กล้องวีดีโอตรวจจับเป้าหมาย

Camera Detective Target

โดย

นาย ฉัตรชัย ทองระหมาน

นาย ถาวร พรหมมาศ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

กล้องวีดีโอตรวจจับเป้าหมาย

Camera Detective Target

โดย

นาย ฉัตรชัย ทองระหมาน รหัสประจำตัว 46015257

นาย ถาวร พรหมมาศ รหัสประจำตัว 46015261

อาจารย์ที่ปรึกษา

ดร. ยุทธนา คิดใจเดียว

ปริญญานิพนธ์นี้สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ ปีการศึกษา 2548

สาขาวิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง กล้องวีดีโอตรวจจับเป้าหมาย

ผู้จัดทำ

1. นาย ฉัตรชัย ทองระหมาย
2. นาย ณาวุ พรหมมาศ

.....อาจารย์ที่ปรึกษา

(ดร. ยุทธนา คิดใจเดียว)

กล้องวีดีโอตรวจจับเป้าหมาย

นาย ฉัตรชัย ทองระหมาน รหัส 46015257
นาย ถาวร พรหมมาศ รหัส 46015261
ดร. ยุทธนา คิดใจเดียว อาจารย์ที่ปรึกษา
ปีการศึกษา 2548

บทคัดย่อ

สำหรับการพัฒนาระบบด้วยการใช้กล้องวีดีโอในการประมวลผลภาพ ปัจจุบันได้นำมาใช้งานอย่างกว้างขวาง ทั้งในองค์กรที่มีขนาดเล็ก ตลอดไปจนถึงองค์กรที่มีขนาดใหญ่ และมีแนวโน้มว่าจะมีการนำไปใช้งานมากขึ้นเรื่อยๆในอนาคต

และนี่ก็เป็นอีกหนึ่งในโครงงาน ที่นำเอากล้องวีดีโอ มาทำการประมวลผลภาพวัตถุ เพื่อใช้ในระบบรักษาความปลอดภัย และอาจนำไปประยุกต์ใช้งานด้านอื่นได้อีกด้วย โดยการทำงานของระบบเริ่มจากการรับภาพเป้าหมายที่ต้องการ ผ่านมาทางกล้องวีดีโอ แล้วให้คอมพิวเตอร์ทำการเก็บค่าที่ได้ แล้วทำการประมวลผลค่าภาพนั้นเพื่อหาตำแหน่งของเป้าหมาย จากนั้นคอมพิวเตอร์จะทำการส่งค่าข้อมูลสั่งให้ไมโครคอนโทรลเลอร์ทำการควบคุมกล้องหมุนเพื่อตามเป้าหมายนั้นไป

Camera Detective Target

Mr.Chatchai Thongramarn ID. 46015257

Mr.Tharworn Prommart ID. 46015261

Dr.Yutthana Kidjaidiew Advisor

Educational Year 2005

Abstract

For the system development on camera usage for image processing , it is widly used in small and large organization. In the present day furthermore the tendency is increasing in the future.

This is one project that uses a camera for image processing in a security system and it also applies to another purpose . The system begins with receiving image targets from a camera and transmiting to a computer for image processing . Then the computer acknowledges and transfers data to microcontroller , which allows the camera to change its position for capturing the target .

กิตติกรรมประกาศ

ขอขอบคุณ ดร.ยุทธนา คิดใจเดียว เป็นอย่างสูงที่ช่วยให้คำปรึกษา คำแนะนำที่ดีต่างๆ มาโดยตลอด และขอขอบคุณพี่ๆ เพื่อนๆที่ให้คำแนะนำ ช่วยเหลือทุกครั้งที่ต้องการคำปรึกษา พร้อมทั้งคอยเป็นกำลังใจที่ดีมาเสมอ

(นายฉัตรชัย ทองระหมาน)

(นายถาวร พรหมมาศ)

ผู้จัดทำ

สารบัญ

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

ในปัจจุบันนี้ มีการพัฒนาวิวัฒนาการเกี่ยวกับการใช้งานอุปกรณ์เทคโนโลยี เป็นไปอย่างรวดเร็ว และเป็นที่แพร่หลายในปัจจุบัน ไม่ว่าจะเป็นการใช้งานในรูปแบบต่างๆ แต่ก็จำกัดอยู่แค่เพียงเป็นการนำเอาเทคโนโลยีไปใช้งาน ในวัตถุประสงค์ที่ทางฝ่ายผู้ผลิตกำหนดมาแค่นั้นเอง

