direction(no - 1)
Turn(no - 1) = fturn(Turn(no))
Next no
Turn(i) = Direction(1) - Direction(i - 1)
Turn(i - 1) = fturn(Turn(i))
Programok = True
Res = MsgBox("You Program route of Robot suscssfully ", vbOKCancel)
If Res = vbCancel Then
```

```
Me.mnew_Click
End If
Call newdraw
Me.MousePointer = 0
End Sub

Private Sub subpicture()
Picture2.Picture = LoadPicture(ofilePicture)
Me.Width = (Picture2.Width * MM) + 100
Me.Height = (Picture2.Height * MM) + 100
Me.Move (frmMain.ScaleWidth - Me.Width) / 2, (frmMain.ScaleHeight - Me.Height) / 2
Me.Picture = LoadPicture(ofilePicture)
End Sub

Public Sub newdraw()
Cls
Call setscale
FillStyle = 0
For no = 1 To i
If no = 1 Then
FillColor = QBColor(9)
Circle (XX(no), YY(no)), 1, QBColor(9)
Print no
Else
Line (XX(no), YY(no))-(XX(no - 1), YY(no - 1)), (QBColor(11))
FillColor = QBColor(13)
Circle (XX(no), YY(no)), 1, QBColor(13)
If no <> i Then
Print no
End If
End If
Next no
End Sub

Private Sub Form_Resize()
```

```
Call newdraw
End Sub

Private Sub setscale()
Dim nox As Long, noy As Long
If Gridvisible = False Then
Exit Sub
End If
Vgridcolor = frmOptions.gridcolor.BackColor
If cgridline = True Then
For no = 0 To ScaleHeight Step Gridrang
Line (0, ScaleHeight - no)-(Me.ScaleWidth, ScaleHeight - no), Vgridcolor
Next no
For no = 0 To ScaleWidth Step Gridrang
Line (no, 0)-(no, ScaleHeight), Vgridcolor
Next no
Else
For noy = 0 To ScaleHeight Step Gridrang
For nox = 0 To ScaleWidth Step Gridrang
PSet (nox, noy), Vgridcolor
Next nox
Next noy
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
frmMain.Caption = "ROBOT'S CONTROLLER "
frmWidth = (Me.Width / MM) * ScaleMap / 1000
frmHeight = (Me.Height / MM) * ScaleMap / 1000
End Sub

Public Sub mclear_Click()
Res = MsgBox("Display no Picture Blackground", vbOKCancel + vbQuestion)
If Res = vbOK Then
ofilePicture = ""
```

```
Picture = LoadPicture()
Call newdraw
End If
End Sub

Private Sub mh_Click()
frmMain.mhelp_Click
End Sub

Public Sub minsert_Click()
If i > 1 Then
MsgBox "Can't Insert Map"
Exit Sub
Else
frmMain.sb.Panels(1) = "คุณต้องการBiack Groundรูปภาพหรือไม่ Click 'Open' หรือ 'Cancel'"
With frmMain.dlg
.DialogTitle = "Load Picture"
.Flags = cdlOFNHideReadOnly
.Filter = "All Picture Files (*.bmp,*.ico,*.jpg,*.gif)|*.bmp; *.ico;*.jpg;*.gif|"
.FilterIndex = 2
.ShowOpen
ofilePicture = .FileName
End With
If ofilePicture <> "" Then
Call subpicture
Call newdraw
End If
End If
End Sub

Public Sub mnew_Click()
i = 0
ofilePicture = ""
Me.Picture = LoadPicture(ofilePicture)
Programok = False
```

```
Line1.Visible = False
frmMain.sb.Panels(1) = "Select point by click LeftButton or deleat by RightButton "
Me.Cls
Call setscale
End Sub

Private Sub mnuop_Click()
frmMain.mnuoption_Click
End Sub

Public Sub msave_Click()
If i < 3 And Programok = False Then
MsgBox "Can 't Open File Plese program Route of Robot", vbQuestion
Exit Sub
End If
With frmMain.dlg
.DialogTitle = "Save"
.CancelError = False
.Flags = cdlOFNHideReadOnly
.Filter = "All Files (*.*)|*.*|Robot 's Controller(*.rbc)|*.rbc"
.FilterIndex = 2
.ShowSave
If Len(.FileName) = 0 Then
Exit Sub
End If
sFile = .FileName
End With
Open sFile For Random As #2
Put #2, 1, i: Put #2, 2, ofilePicture: Put #2, 3, Turn(0)
Put #2, 4, frmWidth:  Put #2, 5, frmHeight: Put #2, 6, ScaleMap
Address = 7
For no = 1 To i - 1
Put #2, Address, XX(no)
Address = Address + 1
Next no
```

```
For no = 1 To i - 1
Put #2, Address, YY(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Put #2, Address, Direction(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Put #2, Address, Distance(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Put #2, Address, Turn(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Put #2, Address, Timestop(no)
Address = Address + 1
Next no
Close #2
End Sub

Public Sub mundo_Click()
If i < 2 Then
Exit Sub
End If
Line1.Visible = True
Res = MsgBox("You want deleae ?", vbQuestion + vbOKCancel, "Please answer me")
If Res = vbOK Then
i = i - 1
Call newdraw
End If
End Sub
```

FormDisplay

```
Option Explicit
Dim buffer As String
Dim cdtmanual As Long
Dim cdt As Byte
Dim dx As Single
Dim dy As Single
Dim s As Long
Dim Turn1 As Single
Dim send As String
Dim manualck As Boolean
Dim direct As Single
Dim d As Single
Dim noturn As Integer
Dim nodistance As Long
Dim times As Boolean
Dim exitp As Boolean
Dim frmX As Single, frmY As Single
Dim picx As Single, picy As Single
Dim x1 As Single, x2 As Single
Dim y1 As Single, y2 As Single
Dim Rang As Boolean
Const PId = Pi / 180

