รถควบคุมด้วยคอมพิวเตอร์

COMPUTER CONTROLLED CAR

โดย

นาย เฉลิมเกียรติ ถีอาสนา

นาย สถิตย์ ฤทธิยงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา อุตสาหกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

หัวข้อปริญญานิพนธ์	รถควบคุมด้วยคอมพิวเตอร์	
นักศึกษา	นายเฉลิมเกียรติ ถีอาสนา	รหัสประจำตัว 43015720
	นายสถิตย์ ฤทธิยงค์	รหัสประจำตัว 43015748
อาจารย์ผู้ควบคุมวิทยานิพนธ์	อาจารย์สมภพ แก้วมีชัย	
	อาจารย์บุณย์ชนะ ภู่ระหงษ์	
ระดับการศึกษา	ปริญญาอุตสาหกรรมศาสตร์บัณฑิต	
ภาควิชา	วิศวกรรมสารสนเทศ	
ปีการศึกษา	2544	

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(อาจารย์สมภพ แก้วมีชัย)

อาจารย์ผู้ควบคุมวิทยานิพนธ์

.....

(อาจารย์บุณย์ชนะ ภู่ระหงษ์)

อาจารย์ผู้ควบคุมวิทยานิพนธ์

หัวข้อปริญญานิพนธ์	รถควบคุมด้วยคอมพิวเตอร์		
นักศึกษา	นายเฉลิมเกียรติ	ถืออาสนา	รหัสประจำตัว 43015720
	นายสถิตย์	ฤทธิยงค์	รหัสประจำตัว 43015748
อาจารย์ผู้ควบคุมวิทยานิพนธ์	อาจารย์สมภพ	แก้วมีชัย	
	อาจารย์บุณย์ชนะ	ภู่ระหงษ์	
ระดับการศึกษา	ปริญญาอุตสาหกรรมศาสตร์บัณฑิต		
ภาควิชา	วิศวกรรมสารสนเทศ		
ปีการศึกษา	2544		

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการศึกษาการออกแบบและการสร้างระบบควบคุมรถให้ เคลื่อนที่ไปยังตำแหน่งที่ต้องการ โดยการส่งสัญญาณควบคุมไร้สายให้กับไมโครคอนโทรล 89C51 ควบคุมการเคลื่อนที่ของรถ กล้องวิดีโอจะคอยจับภาพรถ และส่งให้เครื่องคอมพิวเตอร์ส่วนบุคคลแสดงผล

THESIS TITLE	COMPUTER CONTROLLED CAR
STUDENT	Mr. Chalemkiat Theeasana No. 43015720
	Mr. Satit Rittiyong No. 43015748
ADVISOR	Mr. Sompoph Kaewmechai
	Mr. Boonchana Poorahong
COURSE	Bachelor of Industrial Engineering
DEPARTMENT	Informatiom Engineering
YEAR	2001

ABSTRACT

This project presents the design and construction of car control system in order to move car to the target position within sends the wireless control signal to the microcontroller 89C51 to the movement of the car. A video camera captures of car obstruction and sends to the personal computer for display the captured picture.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่สำเร็จด้วยดี หากไม่ได้รับความร่วมมือจากอาจารย์ รุ่นพี่ และ เพื่อนๆ ขอขอบพระคุณอย่างสูงอาจารย์สมภพ แก้วมีชัย อาจารย์บุณย์ชนะ ภู่ระหงษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้คำปรึกษา คำแนะนำ และคำติชมต่างๆ ขอขอบคุณพี่จตุรวิทย์ จันไพบูลย์ ที่ให้คำปรึกษาด้านโปรแกรมและข้อมูล ขอบใจเพื่อนๆที่ให้ยืมโปแกรมต่างๆและอุปกรณ์ และสุดท้ายขอขอบพระคุณ บิดา มารดา ที่ให้ความเอาใจใส่และกำลังใจผู้เขียนเสมอมา

เฉลิมเกียรติ ถีอาสนา
สถิตย์ ฤทธิยงค์

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	
สารบัญรูป	
สารบัญตาราง	
บทที่ 1 บทนำ	1
1.1 วัตถุประสงค์	1
1.2 ขอบข่ายของโครงงาน	1
บทที่ 2 ทฤษฎีและหลักการ	2-22
2.1 ขั้นตอนการทำงาน	2
2.2 ไมโครคอนโทรเลอร์	3
2.2.1 คุณสมบัติทั่วไปของไมโครคอนโทรลเลอร์ MCS - 51	3
2.2.2 โครงสร้างภายนอกของ MCS - 51	4
2.2.3 การจัดหน่วยความจำ	7
2.2.4 รีจิสเตอร์หน้าที่พิเศษ (SFR)	9
2.2.5 รีจีสเตอร์ใช้งานทั่วไป	9
2.3 การรับส่งข้อมูลผ่าน Serial port	9
2.3.1 พอร์ตสื่อสารข้อมูลอนุกรม RS - 232C	10
2.3.2 วงจรภายในและรีจิสเตอร์ของพอร์ตอนุกรม RS-232	12
2.3.3 ลักษณะสัญญาณอินพุทและเอาต์พุทของพอรต์ RS-232	14
2.3.4 แอดเดรสของพอร์ตอนุกรม	14
2.3.5 การใช้งานพอร์ตอนุกรมของ 89C51	15
2.4 กล้องอินฟราเรด	17
2.5 สเต็ปมอเตอร์ (STEPPING MOTOR)	17
2.5.1 ชนิดและโครงสร้างของสเต็ปมอเตอร์	17
2.6 หลักการของรีโมตคอนโทรล	21

สารบัญ(ต่อ)

	หน้าที่
บทที่ 3 ออกแบบและส่วนประกอบของโครงงาน	23-40
3.1 หุ่นยนต์	23
3.2 วงจรรับส่งอินฟราเรด	25
3.21 วงจรส่งอินฟราเรด	25
3.22 ตัวรับอินฟราเรด	27
3.4 การควบคุมรถ	29
3.5 ส่วนการติดต่อกับผู้ใช้	30
3.6 ส่วนการประมวลผล	31
บทที่ 4 ผลการทดลอง	41-42
4.1 การทดลอง	41
4.2 การทดลองการทำงานของรถ	41
บทที่ 5 บทสรุปและวิจารณ์	43
5.1 สรุปผลการดำเนินงาน	43
5.2 ปัญหาที่เกิดและแนวทางแก้ไข	43
5.3 แนวทางพัฒนาในอนาคต	43
บรรณานุกรม	
ภาคผนวก	

สารบัญภาพ

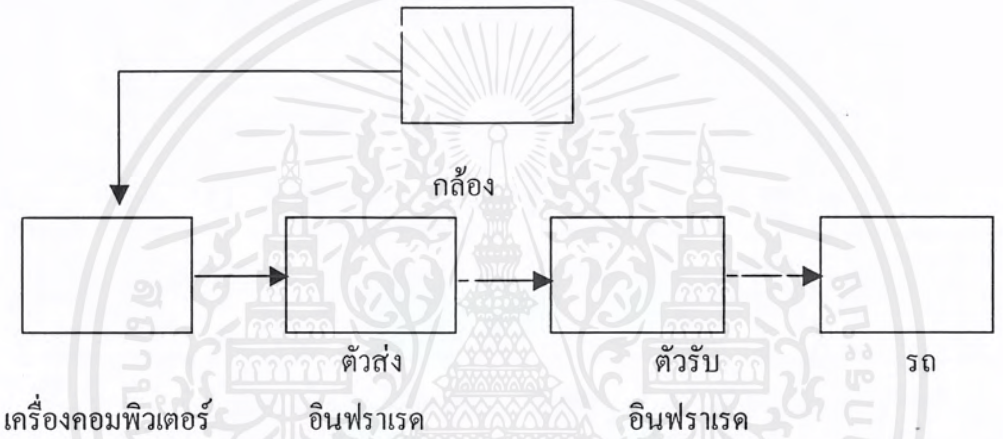
รูปที่	หน้าที่
รูปที่ 2.14แสดงมุมของโรเตอร์เทียบกับกระแสไฟฟ้าที่จ่ายแก่เฟสต่างๆ 8 ตำแหน่ง	20
รูปที่ 2.15 บล็อกไดอะแกรมแสดงโครงสร้างและหลักการทำงานของระบบรีโมตคอนโทรล	21
รูปที่ 3.1 แสดงรูปโครงสร้างของหุ่นยนต์เมื่อมองมาจากข้างบน	23
รูปที่ 3.2 แสดงรูปโครงสร้างของหุ่นยนต์เมื่อมองจกด้านข้าง	24
รูปที่ 3.3 แสดงการเคลื่อนที่ในรูปแบบต่างๆ	24
รูปที่ 3.4 ลักษณะสัญญาณโทนเบิรสต์	25
(ก) สัญญาณพัลส์ข้อมูล	
(ข) สัญญาณโทนเบิรสต์	
รูปที่ 3.5 รูปวงจรตัวส่งอินฟราเรด	26
รูปที่ 3.6 รูปวงจรตัวรับอินฟราเรด	27
รูปที่ 3.7 รูปวงจรแหล่งจ่ายไฟของหุ่นยนต์	28
รูปที่ 3.8 แสดงส่วนประกอบของโปรแกรมที่เขียนด้วย Visual Basic 6	30
รูปที่ 3.9 ประกอบการอธิบายการหาฟังก์ชั่น Point 2 Point	32
รูปที่ 3.10 แสดงจุดที่ใช้ในการตรวจสอบของฟังก์ชั่น CarArea	32
รูปที่ 3.11 แสดงการจำลองจุดที่ใช้แทนหัวและท้ายของรถ	35
รูปที่ 3.12 แสดงเครื่องหมายอง Dx ละ Dy ที่มุมต่างๆกัน	35
รูปที่ 3.13 แสดงตำแหน่งเมื่อรถหยุดเดิน	36
รูปที่ 3.14 แสดงการหันขวาของรถ	37
รูปที่ 3.15 แสดงการหันซ้ายของรถ	37
รูปที่ 4.1 แสดงการทดลองคลิ๊กเมาส์	41
รูปที่ 4.2 (A1,A2,A3,A4) แสดงผลการทดลองการหลบสิ่งกีดขวางของหุ่นยนต์	41-42

บทที่2
ทฤษฎีและหลักการ

ขั้นตอนการทำงาน

เริ่มต้นที่เครื่องคอมพิวเตอร์ป้อนคำสั่งที่ใช้ควบคุมหุ่นยนต์โดยส่งข้อมูลออกทางพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ผ่านตัวส่งอินฟราเรด เมื่อไมโครคอนโทรลเลอร์ในรถได้รับคำสั่งก็จะทำการเดินตามคำสั่งที่รับมา

บล็อกไดอะแกรมการทำงาน



รูปที่2.1บล็อกไดอะแกรมแสดงการทำงานของรถควบคุมด้วยคอมพิวเตอร์

การทำงานแบ่งออกเป็น 4 ส่วน

ส่วนคอมพิวเตอร์ ส่งข้อมูลในการควบคุมรถใช้พอร์ตอนุกรมที่จะส่งให้ตัวส่งอินฟราเรด

ส่วนรับภาพ ใช้กล้องดิจิตอลเป็นตัวจับภาพจากทางทางด้านบนโดยครอบคลุมพื้นที่สำรวจส่งสัญญาณภาพมาให้คอมพิวเตอร์ผ่านทางพอร์ต ตัวกล้องจะถูกยึดติดกับเสาเพื่อให้กล้อสามารถมองจากมุมบนได้

ส่วนการขับเคลื่อนสเป็ปปิ้งมอเตอร์ ส่วนที่ทำให้รถเคลื่อนที่ได้ ใช้สเต็ปปิ้งมอเตอร์ 2 ตัว สามารถควบคุมทิศทางได้จากพัลส์ที่ส่งไปให้มอเตอร์

ส่วนไมโครคอนโทรลเลอร์ ควบคุมการทำงานของรถ โดยรับคำสั่งการควบคุมจาดคอมพิวเตอร์ที่ส่งมาทางพอร์ตอนุกรม ผ่าตัวรับอินฟราเรด ทำกาควบคุมการหมุนมอเตอร์ให้เป็นไปตามคำสั่งโดยการส่งพัลส์

ส่วนการติดต่อข่าวสาร ทำหน้าที่เป็นตัวเชื่อมติดต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ผ่านพอร์ตอนุกรมด้วยตัวตัวส่งอินฟราเรด

2.2 ไมโครคอนโทรเลอร์

ไมโครคอนโทรเลอร์ในตระกูล MCS 51 เป็นไมโครคอนโทรเลอร์ขนาด 8 บิตที่มีอุปกรณ์สนับสนุนประกอบอยู่ภายใน หลายอย่างได้แก่ หน่วยความจำ สำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บโปรแกรม ตัวตั้งเวลา/ตัวนับ อุปกรณ์รับส่งข้อมูลแบบอนุกรม เนื่องจากโครงสร้างของไมโครคอนโทรเลอร์มีอุปกรณ์สนับสนุนประกอบอยู่ภายในนี้นี่เอง ทำให้การใช้งานเพิ่มขึ้นและมีประสิทธิภาพมากขึ้นโดยไม่ต้องมีการเชื่อมต่ออุปกรณ์เพิ่มเติมมากเหมือนกับตัวไมโครโปรเซสเซอร์ทั่วไป นอกจากนี้เราต้องการใช้งานไมโครคอนโทรลเลอร์ร่วกับอุปกรณ์อื่นเพิ่มเติมเช่น ไอซี /8255 หรือหน่วยความจำภายนอก เรายังสามารถนำมาเชื่อมต่อเพิ่มเติมเข้ากับไมโครคอนโทรเลอร์ได้อีก

ตาราง 2.1 ไมโครคอนโทรเลอร์ตระกูล MCS 51 มีให้เลือกใช้หลายเบอร์แสดงดังรูป

ชื่อเบอร์	หน่วยความจำภายใน		จำนวนไทเมอร์ / เคาน์เตอร์	จำนวน อินเตอร์รัปต์
	เก็บโปรแกรม	เก็บข้อมูล		
8052AH	8k*8 ROM	256*8 RAM	3*16 - Bit	6
8051AH	4k*8 ROM	128*8 RAM	2*16 - Bit	5
8051	4k*8 ROM	128*8 RAM	2*16 – Bit	5
8032AH	ไม่มี	256*8 RAM	3*16 - Bit	6
8031AH	ไม่มี	128*8 RAM	2*16 – Bit	5
8031	ไม่มี	128*8 RAM	2*16 – Bit	5
8751H	4k*8 EPROM	128*8 RAM	2*16 – Bit	5
8751H – 12	4k*8 EPROM	128*8 RAM	2*16 - Bit	5

2.3 คุณสมบัติทั่วไปของไมโครคอนโทรลเลอร์ MCS – 51

คุณสมบัติทั่วไปที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล MCS – 51 มีดังนี้

- เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต
- มีวางจรออสซิลเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
- มีขาสัญญาณอินพุทเอาท์พุทจำนวน 32 บิต
- สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (external data memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 k

- สามารถเชื่อมต่อหน่วยความจำโปรแกรมภายนอก (external program memory) โดยอ้างตำแหน่ง แอดเดรสได้ถึง 64 k
- มีหน่วยความจำโปรแกรมภายในตัว (on – chip data memory) ขนาด 128 ไบท์ โดยเฉพาะเบอร์ 8032 และ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 ไบท์
- หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะบิตทำได้ง่าย ส่งผลให้การเขียนโปรแกรมทำได้ง่ายมากขึ้น
- มีไทเมอร์/เคาน์เตอร์ (timer/counter) ขนาด 16 บิต จำนวน 2 ตัว โดยเฉพาะเบอร์ 8032 และ8052 จะมีไทเมอร์/เคาน์เตอร์ จำนวน 3 ตัว
- การอินเตอร์รัปต์สามารถทำได้จาก5 แหล่งกำเนิดโดยเฉพาะเบอร์ 8032 และ 8052 จะทำการอินเตอร์รัปต์ได้จาก 6 แหล่งกำเนิด โดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้เป็น 2 ระดับ
- มีพอร์ตสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานเป็นแบบฟูลดูเพล็กซ์ (full duplex)
- มีคำสั่งในการคำนวณทางคณิตศาสตร์และทางตรรกศาสตร์
- คำสั่งโดยส่วนใหญ่ใช้เวลาทำงานเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกะเฮิรตต้องการแหล่งจ่ายไฟ 5 โวลท์ เพียงชุดเดียว

2.4 โครงสร้างภายนอกของ MCS – 51

ไมโครคอนโทรลเลอร์ตระกูล MCS – 51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูป 2.2 สำหรับหน้าที่การใช้งานของแต่ละขามีดังนี้

รูปที่ 2.2 แสดงการจัดตำแหน่งขาต่างๆของไมโครคอนโทรลเลอร์ตระกูล MCS - 51

- ขา V_{CC} เป็นขาป้อนแรงดันไฟเลี้ยง +5 โวลท์
- ขา V_{SS} เป็นขากราวด์
- ขาพอร์ต 0 (Port 0) มี 8 ขา ได้แก่ ขา $P_{0.0} - P_{0.7}$ เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้ขาพอร์ตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่สามารถนำมาใช้เป็นพอร์ตอินพุทอิมพีแดนซ์สูงได้

นอกจากพอร์ตนี้จะใช้งานเป็นอินพุทเอาท์พุทแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบต์ต่ำ($A_0 - A_7$) ซึ่งจะใช้งานเป็นแบบมัลติเพล็กซ์กับการรับส่งข้อมูลขนาด 8 บิต ($D_0 - D_7$)

- ขาพอร์ต 1 (Port 1) มี 8 ขา ได้แก่ ขา $P_{1.0} - P_{1.7}$ เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ตเพื่อกำหนดให้เป็นพอร์ตอินพุท นอกจากนี้สำหรับเบอร์ 8032 และ 8052 ขาพอร์ต $P_{1.0}$ และ $P_{1.1}$ จะถูกนำมาใช้งานเป็นขา T2 และ T2EX ตามลำดับด้วย
- ขาพอร์ต 2 (Port 2) มี 8 ขา ได้แก่ $P_{2.0} - P_{2.7}$ เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุท นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุทเอาท์พุทแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบท์สูง ($A_8 - A_{15}$)
- ขาพอร์ต 3 (Port 3) มี 8 ขา ได้แก่ $P_{3.0} - P_{3.7}$ เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุท นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุทเอาท์พุทแล้วมันยังถูกใช้งานในหน้าที่พิเศษต่างๆดังแสดงในตารางที่ 2.2

ตาราง 2.2 แสดงหน้าที่พิเศษของแต่ละขาของพอร์ต P3

ขาพอร์ต	หน้าที่พิเศษ
$P_{3.0}$	RXD (serial input port)
$P_{3.1}$	TXD (serial output port)
$P_{3.2}$	$\overline{\text{INT0}}$ (external interrupt 0)
$P_{3.3}$	$\overline{\text{INT1}}$ (external interrupt1)
$P_{3.4}$	T0 (Timer 0 external input)
$P_{3.5}$	T1(Timer 1 external input)
$P_{3.6}$	$\overline{\text{WR}}$ (external data memory write strobe)
$P_{3.7}$	$\overline{\text{RD}}$ (external data memory write strobe)

- ขารีเซต (RST) ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสถานะเป็น 1 อย่างน้อยนาน 2 แมชีนไซเคิล ในขณะที่ออสซิเลเตอร์ยังทำงานอยู่

- ขา ALE / $\overline{\text{PROG}}$ เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตซ์ (latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่เป็นอินพุทรับพัลซ์ในการโปรแกรม (program pulse input) ในส่วนของหน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ในตระกูล MCS – 51 ที่มีหน่วยความจำภายในเป็น EPROM

- ขา $\overline{\text{PSEN}}$ (Program Store Enable) ทำหน้าที่เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรบจำนวน 2 ครั้ง ในแต่ละแมชีนไซเกิล แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณสโตรบแต่อย่างใด

- ขา $\overline{\text{EA}}$ /VPP (External Access enable /VPP) ขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมจากภายในหรือภายนอก โดยถ้ามีสถานะเป็น 0 จะหมายถึงให้ไมโครคอนโทรลเลอร์รับคำสั่งจากหน่วยความจำภายนอกที่ตำแหน่งแอดเดรส 0 - 0FFH (0 – 1FFH ถ้าเป็นเบอร์ 8052)
อย่างไรก็ตาม ถ้าบิตป้องกัน (security bit) ในหน่วยความจำ EPROM ถูกโปรแกรมไว้ ไมโครคอนโทรลเลอร์จะไม่รับคำสั่งจากหน่วยความจำภายนอกเลย นอกจากนี้ขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (V_{PP}) ขนาด 21 โวลท์ เพื่อใช้ในระหว่างการโปรแกรม EPROM