แต่ก็ยังมีอีกส่วนหนึ่ง ซึ่งนำเอาความเจริญก้าวหน้าทางเทคโนโลยีนี้ มาดัดแปลงเพื่อประยุกต์ใช้งานในด้านที่ต่างกัน นอกเหนือข้อจำกัดหรือวัตถุประสงค์ที่ทางฝ่ายผู้ผลิตกำหนดมา คือ นำเอาขีดความสามารถของอุปกรณ์ ที่มีความทันสมัย มาดัดแปลงใช้งานร่วมกับ การทำงานในรูปแบบต่างๆมากมาย และนี่ก็เป็นอีกโครงงานหนึ่งที่นำเอากล้องวีดีโอ ซึ่งเป็นอุปกรณ์ที่มีความทันสมัยมาประยุกต์ ใช้งานร่วมกับ การทำงานของไมโครคอนโทรลเลอร์ เพื่อใช้ในงานระบบรักษาความปลอดภัย หรืออาจนำไปใช้งานในด้านอุตสาหกรรม และอื่นๆได้อีกด้วย ทั้งนี้เพื่อหลีกหนีข้อจำกัดของการใช้งานกล้องวีดีโอที่อาจเป็นเพียงแค่อุปกรณ์ที่ใช้ในการบันทึกภาพเท่านั้น และการทำงานของกล้องวีดีโอมีการเชื่อมต่อเข้ากับ พอร์ตอนุกรมของคอมพิวเตอร์เพื่อทำการแสดงผลภาพ และ ทำการควบคุมกล้องวีดีโอผ่านทางการทำงานของ ไมโครคอนโทรลเลอร์ โดยในตอนแรกคอมพิวเตอร์จะเริ่มทำการเก็บค่าภาพ โดยการใช้การ์ดวีดีโอทำการ Capture ภาพที่ต้องการตรวจสอบ หรือบุคคลที่ต้องสงสัย จากนั้นคอมพิวเตอร์จะทำการส่งค่าภาพที่ได้ ให้กับไมโครคอนโทรลเลอร์ เพื่อให้ไมโครคอนโทรลเลอร์ ทำการควบคุมทิศทางของการหมุนของกล้องวีดีโอ เพื่อตรวจสอบเป้าหมายนั้นไป หรืออาจเป็นการประยุกต์ใช้งาน โดยนำเอาภาพของบุคคลที่ต้องสงสัยเก็บค่าไว้ เปรียบเหมือนเป็นฐานข้อมูล หากบุคคลต้องสงสัยนั้น เข้ามาในเขตหรือพื้นที่ความสามารถของกล้องวีดีโอ กล้องก็จะตรวจจับและรายงานให้กับผู้ที่มีส่วนเกี่ยวข้องทราบในทันที จะเห็นได้ว่าจากขีดความสามารถนี้ เป็นการใช้งานที่มีประโยชน์อย่างยิ่ง ในด้านมาตรการที่เกี่ยวข้องกับการรักษาความปลอดภัย

วัตถุประสงค์ของโครงงาน

1. เพื่อเป็นการศึกษาและประยุกต์ใช้งาน เกี่ยวกับการประมวลผลภาพผ่านคอมพิวเตอร์
2. เพื่อเป็นการศึกษาการทำงานของไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม
3. เพื่อการประยุกต์ใช้งานกล้องวีดีโอ ผ่านระบบรักษาความปลอดภัย

บทที่ 2
ทฤษฎีไมโครคอนโทรลเลอร์

2.1ประวัติของไมโครคอนโทลเลอร์

อุปกรณ์อิเล็กทรอนิกส์ในปัจจุบันจะถูกควบคุมด้วยระบบคอมพิวเตอร์เล็กๆหรือเรียกว่าไมโครโปรเซสเซอร์เกือบทั้งสิ้น ในปัจจุบันจะเรียกชื่อเป็นศัพท์เทคนิคว่าระบบฝังตัว (Embedded System) ระบบควบคุมด้วยคอมพิวเตอร์จะต้องมีหน่วยประมวลผลกลางที่เรียกว่าไมโครโปรเซสเซอร์เป็นหัวใจหลักของการทำงาน โดยบริษัทอินเทลได้สร้างไมโครโปรเซสเซอร์เบอร์ 4004 ซึ่งประมวลผลแบบ 8 บิต ออกมาเป็นรุ่นแรก ต่อมาได้ออกรุ่นที่ประมวลผลแบบ 8 บิตตามมาได้แก่ 8008, 8080 และ 8085 ทำให้การประมวลผลได้รวดเร็วขึ้น ส่วนบริษัทโมโตลาได้ออกเบอร์ 6800 และบริษัทไซลอกได้ออกเบอร์ Z80 ซึ่งจะประมวลผลแบบ 8 บิต เช่นกัน ไมโครโปรเซสเซอร์รุ่นต่อมา จะมีประสิทธิภาพในการทำงานมากขึ้นและได้มีการผลิตรุ่นใหม่ๆออกมามากขึ้น