Private Sub comm_OnComm()
If comEvReceive = 2 Then
buffer = comm.Input
If manualck = False Then
If buffer = "aa" Then
Sleep (Speed)
Select Case cdt
Case 0
If nodistance = 0 And s = 1 And Refcheck = True Then
frmMain.sb.Panels(1) = "Robot finding reference "
```

```
comm.Output = "at"
Distance(s) = Distance(s) + Step
nodistance = nodistance + 1
Exit Sub
End If
nodistance = nodistance + 1
comm.Output = "af"
If nodistance >= Round((Distance(s) / Step), 0) Then
cdt = 1
frmMain.sb.Panels(1) = "Robot stop to work station"
End If
Case 1
comm.DTREnable = False
If Timestop(s + 1) = 0 Then
Timer2.Interval = 10
Else
Timer2.Interval = 1000 * Timestop(s + 1)
End If
Timer2.Enabled = True
Case 2
comm.Output = "al"
noturn = noturn + 1
If noturn >= Abs(Round(Turn(s) * turnError, 0)) Then
cdt = 0
s = s + 1
nodistance = 0
noturn = 0
frmMain.sb.Panels(1) = "Robot moving"
If s = i Then
If Loopcheck = False Then
Me.mdstop_Click
Exit Sub
End If
s = 1
End If
```

```
End If
Case 3
comm.Output = "ar"
noturn = noturn + 1
If noturn >= Abs(Round(Turn(s) * turnError, 0)) Then
cdt = 0
s = s + 1
nodistance = 0: noturn = 0
frmMain.sb.Panels(1) = "Robot moving"
If s = i Then
If Loopcheck = False Then
Me.mdstop_Click
Exit Sub
End If
s = 1
End If
End If
Case 4
comm.Output = "as"
comm.PortOpen = False
End Select
ElseIf buffer = "af" Then
dx = dx + (Cos(Direction(s) * PId)) * Step / ScaleMap
dy = dy - (Sin(Direction(s) * PId)) * Step / ScaleMap
ElseIf buffer = "al" Then
direct = direct + (1 / turnError)
ElseIf buffer = "ar" Then
direct = direct - (1 / turnError)
ElseIf buffer = "ap" Then
dx = XX(1)
dy = YY(1)
direct = Direction(1)
frmMain.sb.Panels(1) = "Robot moving"
ElseIf buffer = "ae" Then
MsgBox "ERRRRR"
```

```
End If
Else
If buffer = "aa" Then
Sleep (Speed)
Select Case cdtmanual
Case 1
comm.Output = "af"
Case 2
comm.Output = "al"
Case 3
comm.Output = "ar"
Case 4
comm.Output = "ab"
End Select
End If
End If
End If
End Sub

Private Sub Form_GotFocus()
frmMain.Caption = "ROBOT'S CONTROLLER - Display route of Robot"
If frmdraw.Visible = True Then
Me.Width = frmdraw.Width
Me.Height = frmdraw.Height
Else
Me.Width = (frmWidth * 1000 / ScaleMap) * MM
Me.Height = (frmHeight * 1000 / ScaleMap) * MM
End If
Me.Move (frmMain.ScaleWidth - Me.Width) / 2, (frmMain.ScaleHeight - Me.Height) / 2
frmMain.sb.Panels(1) = " "
On Error Resume Next
Me.Picture = LoadPicture(ofilePicture)
Call newdraw
End Sub
```

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeySpace Then
mdp_Click
End If
End Sub

Private Sub Form_LostFocus()
frmWidth = (Me.Width / MM) * ScaleMap / 1000
frmHeight = (Me.Height / MM) * ScaleMap / 1000
End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim delx As Single, dely As Single
If Button = vbRightButton Then
Me.MousePointer = 99
frmX = X: frmY = Y
ElseIf Button = vbLeftButton Then
If Rang = False Then
x1 = X: y1 = Y
Me.MousePointer = 2
Rang = True
Else
x2 = X: y2 = Y
delx = x2 - x1
dely = y2 - y1
Rd = frmdraw.fdistance(delx, dely)
Unitdis = FUNIT(Rd)
Rd = Fnumber(Rd)
MsgBox "Distance betaween two point : " & Rd & Unitdis
Me.MousePointer = 0
Rang = False
End If
End If
End Sub
```

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Me.Move Left + ((X - frmX) * MM), Top + ((Y - frmY) * MM)
End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Me.MousePointer = 0
End If
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
If Runautocheck = True Then
Cancel = True
End If
End Sub

Public Sub Form_Resize()
Call newdraw
End Sub

Private Sub Form_Unload(Cancel As Integer)
frmWidth = (Me.Width / MM) * ScaleMap / 1000
frmHeight = (Me.Height / MM) * ScaleMap / 1000
frmMain.Caption = "ROBOT'S CONTROLLER "
End Sub

Private Sub Image1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 4
End Sub

Private Sub Image1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 0
End Sub
```

```
Private Sub Image2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 1
End Sub

Private Sub Image2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 0
End Sub

Private Sub Image3_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 2
End Sub

Private Sub Image3_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 0
End Sub

Private Sub Image4_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 3
End Sub

Private Sub Image4_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
cdtmanual = 0
End Sub

Private Sub mdc_Click()
Unload Me
End Sub

Private Sub mdcr_Click()
frmMain.dlg.DialogTitle = "Color"
frmMain.dlg.CancelError = False
frmMain.dlg.ShowColor
frmdisplay.rbt.BackColor = frmMain.dlg.Color
frmdisplay.rbt.BorderColor = frmMain.dlg.Color
```

```
frmdisplay.Line1.BorderColor = frmMain.dlg.Color
End Sub

Public Sub mdma_Click()
If Programok = False Then
Exit Sub
End If
If Picture1.Visible = False Then
If Runautocheck = False Then
Step = (Wheel * Pi) / Nostep: Direction(0) = 0
dx = Me.ScaleWidth / 2: dy = Me.ScaleHeight / 2: direct = 0
comm.PortOpen = True
End If
manualck = True
Picture1.Visible = True
Picture1.Move (ScaleWidth - Picture1.Width) / 2, (ScaleHeight - Picture1.Height) / 2
frmMain.sb.Panels(1) = "Manual mode "
Else
Picture1.Visible = False
manualck = False
If Runautocheck = False Then
comm.PortOpen = False
End If
End If
End Sub