- ขา $XTAL_1$ และขา $XTAL_2$ เป็นขาอินพุทและเอาท์พุทของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (inverting oscillator amplifier) สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

2.5 การจัดหน่วยความจำ

ในไมโครคอนโทรลเลอร์ตระกูล MCS – 51 แบ่งชนิดหรือหน้าที่ของหน่วยความจำออกเป็น 2 ส่วนคือ หน่วยความจำโปรแกรม (Program Memory) และหน่วยความจำข้อมูล (Data Memory)

หน่วยความจำโปรแกรมจะใช้สำหรับเก็บโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ซึ่งบางเบอร์จะมีหน่วยความจำในส่วนนี้อยู่ภายในตัว โดยอาจจะมีขนาดไม่เท่ากัน หรือเป็นหน่วยความจำต่างชนิดกัน เช่น บางเบอร์เป็น ROM และบางเบอร์อาจเป็น EPROM และบางเบอร์อาจไม่มีหน่วยความจำในส่วนนี้เลย โปรแกรมการทำงานจะถูกเก็บไว้ยังหน่วยความจำโปรแกรมภายนอกทั้งหมด

สำหรับหน่วยความจำข้อมูลจะใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่างๆจากการทำงานของโปรแกรม ซึ่งใน MCS – 51 ทุกเบอร์จะมีหน่วยความจำในส่วนนี้อยู่จำนวนหนึ่งแต่อาจมีขนาดมากน้อยต่างกันไปในแต่ละเบอร์สำหรับการจัดโครงสร้างของหน่วยความจำทั้งในส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล

2.6 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมสามารถแบ่งออกได้เป็น2 ส่วนคือ หน่วยความจำโปรแกรมภายในและหน่วยความจำโปรแกรมภายนอก หน่วยความจำโปรแกรมภายในจะถูกเลือกใช้งานถ้าขาสัญญาณ $\overline{EA}$ มีค่าเป็น 1 โดยจะถูกใช้งานในช่วงแอดเดรส 0 – 0FFFH (หรือช่วงแอดเดรส 0 – 0FFFH ในเบอร์ 8052) นอกเหนือจากช่วงแอดเดรสนี้จะใช้หน่วยความจำโปรแกรมภายนอกทั้งหมด ในกรณีตรงกันข้ามถ้าขาสัญญาณ $\overline{EA}$ มีค่าเป็น 0 ในช่วงแอดเดรส 0 – 0FFFH (หรือช่วงแอดเดรส 0 – 0FFFH ในเบอร์ 8052) จะถูกใช้จากหน่วยความจำภายนอก หรือกล่าวได้ว่าถ้าขาสัญญาณ $\overline{EA}$ มีค่าเป็น 0 จะเป็นการเลือกใช้หน่วยความจำโปรแกรมภายนอกทั้งหมดตลอดช่วงแอดเดรส

2.7 หน่วยความจำข้อมูล

หน่วยความจำข้อมูลสามารถแบ่งออกได้เป็น 2 ส่วน คือ หน่วยความจำข้อมูลภายในและหน่วยความจำข้อมูลภายนอก สำหรับหน่วยความจำข้อมูลภายในยังแบ่งออกได้เป็น 2 ส่วนย่อย คือ ส่วนที่ใช้เก็บข้อมูลทั่วไปและส่วนที่ใช้เป็นรีจีสเตอร์หน้าที่พิเศษหรือ SFR (Special Function Register) โดยส่วนที่ใช้เก็บข้อมูลทั่วไปจะถูกใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่างๆจากการทำงานของโปรแกรม ส่วนรีจีสเตอร์หน้าที่พิเศษจะถูกใช้งานเป็นรีจีสเตอร์ควบคุมการทำงานและบอกสถานะการทำงานของไมโครคอนโทรลเลอร์

ตำแหน่งแอดเดรส	(MSB) บิตแอดเดรส (LSB)								รีจิสเตอร์หน้าที่พิเศษ
	WDT	T32	SERR	IZC	P3HZ	P2HZ	P1HZ	ALE	
0F8H	FF	FE	FD	FC	FB	FA	F9	F8	IOCON
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	CY	AC	F0	RS1	RS0	OV	F1	P	
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0CDH	ไม่สามารถเข้าถึงได้ระดับบิต								TH2
0CCH	ไม่สามารถเข้าถึงได้ระดับบิต								TL2
0CBH	ไม่สามารถเข้าถึงได้ระดับบิต								RCAP2H
0CAH	ไม่สามารถเข้าถึงได้ระดับบิต								RCAP2L
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	
0C8H	CF	CE	CD	CC	CB	CA	C9	C8	T2CON
	PCT	-	PT2	PS	PT1	PX1	PT0	PX0	
0B8H	BF	-	BD	PC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA		ET2	ES	ET1	EX1	ET0	EX0	
0A8H	AF	-	AD	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
99H	ไม่สามารถเข้าถึงได้ระดับบิต								SBUF
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
8DH	ไม่สามารถเข้าถึงได้ระดับบิต								TH1
8CH	ไม่สามารถเข้าถึงได้ระดับบิต								TH0
8BH	ไม่สามารถเข้าถึงได้ระดับบิต								TL1
8AH	ไม่สามารถเข้าถึงได้ระดับบิต								TL0
89H	ไม่สามารถเข้าถึงได้ระดับบิต								TMOD
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
87H	ไม่สามารถเข้าถึงได้ระดับบิต								PCON
83H	ไม่สามารถเข้าถึงได้ระดับบิต								DPH
82H	ไม่สามารถเข้าถึงได้ระดับบิต								DPL
81H	ไม่สามารถเข้าถึงได้ระดับบิต								SP
80H	87	86	85	84	83	82	81	80	P0

รูปที่ 2.3 แสดงการจัดหน่วยความจำและตำแหน่งของรีจีสเตอร์หน้าที่พิเศษต่างๆ

ไมโครคอนโทรลเลอร์ตระกูล MCS – 51 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในขนาด 128 ไบท์เป็นอย่างน้อย และบางเบอร์อาจมีถึงขนาด 256 ไบต์

2.8 รีจิสเตอร์หน้าที่พิเศษ (SFR)

รีจีสเตอร์หน้าที่พิเศษมีบทบาทอย่างมากในการควบคุมการทำงานของไมโครคอนโทรลเลอร์ และทำให้การเขียนโปรแกรมสามารถทำได้สะดวกมากขึ้น รีจีสเตอร์หน้าที่พิเศษทำหน้าที่สำคัญคือควบคุมการทำงานในส่วนต่างๆภายในไมโครคอนโทรลเลอร์และทำหน้าที่แสดงสถานะการทำงาน ซึ่งในรีจีสเตอร์หน้าที่พิเศษบางตัวยังสามารถเข้าถึงได้ในระดับบิต (bit addressable) ด้วย ดังแสดงรูปการจัดหน่วยความจำและตำแหน่งของรีจีสเตอร์หน้าที่พิเศษต่างๆในรูปที่ 2.5

2.9 รีจีสเตอร์ใช้งานทั่วไป

รีจีสเตอร์ใช้งานทั่วไปมีไว้สำหรับให้ผู้เขียนโปรแกรมสามารถนำข้อมูลไปพักไว้ชั่วคราว หรือใช้งานทั่วไปได้ตามต้องการ ซึ่งรีจิสเตอร์ใช้งานทั่วไปนี้มีอยู่ด้วยกัน 8 ตัว คือรีจีสเตอร์ R_0 - R_7 โดยรีจีสเตอร์ทั้ง 8 ตัว ถูกจัดให้อยู่รวมกันและมีให้เลือกใช้ถึง 4 แบงค์ (bank) นั่นคือมีรีจีสเตอร์ใช้งานทั่วไปถึง 32 ตัวให้ใช้งาน เพียงแต่การเลือกใช้รีจิสเตอร์ R_0 - R_7ในแบงค์ใดแบงค์หนึ่งจะถูกกำหนดจากบิต RS0 , RS1 ในรีจีสเตอร์หน้าที่พิเศษ PSW ดังนั้นการเลือกใช้จึงเลือกได้เพียงแบงค์เดียวในขณะใดขณะหนึ่ง อย่างไรก็ตาม ค่าข้อมูลที่เก็บได้ไว้ในรีจิสเตอร์แบงค์ใดก็ตามที่มีชื่อเดียวกันแต่อยู่คนละแบงค์จะไม่มีผลซึ่งกันและกันเลย ทำให้ผู้เขียนโปรแกรมใช้งานรีจิสเตอร์ทั่วไปนี้ได้ทั้ง 32 ตัวอย่างเต็มที่และไม่ยุ่งยากในการเขียนโปรแกรม

2.10 การรับส่งข้อมูลผ่าน Serial port

Microcontroller MCS-51 มี Serail Port ซึ่งสามารถรับและส่งข้อมูลแบบอนุกรมได้โดยใช้ไม่ต้องต่อไอซีเพิ่มเติมเข้าไป ทำให้มีความสะดวกในการนำไปประยุกต์ใช้งานที่ต้องมีการติดต่อข้อมูลแบบอนุกรม Serail Port ที่มีใน MCS-51 สามารถทำงานได้ในแบบ Full Duplex หมายความว่ามันสามารถรับและส่งข้อมูลได้พร้อม ๆ กัน โดยมีการ Bufffer ในการรับข้อมูลให้ด้วย กล่าวคือ มันสามารถกำหนดการรับของไบต์ที่สองที่ถูกรับเข้ามา ก่อนที่ไบต์แรกซึ่งได้รับเข้ามาก่อนจะถูกอ่านจาก Receive Register (แต่ถ้าไบต์แรกยังไม่ถูกอ่านเมื่อเวลาที่การรับของไบต์ที่สองสิ้นสุดลง หนึ่งไบต์ในสองไบต์จะหายไป) Serail Port จริง ๆ แล้ว ประกอบไปด้วยรีจิสเตอร์ขนาด 8 บิทจำนวนสองตัว ซึ่งมีชื่อเรียกดังนี้ คือ Receive Register และ Transmit Register ซึ่งรีจิสเตอร์ทั้งสองมีตำแหน่งเดียวกันใน SFR คือ รีจิสเตอร์ SBUF โดยการเข้าถึงรีจิสเตอร์แต่ละตัว ซีพียูจะรู้เองว่าใช้ต้องการติดต่อกับรีจิสเตอร์ตัวใด เพราะในการเขียนข้อมูลไปที่รีจิสเตอร์ SBUF จะหมายถึงโหลด ค่าไปยัง Transmit Register

ส่วนการอ่านข้อมูลในรีจิสเตอร์ SBUF หมายถึงการรับข้อมูลจาก Receive Register การสื่อสารข้อมูลอนุกรมนิยมนำมาใช้ในการโอนถ่ายข้อมูล ระหว่างระบบควบคุมด้วยไมโครคอนโทรลเลอร์หรืออุปกรณ์คอมพิวเตอร์ต่างๆ เพราะใช้เส้นสัญญาณเพียง 2 หรือ 3 เส้น สำหรับการรับส่งข้อมูลเท่านั้น โดยจะเป็นการรับส่งข้อมูลทีละบิต จนครบหนึ่งไบต์ คือ $D_0 - D_7$ โดยจะส่ง D_0 ออกไปก่อนแล้วตามด้วย D_1 ไปเรื่อยๆจนถึง D_7 การส่งข้อมูลแบบอนุกรมนี้จะช้ากว่าการส่งแบบขนาน แต่มีข้อดีคือถ้าเป็นการส่งแบบขนาน ที่ต้องส่งข้อมูลระยะไกลจะทำให้สิ้นเปลืองสายสัญญาณมาก แต่เนื่องจากการประมวลข้อมูลในระบบคอมพิวเตอร์เป็นลักษณะแบบขนาน ทำให้ต้องมีการแปลงผันข้อมูลให้เหมาะสมก่อนการส่งข้อมูล รวมทั้งภายหลังการรับข้อมูลแบบอนุกรม

ภายใน MCS – 51 จะมีตัวรับส่งข้อมูลอนุกรมอยู่ภายใน โดยตัวส่งข้อมูลจะทำหน้าที่แหล่งข้อมูลจากการประมวลผลที่เป็นแบบขนานให้เป็นข้อมูลแบบอนุกรมก่อน(Parallel – to – serial conversion) แล้วส่งข้อมูลออกมาโดยมีสัญญาณนาฬิกาเป็นตัวกระตุ้น ตัวแปลงข้อมูลนี้อาจพิจารณาง่ายๆว่าเป็นชิฟรีจีสเตอร์ (shift register) ที่ทำหน้าที่เลื่อนข้อมูลออกมา ถ้าสัญญาณนาฬิกา
มีความถี่สูงจะทำให้ส่งข้อมูลได้เร็ว ส่วนตัวรับข้อมูลจะเป็นการรับข้อมูลแบบอนุกรมและแปลงให้เป็นข้อมูลแบบขนานเพื่อใช้ในการประมวลผล (Serial – to – Parallel conversion) ใน MCS – 51 ตัวแปลงข้อมูลทั้งสองประเภทนี้จะเป็นรีจิสเตอร์ภายใน

การส่งข้อมูลอนุกรมด้วยชุดคำสั่งของ MCS – 51 นั้น ทำได้ง่ายเพียงการเขียนโปรแกรมโอนถ่ายข้อมูลแบบขนานให้กับรีจิสเตอร์ที่จัดการข้อมูลอนุกรมเท่านั้น หรือการรับข้อมูลก็อ่านจาก รีจิสเตอร์นี้เช่นกัน โดยไม่ต้องเขียนโปรแกรมเพื่อแปลงผันข้อมูลระหว่างอนุกรมและขนาน รวมทั้งการเลื่อนบิต (shift bit) จากขาสัญญาณแต่อย่างใด จึงนับว่าอำนวยความสะดวกให้กับผู้ใช้งานมาก

2.11 พอร์ตสื่อสารข้อมูลอนุกรม RS – 232C

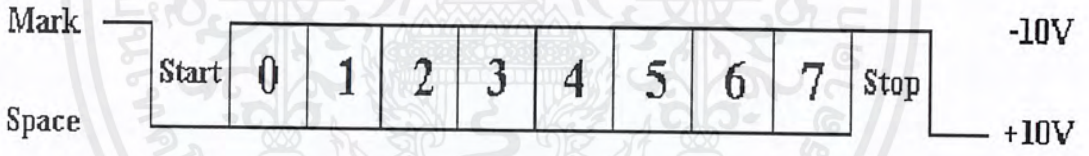
มาตฐานการเชื่อมต่อต่อแบบอนุกรม RS 232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอซิงโครนัส 2 ทิศทาง โดยมาตฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อใช้ในการส่งการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากมเด็มนั้นนี้สื่อสารผ่านสายโทรศัพย์ไปยังคอมพิวเตอร์อีกชุดหนึ่งซึ่งอยู่ห่างไกล โดยคณะกรรมการที่เรียกว่าสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุตมีระดับสัญญาณตั้งแต่ –3 ถึง –12 V แสดงว่ามีข้อมูล(Mark)และ +3 ถึง +12 แสดงว่าเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment:DTE)กับวงจรข้อมูลปลายทาง(Data Terminating:DCE) ไว้ว่าอุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวอย่างเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสารในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DCE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

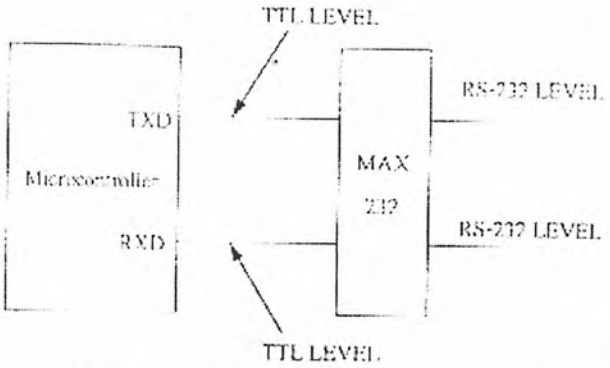
สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม, RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

รูปแบบของข้อมูลอนุกรมที่ใช้งานกันอย่างแพร่หลายเป็นลักษณะ การสื่อสารแบบอะซิงโครนัส (Asyncronous communication) และเป็นไปตามมาตรฐาน RS –232C รูปแบบของข้อมูลภายในเส้นสัญญาณจะมีบิตข้อมูลเพิ่มขึ้นจากบิตข้อมูลสื่อสารจริง เช่น บิตเริ่มต้น (start bit) และบิตสิ้นสุด (stop bit) เพื่อนำมาใช้ในการบอกจังหวะการเริ่มต้นและสิ้นสุดข้อมูลแต่ละไบต์



รูปที่ 2.4 รูปแบบสัญญาณข้อมูลอนุกรมแบบอะซิงโครนัส โดยมีระดับสัญญาณแบบ RS – 232C

ตามมาตรฐาน RS – 232C ระดับสัญญาณข้อมูลของวงจร (TTL voltage levels) ซึ่งมีค่าอยู่ในช่วง 0 ถึง 5 โวลท์ จะต้องนำมาปรับให้เป็นระดับแบบ RS – 232C (RS – 232 voltage levels) ซึ่งมีค่าสูงขึ้นในช่วงจาก + 15 ถึง – 15 โวลท์ รูปที่ 2.7 แสดงการเชื่อมต่อ RS – 232C ของ
ไมโครคอนโทรลเลอร์โดยมี IC MAX232 เป็นตัวปรับระดับสัญญาณ (RS – 232C Line driver and receiver)



รูปที่ 2.5 การเชื่อมต่อสัญญาณอนุกรมของ Microcontroller

- Recive Data: RXD หรือ RX ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์บัฟเฟอร์

- Transmitted Data:TD sinv TXD ขานี้ใช้ส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

- Singnal Ground :GND ขากราวด์ของระบบ

2.12 วงจรภายในและรีจิสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์โดยทั่วไปสมารถต่อพอร์ตอนุกรม RS-232 สูงสุดได้ 4 พอร์ต ซึ่งจะชื่อเรียกเป็น COM1,COM2,COM3 และ COM4 ซึ่งอนุกรมแต่ละตัวต่างก็ใช้งาน UART ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอกเช่นเดียวกัน

การทำงานภายในของพอร์ตอนุกรม ซึ่งประกอบไปด้วยรีจีสเตอร์ขนาด 8 บิต ค ตัวที่ ใช้งานร่วมกับ UART แอดเดรสของรีจิสเตอร์ภายในของพอรต์อนุกรมสามารถคำนวณได้จากค่ารีจิสเตอร์พื้นฐานของพอร์ตอนุกรม ยกตัวอย่างพอร์ตอนุกรม COM1 มีแอดเดรสอยู่ที่ 3F8H ตำแน่งของรีจิสเตอร์ต่างๆจะเป็นตำแหน่งที่บวกเข้าไปกับค่า 3F8H โดยรีจิสเตอร์ที่ใช้งานกับพอร์ตอนุกรมมีดังนี้

00H เป็นรีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาหรือเตรียมข้อมูลก่อนที่จะส่งไป

01H รีจิสเตอร์อีนาเบิลการอินเตอร์รัปต์ ใช้ในการเซตโหมดการอินเตอร์รัปของพอร์ตอนุกรม

02H รีจิสเตอร์แสดงโหมดการอินเตอร์รัปต์ ใช้เพื่อตรวจสอบโหมดของการอินเตอรรัปเมื่อมีการอินเตอร์รัปต์เกิดขึ้น