Public Sub mdn_Click()
If Runautocheck = False Then
Me.Cls
i = 0
ofilePicture = ""
Me.Picture = LoadPicture(ofilePicture)
rbt.Visible = False
Line1.Visible = False
Programok = False
```

```
frmMain.sb.Panels(1) = "Plese select menu"
End If
End Sub

Private Sub mdop_Click()
frmMain.mnuoption_Click
End Sub

Public Sub mdorf_Click()
If Runautocheck = False Then
If i > 2 And Programok = True Then
MsgBox "Can 't Open File Plese clear all point", vbQuestion
Exit Sub
End If
With frmMain.dlg
.DialogTitle = "Open"
.CancelError = False
.Flags = cdlOFNHideReadOnly
.Filter = "All Files (*.*)|*.*|Robot 's Controller(*.rbc)|*.rbc"
.FilterIndex = 2
.ShowOpen
If Len(.FileName) = 0 Then
Exit Sub
End If
sFile = .FileName
End With
Open sFile For Random As #1
Get #1, 1, i
Call seti
Get #1, 2, ofilePicture: Get #1, 3, Turn(0)
Get #1, 4, frmWidth: Get #1, 5, frmHeight: Get #1, 6, ScaleMap
Address = 7
For no = 1 To i - 1
Get #1, Address, XX(no)
Address = Address + 1
```

```
Next no
For no = 1 To i - 1
Get #1, Address, YY(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Get #1, Address, Direction(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Get #1, Address, Distance(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Get #1, Address, Turn(no)
Address = Address + 1
Next no
For no = 1 To i - 1
Get #1, Address, Timestop(no)
Address = Address + 1
Next no
Close #1
Programok = True
Me.Width = (frmWidth * 1000 / ScaleMap) * MM
Me.Height = (frmHeight * 1000 / ScaleMap) * MM
Me.Move (frmMain.ScaleWidth - Me.Width) / 2, (frmMain.ScaleHeight - Me.Height) / 2
Call Me.newdraw
End If
End Sub

Public Sub mdp_Click()
If Runautocheck = True Then
On Error Resume Next
comm.PortOpen = False
Res = MsgBox("Pause run Robot ! You wan to  continue ?", vbQuestion + vbYesNo)
```

```
If Res = vbYes Then
comm.PortOpen = True
Exit Sub
Elself Res = vbNo Then
mdstop_Click
End If
End If
End Sub

Private Sub mdr_Click()
If Runautocheck = False Then
If Programok = True Then
frmMain.sb.Panels(1) = "Press Key 'Space' to pause "
Runautocheck = True
dx = XX(1): dy = YY(1): s = 1: cdt = 0: direct = Direction(1)
Timer1.Enabled = True
comm.PortOpen = True
Else
MsgBox "Plese Program Route of Robot befor run ! "
End If
End If
End Sub

Public Sub mdra_Click()
If Runautocheck = False And Programok = True Then
Step = (Wheel * Pi) / Nostep
Runautocheck = True
dx = XX(1)
dy = YY(1)
s = 1
cdt = 0
direct = Direction(1)
frmMain.sb.Panels(1) = "Robot Run "
nodistance = 0
noturn = 0
```

```
Timer1.Enabled = True
comm.PortOpen = True
End If
End Sub

Private Sub mdsh_Click()
frmMain.mnudata_Click
End Sub

Public Sub mdstop_Click()
If Runautocheck = True Then
cdt = 4
Runautocheck = False
Timer1.Enabled = False
rbt.Visible = True
Line1.Visible = True
frmMain.sb.Panels(1) = "Robot stop "
End If
End Sub

Private Sub mh_Click()
frmMain.mhelp_Click
End Sub

Public Sub mspeed_Click()
If Slider1.Visible = False Then
Slider1.Move (ScaleWidth - Slider1.Width) / 2, (ScaleHeight - Slider1.Height) / 2
Slider1.Visible = True
Slider1.Value = (20 - Speed) / 2
Else
Slider1.Visible = False
End If
End Sub

Private Sub Picture1_GotFocus()
```

```
Sled.Visible = True
End Sub

Private Sub Picture1_KeyDown(KeyCode As Integer, Shift As Integer)
Select Case KeyCode
Case vbKeyLeft
cdtmanual = 2
Case vbKeyDown
cdtmanual = 4
Case vbKeyRight
cdtmanual = 3
Case vbKeyUp
cdtmanual = 1
End Select
End Sub

Private Sub Picture1_KeyUp(KeyCode As Integer, Shift As Integer)
Select Case KeyCode
Case vbKeyLeft
cdtmanual = 0
Case vbKeyDown
cdtmanual = 0
Case vbKeyRight
cdtmanual = 0
Case vbKeyUp
cdtmanual = 0
End Select
End Sub

Private Sub Picture1_LostFocus()
Sled.Visible = False
End Sub

Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
```

```
Me.MousePointer = 99
picx = X
picy = Y
End If
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Picture1.Move Picture1.Left + ((X - picx)), Picture1.Top + ((Y - picy))
Me.MousePointer = 0
End If
End Sub

Private Sub Slider1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Me.MousePointer = 99
picx = X
picy = Y
End If
End Sub

Private Sub Slider1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
Slider1.Move Slider1.Left + ((X - picx) / MM), Slider1.Top + ((Y - picy) / MM)
Me.MousePointer = 0
End If
End Sub

Private Sub Slider1_Scroll()
Speed = 20 - Slider1.Value * 2
End Sub

Private Sub Timer1_Timer()
rbt.Move dx - 2, dy - 2
Line1.x1 = dx: Line1.y1 = dy
```

```
Line1.x2 = dx + (3 * Cos(direct * Pid))
Line1.y2 = dy - (3 * Sin(direct * Pid))
If direct = 360 Then
direct = 0
ElseIf direct = -1 Then
direct = 359
End If
If rbt.Visible = False Then
rbt.Visible = True
Line1.Visible = True
Else
rbt.Visible = False
Line1.Visible = False
End If
End Sub

Private Sub setscale()
Dim nox As Long, noy As Long
If Gridvisible = False Then
Exit Sub
End If
Vgridcolor = frmOptions.gridcolor.BackColor
If cgridline = True Then
For no = 0 To ScaleHeight Step Gridrang
Line (0, ScaleHeight - no)-(Me.ScaleWidth, ScaleHeight - no), Vgridcolor
Next no
For no = 0 To ScaleWidth Step Gridrang
Line (no, 0)-(no, ScaleHeight), Vgridcolor
Next no
Else
For noy = 0 To ScaleHeight Step Gridrang
For nox = 0 To ScaleWidth Step Gridrang
PSet (nox, noy), Vgridcolor
Next nox
Next noy
```

```
End If
End Sub

Public Sub newdraw()
Me.Cls
Me.Picture = LoadPicture(ofilePicture)
Call setscale
FillStyle = 0
For no = 1 To i - 1
If no = 1 Then
FillStyle = 0: FillColor = QBColor(9)
Me.Circle (XX(no), YY(no)), 1, QBColor(9)
Else
Me.DrawWidth = 3
Line (XX(no), YY(no))-(XX(no - 1), YY(no - 1)), (QBColor(11)): Print no - 1
If Timestop(no) = 0 Then
FillColor = QBColor(11): Circle (XX(no), YY(no)), 1, QBColor(11)
Else
FillColor = QBColor(13): Circle (XX(no), YY(no)), 1, QBColor(13)
End If
Line (XX(1), YY(1))-(XX(i - 1), YY(i - 1)), (QBColor(11)): Print i - 1
Me.DrawWidth = 1
End If
Next no
End Sub

Private Sub Timer2_Timer()
Select Case Turn(s)
Case 0
cdt = 0: s = s + 1
If s = i Then
s = 1
End If
Case Is > 0
cdt = 2
```

```
frmMain.sb.Panels(1) = "Robot turn left"
Case Is < 0
cdt = 3
frmMain.sb.Panels(1) = "Robot turn right"
End Select
comm.DTREnable = True
Timer2.Enabled = False
End Sub
```

FormData

```
Option Explicit
Dim grdx As Single, grdy As Single

Private Sub Form_Load()
Me.Move (frmMain.ScaleWidth - Me.Width) / 2, 0 'Set form to center
End Sub

Public Sub Form_Resize()
grd.TextMatrix(0, 1) = "  PositionX"
grd.TextMatrix(0, 2) = "  PositionY"
grd.TextMatrix(0, 3) = "  Direction"
grd.TextMatrix(0, 4) = "  Distance"
grd.TextMatrix(0, 5) = "     Turn"
grd.TextMatrix(0, 6) = "     Turn"
grd.Rows = i
For no = 1 To i - 1
grd.TextMatrix(no, 0) = no
Next no
For no = 1 To i - 1
Rx = XX(no) * ScaleMap
Unitx = FUNIT(Rx)
Rx = Fnumber(Rx)
grd.TextMatrix(no, 1) = Rx & Unitx
Next no
For no = 1 To i - 1
Ry = ((frmHeight / MM) - YY(no)) * ScaleMap
Unity = FUNIT(Ry)
Ry = Fnumber(Ry)
grd.TextMatrix(no, 2) = Ry & Unity
Next no
For no = 1 To i - 1
 grd.TextMatrix(no, 3) = Direction(no)
Next no
```

```
For no = 1 To i - 1
Rd = Distance(no)
 Unitdis = FUNIT(Rd)
Rd = Fnumber(Rd)
grd.TextMatrix(no, 4) = Rd & Unitdis
Next no
For no = 1 To i - 1
grd.TextMatrix(no, 5) = Turn(no)
Next no
For no = 1 To i - 1
grd.TextMatrix(no, 6) = Timestop(no)
Next no
End Sub

Private Sub grd_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If X < 585 Or Y < 255 Then
Exit Sub
End If
grdx = (X - 585) \ 1020
grdy = Y \ 255
Select Case grdx
Case 0
Label1.Caption = "PositionX ( " & grdy & " )"
Text1.Text = grd.TextMatrix(grdy, 1)
Case 1
Label1.Caption = "PositionY ( " & grdy & " )"
Text1.Text = grd.TextMatrix(grdy, 2)
Case 2
Label1.Caption = "Direction ( " & grdy & " )"
Text1.Text = grd.TextMatrix(grdy, 3)
Case 3
Label1.Caption = "Distance ( " & grdy & " )"
Text1.Text = grd.TextMatrix(grdy, 4)
Case 4
Label1.Caption = "Turn ( " & grdy & " )"
```

```
Text1.Text = Turn(grdy)
Text1.Text = grd.TextMatrix(grdy, 5)
Case 5
Label1.Caption = "Timestop ( " & grdy & " )"
Text1.Text = grd.TextMatrix(grdy, 6)
End Select
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 And grdx = 5 Then
If Val(Text1.Text) > 100 Then
MsgBox "Over rang", vbCritical
Timestop(grdy) = 100
grd.TextMatrix(grdy, 6) = 100
Text1.Text = 100
Exit Sub
End If
Timestop(grdy) = CLng(Text1.Text)
grd.TextMatrix(grdy, 6) = Timestop(grdy)
End If
End Sub
```

FormOptions

```
Option Explicit
Dim serialport As Byte
Dim timeok  As Byte
Dim  timeopt As Byte
Dim modetime1 As Boolean