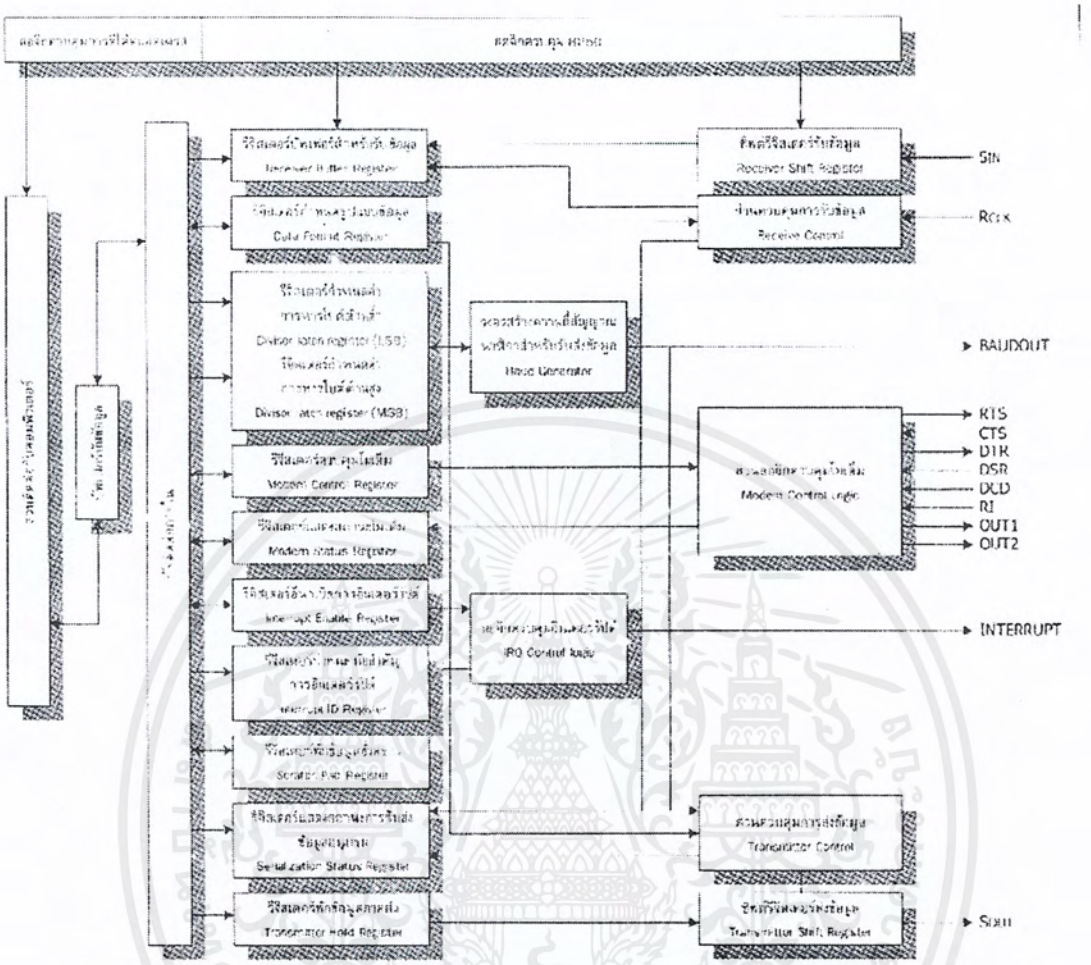
03H รีจิสเตอร์กำหนดรูปแบบของข้อมูล

04H รีจิสเตอร์ควบคุมโมเด็ม ใช้ตรวจสอบบิตสำหรับติดต่อกับโมเด็มเช่น RTS หรือ DTR

05H รีจิสเตอร์แสดงสถานะการรับและการส่งแบบอนุกรม

06 Hริจิสเตอร์แสดงสถานะของโมเด็ม ซึ่งจะแสดงสถานะของขา DCD,RI,DSR และ CTS

07H รีจิสเตอร์สำหรับการเก็บข้อมูล



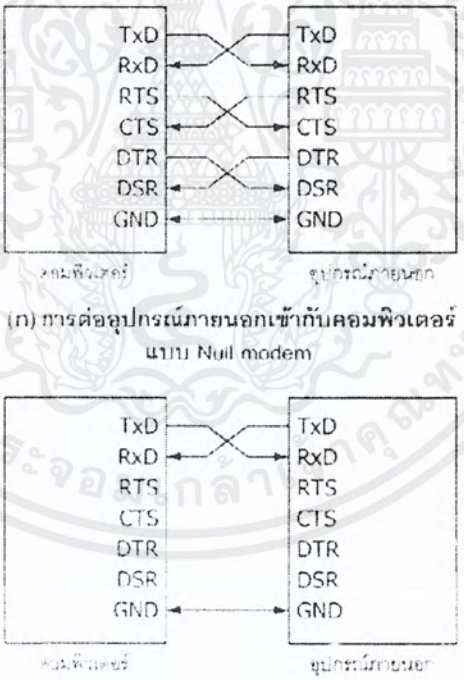
รูปที่2.6 ไดอะแกรมการทำงาน ภายในของพอร์ตอนุกรมของเครื่องคอมพิวเตอร์

2.13 ลักษณะสัญญาณอินพุตและเอาต์พุต ของพอรต์ RS-232

สัญญาณเอาต์พุต ที่ใช้ควบคุม (RTS และ DTR) และสัญญาณแสดงสถานะอินพุต (CTR,DSR และ DCD) ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและรับจะไม่ถูกกลับสถานะ UART จะให้ระดับสัญญาณเอาต์พุตออกมาเป็นทีทีแอล เท่านั้น ดังนั้นเมื่อสัญญาณถูกส่งออกมาจาก UART จึงต้องส่งเข้าสู่วงจรขับเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรขับที่ใช้ทั้งภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะ ดังแสดงเป็นรูปบล็อกไดอะแกรม

2.14 แอดเดรสของพอร์ตอนุกรม

แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแน่งดังนี้คือ



รูปที่ 2.7 แอดเดรสพื้นฐานของพอร์ตอนุกรม

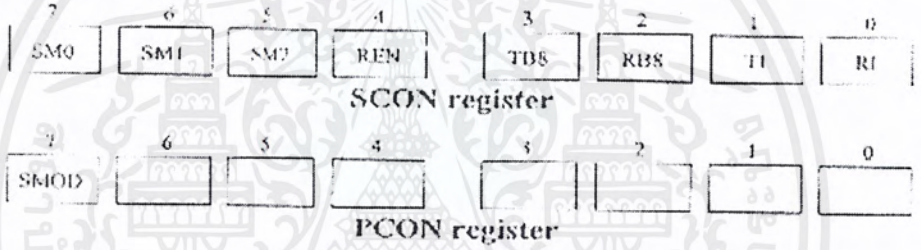
สำหรับความเร็วของการส่งข้อมูลอนุกรมมักจะเรียกกันว่า อัตราบอด (baud rate) ซึ่งอธิบายโดยง่ายสำหรับรูปแบบข้อมูลอนุกรมแบบอะซิงโครนัสว่า เป็นจำนวนของบิตข้อมูลภายในหนึ่งช่วงเวลา อัตราบอดที่ใช้งานกันโดยทั่วไป เช่น 1200 , 2400, 4800 และ 9600 เป็นต้น

2.15 การใช้งานพอร์ตอนุกรมของ 89C51

รี จิสเตอร์ที่เกี่ยวข้องกับการสื่อสารอนุกรมของ 89C51 ประกอบด้วย รีจิสเตอร์ SBUF สำหรับเก็บข้อมูลสื่อสาร รีจิสเตอร์ SCON ควบคุมการสื่อสาร รีจิสเตอร์ PCON ควบคุมความเร็วการสื่อสาร และขาสัญญาณ RXD ($P_{3.1}$) เป็นขาสัญญาณที่ต่อเข้ากับอุปกรณ์หรือวงจรภายนอก

เนื่องจากการส่งข้อมูลหนึ่งไบต์ออกไปทางพอร์ตอนุกรมในคราวหนึ่งๆจะใช้เวลามาก ดังนั้น เพื่อไม่ให้เสียเวลาในการประมวลผลไป 89C51 จึงนำแฟล็ก TI และ RI ซึ่งเป็นตำแหน่งบิต อยู่ภายในรีจิสเตอร์ SCON เพื่อแจ้งให้กับผู้ใช้งานทราบ โดยเมื่อใดก็ตามที่มีการส่งข้อมูลแบบอนุกรมเสร็จสิ้น บิตแฟลก TI จะมีค่าเป็น 1 หรือเมื่อได้รับข้อมูลอนุกรมเข้ามาแฟลก RI ก็จะมีค่าเป็น 1 เช่นเดียวกัน

การทำงานของพอร์ตอนุกรมของ MCS –51 มีหลายโหมด โดยเราสามารถโปรแกรมการทำงานได้ในรีจิสเตอร์ SCON โดยความหมายของแต่ละบิตเป็นดังนี้



รูปที่ 2.8 ชื่อและตำแหน่งของบิตต่างๆภายในรี จิสเตอร์ SCON และ PCON (เฉพาะเกี่ยวกับการส่งข้อมูลอนุกรม)

ตารางที่ 2.3 ความหมายบิตต่างๆของรี จิสเตอร์ SCON

ชื่อบิต	ตำแหน่งบิต	ความหมาย
SMO	9FH	บิตเลือกโหมดการทำงานบิต 0
SM1	9EH	บิตเลือกโหมดการทำงานบิต 1
SM2	9DH	บิตเลือกโหมดการทำงานบิต 2
REN	9CH	บิตแฟลกกำหนดการยอมให้มีการรับข้อมูล
TB8	9BH	ค่าของบิต 9 สำหรับการส่งข้อมูลในโหมด 2 และ 3 สามารถ set และ clear ได้โดยโปรแกรม
RB8	9AH	ค่าของบิต 9 เมื่อรับข้อมูลเข้ามา
TI	99H	บิตแฟลกแสดงการอินเตอร์รัปต์ภายหลังการส่งข้อมูลออกไปโดยจะ set เมื่อส่งข้อมูลออกไปหมดแล้ว และสามารถ clear ได้โดยโปรแกรม
RI	98H	แฟลกแสดงการอินเตอร์รัปต์ภายหลังรับข้อมูลเข้ามา สามารถ clear ได้โดยโปรแกรม

ตารางที่ 2.4 โหมดต่างๆของการรับส่งแบบอนุกรม

SM0	SM1	โหมด	ความหมาย	อัตราบอด
0	0	0	ชิพรีจิสเตอร์	เปลี่ยนแปลงไม่ได้
0	1	1	8 บิต UART	เปลี่ยนแปลงได้โดยการกำหนดจาก Timer
1	0	2	9 บิต UART	เปลี่ยนแปลงไม่ได้
1	1	3	9 บิต UART	เปลี่ยนแปลงได้โดยการกำหนดจาก Timer

ในการใช้พอร์อนุกรมจะต้องโปรแกรมให้กับรีจิสเตอร์ SCON เสียก่อนเพื่อกำหนดโหมดการทำงานและลักษณะต่างๆ เช่น

```
MOV SCON, #01010010B
```

เป็นการกำหนดให้พอร์ตอนุกรมทำงานในโหมด 1 และอีนาเบิลให้มีการรับข้อมูล พร้อมกับกำหนดให้ TI เป็นลอจิก 1

ในการส่งข้อมูลทุกโหมดสามารถทำได้โดยเขียนข้อมูลไปยัง SBUF เมื่อข้อมูลถูกส่งไปแล้ว บิต TI จะถูกเซ็ตเป็น 1 ในการส่งข้อมูลจะต้องคอยตรวจสอบบิต TI เพราะว่าถ้า TI ยังไม่เป็น 1 แสดงว่าข้อมูลยังส่งไม่หมด ถ้าหากมีการเขียนข้อมูลไบต์ต่อไปไปยังรีจิสเตอร์ SBUF จะทำให้เกิดข้อผิดพลาดขึ้น สำหรับในการรับข้อมูลบิต REN จะต้องเซ็ตให้เป็น 1 ยกเว้นโหมด 0 เพื่ออนุญาตให้รอรับข้อมูลได้ เมื่อข้อมูลรับเข้ามาเรียบร้อยแล้ว บิต RI จะถูกเซ็ตให้เป็น 1

2.16 กล้อง Infrared

สิ่งที่จำเป็นต้องมีการใช้งานกล้อง Infrared ได้แก่

- เครื่องคอมพิวเตอร์ตั้งแต่รุ่น Pentium 200 ขึ้นไป
- Microsoft Windows 98
- หน่วยความจำ 16 MB
- Video 16-bit color display
- CD-ROM drive
- พอร์ต USB 1 พอร์ต

2.17 สเต็ปมอเตอร์ (STEPPING MOTOR)

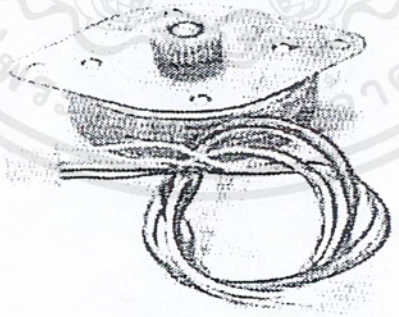
สเต็ปมอเตอร์เป็นมอเตอร์ที่ขับเครื่องด้วยพัลส์ ลักษณะการขับเคลื่อน จะหมุนรอบแกนได้ 360 องศา มีลักษณะไม่ต่อเนื่อง แต่มีลักษณะเป็นสเต็ป โดยแต่ ละสเต็ปจะขับเคลื่อนได้ 1,1.5,1.8 หรือ 2 องศาแล้วแต่ละโครงสร้างของมอเตอร์ที่นำมอเตอร์ไปใช้งานที่ต้องการตำแหน่งแม่นยำ เช่นระบบขับเคลื่อนหัวแม่พิมพ์ในเครื่องพิมพ์ (PRINTER) ระบบขับเคลื่อนหัวอ่านในเครื่องอ่านบันทึกเหล็ก ระบบขับเคลื่อนตำแหน่งของปากกาใน X-Y PLOTER เป็นต้น

2.18 ชนิดและโครงสร้างของสเต็ปมอเตอร์

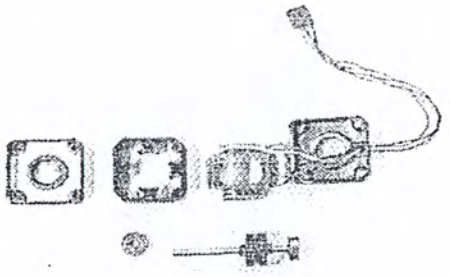
สเต็ปมอเตอร์มีลักษณะดังรูป

รูปที่2.9 สเต็ปมอเตอร์แบบมีสาย 5 เส้น

รูปที่ 2.10 สเต็ปมอเตอร์แบบมีสาย 6 เส้น



รูปที่ 2.11 สเต็ปมอเตอร์แบบไบโพลาร์



รูปที่ 2.12 แสดงภาพถ่ายโครงสร้างสเต็ปมอเตอร์

2.19 สเต็ปมอเตอร์ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

1.แบบแม่เล็กถาวร (PERMANENT MAGNEY_PM)

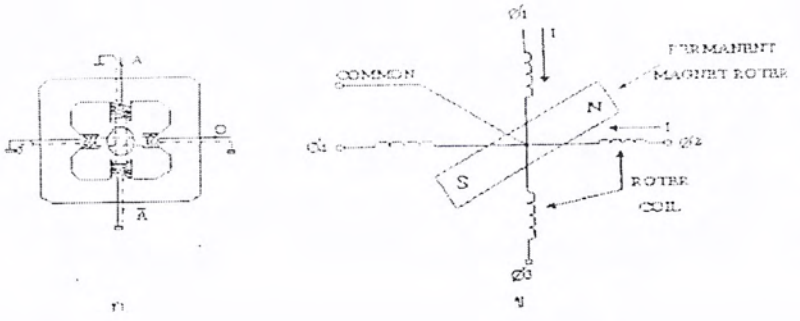
สเต็ปมอเตอร์แบบ PM มีสเตเตอร์ (STATOR)ที่พันขดลวดหลายๆโพลโดยโรเตอร์(ROTOR) เป็นทรงกระบอกฟันเลื่อย และโรเตอร์ทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรงให้กับขดสเตอร์ จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ ทำให้มอเตอร์หมุน มอเตอร์แบบ PM จะเกิดแรงฉุดหยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด

2.แบบแปรค่ารีลักแตนซ์ (VARIABLE RELUCTANCE –VR)

สเต็ปมอเตอร์แบบ VR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟแก่โรเตอร์ทำจากสารเฟอร์โรแมกเน ติก มีลักษณะเป็นฟันเลื่อย รูปทรงกระบอกโดยจะมีความสัมพันธ์ โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะหมุนโรเตอร์ ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆจุดดังนั้นเมื่อป้อนไฟเข้าขดต่างๆในมอเตอร์แตกต่างขดกันไป ก็ทำให้มอเตอร์หมุนไปตำแหน่งต่างๆกันโรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

3.แบบผสม (HYBRID-H)

สเต็ปมอเตอร์แบบ H เป็นลูกผสมๆของ VR กับ PM (โดยจะมีสเตอร์คล้ายกับที่ใช้ใน VR) โรเตอร์มีหมวกหุ้มปลายซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกเหล็กอย่างดีทำให้ได้มุม การหมุนและครั้งน้อยและแม่นยำ ข้อดีก็คือ ให้แรงบิดสูงและมีขนาดกระทัดรัด และให้แรงฉุดโรเตอร์นิ่งกับตอนที่ไม่จ่ายไฟ



รูปที่ 2.13 แสดง (ก) โครงสร้าง (ข) วงจรเทียบเท่า (equivalentcircuit) ของมอเตอร์ชนิด 4 ขด

รูปที่ 2.14แสดงมุมของโรเตอร์เทียบกับกระแสไฟฟ้าที่จ่ายแก่เฟสต่างๆ 8 ตำแหน่ง

จากลักษณะของมุมโรเตอร์หมุนกับกระแสไฟที่ป้อนแก่เฟสต่างๆจะสามารถสั่งงานให้ STEPPING MOTOR หมุนได้อย่างคือ

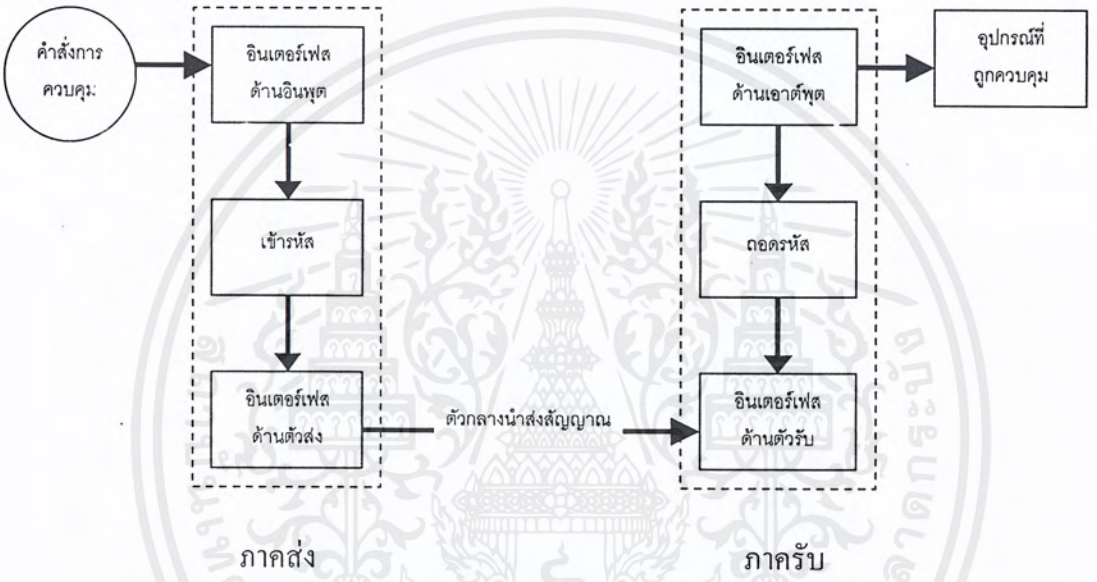
1. แบบจ่ายกระแส ให้ เฟสเดียววนเวียนกันไป เรียก ONE-EXCITATION หรือ HALF DRIVE คือ f1, f2, f3, f4 การ OUT EXCITATION แบบนี้แรงบิดจะน้อย

2. แบบจ่ายกระแสไฟให้พร้อมกันทีเดียว 2f เรียก TWO-EXCITATION หรือ FULL STEP คือ f1f2 , f2f3 , f3f4 , f4f1 หมุนเวียนกันไปแบบนี้แรงบิดจะมาก

3. แบบจ่ายกระแสไฟให้ทีละ 1 เฟส สลับกับ 2 เฟส เรียก ONE-TWO EXCITATION หรือ HALF STEP เหมือนรูปแสดง ของมุมโรเตอร์ แต่แบบนี้จำนวน STEP ทวนเข็มนาฬิกาจะตรงข้าม

2.20 หลักการของรีโมตคอนโทรล

บล็อกไดอะแกรมในรูปที่ 2.13 แสดงโครงสร้างและหลักการทำงานของระบบควบคุมระยะไกลโดยทั่วไป ในลักษณะของการควบคุมแบบทางเดียว เริ่มจากตัวกำหนดคำสั่งที่ใช้สำหรับการควบคุมว่าคำสั่งอะไรบ้าง ชุดคำสั่งทั้งหมดมีกี่คำสั่ง เป็นต้น เมื่อมีการกำหนดรูปแบบของคำสั่งแล้ว รูปแบบของคำสั่งที่ถูกเลือก จะถูกส่งไปยังภาคส่งสัญญาณที่ทำหน้าที่แปลงสัญญาณ หรือรวมสัญญาณควบคุมให้มีรูปแบบที่เหมาะสมกับวงจร



รูปที่ 2.15 บล็อกไดอะแกรมแสดงโครงสร้างและหลักการทำงานของระบบรีโมตคอนโทรล

โดยอาจทำการเข้ารหัสสัญญาณให้แต่ละคำสั่งมีรหัสเฉพาะของตัวมันเองให้เป็นสัญญาณทางไฟฟ้าก่อนที่จะถูกส่งออกไปยังภาครับ โดยตัวอินเตอร์เฟสด้านตัวส่งเพื่อทำหน้าที่ส่งสัญญาณที่ภาครับต้องเข้าใจได้ นั่นก็คือต้องเป็นระบบเดียวกัน สัญญาณไฟฟ้า สัญญาณแสง หรือสัญญาณเสียงความถี่สูง สัญญาณนี้สามารถเดินทางผ่านตัวกลางที่เป็นสายนำสัญญาณ หรือผ่านตัวกลางอากาศ ขึ้นอยู่กับระบบที่ถูกออกแบบ

หากใช้สายสัญญาณเป็นตัวนำสัญญาณจะเรียกว่า ระบบใช้สาย ซึ่งถ้าใช้สัญญาณไฟฟ้าเป็นสัญญาณควบคุม (ที่มีการจัดรูปแบบหรือเข้ารหัสแล้ว) ก็จะใช้สายไฟฟ้าเป็นตัวนำสัญญาณ แต่ถ้าหากใช้สัญญาณแสงเป็นตัวควบคุม ตัวนำสัญญาณจะเป็นเส้นใยแก้วนำแสงหรือไฟเบอร์ออปติก ในกรณีที่สัญญาณควบคุมถูกส่งไปในอากาศ เพื่อเดินทางไปยังเครื่องรับ เช่น การใช้สัญญาณไฟฟ้าในรูปของคลื่นวิทยุ หรือการใช้สัญญาณแสงอินฟาเรดโดยตรง ระบบจะชื่อว่า ระบบไร้สาย ระบบนี้เองที่กำลังเป็นที่นิยมใช้กันอยู่มากในปัจจุบัน

สัญญาณที่เข้ามายังเครื่องรับหรือภาครับ จะถูกตัวอินเตอร์เฟสทำหน้าที่แปลงสัญญาณให้อยู่ในรูปของสัญญาณไฟฟ้าที่เข้ากับระบบของตัวรับ ก่อนถูกถอดรหัสเพื่อทราบวัตถุประสงค์ของคำสั่งจากนั้นส่วนของวงจรอินเตอร์เฟสด้านเอาท์พุทจะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ที่ต้องการ ตามลักษณะคำสั่งที่ได้รับ

ระบบที่กล่าวมาถึงนั้น เป็นระบบรีโมตคอนโทรล หรือการควบคุมระยะไกลแบบทางเดียวที่มีการสั่งงานจากจุดหนึ่งแล้วเกิดการทำงานขึ้นอีกจุดหนึ่ง หากจุดที่ถูกสั่งให้ทำงาน มีความสามารถในการสั่งการกลับมายังจุดเริ่มต้นให้ทำงานได้ด้วยแล้ว แสดงว่าการทำงานของจุดหรือตำแหน่งทั้งสองมีความเสมอภาคกัน อย่างนี้ถือเป็นระบบควบคุมระยะไกลแบบสองทางซึ่งมักปรากฏให้เห็นอย่างชัดเจนในระบบการสื่อสารทั่วไป

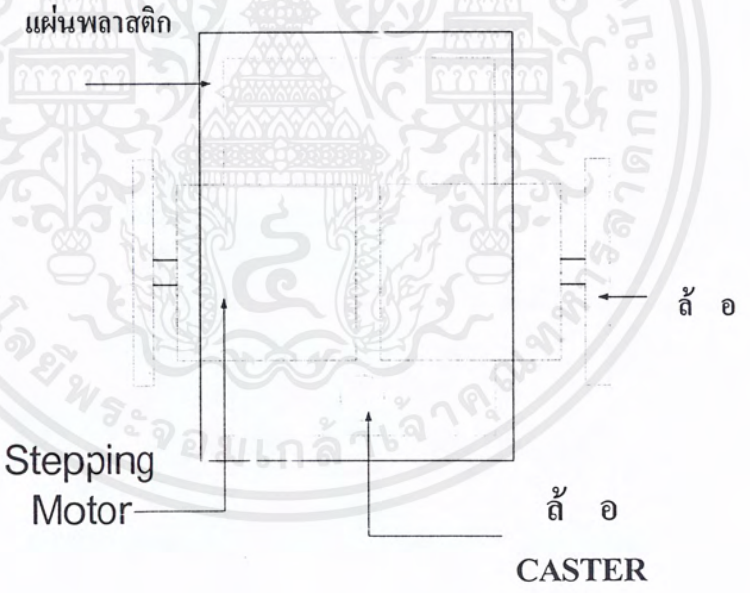
บทที่ 3
ออกแบบและส่วนประกอบของโครงงาน

3.1 รถ

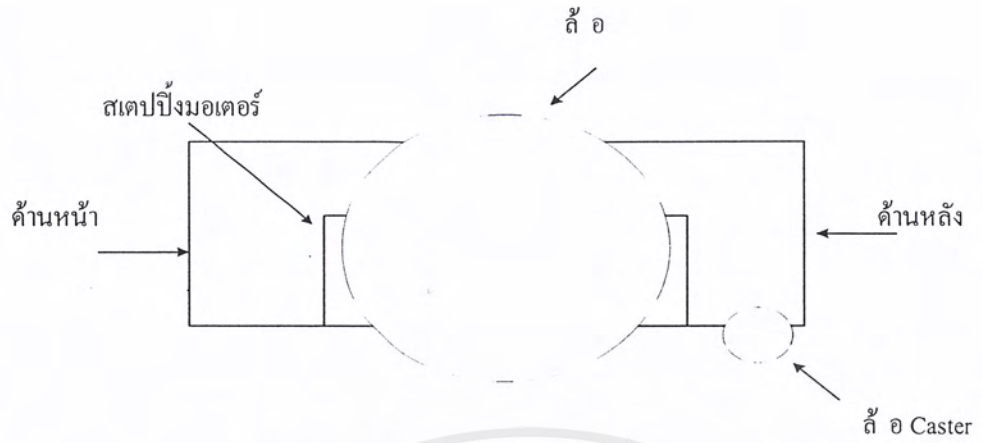
โครงรถทำจากกล่องพลาสติกใสขนาดเล็กซึ่งเจาะรูให้มอเตอร์ยื่นออกมาได้ รถจะประกอบด้วยล้อขับเคลื่อน 2 ล้อ และล้ออิสระอีก 2 ล้อ ล้อขับเคลื่อนจะวางตัวอยู่ตรงแนวกลางของรถเพื่อให้รถหมุนเป็นวงกลมได้ ส่วนล้ออิสระจะวางในแนวตั้งฉากกับล้อขับเคลื่อนเพื่อให้หุ่นยนต์สมดุล

ใช้ถ่านไฟฉาย Alkaline ขนาด AA 6 ก้อนเป็นแหล่งจ่ายไฟ จ่ายไฟรวมทั้งหมด 9 โวลท์ โดยเราจะติดลังถ่าน 2 อันไว้ที่ด้านหน้าและด้านหลังของหุ่นยนต์ ส่วนอีกอันติดที่ข้างใต้ เหตุผลที่เราติดลังถ่านไว้แบบนี้ เพื่อให้รถอยู่ในสภาพสมดุล ลดภาระของล้อไม่ให้ล้อใดล้อหนึ่งรับน้ำหนักมากเกินไป

รูปข้างล่างแสดงโครงสร้างของรถที่ออกแบบไว้ โดยจะแสดงเฉพาะมุมมองด้านบนและด้านข้างเท่านั้น

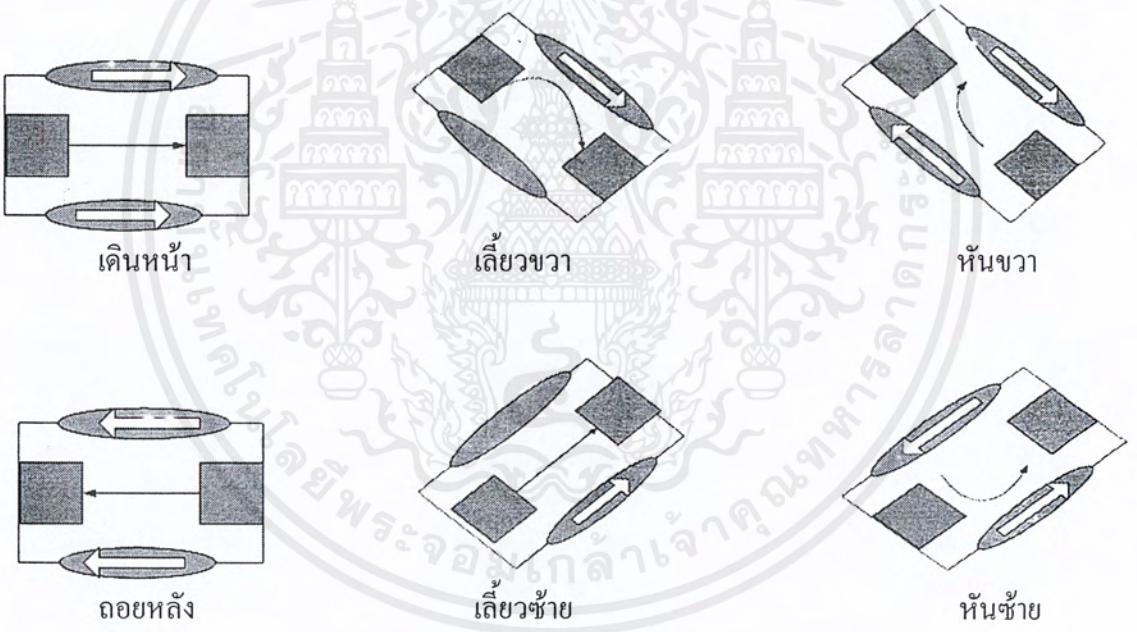


รูปที่ 3.1 แสดงรูปโครงสร้างของหุ่นยนต์เมื่อมองมาจากข้างบน



รูปที่ 3.2 แสดงรูปโครงสร้างของหุ่นยนต์เมื่อมองจากด้านข้าง

การออกแบบการวางล้อแบบนี้เพื่อให้ง่ายต่อการควบคุมทิศทางของรถ พิจารณาได้ จากรูปข้างล่าง



รูปที่ 3.3 แสดงการเคลื่อนที่ในรูปแบบต่างๆ

จากรูปที่ผ่านมาเราสามารถแบ่งการเคลื่อนที่ของหุ่นยนต์ได้เป็น 6 แบบ

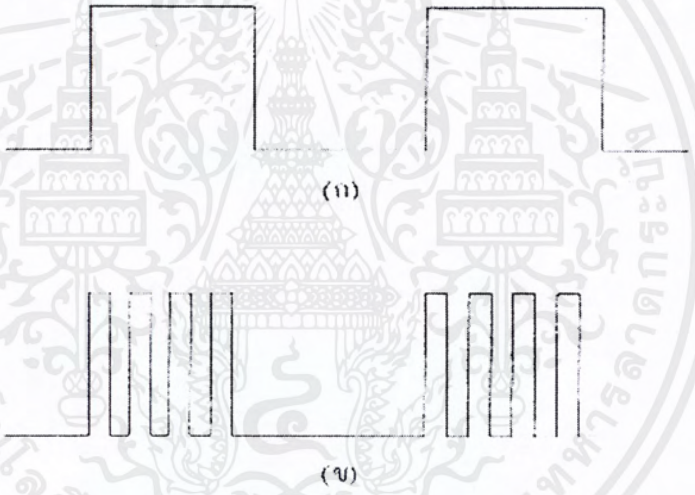
1. **เดินหน้า** สามารถทำได้โดยการสั่งให้มอเตอร์ทั้งคู่หมุนไปข้างหน้า
2. **ถอยหลัง** ทำได้โดยการสั่งให้มอเตอร์ทั้งคู่หมุนไปข้างหลัง

3. **เลี้ยวขวา** ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนไปข้างหน้า ส่วนมอเตอร์ทางขวาอยู่กับที่
4. **เลี้ยวซ้าย** ทำได้โดยการสั่งให้มอเตอร์ทางขวาหมุนไปข้างหน้า ส่วนมอเตอร์ทางซ้ายอยู่กับที่
5. **หันขวา** ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนไปข้างหน้า ส่วนมอเตอร์ทางขวาหมุนกับหลัง
6. **หันซ้าย** ทำได้โดยการสั่งให้มอเตอร์ทางขวาหมุนไปข้างหน้า ส่วนมอเตอร์ทางซ้ายหมุนกับหลัง

3.2 วงจรรับส่งอินฟราเรด

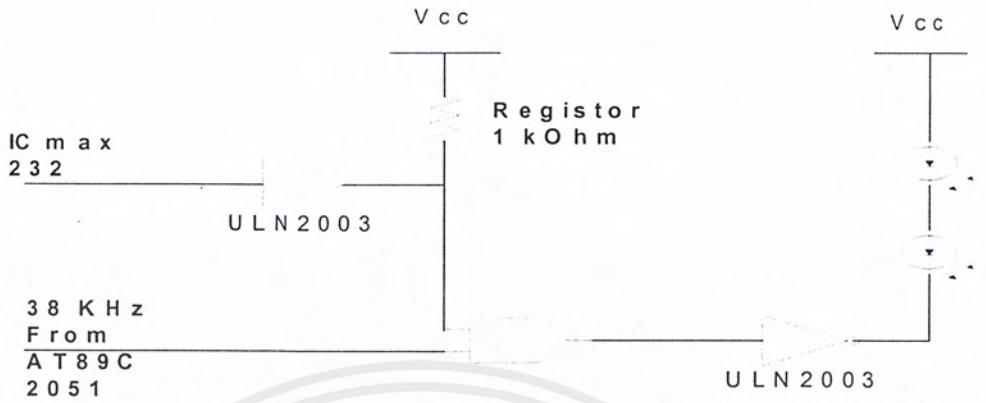
3.21 วงจรส่งอินฟราเรด

การออกแบบวงจรโดยใช้การส่งสัญญาณแบบโทนเบิรสต์ (tone burst) ซึ่งประกอบด้วยพัลส์ความถี่สูงแบบต่อเนื่องตลอดช่วงความกว้างของบิตข้อมูลที่เป็น "1" ในขณะที่บิตข้อมูลอยู่ในสภาวะต่ำ สัญญาณจะคงเดิมไม่มีการเปลี่ยนแปลงแต่อย่างใด



รูปที่ 3.4 ลักษณะสัญญาณโทนเบิรสต์
(ก) สัญญาณพัลส์ข้อมูล
(ข) สัญญาณโทนเบิรสต์

วงจรส่งตัวอินฟราเรดที่ใช้มีรูปดังนี้

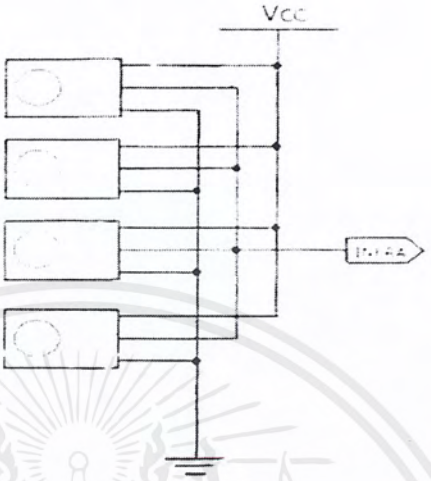


รูปที่ 3.5 รูปวงจรตัวส่งอินฟราเรด

รูปข้างบนจะเห็นได้ว่า สัญญาณที่ออกมาจาก IC MAX232 จะเป็นสัญญาณพัลส์ข้อมูล เราจะนำมารวมกับสัญญาณความถี่สูงที่เกิดจากไมโครคอนโทรลเลอร์ 89C51 กำเนิดพัลส์ความถี่ 38 kHz การรวมกันของ 2 สัญญาณนี้จะถูกรวมด้วย AND Gate จะได้สัญญาณโทนเบิรสต์ออกมา แล้วส่งต่อให้ภาคขับ LED อินฟราเรดต่อไป ตัว NOT Gate ตัวแรกมีไว้เพื่อกลับสถานะของสัญญาณที่ออกมาจาก MAX232 เพราะเนื่องจากว่าที่ภาครับอินฟราเรดมีการกลับสัญญาณเกิดขึ้น จึงต้องใช้ NOT Gate เพื่อให้ค่าที่ถูกต้อง ส่วนตัวต้านทาน 1k เป็นตัวพูลอัพ (pull – up) เพื่อให้สัญญาณมีการชดเชยกันอย่างเหมาะสม

3.22 คัวรับอินฟราเรด

วงจรรับอินฟราเรด



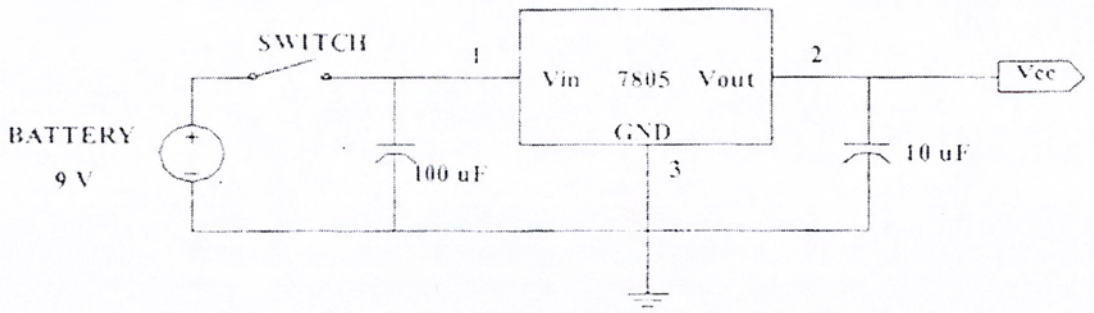
รูปที่ 3.6 รูปวงจรตัวรับอินฟราเรด

วงจรรับอินฟราเรดถูกออกแบบโดยใช้โมดูลอินฟราเรดเบอร์ TSOP1831 โมดูลตัวนี้จะตอบสนองเฉพาะสัญญาณความถี่ที่ 38 kHz เท่านั้น โดยจะทำการกรองความถี่พาหะ 38 kHz ออกแล้วเอาเฉพาะสัญญาณข้อมูลส่งให้ภาคขยายสัญญาณเพื่อส่งเป็นเอาท์พุทออกไป แต่เอาท์พุทที่ได้จะมีสถานะกลับกันกับสัญญาณข้อมูลจริง

เนื่องจากต้องการให้ตัวรับอินฟราเรดสามารถรับอินฟราเรดได้ทุกทิศทาง เราจึงออกแบบติดตั้งโมดูลอินฟราเรดให้หันหน้าออกจากกัน โมดูลอินฟราเรด 1 ตัวมีรัศมีการรับ 90 องศา ฉะนั้นเราจึงใช้โมดูลทั้งหมด 4 ตัว ในการรับข้อมูลจากทุกทิศทาง