Private Sub Check1_Click()
If Check1.Value = 1 Then
Gridvisible = True
Textgrd.Visible = True
Checkpoint.Visible = True
ElseIf Check1.Value = 0 Then
Gridvisible = False
Textgrd.Visible = False
Checkpoint.Visible = False
End If
End Sub

Private Sub Check2_Click()
If Check2.Value = 0 Then
Refcheck = False
ElseIf Check2.Value = 1 Then
Refcheck = True
End If
End Sub

Private Sub Checkpoint_Click()
If Checkpoint.Value = 0 Then
cgridline = False
ElseIf Checkpoint = 1 Then
cgridline = True
End If
End Sub
```

```
Private Sub cmdCancel_Click()
Unload Me
End Sub

Private Sub cmdOK_Click()
Call optionok
Unload Me
End Sub

Private Sub cmdwheel_Click()
Text3.Text = (CVar(Textw.Text) / CVar(Textstep.Text)) * 360
End Sub

Private Sub Command3_Click()
Call optionok
End Sub

Private Sub Form_Load()
If Loopcheck = False Then
Option1.Value = True
Option2.Value = False
Else
Option1.Value = False
Option2.Value = True
End If
If Refcheck = False Then
Check2.Value = 0
Else
Check2.Value = 1
End If
End Sub

Private Sub Form_Resize()
Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2
```

```
Checkpoint = Abs(CInt(cgridline))
frmcolor.BackColor = frmdraw.BackColor
gridcolor.BackColor = Vgridcolor
If Gridvisible = True Then
Check1.Value = 1
Else
Check1.Value = 0
End If
Select Case Portno
Case 1
Ops1.Value = True
Case 2
Ops2.Value = True
End Select
ReDim Preserve Timestop(i)
Select Case timeok
Case 0: Optn.Value = True
Case 1:  Opta.Value = True
Text.Text = Timestop(1)
Lat.Visible = True
End Select
If frmdraw.Visible = True Then
txtwidth.Text = (frmdraw.Width / MM) * ScaleMap / 1000
txtheight.Text = (frmdraw.Height / MM) * ScaleMap / 1000
ElseIf frmdisplay.Visible = True Then
txtwidth.Text = (frmdisplay.Width / MM) * ScaleMap / 1000
txtheight.Text = (frmdisplay.Height / MM) * ScaleMap / 1000
Else
txtwidth.Text = Round(frmWidth, 2)
txtheight.Text = Round(frmHeight, 2)
End If
txtwidth.Text = Round(txtwidth.Text, 2)
txtheight.Text = Round(txtheight.Text, 2)
Textstep.Text = Nostep
Textw.Text = Wheel
```

```
Textscl.Text = ScaleMap
Textgrd.Text = Gridrang
Texterror = turnError
End Sub

Private Sub frmcolor_Click()
frmMain.dlg.DialogTitle = "Form Color"
frmMain.dlg.CancelError = False
frmMain.dlg.ShowColor
frmcolor.BackColor = frmMain.dlg.Color
End Sub

Private Sub gridcolor_Click()
frmMain.dlg.DialogTitle = "Grid Color"
frmMain.dlg.CancelError = False
frmMain.dlg.ShowColor
gridcolor.BackColor = frmMain.dlg.Color
End Sub

Private Sub ops1_Click()
Ops1.Value = True
serialport = 1
End Sub

Private Sub Ops2_Click()
Ops2.Value = True
serialport = 2
End Sub

Private Sub Opta_Click()
Lat.Visible = True
Label12.Visible = True
Text.Visible = True
On Error Resume Next
Text.Text = CStr(Timestop(1))
```

```
Opta.Value = True
timeopt = 1
modetime1 = False
End Sub

Private Sub Optipic_Click()
modetime1 = True
End Sub

Private Sub Option1_Click()
Loopcheck = False
End Sub

Private Sub Option2_Click()
Loopcheck = True
End Sub

Private Sub Optn_Click()
Lat.Visible = False
Text.Visible = False
Label12.Visible = False
Optn.Value = True
timeopt = 0
modetime1 = False
End Sub

Private Sub Opts_Click()
Lat.Visible = False
Label12.Visible = False
Text.Visible = False
If i < 3 Then
Opts.Value = False
Exit Sub
End If
Res = MsgBox("You want to Program time  stop ? ", vbQuestion + vbYesNo, "Plese answer")
```

```
If Res = vbYes Then
timeok = 2
Unload Me
modetime1 = False
frmdata.Show
Else
Opts.Value = False
Exit Sub
End If
End Sub

Private Sub optionok()
If Check1.Value = 1 Then
Gridvisible = True
Checkpoint.Visible = True
ElseIf Check1.Value = 0 Then
Gridvisible = False
Checkpoint.Visible = False
End If
frmdraw.BackColor = frmcolor.BackColor
Vgridcolor = gridcolor.BackColor
Gridrang = CLng(Textgrd.Text)
If Gridrang < 1 Then
Gridrang = 1
End If
If Val(Textscl.Text) > 1 Then
ScaleMap = Val(Textscl.Text)
Else
Textscl.Text = 2
End If
frmWidth = CSng(txtwidth.Text)
frmHeight = CSng(txtheight.Text)
If frmdraw.Visible = True Then
Call frmdraw.newdraw
End If
```

```
If frmdisplay.Visible = True Then
Call frmdisplay.newdraw
End If
Select Case serialport
Case 1
Portno = 1
Case 2
Portno = 2
End Select
If i > 2 Then
timeok = timeopt
Select Case timeok
Case 0
For no = 1 To i - 1
Timestop(no) = 0
Next no
Case 1
If CVar(Text.Text) > 121 Or Text.Text = "" Then
MsgBox "Please insert value in textbox or new choie ", vbQuestion
Exit Sub
End If
For no = 1 To i - 1
Timestop(no) = CVar(Text.Text)
Next no
End Select
End If
Nostep = Val(Textstep.Text)
Wheel = Val(Textw.Text)
Step = (Wheel * Pi) / Nostep
turnError = Val(Texterror)
Modetime = modetime1
End Sub
```

FormModule