3.3 วงจรควบคุมไมโครคอนโทรลเลอร์

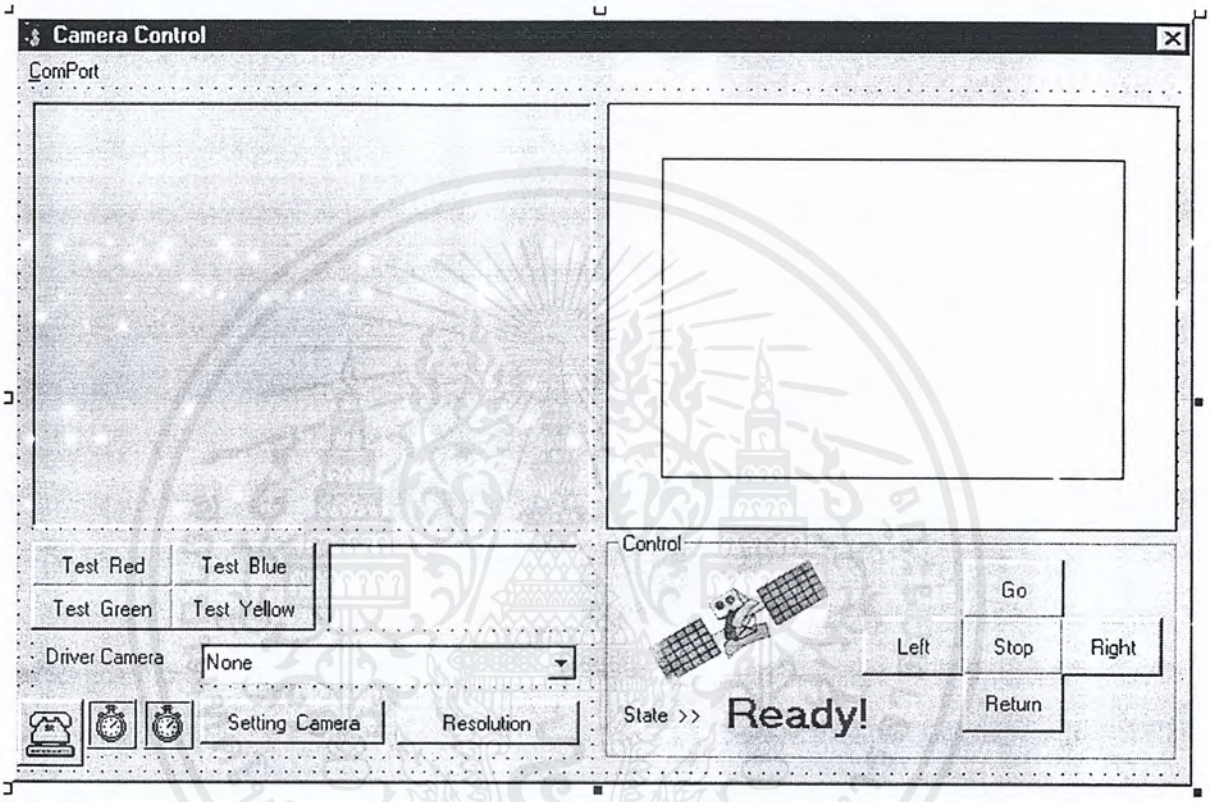
ไมโครคอนโทรลเลอร์ที่ใช้เป็นเบอร์ 89C51 ของบริษัท Atmel ซึ่งอยู่ในตระกูล MCS – 51 ไมโครคอนโทรลเลอร์ตัวนี้มี ROM ภายใน 4k ซึ่งมีมากพอในการโปรแกรมที่พอร์ต 1 ขา 0 เราจะต่อกับมอเตอร์ตัวที่ 1 ส่วน พอร์ต 1 ขา 1 เราจะต่อเข้ากับ มอเตอร์ตัวที่ 2 โดยสองขานี้จะเป็นขาส่งพัลส์ไปให้มอเตอร์ส่วนพอร์ต 0 จะต่อกับ DIP Switch เผื่อไว้สำหรับปรับโหมดการทำงานที่รองรในอนาคต พอร์ต 2 จะต่อกับวงจร LED มีไว้สำหรับตรวจสอบค่าที่เราส่งมาให้หุ่นผ่านพอร์ตอนุกรมรูปต่อไปนี้แสดงวงจรจ่ายไฟของหุ่นยนต์



รูปที่ 3.7 รูปวงจรแหล่งจ่ายไฟของหุ่นยนต์

3.5 ส่วนการติดต่อกับผู้ใช้

ใช้โปรแกรม Microsoft Visual Basic สร้างโปรแกรมทั้งในส่วนติดต่อกับผู้ใช้และในส่วนการประมวลผล รูปข้างล่างเป็นรูปแบบฟอร์มที่สร้างขึ้น



รูปที่3.8 แสดงส่วนประกอบของโปรแกรมที่เขียนด้วย Visual Basic 6

จากรูปฟอร์มข้าบน สามารถแบ่งส่วนประกอบออกได้ส่วนประกอบออกได้เป็นส่วนย่อยๆดังนี้

1. **MnuPort :** เป็นออบเจ็คชนิด Menu เป็นส่วนให้ผู้เลือกว่าจะส่งคำสั่งควบคุมรถโดยผ่านพอร์ตอนุกรมไหน ซึ่งจะมีให้เลือกใช้ 2 พอร์ตคือ พอร์ต Com1 และ พอร์ต Com2

2. **ezVidCap1 :** เป็นออบเจ็คชนิด ezVidCap สามารถดึงภาพจากกล้องมาได้ ตามปกติใน VB6 จะไม่มี ออบเจ็คนี้ต้อง Add Components เข้ามาเอง ออบเจ็คนี้สามารถ Download ได้มาจาก http://i.am/shrinkwrapvb

3. **Picture1 :** เป็นออบเจคชนิด Picture Box เป็นออบเจ็คที่สามารถแสดงภาพในรูปแบบขอบิทแมท (Bitmap) , ไอคอน หรือเมตาไฟล์ (Metafile เป็นไฟล์ที่เก็บคำอธิบายว่าจะสร้างภาพในลักษณะสร้างภาพในลักษณะใด ณ ตำแหน่งใด) ใช้สำหรับวาดรูปภาพจำลองเส้นทางเดินของรถ

4. **Text1:** เป็นออปเจ็คชนิด Text Box ใช้สำหรับแสดงรายละเอียดตำแหน่งของรถโดยจะแสดงค่าทิศทางของรถยนต์ ทิศทางของจุดรถจะต้องไป และโคออดิเนตที่รถอยู่

5. **ComRed, ComGreen,ComBule,ComYellow :**เป็นออบเจ็คชนิด CommandButton ใช้เป็นปุ่มในการสั่งทำงาน โดยถ้ากดปุ่ม ComRed จะเป็นการสำเนาภาพจาก ezVidCap มาลงมาใน Picture 1 โดยจะไม่นำภาพสิแดงมาด้วย ส่วน ComGreen , ComBlue ,ComYellow จะมีลักษณะการทำงานคล้ายกัน ต่างกันเพียงการทำสำเนาภาพจะไม่นำภาพสีเขียว สีน้ำเงิน สีเหลือง มาด้วย

6. **Label4 :** เป็นออบเจ็คชนิด Label 1 ใช้แสดงข้อความบ่งบอกสถานะการสั่งงานรถโดยถ้าข้อความแสดงว่า Ready ! จะแสดงว่าตอนนี้รถอยู่กับที่ พร้อมที่จะสั่งให้รถทำงาน ถ้าแสดงข้อความว่า Go Baby ! จะแสดงว่าตอนนี้รถกำลังเคลื่อนที่อยู่ แต่ผู้ใช้ก็สามารถสั่งใหม่ให้รถได้

7. **Command 2-Command 6 :**เป็นออบเจ็คชนิด CommandButtonใช้เป็นปุ่มคำสั่งบังคับรถโดยตรงโดยจะมีปุ่มเดินหน้า ถอยหลัง หันหน้า หันซ้าย หันขวา

8. **Commmand7 :** เป็นออบเจ็คชนิด CommandButton ใช้เป็นปุ่มคำสั่งบังคับรถหยุดเดิน

9. **Label 3 :**เป็นออบเจ็คชนิด Label ใช้แสดงข้อความบอกจุดที่รถต้องเดินผ่าน

10. **CbDriver :** เนื่องจากภายในคอมพิวเตอร์มี Driver สำหรับกล้องวิดีโออยู่หลายชนิด ดังนั้นจึงสร้าง cbDriver ซึ่งเป็นออปเจ็คชนิด ComBox ขึ้น เพื่อช่วยผู้ใช้โปรแกรมในการเลือกไดรเวอร์ที่เหมาะสำหรับการใช้งานนั้นๆ

11. **CmdSetting :** เป็นออบเจ็คชนิด CommmandButton สร้างขึ้นเพื่อช่วยในการปรับแสงสี รวมถึงความชัดเจนของภาพโดยเรียกจากไดเวอร์ Logitech USB Video Camera จากผู้ใช้งานสามารถปรับแต่งภาพได้ตาทต้องการ

12. **CmdSol :** เป็นออบเจ็คชนิด CommandButton สร้างขึ้นเพื่อปรับความละเอียดของภาพ (Resolution) โดยมีหน่วยเป็น Pixel ซึ่งแดงค่าความละเอียดของภาพตามที่เราต้องการ

3.6 ส่วนการประมวลผล

ในโปรแกรม VB6 สามารถแบ่งโปรแกรมออกเป็น 3 ส่วนคือ

1.ส่วนฟังก์ชั่น

2.ส่วนโปแกรมย่อย

3.ส่วนตอบสนองเหตุการณ์

1.ส่วนฟังก์ชั่น มีดังต่อไปนี้

1. Point 2 Point : เป็นฟังก์ชั่นที่ถูกสร้างขึ้นเพื่อตรวจสอบความเป็นไปได้ที่ตัวรถจะเดินทางจากจุดหนึ่งไปยังอีกจุดกนึ่งโดยมีกระบวนการดังนี้

รูปที่ 3.9 ประกอบการอธิบายการหาฟังก์ชั่น Point 2 Point

สมมุติว่าต้องการให้รถเดินทางจากจุด P1 ไปยัง P2 เราจะใช้ฟังก์ชั่นนี้ตรวจสอบดูว่ารถเดินทางไปได้หรือไม่ โดยจะแบ่งเส้นตรงระหว่างจุดสองจุดออกเป็นส่วนย่อยๆ แล้วพิจารณา ณ จุดที่ถูกแบ่ง (คือจุด 1,2,3,4,5,6) แล้วเรียกฟังก์ชั่น CarArea เพื่อตรวจดูว่าจุดที่ถูกแบ่งนั้นรถสมารถอยู่ได้หรือไม่ ถ้าทุกจุดรถสามารถอยู่ได้ ฟังก์ชั่น Point2Point นี้จะให้ค่า True หมายความว่ารถสามารถเดินไปได้ แต่ถ้าผลออกมาเป็น Falseหมายความว่ามีสิ่งขีดขวางอยู่อยู่ทำให้รถไม่สามารถเดินทางเป็นเส้นตรงตามทางที่ต้องการได้

1. Object : เป็นฟังก์ชั่นสำหรับตรวจสอบว่า ณ. จุดนั้นมีสิ่งกีดขวางหรือไม่ (ในที่นี้สมมุติให้จุดเหลืองเป็นสิ่งกีดขวาง) ถ้าจุดนั้นเป็นสิ่งขวางฟังก์ชั่นจะแสดงค่า Ture แต่ถ้าไม่ใช่ก็จะให้ค่า False ออกมา

1. CarArea : เป็นฟังก์ชั่นสำหรับตรวจสอบว่า ณ จุดที่พิจารณาหุ่นยนต์สามารถเคลื่อนที่ไปได้หรือไม่ สามารถอธิบายได้ดังรูปต่อไปนี้

รูปที่ 3.10 แสดงจุดที่ใช้ในการตรวจสอบของฟังก์ชั่น CarArea

ทำการสร้างจุด 16 จุด รอบจุดที่พิจารณาแล้วตรวจดูว่าทั้ง 16 จุดนี้โดนสิ่งกีดขวางหรือไม่ต้องเรียก Object ขึ้นมาตรวจสอบ ถ้าทั้ง 16 จุดนี้ไม่โดนสิ่งกีดขวางฟังก์ชั่นCarArea จะให่ค่า Ture ออกมา แต่ถ้ามีจุดใดจุดหนึ่งโดนสิ่งกีดขวางฟังก์ชั่นดังกล่าวจะให้ผลเป็น False

2. ColorRGB : เป็นฟังก์ชั่นให้ค่าแม่สีหลัก แดง เขียว น้ำเงิน โดยค่าที่ให้เป็นค่ารหัสสีของจุดพิจารณา

ส่วนโปรแกรมย่อย มีดังนี้

1. CarGo : จะทำการส่งคำสั่งให้รถเดินหน้า
2. CarReturn : จะทำการส่งคำสั่งให้รถหันซ้าย หันขวา
3. CarTurnLeft : จะทำการส่งคำสั่งให้รถเลี้ยวซ้าย
4. CarTurnRight : จะทำการส่งคำสั่งให้รถเลี้ยวขวา
5. CarStop : จะทำการส่งคำสั่งให้รถหยุด
6. PathA :เป็นกระบวนการหาเส้นทางที่สั้นที่สุดเพื่อให้รถไปถึงจุดหมายได้เร็วที่สุด

ส่วนตอบสนองเหตุการณ์ มีดังนี้

1. Picture1_MouseDown

เมื่อเราทำการคลิ๊กเมาส์บน Picture 1 โปรแกรมจะทำการตรวจดูก่อนว่า ณ. จุดที่ คลิ๊กนั้นรถสามารถอยู่ได้หรือไม่ ถ้าได้ก็เริ่มต้นคำนวณหาจุดที่ รถจะเดินผ่าน โดนคอมพิวเตอร์มีขั้นตอนการทำงานดังนี้

- สแกนภาพจากกล้องทีละจุดถ้าพบสีแดงจะทำการเก็บค่าตำแหน่งสีแดงไว้ ถ้าเจอสีเขียวจะเก็บค่าตำแหน่งสีเขียวไว้ ถ้าเจอสีเหลืองก็จะทำการตรวจสอบว่า ณ จุดนั้นเป็นจุดมุมของสิ่งกีดขวางหรือไม่โดยพิจารณาได้จากจุดรอบข้างจุดนั้นๆ ถ้าจุดรอบข้างทั้ง 8 จุด มีสีเหลืองน้อยกว่า 4 จุด ก็แสดงว่าจุดพิจารณานั้นเป็นจุดมุมของสิ่งกีดขวาง

2	1	8
3	X	7
4	5	6

ตารางที่ 3.1 แสดงการพิจารณาจุดรอบข้าง

ถ้าจุดพิจารณาเป็นจุดมุมของสิ่งกีดขวาง ก็จะทำการสร้างจุดสีแดงขึ้นรอบจุดนั้น 8 จุดและทำการพิจารณาทีละจุดนั้นๆ รถสามารถอยู่ได้หรือไม่ ถ้าอยู่ได้ก็คงจุดนั้นไว้ แต่ถ้าอยู่ไม่ได้ก็จะลบจุดนั้นทิ้ง สุดท้ายจะได้จุดสีแดงรอบสิ่งกีดขวางจุดสีแดงที่เราสร้างขึ้นนี้จะถูกเก็บค่าตำแหน่งไว้เพื่อไว้เพื่อใช้ในการคำนวณหาระยะที่สั้นที่สุดต่อไป

หาจุดศูนย์กลางของรถ โดยการหาจุดศูนย์กลางของสีแดงหาได้จาก

$$red\_center.X = (\sum Xred)/n$$

$$red\_center.Y = (\sum Yred)/n$$

โดยที่ n คือ จำนวนจุดสีแดงทั้งหมด

จุดศูนย์กลางของสีเขียวหาได้จาก

$$green\_center.X = (\sum Xgreen)/n$$

$$green\_center.Y = (\sum Ygreen)/n$$

หาจุดศูนย์กลางของรถได้จาก

$$X=(red\_center.X+ green\_center.X)/2$$

$$Y=(red\_center.Y+ green\_center.Y)/2$$

เมื่อหาตำแหน่งของรถได้ ก็ ทำการวาดวงกลมลงใน Piciture1เพื่อบ่งบอกตำแหน่งเริ่มต้นเดินของรถ แล้วทำการหาเส้นทางเดินที่สันที่สุดโดนเรียกใช้โปรแกรมย่อย PathA ในการคำนวณ ถ้าจุดเป้าหมายที่จะให้รถเดินไปเป็นจุดอับหรือโดนสิ่งกีดขวางบังทางที่สั้นที่สุดได้ก็ทำการวาดเส้นทางที่รถจะองเดินโดยจะวาดบน Picture1

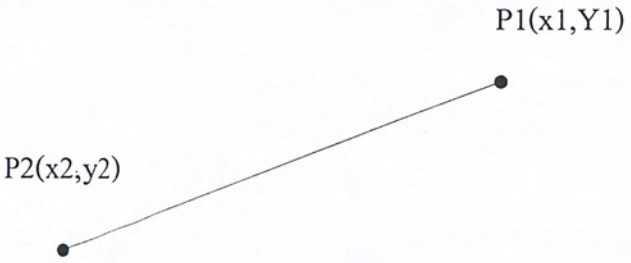
หลังจากนั้นก็จะเรียกใช้ Timer2 เพื่อให้เกิดการตอบสนองเหตุการณ์เป็นคาบเวลา โดนเราจะต้องตั้งเวลาให้เกิดเหตุการณ์ Timer2_Timer ทุกๆ 300 mS

2. Timer2_Timer

เมื่อเหตุการณ์นี้ จะเกิดขั้นตอนการทำงานดังนี้

- ทำการสแกนภาพขนาด 60x60 โดยมีจุดศูนย์กลางของภาพเป็นจุดเดียวกับจุดศูนย์กลางของรถครั้งล่าสุด สาเหตุที่สแกนภาพภาพขนาดนี้เพื่อให้โปรแกรมทำงานเร็วขึ้นการสแกนภาพนี้ถ้าเจอจุดสีแดงและสีเขียวก็จะเก็บค่าตำแหน่งไว้
- เมื่อสแกนจนครบแล้วก็จะทำการคำนวณหาจุดศูนย์กลางของสีแดงและสีเขียวแล้วคำนวณหาจุดศูนย์กลางของรถ

- ทำการหาทิศทางการหันของหุ่นยนต์โดยพิจารณาจากรูปต่อไปนี้



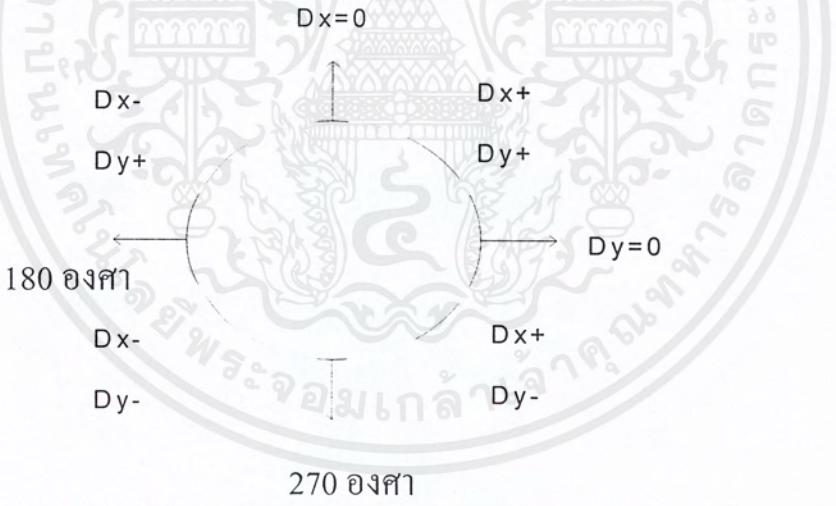
รูปที่ 3.11 แสดงการจำลองจุดที่ใช้แทนหัวและท้ายของรถ

สมมุติให้ P1 เป็นด้านหน้าของรถและกำหนดให้

$$Dx = X1 - X2$$

$$Dy = Y1 - Y2$$

เมื่อพิจารณาร่วมกับวงกลม 1 หน่วย จะสามารถแบ่งเงื่อนไขในการคำนวณได้เป็น



รูปที่ 3.12 แสดงเครื่องหมายของ Dx และ Dy ที่มุมต่างๆกัน

- ถ้า Dx มีเครื่องหมายเป็น – ทิศทางของรถสามารถได้จาก

$$DeCar = 180 + 180/\pi * \arctan(Dy/Dx) \text{ องศา}$$

- ถ้า Dx=0 แสดงว่ารถวางตัวอยู่ตามแนวแกน Y กรณีนี้ arctan(Dy/Dx) จะหาไม่ได้ จึงต้องใช้วิธี พิจารณาเครื่องหมาย Dy แทน ซึ่งจะได้ว่า

- ถ้า Dy มีเครื่องหมายเป็น + จะได้ทิศทางของรถเท่ากับ 90 องศา

- ถ้า Dy มีเครื่องหมายเป็น - จะได้ทิศทางของรถเท่ากับ 270 องศา

- ถ้า Dx มีเครื่องหมาย + การหาทิศทางของรถแบ่งเป็น 2 กรณี

- ถ้า Dy มีเครื่องหมายเป็น – จะสามารถคำนวณหาทิศทางได้ คือ

$$DeCar = 360+180/\pi * \arctan(Dy/Dx) \quad \text{องศา}$$

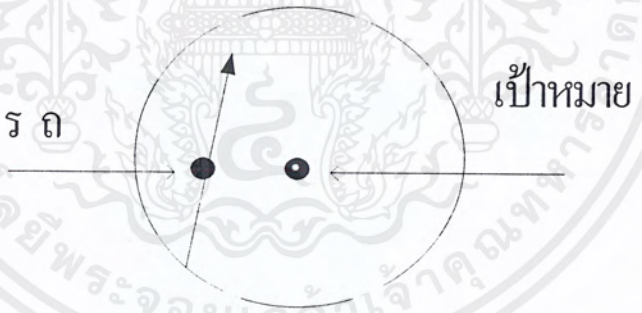
- ถ้า Dy มีเครื่องหมายเป็น + หรือมีค่าเป็น 0 จะสามารถคำนวณหาทิศทางได้ คือ

$$DeCar = 180/\pi * \arctan(Dy/Dx) \quad \text{องศา}$$

- ทำการทิศทางของจุดที่เราต้องไปโดยใช้จุดศูนย์กลางของหุ่นยนต์กับจุดเป้าหมายนำคำนวณเหมือนกับการหาทิศทางการหันของรถ