```
Option Explicit
Public XX() As Single
Public YY() As Single
Public Distance() As Single
Public Step As Single
Public Direction() As Single
Public Turn() As Single
Public Timestop() As Long
Public Speed As Long
Public Wheel As Single
Public Portno As Long
Public ScaleMap As Long
Public Nostep As Long
Public turnError As Single
Public Ry As Single
Public Rx As Single
Public Rd As Single
Public Unitx As String
Public Unity As String
Public Unitdis As String
Public frmHeight As Single
Public frmWidth As Single
Public i As Long
Public Programok As Boolean
Public Modetime As Boolean
Public Gridvisible As Boolean
Public Loopcheck As Boolean
Public Refcheck As Boolean
Public Gridrang As Long
Public Runautocheck As Boolean
Public cgridline As Boolean
Public Vgridcolor As Variant
Public Res As Variant
```

```
Public no As Long
Public ofilePicture As String
Public sFile As String
Public Address As Long
Public Const MM = 144 / 2.54
Public Const Pi = 3.141592654
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Public Sub seti()
ReDim Preserve XX(i): ReDim Preserve YY(i): ReDim Preserve Distance(i)
ReDim Preserve Turn(i): ReDim Preserve Direction(i)
ReDim Preserve Timestop(i)
End Sub

Public Function FUNIT(a As Single) As String
If a > 1000 Then
FUNIT = " m"
Else
FUNIT = " cm"
End If
End Function

Public Function Fnumber(a As Single) As Single
If a > 1000 Then
a = a / 1000
Else
a = a / 10
End If
Fnumber = Round(a, 2)
End Function
```

โปรแกรมหุ่นยนต์

```
'PROGRAM ROBOT'S CONTROLLER
PDI     CON     0
PLT     CON     1
PCK     CON     2
PTL1    CON     4
PTL2    CON     5
PTL3    CON     6
PTL4    CON     3
PSP     CON     7
PSW     CON     12
PCKD    CON     14
PLTD    CON     15
PDID    CON     13
PTX     CON     8
PRX     CON     9
REF     CON     10
BAUDR   CON     84              ' SET BUAD RATE =9600
BAUDT   CON     16468
LED     VAR BYTE
DATA1   VAR BYTE
L VAR NIB
R VAR NIB
LOOP VAR NIB
REC VAR BYTE
RES VAR WORD
RESL VAR BYTE
RESR VAR BYTE
RUNCHECK VAR BIT
FREQOUT PSP,100,2000            'BEGIN PROGRAM
FREQOUT PSP,100,2100
SHIFTOUT PDID,PCKD,0,[LED]      ' SEND THE BITS.
PULSOUT PLTD,1
SHIFTOUT PDI,PCK,0,[DATA1]      ' SEND THE BITS.
PULSOUT PLT,1
```

```
RUN:
SEROUT PTX,BAUDT,["aa"]                    'request data to Computer
SERIN PRX,BAUDR,100,RUN,[WAIT ("a"),REC] ' Wait data
RUN1:
IF REC="f" THEN FOW
IF REC="l" THEN LEFT
IF REC="r" THEN RIGHT
IF REC="b" THEN BLACK
IF REC="t" THEN BEGINTRACK
IF REC="c" THEN TRACK
IF REC="s" THEN STOPROBOT
ERROR:
SEROUT PTX,BAUDT,["ae"]
FREQOUT PSP,100,2000
FREQOUT PSP,100,1500
FREQOUT PSP,100,2100
FREQOUT PSP,100,3000
GOTO ERROR

FOW:              'forword
L=L>> 1
R=R << 1
IF L=0 THEN FSETL
FCKR:
IF R=0 THEN FSETR
GOTO DRIVE
FSETL:
L=8
GOTO FCKR
FSETR:
R=1
GOTO DRIVE
```

```
LEFT:           'Turn left
L=L>>1
R=R >> 1
IF L=0 THEN LSETL
LCKR:
IF R=0 THEN LSETR
GOTO DRIVE
LSETL:
L=8
GOTO LCKR
LSETR:
R=8
GOTO DRIVE

RIGHT:          'turn right
L=L<<1
R=R << 1
IF L=0 THEN RSETL
RCKR:
IF R=0 THEN RSETR
GOTO DRIVE
RSETL:
L=1
GOTO RCKR
RSETR:
R=1
GOTO DRIVE

BLACK:          'revert
L=L<<1
R=R >> 1
IF L=0 THEN BSETL
BCKR:
IF R=0 THEN BSETR
GOTO DRIVE
```

```
BSETL:
L=1
GOTO BCKR
BSETR:
R=8
GOTO DRIVE

FOWL:
L=L>> 1
IF L=0 THEN FLSETL
GOTO DRIVE
FLSETL:
L=8
GOTO DRIVE

FOWR:
R=R << 1
IF R=0 THEN FRSETR
GOTO DRIVE
FRSETR:
R=1
GOTO DRIVE

BLACKL:
L=L<<1
IF L=0 THEN BLSETL
GOTO DRIVE
BLSETL:
L=1
GOTO DRIVE

BLACKR:
R=R >> 1
IF R=0 THEN BRSETR
GOTO DRIVE
```

```
BRSETR:
R=8
GOTO DRIVE

DRIVE:                    'drive motor
DATA1=R
DATA1=DATA1<<4
DATA1=DATA1+L
SHIFTOUT PDID,PCKD,0,[DATA1] ' SEND THE BITS.
PULSOUT PLTD,1
IF RUNCHECK=1 THEN RUN2          ' TRANSFER TO OUTPUTS.
RETURN

RUN2:
SEROUT PTX,BAUDT,["a",REC]
REC=0
GOTO RUN

BEGINTRACK: 'fine reference
FREQOUT PSP,100,2400
LED=%00010000
SHIFTOUT PDI,PCK,0,[LED]
PULSOUT PLT,1
LOOP=0
REC="c"
GOTO RUN1

TRACK:
RUNCHECK=0
LED=%00001000'ON LED1
SHIFTOUT PDI,PCK,0,[LED]
PULSOUT PLT,1
LED=LED | %10000000
SHIFTOUT PDI,PCK,0,[LED]
PULSOUT PLT,1
```

```
TRACK0:
LOW     PTL3
LOW PTL4
PAUSE 20
RCTIME PTL3,0,RES
RESL=RES
RCTIME PTL4,0,RES
RESR=RES
IF RESL < REF AND RESR < REF THEN TRACK1
IF RESL > REF AND RESR < REF THEN TRACK2
IF RESL < REF AND RESR > REF THEN TRACK3
IF RESL > REF AND RESR > REF THEN TRACK4

TRACK1
GOSUB FOW
GOTO TRACK0

TRACK2:
GOSUB FOWL
GOTO TRACK0

TRACK3:
GOSUB FOWR
GOTO TRACK0

TRACK4:
LOW     PTL3
PAUSE 20
RCTIME PTL3,0,RES
RESL=RES
IF RESL< REF THEN TRACK5
GOSUB BLACKR
GOTO TRACK4
```

```
TRACK5:
LOW     PTL4
PAUSE 20
RCTIME PTL4,0,RES
RESR=RES
IF RESR<REF THEN TRACK6
GOSUB BLACKL
GOTO TRACK5


TRACK6:
LOOP=LOOP+1
IF LOOP>1 THEN TRACK7
GOTO TRACK0

TRACK7:
LED=LED & %01111111
SHIFTOUT PDI,PCK,0,[LED]
PULSOUT PLT,1
FREQOUT PSP,100,2000
FREQOUT PSP,100,3000
SEROUT PTX,BAUDT,["AP"]
RUNCHECK=1
GOTO RUN