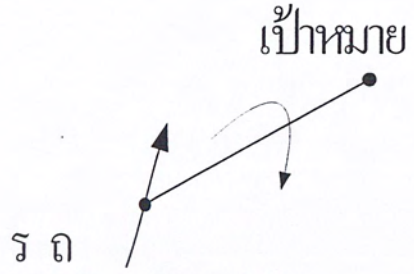
- เลือกคำสั่งที่จะสั่งให้กับรถโดยพิจารณาจาก

1.ถ้าจุดศูนย์กลางของรถอยู่ใกล้จุดเป้าหมายรัศมีที่กำหนด ก็จะสั่งให้รถหยุดเดินแล้วพร้อมที่จะเดินไปยังจุดเป้าหมายอันต่อไป



รูปที่ 3.13 แสดงตำแหน่งเมื่อรถหยุดเดิน

2 .ถ้ามุมของรถมากกว่ามุมของจุดเป้าหมายก็จะสั่งให้รถหันขวา

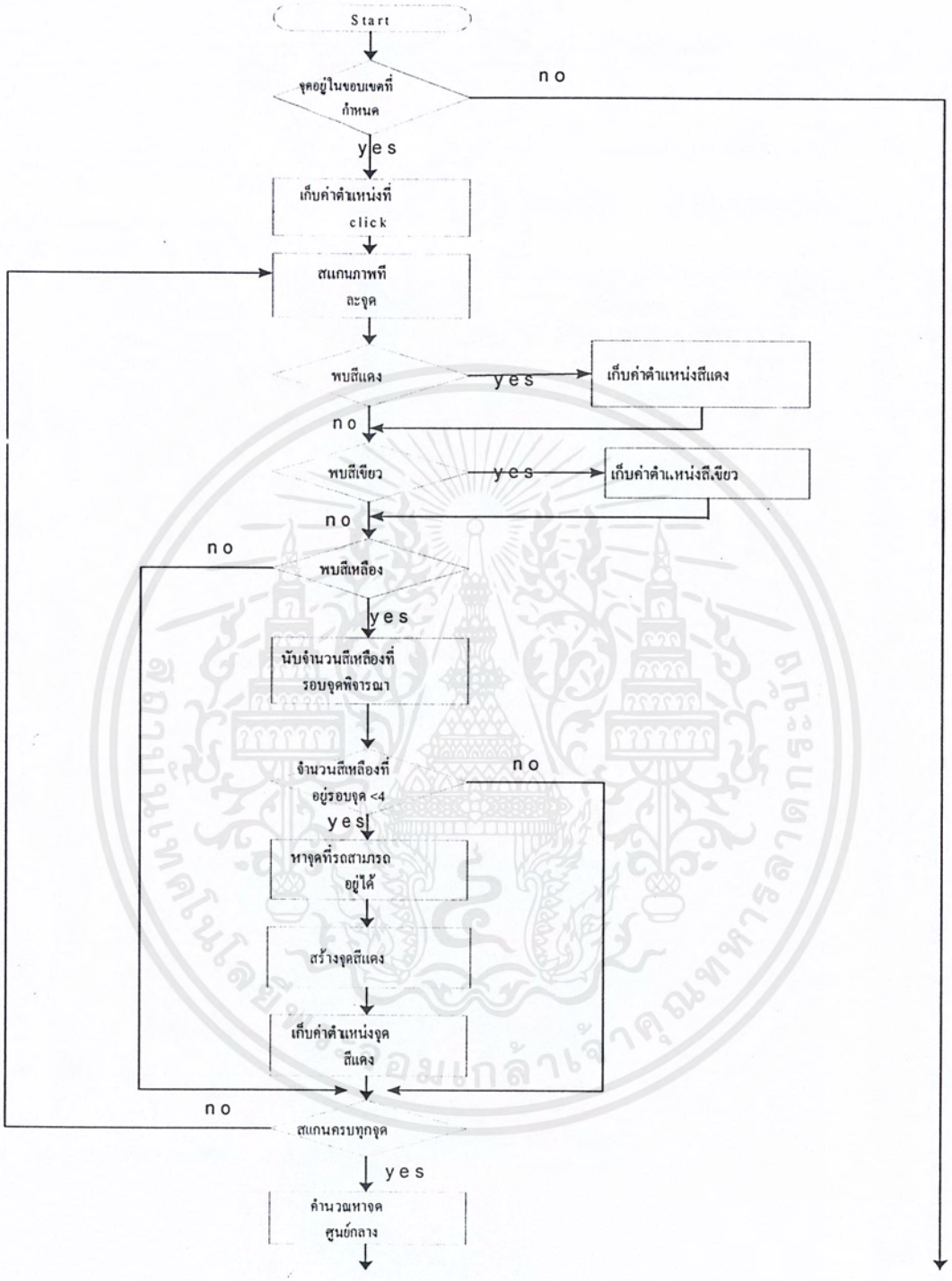


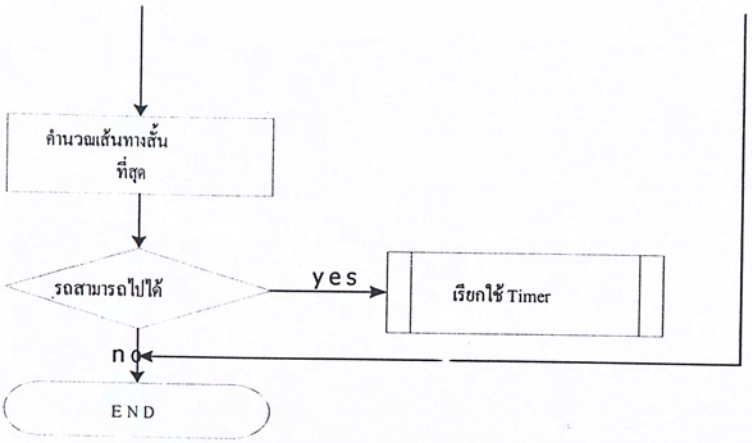
รูปที่ 3.14 แสดงการหันขวาของรถ

3.ถ้ามุมของรถน้อยกว่ามุมของจุดเป้าหมายก็จะสั่งให้รถหันซ้าย

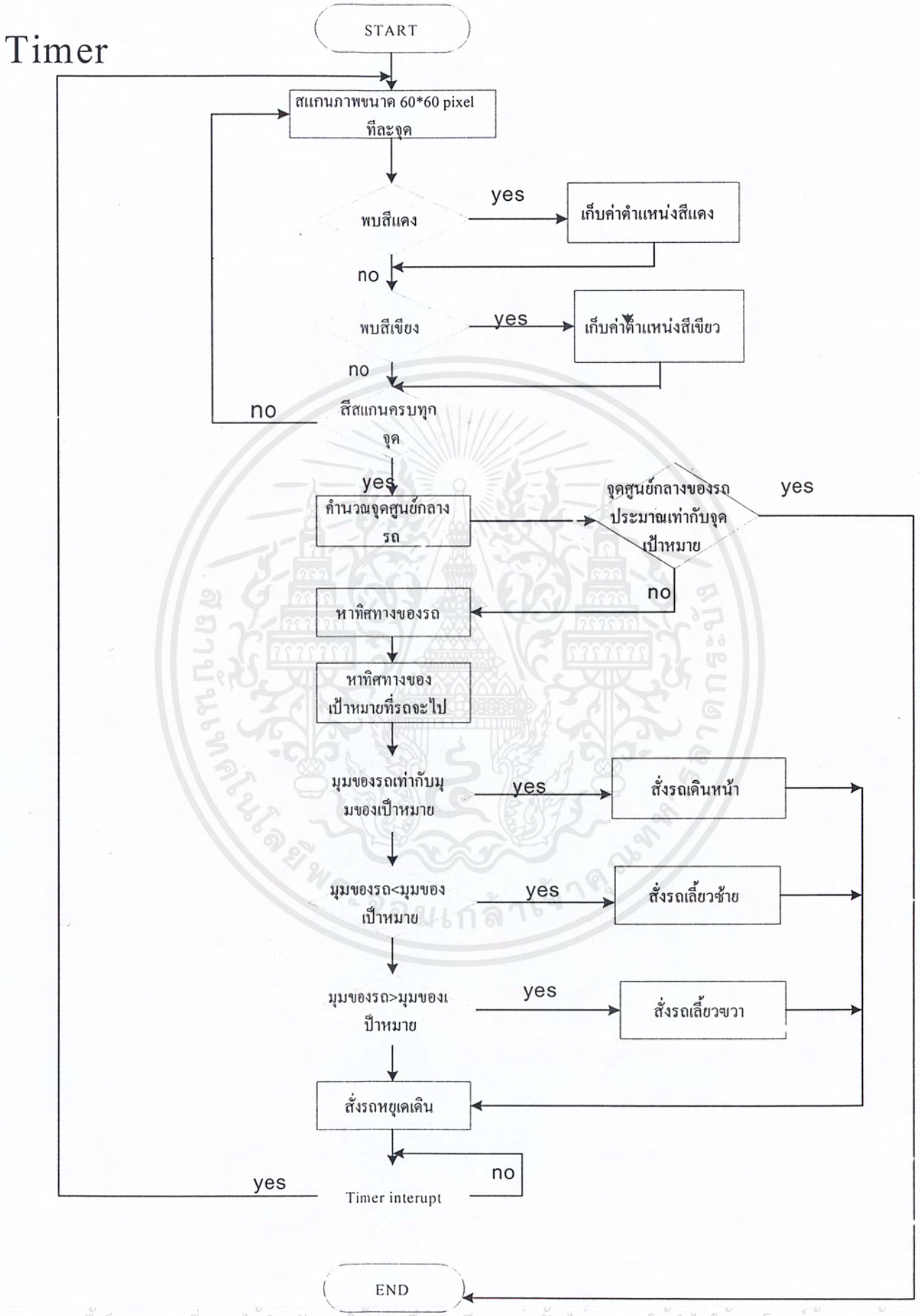
รูปที่ 3.15 แสดงการหันซ้ายของรถ

4. ถ้ารถอยู่ที่จุดเป้าหมายอันสุดท้ายแล้ว (จุดคลิ๊กเมาส์) ก็สั่งให้หุ่นยนต์หยุดเดิน (หยุดการตอบสนองต่อเหตุการณ์ Timer2_Timer) แล้วรอคำสั่งใหม่อีกที





รูปโฟลว์ชาร์ตต่อไปแสดงของเหตุการณ์ ของTimer2_Timer

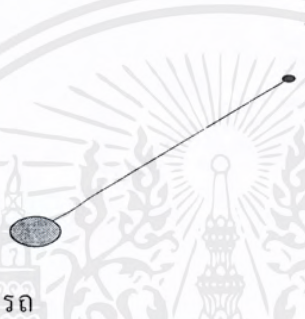


บทที่ 4

ผลการทดลอง

4.1 การทดลอง

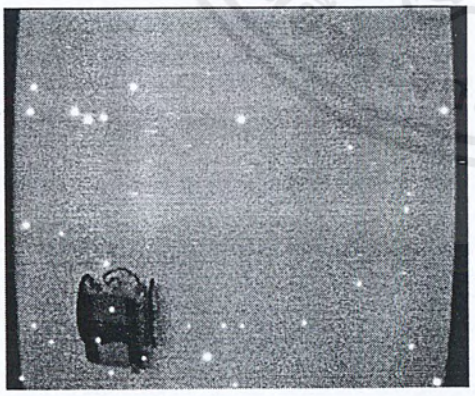
คลิ๊กเมาส์สั่งให้รถไปยังเป้าหมาย โดยโปรแกรมจะลากเส้นจากรถไปยังเป้าหมาย ดังรูปข้างล่างเ



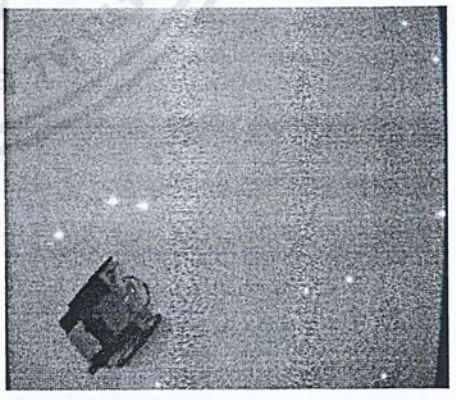
รูปที่ 4.1 แสดงการทดลองคลิ๊กเมาส์

4.2 การทดลองการทำงานของรถ

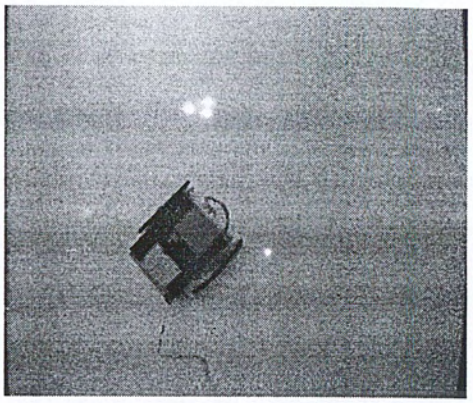
การทำงานการเดินทางของรถใช้วิธีการดูเส้นทางของรถว่าสามารถเคลื่อนที่ได้หรือไม่



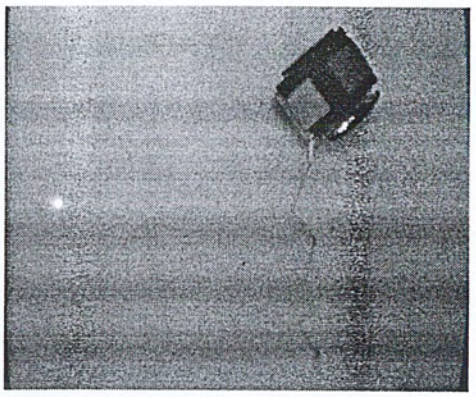
A 1



A 2



A 3



A 4

รูปที่ 4.2 (A1,A2,A3,A4) แสดงผลการทดลองเคลื่อนที่ของรถ

บทที่ 5
บทสรุปและวิจารณ์

5.1 สรุปผลการดำเนินงาน

โครงการนี้เกี่ยวกับการสร้างรถที่ควบคุมด้วยคอมพิวเตอร์โดนใช้รีโมตคอนโทรล แบบอินฟราเรด เป็นตัวส่งข้อมูลระหว่างรถกับคอมพิวเตอร์ การควบคุมรถควบคุมผ่านทางคอมพิวเตอร์โดยใช้กล้องวิดีโอจับภาพการเคลื่อนไหว ตำแหน่งของรถเพื่อนำไปใช้งานในการออกคำสั่งกับรถ

จากการดำเนินงาน รถสามารถเดินทางตามที่กำหนดโดยมีความสามารถ เลี้ยวซ้าย ขวา หันหน้า หันหลัง

5.2 ปัญหาที่เกิดและแนวทางแก้ไข

ปัญหา ใช้เวลานานในการออกคำสั่งให้เดินรถ

แนวทางแก้ไข : เขียนโปรแกรมให้มีการทำงานหรือเปลี่ยนภาษาโปรแกรมที่ใช้เขียน เช่น อาจใช้ Visual C++ ซึ่งทำงานได้เร็วขึ้น

ปัญหา : ภาพที่ได้จากกล้องวิดีโอไม่คมชัดเท่าที่ควร ทำให้ดำเนินหาเส้นทางมีการผิดพลาดขึ้น อีกทั้งภาพที่ส่งมามีการกระตุก

แนวทางแก้ไข : ใช้วิดีโอที่ให้ภาพคมชัดกว่ากล้อง Quick Cam ที่ใช้ในการทดลอง

ปัญหา : ภาพที่ได้จากกล้องวิดีโอมีการผิดเพี้ยนของสีที่ Capture

แนวทางแก้ไข : ควรใช้กล้องอินฟราเรดไม่คำนึงถึงผลของแสงหรือความละเอียดของสีสูงกว่า 24 bitขึ้นไป

5.3 แนวทางพัฒนาในอนาคต

1.นำระบบการตรวจจับต่างๆติดตั้งบนรถ เช่น ตัวตรวจจับกลิ่น ตัวจับความร้อน ตัวจับคลื่นความถี่

2.ใช้หน่วยประมวลผล CPU ที่มีความสามารถสูงติดตั้งบนรถ เพื่อที่รถจะสามารถเคลื่อนที่ได้เองโดยไม่ต้องมีคนควบคุม คอยควบคุมบนจอคอมพิวเตอร์

3.ทำระบบการสื่อสารด้วยคลื่นวิทยุ เพื่อลดข้อจำกัดการส่งข้อมูลเป็นเส้นตรง

บรรณานุกรม

[1] ธาริน สิทธิธรรมชารี , " คู่มือการเขียนโปแกรม Visaul Basic 6 " , บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน),1989

[2] กฤษดา ใจเย็น, "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม " , บริษัท อินโนเวตีฟเอ็กเพอริเมนต์ จำกัด

[3] รองศาสตราจารย์สมยศ จุณณะปิยะ , " การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS 51 " , สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2543

บรรณานุกรม

[1] ธาริน สิทธิธรรมชารี,“ คู่มือการเขียนโปแกรม Visaul Basic 6 ” , บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน),1989

[2] กฤษดา ใจเย็น, “ เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม ” ,บริษัท อินโนเวตีฟเอ็กเพอริเมนต์ จำกัด

[3] รองศาสตราจารย์สมยศ จุณณะปิยะ , “ การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS 51 ” , สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2543

ภาคผนวก

Code ของโปรแกรม Assembly

```
            ORG         0000H
AJMP        MAIN
CR          EQU         13
LF          EQU         10
NL          EQU         12
ADC_W       EQU         090H
ADC_R       EQU         091H
EE_W        EQU         0A0H
EE_R        EQU         0A1H
RTC_R       EQU         0D1H
RTC_W       EQU         0D0H
I2C_ADDR    EQU         30H
I2C_DATA    EQU         31H
SCL         EQU         P1.0
SDA         EQU         P1.1
CS_LCD      EQU     P1.3        ; E  LCD (PIN INT1)
RS_LCD      EQU     P1.2        ; RS LCD (Pin T0)

    ORG         0020H
FLAG:       DS          1
N_ACK       EQU         FLAG.0
RST         EQU         P1.0
CLK         EQU         P1.2
IO          EQU         P1.1
;           CAR   ROBOT
INPUT       EQU         P0
LED         EQU         P2.0
OP          EQU         P2.1
;
BUFFER      EQU         30H
```

```
;********************* MAIN ************************ *************

MAIN:  MOV          p2,#0
       MOV          p1,#0
       MOV          DPTR,#07FFFH
       LCALL        POR
       MOV          p2,#0ffh
       MOV          INPUT,#0FFH
       MOV          TMOD,#022H
       MOV          SCON,#052H
       MOV          TH0,#-15
       MOV          TL0,#-15
       MOV          TH1,#-96
       MOV          TL1,#-96
       SETB         TR1
;      input
;      128''
       MOV          BUFFER,#20H
       MOV          P1,#0     ;INITIAL STEP

ML:        JNB          RI,$
           CLR          RI
           MOV          A,SBUF
           CPL          A
           MOV          P2,A
           MOV          A,SBUF
           CJNE         A,#'2',CP0
           MOV          DPTR,#T_L
           AJMP         DRIV
CP0:       CJNE         A,#'0',CP1
           MOV          DPTR,#T_S
```

```
                AJMP        DRIV
CP1:            CJNE        A,#'3',CP2
                MOV         DPTR,#T_R
                AJMP        DRIV
CP2:            CJNE        A,#'U',CP3
                DEC         BUFFER
                AJMP        DRIV
CP3:            CJNE        A,#'D',CP4
                INC         BUFFER
                AJMP        DRIV
CP4:            CJNE        A,#'1',CP5
                MOV         DPTR,#T_F
                AJMP        DRIV
CP5:            CJNE        A,#'4',ML
                MOV         DPTR,#T_G
                AJMP        DRIV
;*****************************************
DELL:           MOV         R7,BUFFER
D1:             MOV         R6,#05H
D2:             MOV         R5,#050H
                DJNZ        R5,$
                DJNZ        R6,D2
                DJNZ        R7,D1
                RET
;*****************************************
DRIV:           CLR         A
                MOV         R0,#4
DRV1:           PUSH        ACC
                MOVC        A,@A+DPTR
                MOV         P1,A
                ACALL       DELL
                POP         ACC
```

```
                INC             A
                DJNZ            R0,DRV1
                1 JNB           RI,DRIV
                AJMP            ML
T_L:            DB                  88H,41H,22H,14H
T_R:            DB                  88H,14H,22H,41H
T_F:            DB                  88H,44H,22H,11H
T_G:            DB                  88H,11H,22H,44H
T_S:            DB                  00H,00H,00H,00H
;
;*************************************************
;*      SUB FUNC TION                  *
;*                                     *
;*          -POR        -CK_ENT    *
;*          -WR_EEP     -GET       *
;*          -RD_EEP     -GETCH     *
;*          -WR_RTC     -GET_2D    *
;*          -RD_RTC     -CLRSCR    *
;*          -INIT_I2C   -ENTER     *
;*          -START_I2C   SPACE     *
;*          -STOP_I2C   -WR_ASC    *
;*          -WR_I2C     -WR_CHAR   *
;*          -RD_I2C     -WR_LINE   *
;*          -I2C_DEL               *
;*          -INIT_LCD              *
;*          -EN_LCD                *
;*          -BUSY                  *
;*          -GOTO_LCD              *
;*          -WR_LCD                *
;*          -WR_INS                *
;*************************************************
```

```
POR:            INC       DPTR
POR1:           MOV       A,DPH
                CJNE      A,#0,POR
                RET
;*****************************************
;*        I/P       I2C_ADDA *
;*        I2C   _DATA        *
;*        REG:       A      *
;*****************************************

WR_EEP:         PUSH      ACC
WR_EEP1:        ACALL     START_I2C
                MOV       A,#EE_W
                ACALL     WR_I2C
                JB        N_ACK,WR_EEP1
                MOV       A,I2C_ADDR
                ACALL     WR_I2C
                JB        N_ACK,WR_EEP1
                MOV       A,I2C_DATA
                ACALL     WR_I2C
                ACALL     STOP_I2C
                POP       ACC
                RET
;*********************************************
;*        I/P       I2C_ADDA   *
;*                  I2C_DATA   *
;*        REG:       A(DATA)   *
;*********************************************
RD_EEP:         ;PUSH     ACC
RD_EEP1:        ACALL     START_I2C
                MOV       A,#EE_W
                ACALL     WR_I2C
```

```
                JB      N_ACK,RD_EEP1
                MOV     A,I2C_ADDR
                ACALL   WR_I2C
                JB      N_ACK,RD_EEP1
                ACALL   START_I2C
                MOV     A,#EE_R
                ACALL   WR_I2C
                JB      N_ACK,RD_EEP1
                ACALL   RD_I2C
                ACALL   STOP_I2C
                RET
;***********************************************
;           A   =   DATA
;       I2C_ADDR    =   ADDRESS TO SL
;***********************************************
            PUSH    ACC
WR_RTC1:    ACALL   START_I2C
            MOV     A,#RTC_W
            ACALL   WR_I2C
            JB      N_ACK,WR_RTC1
            MOV     A,I2C_ADDR
            ACALL   WR_I2C
            JB      N_ACK,WR_RTC1
            POP     ACC
            ACALL   WR_I2C
            ACALL   STOP_I2C
            RET
;***********************************************
;       I2C_ADDR    =   ADDRESS IN SLAVE
;       A       =   DATA FORM   SLAVE
;***********************************************
```

```
RD_RTC:     ;PUSH     ACC
RD_RTC1:    ACALL     START_I2C
            MOV       A,#RTC_W
            ACALL     WR_I2C
            JB        N_ACK,RD_RTC1
            MOV       A,I2C_ADDR
            ACALL     WR_I2C
            JB        N_ACK,RD_RTC1
            ACALL     START_I2C
            MOV       A,#RTC_R
            ACALL     WR_I2C
            JB        N_ACK,RD_RTC1
            ACALL     RD_I2C
            ACALL     STOP_I2C
            RET
;*********************************************************
;*      I2C SUB FUNCTION                 *
;*********************************************************
INIT_I2C:   SETB      SDA
            SETB      SCL
            RET
;*********************************************************
START_I2C:  JNB       SCL,START_I2C1
            CLR       SCL
START_I2C1: SETB      SDA
            SETB      SCL
            ACALL     I2C_DEL
            CLR       SDA
            ACALL     I2C_DEL
            CLR       SCL
            RET
;*********************************************************
```

```
STOP_I2C:       JNB       SCL,STOP_I2C1
                CLR       SCL
STOP_I2C1:      CLR       SDA
                ACALL      I2C_DEL
                SETB      SCL
                ACALL      I2C_DEL
                SETB      SDA
                 RET
;*******************************************************
WR_I2C:         CLR       N_ACK       ;ERROR FLAG 1=ERROR (RETURN)
                PUSH      0
                MOV       R0,#8
WR_I2C1:        RLC       A

                 MOV       SDA,C
                 ACALL      I2C_DEL
                 SETB      SCL
                 ACALL      I2C_DEL
                 CLR       SCL
                 DJNZ      R0,WR_I2C1
                 SETB      SDA
                 ACALL      I2C_DEL
                 SETB      SCL
                 ACALL      I2C_DEL
                 JNB       SDA,WR_I2C2
                 SETB       N_ACK
                WR_I2C2:     CLR        SCL
                 POP       0
                 RET
;*******************************************************
        RD_I2C:     CLR       N_ACK       ;CLR ERROR 1=ERROR (RETURN)
                    PUSH      0
```

```
                MOV         R0,#8
RD_I2C1:        SETB        SCL
                ACALL       I2C_DEL
                MOV         C,SDA
                ACALL       I2C_DEL
                CLR         SCL
                RLC         A
                DJNZ        R0,RD_I2C1
                SETB        SDA
                ACALL       I2C_DEL
                SETB        SCL
                ACALL       I2C_DEL
                JNB         SDA,RD_I2C2
                SETB        N_ACK
RD_I2C2:        CLR         SCL
                ACALL       I2C_DEL
                CLR         SDA
                POP         0
                RET
;*****************************************************
I2C_DEL:        PUSH        0
                MOV         0,#10H
                DJNZ        R0,$
                POP             0
                RET
;*****************************************************
;           8051 IO
;*****************************************************
CK_ENT:         ACALL       GET
                MOV             A,SBUF
                CJNE        A,#0DH,CK_ENT
                RET
```

```
;*********************************************************
GET:        JNB     RI,$
            CLR     RI
            RET
;*********************************************************
GETCH:      JNB     RI,$
            CLR     RI
            MOV     A,SBUF
            PUSH    ACC
            ACALL   WR_CHAR
            CJNE    A,#08H,GETCH1
            ACALL   SPACE
            POP     ACC

            ACALL   WR_CHAR
            RET
GETCH1:     CJNE    A,#0DH,GETCH2
            ACALL   ENTER
            RET
GETCH2:     POP     ACC
            RET
;*********************************************
GET_2D:     ACALL   GETCH
            CLR     C
            SUBB    A,#'0'
            SWAP    A
            MOV     R0,A
            ACALL   GETCH
            SUBB    A,#'0'
            ORL     A,R0
            RET
;*********************************************
```

```
CLRSCR:     MOV       DPTR,#CLR_TAB
            ACALL      WR_LINE
            RET
CLR_TAB:    DB        CR,LF,NL,0
;*****************************************************
ENTER:      MOV       A,#CR
            ACALL      WR_CHAR
            MOV       A,#LF
            ACALL      WR_CHAR
            RET

;*****************************************************
SPACE:      MOV       A,#' '
            ACALL      WR_CHAR
            RET
;*****************************************************
WR_CHAR:    MOV       SBUF,A
            JNB       TI,$
            CLR       TI
            RET
;*****************************************************
WR_ASC:     PUSH      ACC
            PUSH      ACC
            SWAP       A
            ANL       A,#0FH
            MOV       DPTR,#ASC_TAB
            MOVC       A,@A+DPTR
            ACALL      WR_CHAR
            POP       ACC
```

```
                ANL         A,#0FH
                MOVC        A,@A+DPTR
                ACALL       WR_CHAR
                POP         ACC
                RET
ASC_TAB:        DB          '0123456789ABCDEF'
;************************************************
WR_LINE:        CLR         A
WR_L1:          PUSH        ACC
                MOVC        A,@A+DPTR
                CJNE        A,#0,WR_LINE_1
                POP         ACC
                RET

WR_LINE_1:      ACALL       WR_CHAR
                POP         ACC
                INC         A
                SJMP        WR_L1
;************************************************
;*              Initial LCD         *
;*              4-Bit Interface     *
;************************************************
INIT_LCD:       CLR   RS_LCD
                MOV   A,#33H        ; Set DL = 1 3-time
                ACALL         BUSY
                LCALL  WR_INS
                MOV   A,#32H        ; Clear DL = 0 1-time
                LCALL  WR_INS
                MOV   A,#00100011B        ; Function set
                LCALL  WR_INS       ; DL=0 4Bit,N=1 2Line,F=0 5X7
                MOV   A,#0CH
                LCALL  WR_INS       ; Entry display,cursor off,cursor not blink
```

```
                MOV     A,#06H          ; Entry mode set
                LCALL   WR_INS          ; I/D=1 Increment,S=0 cursor shift
                MOV     A,#01H          ; Clear display
                LCALL   WR_INS          ; Clear display,set DD RAM address=0
                RET
;*****************************************************
WR_LCD:         MOV     B,A
                ANL     A,#0F0H
                ORL     A,#00001100B    ;RS/CS HI
                MOV     R2,A
                MOV     A,P1
                ANL     A,#00000011B
                ORL     A,R2
                MOV     P1,A
                LCALL   EN_LCD
                MOV     A,B             ;Low byte
                SWAP    A
                ANL     A,#0F0H
                ORL     A,#00001100B
                MOV     R2,A
                MOV     A,P1
                ANL     A,#00000011B
                ORL     A,R2
                MOV     P1,A
                LCALL   EN_LCD
                 RET
;*****************************************************
WR_LINE1_LCD:   CLR             A
                ACALL           GOTO_LCD
                SJMP            WR_LINE_LCD
;*****************************************************
WR_LINE2_LCD:   MOV             A,#40H
```

```
                    ACALL       GOTO_LCD
;*****************************************************
WR_LINE_LCD:        PUSH          ACC
                    CLR         A
WR_LINE_LCD1:       PUSH        ACC
                    MOVC        A,@A+DPTR
                    JZ          WR_LINE_E
                    ACALL       WR_LCD
                    POP         ACC
                    INC         A
                    SJMP        WR_LINE_LCD1
WR_LINE_E:          POP         ACC
                    POP         ACC

                    RET
;*****************************************************
WR_INS:     MOV   B,A
            ANL   A,#0F0H
            SETB  ACC.3       ;CS HI
            MOV   R2,A
            MOV   A,P1
            ANL   A,#00000011B  ;else bit
            ORL   A,R2
            MOV   P1,A        ; High byte
            LCALL EN_LCD
            MOV   A,B         ; Low byte
            SWAP  A
            ANL   A,#0F0H
            SETB  ACC.3
            MOV   R2,A
            MOV   A,P1
            ANL   A,#00000011B
```

```
                ORL    A,R2
                MOV    P1,A
                LCALL  EN_LCD
                RET
;*************************************************
GOTO_LCD:       SETB   ACC.7
                LCALL  WR_INS
                RET
;*************************************************
EN_LCD:         CLR    CS_LCD       ; Enable LCD
                LCALL  BUSY         ; Busy delay time
                SETB   CS_LCD       ; Disable LCD
                RET
;*************************************************
BUSY:           PUSH   07H
                PUSH   06H
                MOV    R6,#0AH
BUSY1:          MOV    R7,#0FFH
                DJNZ   R7,$
                DJNZ   R6,BUSY1
                POP    06H
                POP    07H
                RET
;*************************************************
                END
```