STOPROBOT:
SHIFTOUT PDID,PCKD,0,[0]        ' SEND THE BITS.
PULSOUT PLTD,1
FREQOUT PSP,100,2400
PAUSE 800
GOTO RUN
```

ภาคผนวก ค
การใช้งานโปรแกรม

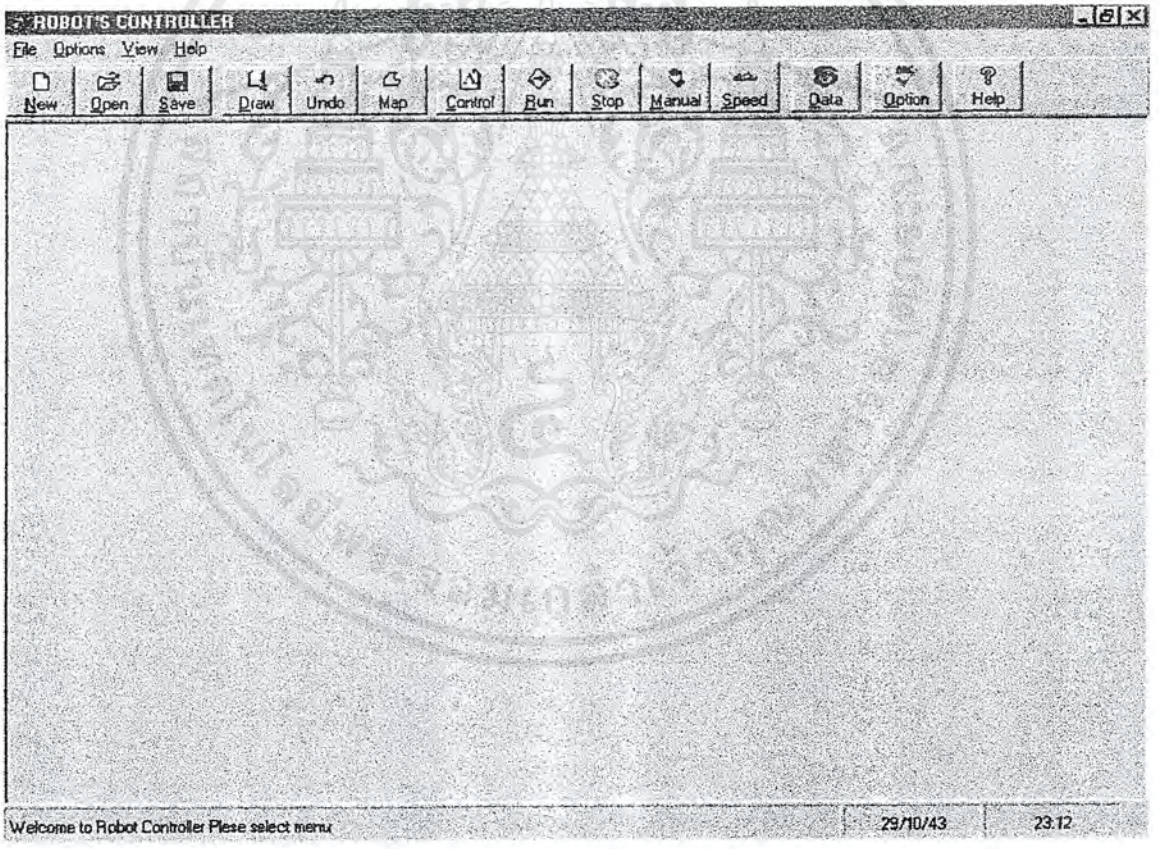
การใช้งานโปรแกรม Robot Controller

การติดตั้งโปรแกรม

เนื่องจากโปรแกรมนี้ทำงานในระบบปฏิบัติการวินโดว์ส ดังนั้นเครื่องคอมพิวเตอร์ที่จะติดตั้งจึงต้องมีระบบปฏิบัติการวินโดว์ส ควรจะเป็นวินโดว์ส 98 ขึ้นไป การติดตั้งเหมือนกับการติดตั้งโปรแกรมอื่นๆลงในวินโดว์ส โดยคลิกที่ ไฟล์ Setup แล้วทำตาม Wizard ที่วินโดว์สกำหนด

การใช้งานโปรแกรม

หลังจากติดตั้งโปรมแกรมเสร็จเรียบร้อยแล้วเริ่มเข้าสู่โปรแกรมโดยคลิก Start>program> Robot Controller เพื่อเข้าสู่โปรแกรม ครั้งแรกจะมีลักษณะดังรูป



รูปแสดงลักษณะหน้าตาของโปรแกรม

คำสั่งต่างๆใน เมนูฟอร์มหลัก

File	
New	ใช้สำหรับเริ่มโปรแกรมใหม่
Program route of robot	ใช้สำหรับเปิดฟอร์มเพื่อโปรแกรมการเคลื่อนที่ของหุ่นยนต์
Display route of robot	ใช้สำหรับเปิดฟอร์มแสดงตำแหน่งและควบคุมการทำงานของหุ่นยนต์
Exit Program	ใช้สำหรับเมื่อต้องการออกจากโปรแกรม
Options	
Data route of robot	ใช้เปิดฟอร์มเพื่อแสดงข้อมูลการเคลื่อนที่ของหุ่นยนต์
Options	ใช้เปิดฟอร์ม Option ของโปรแกรม
View	
Tool bar	ใช้เลือกว่าจะให้ Toolbar แสดงหรือไม่ต้องการให้แสดง
Status bar	ใช้เลือกว่าจะให้ Statusbar แสดงหรือไม่ต้องการให้แสดง
Help	
Contents	ใช้ดูคู่มือการใช้งานโปรแกรม

คำสั่งต่างๆใน เมนูฟอร์ม Draw

File	
New	ใช้สำหรับเคลียร์เมนูเดิมเพื่อเริ่มโปรแกรมใหม่
Save route	ใช้บันทึก File โปรแกรมการเคลื่อนที่ของหุ่นยนต์
Close form	ใช้ปิดฟอร์ม
Edit	
Undo	ใช้ลบเส้นทางเมื่อโปรแกรมเส้นทางผิด
Insert map	ใช้แทรกแผนที่เข้าไปยังฟอร์มวาด โดย File ที่เข้าแทรกจะต้องเป็น File รูปภาพเท่านั้น ได้แก่ .bmp .ico .jpg .gif
Clear map	ใช้เคลียร์แผนที่ออกจากฟอร์ม
Options	
Data route of robot	ใช้เปิดฟอร์มเพื่อแสดงข้อมูลการเคลื่อนที่ของหุ่นยนต์
Option	ใช้เรียกฟอร์ม Option ต่างๆของโปรแกรม
Help	
Help	ใช้ดูคู่มือการใช้งานโปรแกรม

คำสั่งต่างๆใน เมนูฟอร์ม Display

File	
New	ใช้สำหรับเคลียร์โปรแกรมเดิมเพื่อเริ่มโปรแกรมใหม่
Open Route	ใช้เปิด File การเคลื่อนที่ของหุ่นยนต์ที่บันทึกไว้มาใช้งาน
Close form	ใช้ปิดฟอร์มแสดงผล
Run	
Auto Run	คำสั่งที่ให้ Robot ทำงานตามที่เราโปรแกรม
Pause	คำสั่งหยุดการทำงานชั่วขณะ
Stop	คำสั่งหยุดการทำงาน Computer เลิกติดต่อหุ่นยนต์
Manual	ควบคุมหุ่นยนต์แบบ Manual โดยใช้ mouse คลิกที่ Form หรือ กด keyUp keyDown ,keyLeft และ keyRight บน Keybord
Options	
Speed	ใช้ปรับความเร็วในการเคลื่อนที่ของหุ่นยนต์
Show Data	ใช้เปิดฟอร์มเพื่อแสดงข้อมูลการเคลื่อนที่ของหุ่นยนต์
Robot colour	ใช้ปรับแต่งสีการแสดงผลของหุ่นยนต์
Options	ใช้เรียกฟอร์ม Options ของโปรแกรม
Help	
Help	ใช้ดูคู่มือการใช้งานโปรแกรม

การใช้งานฟอร์ม Options

General	
Scale	ใช้ใส่ค่าอัตราส่วนของแผนที่
Grid	
Grid Visible	ใช้เลือกว่าจะแสดง Grid หรือไม่
Grid colour	ใช้ปรับแต่งสีของ Grid
Grid line	ใช้เลือกว่าจะสร้าง Grid แบบสีหรือแบบเส้น
Form	
Form width	ใช้ปรับความกว้างของแผนที่โดยระยะเป็นเมตร
Form height	ใช้ปรับความยาวของแผนที่โดยระยะเป็นเมตร
Form colour	ใช้ปรับแต่งสีของ Form

Time dock	เป็น Option ปรับเวลาการหยุดโดยมีให้เลือก 3 แบบคือ
Unlock	ไม่มีการหยุด
All dock	ทุกสถานีจะมีค่าอยู่เท่าๆกัน โดยสามารถใส่ค่าเวลาหยุดลงไป
Select dock	ใช้เลือกปรับเวลาหยุดของแต่ละสถานีโดยแต่ละสถานีหยุดไม่เท่ากัน
Serial port	ใช้ในการเลือกพอร์ตสื่อสารว่าจะเลือกพอร์ตไหน
Structure of Robot	
Diameter of wheel	ใช้กำหนดเส้นผ่านศูนย์กลางของล้อของหุ่นยนต์
No.step of motor	จำนวน step ของ motor
Turn Error	ใช้กำหนดค่า error จากการเลี้ยว
Wheel to Wheel	ใช้กำหนดระยะระหว่างล้อของหุ่นยนต์
Auto	
Single Round	หุ่นยนต์เคลื่อนที่รอบเดียว
Multi Round	หุ่นยนต์เคลื่อนแบบต่อเนื่อง
Check Reference	หุ่นยนต์ตรวจสอบจุดอ้างอิง และทำการปรับตำแหน่งให้ถูกต้อง