Code ของโปรแกรม Visual Basic 6

```
Option Explicit
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal Hdc As Long)
As Long
Dim CameraH As Long
Dim PicX%, PicY%, i%, AA As Byte, target As Byte
Dim PicM As Boolean
Dim P() As Integer
Dim XX() As Integer
Private Type Point
    X As Integer
    Y As Integer
End Type
Dim Car As Point
Dim clientP As POINTAPI
Dim AAA(0 To 200) As Point
Dim lpPoint As POINTAPI
Dim A As Boolean
Dim DAT As Byte
Private Const Rr = 40, Rrr = 42, pi = 22 / 7

Private Sub CmdSetting_Click()
  ezVidCap1.ShowDlgVideoSource
End Sub

Private Sub CmdSol_Click()
  ezVidCap1.ShowDlgVideoFormat
End Sub

Private Sub Command1_Click()
End Sub
```

```
Private Sub Command2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
  DAT = 1
End Sub

Private Sub Command2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  DAT = 9
End Sub

Private Sub Command3_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
  DAT = 4
End Sub

Private Sub Command3_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  DAT = 9
End Sub

Private Sub Command4_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
  DAT = 2
End Sub

Private Sub Command4_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  DAT = 9
End Sub

Private Sub Command6_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
  DAT = 3
End Sub
```

```
Private Sub Command6_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   DAT = 9
End Sub

Private Sub Command7_Click()
   Timer2.Interval = 0
   Label4.Caption = "Ready!"
   DAT = 9
End Sub

Private Sub ComRed_Click()
    Dim Gx, Gy, Drg%, Drb%, Dgb%, R%, G%, B As Integer
   Dim Gcolor As Long
        For Gx = 0 To 320
         For Gy = 0 To 240
            Gcolor = GetPixel(CameraH, Gx, Gy)
            Call ColorRGB(Gcolor, R, G, B)
            Drg = R - G
            Drb = R - B
            Dgb = G - B
            If Drg > 40 And Drb > 30 Then Gcolor = 16777215
            SetPixel Picture1.Hdc, Gx, Gy, Gcolor
         Next Gy
      Next Gx
End Sub

Private Sub ComGreen_Click()
    Dim Gx, Gy, Dgr%, Drb%, Dgb%, R%, G%, B As Integer
   Dim Gcolor As Long

      For Gx = 0 To 320
         For Gy = 0 To 240
```

```
            Gcolor = GetPixel(CameraH, Gx, Gy)
            Call ColorRGB(Gcolor, R, G, B)
            Dgr = G - R
            Dgb = G - B
            Drb = R - B
            If Dgr > -20 And Dgb > 55 Then Gcolor = 16777215
            SetPixel Picture1.Hdc, Gx, Gy, Gcolor
        Next Gy
    Next Gx
End Sub

Private Sub ComBlur_Click()
    Dim Gx, Gy, Dbg%, Dbr%, Dgb%, R%, G%, B As Integer
    Dim Gcolor As Long
        For Gx = 0 To 320
            For Gy = 0 To 240
            Gcolor = GetPixel(CameraH, Gx, Gy)
            Call ColorRGB(Gcolor, R, G, B)
            Dbg = B - G
            Dbr = B - R
            Dgb = G - B
            If Dbg > -20 And Dbr > 20 Then Gcolor = 16777215
            SetPixel Picture1.Hdc, Gx, Gy, Gcolor
        Next Gy
    Next Gx
End Sub

Private Sub Form_Load()
Dim i As Integer
MSComm1.Settings = "300,n,8,1"
SPEED1.Value = 20
```

```
    CameraH = GetDC(ezVidCap1.hwnd)
    If 0 < ezVidCap1.NumCapDevs Then
        For i = 0 To ezVidCap1.NumCapDevs - 1
            cbDriver.AddItem (ezVidCap1.GetDriverName(i))
        Next i
        cbDriver.ListIndex = ezVidCap1.DriverIndex
    Else
        cbDriver.AddItem ("<none>")
        cbDriver.ListIndex = 0
        MsgBox "No Video Capture Device!", vbInformation, App.Title
    End If
    If MSComm1.PortOpen = False Then MSComm1.PortOpen = True
    List1.AddItem "Car   = XXX" & vbTab & "  Degree"
    List1.AddItem "point = XXX" & vbTab & "  Degree"
    List1.AddItem "X= " & Car.X & vbTab & ", Y= " & Car.Y
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ReleaseDC ezVidCap1.hwnd, CameraH
    MSComm1.PortOpen = False
End Sub

Private Sub cbDriver_Click()
    Dim oldDriver As Long
    oldDriver = ezVidCap1.DriverIndex
    On Error Resume Next
    ezVidCap1.DriverIndex = cbDriver.ListIndex
    If Err Then
        ezVidCap1.DriverIndex = oldDriver
        cbDriver.ListIndex = oldDriver
    End If
End Sub
```

```
Private Sub Key_KeyDown(KeyCode As Integer, Shift As Integer)
End Sub

Private Sub Key_KeyUp(KeyCode As Integer, Shift As Integer)
End Sub

Private Sub MnuCom1_Click()
   MSComm1.PortOpen = False
   MSComm1.CommPort = 1
   MnuCom2.Checked = False
   MnuCom1.Checked = True
   MSComm1.PortOpen = True
End Sub

Private Sub MnuCom2_Click()
   MSComm1.PortOpen = False
   MSComm1.CommPort = 2
   MnuCom2.Checked = True
   MnuCom1.Checked = False
   MSComm1.PortOpen = True
End Sub

Private Sub CarGo()
   DAT = 3
End Sub
Private Sub CarReturn()
   DAT = 2
End Sub
Private Sub CarTurnLeft()
   DAT = 1
End Sub
```

```
Private Sub CarTurnRight()
   DAT = 4
End Sub
Private Sub CarStop()
   DAT = 9
End Sub

Private Sub Picture1_Mouseup(Button As Integer, Shift As Integer, XXX As Single, YYY As
Single)
   Dim X%, Y%, Carry%, Jr%
   Dim A%, B%, C%, h%, k%
   Dim ColorR%, ColorG%, ColorB%, Drg%, Drb%, Dgb%, Dgr%, red_x&, red_y&, green_x&,
green_y&
   Dim Gcolor As Long
   Dim D As Single
   Dim red_center As Point, green_center As Point
      Timer2.Interval = 0
      en = 0
      target = 1
      h = 0
      k = 0
      green_x = 0
      green_y = 0
      Picture1.Cls
      If CarArea(XXX, YYY, Rr) = False Then
         MsgBox "กรุณา click ใหม่", vbExclamation + vbOKOnly, "Error!"
         Exit Sub
      End If
      If Abs(AAA(0).X - XXX) <= 20 And Abs(AAA(0).Y - YYY) <= 20 Then
         Picture1.Line (AAA(0).X, AAA(0).Y)-(XXX, YYY), vbBlue
         Exit Sub
      End If
```

```
D = 3.14 / 8
A = Fix(Rrr * Cos(D))
B = Fix(Rrr * Cos(2 * D))
C = Fix(Rrr * Sin(D))
Carry = 0
i = 1
On Error Resume Next
For Y = 0 To 240
   For X = 0 To 320
      Gcolor = GetPixel(CameraH, X, Y)
      Call ColorRGB(Gcolor, ColorR, ColorG, ColorB)
      Drg = ColorR - ColorG
      Drb = ColorR - ColorB
      Dgb = ColorG - ColorB
      Dgr = ColorG - ColorR
      If Drg > 40 And Drb > 30 Then
         red_x = red_x + X
         red_y = red_y + Y
         h = h + 1
      End If
      If Dgr > -20 And Dgb > 55 Then
         green_x = green_x + X
         green_y = green_y + Y
         k = k + 1
      End If
      If Object(X, Y) = True Then
         SetPixel Picture1.Hdc, X, Y, vbYellow
         If Object(X, Y - 1) = True Then Carry = Carry + 1
         If Object(X - 1, Y - 1) = True Then Carry = Carry + 1
         If Object(X - 1, Y) = True Then Carry = Carry + 1
         If Object(X - 1, Y + 1) = True Then Carry = Carry + 1
         If Object(X, Y + 1) = True Then Carry = Carry + 1
```

```
If Object(X + 1, Y + 1) = True Then Carry = Carry + 1
If Object(X + 1, Y) = True Then Carry = Carry + 1
If Object(X + 1, Y - 1) = True Then Carry = Carry + 1
If Carry < 4 Then
   SetPixel Picture1.Hdc, X, Y, vbGreen
      If CarArea(X, Y - Rrr, Rr) = True Then       'find 8 points
         AAA(i).X = X
         AAA(i).Y = Y - Rrr
         SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
         i = i + 1
      End If
      If CarArea(X + B, Y - B, Rr) = True Then
         AAA(i).X = X + B
         AAA(i).Y = Y - B
         SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
         i = i + 1
      End If
      If CarArea(X + Rrr, Y, Rr) = True Then
         AAA(i).X = X + Rrr
         AAA(i).Y = Y
         SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
         i = i + 1
      End If
      If CarArea(X + B, Y + B, Rr) = True Then
         AAA(i).X = X + B
         AAA(i).Y = Y + B
         SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
         i = i + 1
      End If
      If CarArea(X, Y + Rrr, Rr) = True Then
         AAA(i).X = X
         AAA(i).Y = Y + Rrr
```

```
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
              End If
              If CarArea(X - B, Y + B, Rr) = True Then
                AAA(i).X = X - B
                AAA(i).Y = Y + B
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
              End If
              If CarArea(X - Rrr, Y, Rr) = True Then
                AAA(i).X = X - Rrr
                AAA(i).Y = Y
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
              End If
              If CarArea(X - B, Y - B, Rr) = True Then
                AAA(i).X = X - B
                AAA(i).Y = Y - B
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
              End If
            If i > 200 Then
              Label4.Caption = "!ERROR!"
              Exit Sub
            End If
          End If
          Carry = 0
        End If
      Next X
    Next Y
    AAA(i).X = XXX
    AAA(i).Y = YYY
```

```
If h = 0 Or k = 0 Then Exit Sub
red_center.X = Fix(red_x / h)
red_center.Y = Fix(red_y / h)
green_center.X = Fix(green_x / k)
green_center.Y = Fix(green_y / k)
Car.X = Fix((red_center.X + green_center.X) / 2)
Car.Y = Fix((green_center.Y + red_center.Y) / 2)
Picture1.Circle (Car.X, Car.Y), 5
AAA(0).X = Car.X
AAA(0).Y = Car.Y
ReDim P(i, i)
For Y = 0 To i
  For X = 0 To i
    If X = Y Then
      P(X, Y) = M
    Else
      If Abs(AAA(X).X - AAA(Y).X) < 5 And Abs(AAA(X).Y - AAA(Y).Y) < 5 Then
        P(X, Y) = M
      Else
        If Point2Point(X, Y) Then
          P(X, Y) = Fix(((AAA(X).X - AAA(Y).X) ^ 2 + (AAA(X).Y - AAA(Y).Y) ^ 2) ^
(1 / 2))
        Else
          P(X, Y) = M
        End If
      End If
    End If
  Next X
Next Y
PathA i, P(), XX(), AA
If en = 1 Then
  Label4.Caption = "!ERROR!"
```

```
        Exit Sub
      End If
      For X = 1 To AA
        Picture1.Line (AAA(XX(X - 1)).X, AAA(XX(X - 1)).Y)-(AAA(XX(X)).X, AAA(XX
(X)).Y), vbBlue
      Next X
      Label4.Caption = "Go Baby!"
      Timer2.Interval = 25
End Sub

Private Function Point2Point(ByVal P1%, ByVal P2%) As Boolean
  Dim Cx%, Cy%, Dx%, Dy%, Ex%, Ey%, j%
  Point2Point = True
  Cy = AAA(P2).Y
  Cx = AAA(P2).X
  Dx = AAA(P1).X - AAA(P2).X
  Dy = AAA(P1).Y - AAA(P2).Y
  If Abs(Dx) > Abs(Dy) Then
    If Dx > 0 Then
      For j = 0 To Dx Step 20
        Ey = (Dy / Dx) * j + Cy
        Ex = j + Cx
        If CarArea(Ex, Ey, Rr) = False Then
          Point2Point = False
          Exit Function
        End If
      Next j
    Else
      For j = Dx To 0 Step 20
        Ey = (Dy / Dx) * j + Cy
        Ex = j + Cx
        If CarArea(Ex, Ey, Rr) = False Then
```

```
            Point2Point = False
            Exit Function
          End If
        Next j
      End If
    Else
      If Dy > 0 Then
        For j = 0 To Dy Step 20
          Ex = (Dx / Dy) * j + Cx
          Ey = j + Cy
          If CarArea(Ex, Ey, Rr) = False Then
            Point2Point = False
            Exit Function
          End If
        Next j
      Else
        For j = Dy To 0 Step 20
          Ex = (Dx / Dy) * j + Cx
          Ey = j + Cy
          If CarArea(Ex, Ey, Rr) = False Then
            Point2Point = False
            Exit Function
          End If
        Next j
      End If
    End If

End Function

Private Function Object(ByVal X%, ByVal Y%) As Boolean
Dim Gcolor As Long
Dim ColorR%, ColorG%, ColorB%, Dbg%, Dbr%, Dgb%
```

```
    Object = False
    Gcolor = GetPixel(CameraH, X, Y)
    Call ColorRGB(Gcolor, ColorR, ColorG, ColorB)
    Dbg = ColorB - ColorG
    Dbr = ColorB - ColorR
    Dgb = ColorG - ColorB
    If Dbg > -20 And Dbr > 20 Then Object = True
    End Function

Private Function CarArea(ByVal X%, ByVal Y%, ByVal Rc%) As Boolean
    Dim A%, B%, C%
    Dim D As Single
    D = 3.14 / 8
    A = Fix(Rc * Cos(D))
    B = Fix(Rc * Cos(2 * D))
    C = Fix(Rc * Sin(D))
    CarArea = True
    If X - Rc < 0 Or X + Rc > 320 Or Y - Rc < 0 Or Y + Rc > 240 Then
       CarArea = False
    ElseIf Object(X, Y - Rc) = True Then
       CarArea = False
    ElseIf Object(X + C, Y - A) = True Then
       CarArea = False
    ElseIf Object(X + B, Y - B) = True Then
       CarArea = False
    ElseIf Object(X + A, Y - C) = True Then
       CarArea = False
    ElseIf Object(X + Rc, Y) = True Then
       CarArea = False
    ElseIf Object(X + A, Y + C) = True Then
       CarArea = False
```

```
    ElseIf Object(X + B, Y + B) = True Then
      CarArea = False
    ElseIf Object(X + C, Y + A) = True Then
      CarArea = False
    ElseIf Object(X, Y + Rc) = True Then
      CarArea = False
    ElseIf Object(X - C, Y + A) = True Then
      CarArea = False
    ElseIf Object(X - B, Y + B) = True Then
      CarArea = False
    ElseIf Object(X - A, Y + C) = True Then
      CarArea = False
    ElseIf Object(X - Rc, Y) = True Then
      CarArea = False
    ElseIf Object(X - A, Y - C) = True Then
      CarArea = False
    ElseIf Object(X - B, Y - B) = True Then
      CarArea = False
    ElseIf Object(X - C, Y - A) = True Then
      CarArea = False
    End If
End Function

Private Function ColorRGB(Gcolor As Long, R As Integer, G As Integer, B As Integer)
  R = Gcolor Mod 256
  B = Gcolor \ 65536
  G = (Gcolor \ 256) Mod 256
End Function
Private Sub Timer1_Timer()
  Timer1.Interval = 0
  DAT = 9
End Sub
```

```
Private Sub Timer2_Timer()
Dim Drg%, Drb%, Dgb%, R%, G%, B%, Dx%, Dy%, X%, Y%, red_x&, red_y&, green_x&,
green_y&, h%, k%
Dim Gcolor As Long
Dim red_center As Point, green_center As Point
Dim RadCar As Double, RadPoint As Double
  red_x = 0
  red_y = 0
  h = 0
  k = 0
  green_x = 0
  green_y = 0
  For Y = Car.Y - 30 To Car.Y + 30
    For X = Car.X - 30 To Car.X + 30
      If X < 0 Or X > 320 Or Y < 0 Or Y > 240 Then
        Gcolor = vbBlack
      Else
        Gcolor = GetPixel(CameraH, X, Y)
      End If
      Call ColorRGB(Gcolor, R, G, B)
      Drg = R - G
      Drb = R - B
      Dgb = G - B
      If Drg > 30 And Drb > 30 Then
        red_x = red_x + X
        red_y = red_y + Y
        h = h + 1
      End if
      If Drg < -30 And Dgb > 30 Then
        green_x = green_x + X
        green_y = green_y + Y
        k = k + 1
```

```
        End If
      Next X
    Next Y
Dim Loot As Double, Arcsin As Double, Arccos As Double, Carry As Double
    If h <> 0 And k <> 0 And red_x <> 0 And red_y <> 0 And green_x <> 0 And green_y <> 0
Then
    red_center.X = Fix(red_x / h)
    red_center.Y = Fix(red_y / h)
    green_center.X = Fix(green_x / k)
    green_center.Y = Fix(green_y / k)
    End If
    Car.X = Fix((red_center.X + green_center.X) / 2)
    Car.Y = Fix((green_center.Y + red_center.Y) / 2)
    'Find Car's angle
    Dx = red_center.X - green_center.X
    Dy = green_center.Y - red_center.Y
    Loot = (Dx ^ 2 + Dy ^ 2) ^ 0.5
    If Loot = 0 Then Loot = 0.01
    Carry = Dy / Loot
    If Abs(Carry) = 1 Then Carry = Carry * 0.99
    Arcsin = Atn(Carry / Sqr(-Carry * Carry + 1))
    Carry = Dx / Loot
    If Abs(Carry) = 1 Then Carry = Carry * 0.99
    Arccos = Atn(-Carry / Sqr(-Carry * Carry + 1)) + 1.57
    If Arcsin < 0 Then
      If Carry < 0 Then
        Arccos = Arccos - 2 * Arcsin
      Else
        Arccos = 6.28 - Arccos
      End If
    End If
    RadCar = Arccos * 180 / pi
```

```
'Find Point's angle
Dx = AAA(XX(target)).X - Car.X
Dy = Car.Y - AAA(XX(target)).Y
Loot = (Dx ^ 2 + Dy ^ 2) ^ 0.5
If Loot = 0 Then Loot = 0.01
Carry = Dy / Loot
If Abs(Carry) = 1 Then Carry = Carry * 0.99
Arcsin = Atn(Carry / Sqr(-Carry * Carry + 1))
Carry = Dx / Loot
If Abs(Carry) = 1 Then Carry = Carry * 0.99
Arccos = Atn(-Carry / Sqr(-Carry * Carry + 1)) + 1.57
If Arcsin < 0 Then
   If Carry < 0 Then
      Arccos = Arccos - 2 * Arcsin
   Else
      Arccos = 6.28 - Arccos
   End If
End If
RadPoint = Arccos * 180 / pi
List1.Clear
List1.AddItem "Car   = " & RadCar & vbTab & " Degree"
List1.AddItem "point = " & RadPoint & vbTab & " Degree"
List1.AddItem "X= " & Car.X & vbTab & ", Y= " & Car.Y
If RadCar - RadPoint > 180 Then
   RadPoint = RadPoint + 360
ElseIf RadCar - RadPoint < -180 Then
   RadCar = RadCar + 360
End If
'Select Command
If Abs(Car.X - AAA(XX(target)).X) < 10 And Abs(Car.Y - AAA(XX(target)).Y) < 10 Then
   target = target + 1
```

```
ElseIf RadCar - RadPoint > 15 Then
    DAT = 4
  ElseIf RadCar - RadPoint < -15 Then
    DAT = 1
  Else
    DAT = 3
  End If
  If target > AA Then
    Timer2.Interval = 0
    Label4.Caption = "Ready!"
  End If
  If Car.X = 0 Or Car.Y = 0 Then
    Timer2.Interval = 0
    Label4.Caption = "!ERROR!"
  End If
  Timer1.Interval = 1

End Sub
'*************************( MANNUAL )*************************
Private Sub CmdLeft_Click()   'LEFT
DAT = 1
End Sub
Private Sub CmdBw_Click()   'BACK
DAT = 2
End Sub
Private Sub CmdFw_Click()   'FW
DAT = 3
End Sub
Private Sub CmdRight_Click()   'RIGHT
DAT = 4
End Sub
```

```
Private Sub CmdExit_Click()
Unload Me
End Sub
Private Sub CmdSlow_Click()   'DOWN SPEED
DAT = 0
If SPEED1.Value > 1 Then
SPEED1.Value = SPEED1.Value - 1
DAT = 5
End If
End Sub
Private Sub CmdFast_Click()   'UP SPEED
DAT = 0
If SPEED1.Value < 39 Then
SPEED1.Value = SPEED1.Value + 1
DAT = 6
End If
End Sub
Private Sub CmdStop_Click()   'Stop
   DAT = 9
End Sub
Private Sub Timer3_Timer()
If DAT = 9 Then
MSComm1.Output = "0"
DAT = 0
ElseIf DAT = 1 Then
MSComm1.Output = "1"
DAT = 0
ElseIf DAT = 2 Then
 MSComm1.Output = "2"
DAT = 0
ElseIf DAT = 3 Then
```

```
MSComm1.Output = "3"
DAT = 0
ElseIf DAT = 4 Then
MSComm1.Output = "4"
DAT = 0
ElseIf DAT = 5 Then
MSComm1.Output = "D"
DAT = 0
ElseIf DAT = 6 Then
MSComm1.Output = "U"
DAT = 0
End If
End Sub
Private Sub Timer4_Timer()
Label3.Caption = Time
End Sub

Option Explicit
Public Declare Function GetPixel Lib "gdi32" (ByVal Hdc As Long, ByVal X As Long,
ByVal Y As Long) As Long
Public Declare Function SetPixel Lib "gdi32" (ByVal Hdc As Long, ByVal X As Long,
ByVal Y As Long, ByVal crColor As Long) As Long
Public Const M = 9999
Public en As Byte
Public Type POINTAPI
    X As Long
    Y As Long
End Type
Public Sub PathA(N As Integer, P() As Integer, ByRef A() As Integer, ByRef AA As Byte)
  Dim S As Byte, t As Byte, C As Byte, j As Byte, i As Byte, II As Byte, k As Byte
  Dim dc As Single, smalldist As Single, newdist As Single
  Dim distance() As Single
```

```
Dim precede() As Byte, repath() As Byte
Dim TT As String
ReDim distance(N)
ReDim precede(N)
ReDim repath(N)
Dim perm() As Integer
ReDim perm(2 * N)
  S = 0
  t = N
  i = 0
  perm(0) = 0
  For II = 0 To N
    distance(II) = M
  Next II
  distance(S) = 0
  C = S
  Do While (C <> t)
    smalldist = M
    dc = distance(C)
    For II = 0 To N
      If Not find(II, perm(), i) Then      'II in perm ?
        newdist = dc + P(C, II)
        If newdist > M Then
          newdist = M
        End If
        If newdist < distance(II) Then
          distance(II) = newdist
          precede(II) = C
        End If
        If distance(II) < smalldist Then
          smalldist = distance(II)
          k = II
```

```
            End If
          End If
        Next II
        C = k
        i = i + 1              'count perm
        If i > N Then
          MsgBox "ไม่มีทางไป", vbExclamation + vbOKOnly, "Error"
          en = 1
          Exit Sub
        End If
        perm(i) = C
      Loop
      II = 0
      repath(II) = t
      Do
        II = II + 1
        repath(II) = precede(k)
        k = precede(k)
      Loop Until k = S
      ReDim A(II)
      For j = 0 To II
        A(j) = (repath(II - j))
        TT = TT + " " + CStr(A(j))
      Next j
      AA = II
End Sub
Private Function find(ByVal U As Byte, ByRef V() As Integer, ByVal j1 As Byte) As Boolean
Dim i1 As Byte
find = False
```

```
or i1 = 0 To j1
   If V(i1) = U Then
      find = True
   End If
Next i1
End Function
```

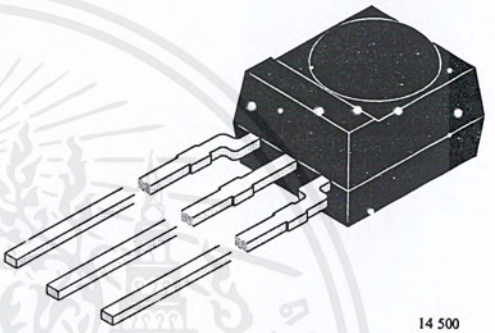
Photo Modules for PCM Remote Control Systems

Available types for different carrier frequencies

Type	fo	Type	fo
TSOP4830	30 kHz	TSOP4833	33 kHz
TSOP4836	36 kHz	TSOP4837	36.7 kHz
TSOP4838	38 kHz	TSOP4840	40 kHz
TSOP4856	56 kHz		

Description

The TSOP48.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.
The demodulated output signal can directly be decoded by a microprocessor. TSOP48.. is the standard IR remote control receiver series, supporting all major transmission codes.

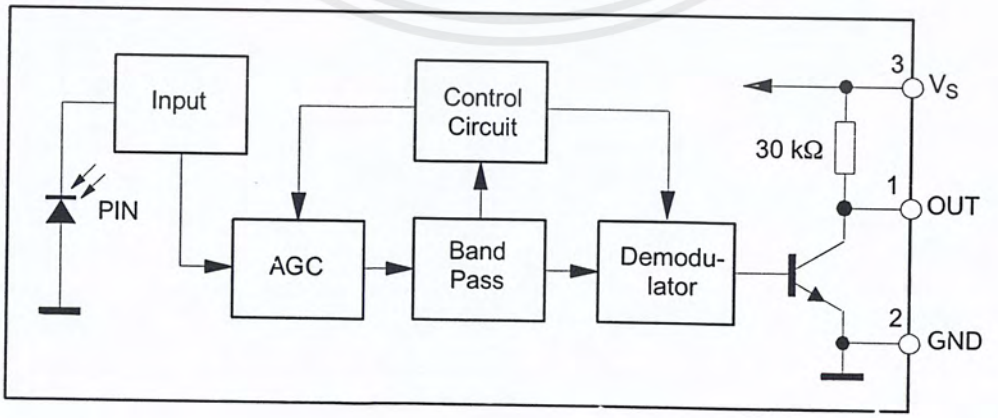


14 500

Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption
- High immunity against ambient light
- Continuous data transmission possible (800 bit/s)
- Suitable burst length ≥10 cycles/burst

Block Diagram



9612226

Absolute Maximum Ratings

T_{amb} = 25°C

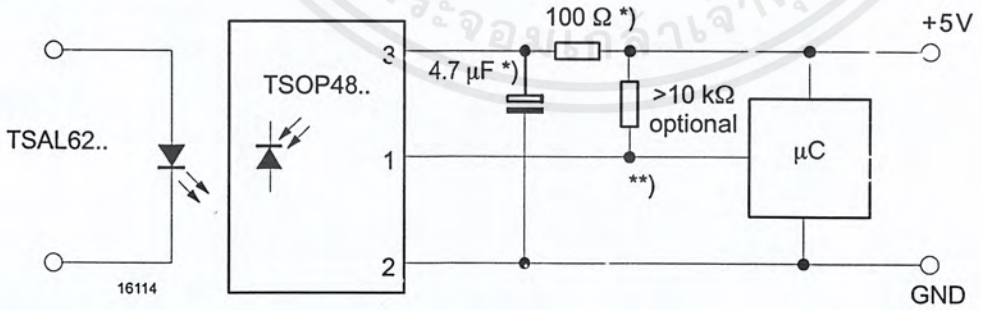
Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 3)	V_S	–0.3...6.0	V
Supply Current	(Pin 3)	I_S	5	mA
Output Voltage	(Pin 1)	V_O	–0.3...6.0	V
Output Current	(Pin 1)	I_O	5	mA
Junction Temperature		T_j	100	°C
Storage Temperature Range		T_{stg}	–25...+85	°C
Operating Temperature Range		T_{amb}	–25...+85	°C
Power Consumption	(T_{amb} ≦ 85 °C)	P_{tot}	50	mW
Soldering Temperature	t ≦ 10 s, 1 mm from case	T_{sd}	260	°C

Basic Characteristics

T_{amb} = 25°C

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 3)	V_S = 5 V, E_v = 0	I_{SD}	0.8	1.1	1.5	mA
	V_S = 5 V, E_v = 40 klx, sunlight	I_{SH}		1.4		mA
Supply Voltage (Pin 3)		V_S	4.5		5.5	V
Transmission Distance	E_v = 0, test signal see fig.7, IR diode TSAL6200, I_F = 250 mA	d		35		m
Output Voltage Low (Pin 1)	I_{OSL} = 0.5 mA, E_e = 0.7 mW/m²	V_{OSL}			250	mV
Irradiance (30 – 40 kHz)	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see fig.7	$E_{e\ min}$		0.2	0.4	mW/m²
Irradiance (56 kHz)	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see fig.7	$E_{e\ min}$		0.3	0.6	mW/m²
Irradiance	$t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$	$E_{e\ max}$	30			W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$		±45		deg

Application Circuit



*) recommended to suppress power supply disturbances
**) The output voltage should not be hold continuously at a voltage below 3.3V by the external circuit.