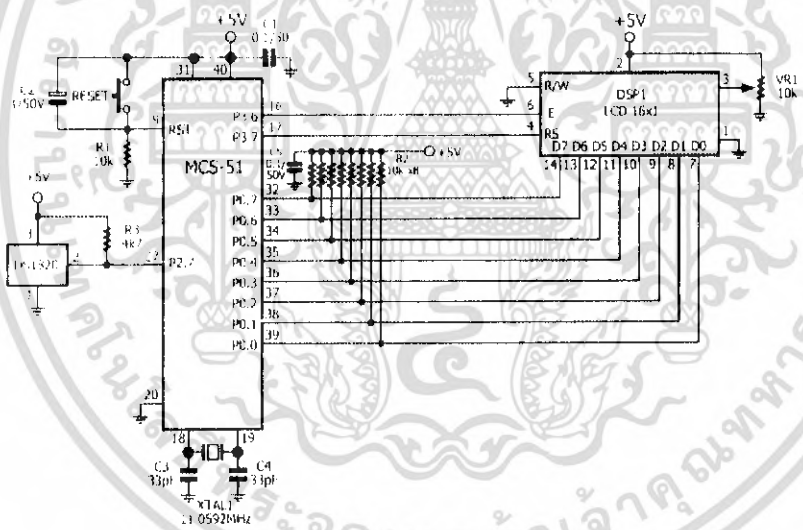
เมื่อนำไมโครโปรเซสเซอร์แบบ 4 บิต มาใช้ในการควบคุมจะทำให้ระบบควบคุมทำงานได้ดีขึ้นฉลาดมากขึ้น ปัจจุบันไมโครโปรเซสเซอร์แบบ 4 บิต นี้จะถูกนำมาใช้ในเตาไมโครเวฟ โทรทัศน์ และของเด็กเล่น เป็นต้น สำหรับระบบควบคุมที่ต้องการประสิทธิภาพมากขึ้นจะใช้ไมโครโปรเซสเซอร์แบบ 8 บิต เป็นตัวประมวลผล แต่ราคาของระบบก็แพงขึ้นตามไปด้วย

ไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์

ไมโครคอมพิวเตอร์ มีส่วนประกอบหลักคือ ไมโครโปรเซสเซอร์ หน่วยความจำ และอินพุทเอาท์พุท หน่วยประมวลผล มีขนาดใหญ่เทียบได้กับห้องขนาดย่อมๆเลยทีเดียว ปัจจุบันนี้เราจะเห็น ว่าหน่วยประมวลผลจะเป็นวงจรรวมขนาดใหญ่มาก (VLSI) บรรจุอยู่ในตัวถังพลาสติกขนาด 40 ขา ตัวอย่างของไมโครโปรเซสเซอร์ที่เป็นที่นิยมกันมานานจนถึงปัจจุบันนี้คือ Z80 CPU ภายในจะมีบัสเชื่อมส่วนของหน่วยประมวลผลทางคณิตศาสตร์และลอจิก (ALU) หน่วยประมวลผล (CPU) รีจิสเตอร์ คำสั่ง และวงจรควบคุมสัญญาณต่างๆเข้าด้วยกัน ส่วนของหน่วยประมวลผลทางคณิตศาสตร์และลอจิก (ALU) ทำหน้าที่ประมวลผลทางคณิตศาสตร์และตรรกศาสตร์ ผลลัพธ์จะส่งไปเก็บยังรีจิสเตอร์ภายใน จะเห็นว่ามีกลุ่มสัญญาณสามกลุ่ม คือ 1) บัสข้อมูลขนาดความกว้าง 8 บิต 2) บัสแอดเดรสขนาด 16 บิต และ 3) สัญญาณควบคุมการติดต่อกับหน่วยความจำ อินพุทเอาท์พุท คือไมโครโปรเซสเซอร์ไม่มีหน่วยความจำที่ใช้เก็บโปรแกรมกล่าวคือหากจะทำให้ไมโครโปรเซสเซอร์ทำงานได้ จะต้องต่อหน่วยความจำภายนอกเพิ่ม และหากต้องการควบคุมอุปกรณ์อินพุทเอาท์พุท ก็ต้องต่ออินพุทเอาท์พุทเพิ่มเติม

ส่วนไมโครคอนโทรลเลอร์หรืออดีตนิยมเรียกว่า ไมโครคอมพิวเตอร์ชิพเดี่ยว (Single Chip Microcomputer) ได้รวมเอาหน่วยความจำ อินพุทเอาท์พุทพอร์ท และวงจรพิเศษ เช่นวงจรตั้งเวลาเข้าไว้บนชิพแผ่นเดียว ทำให้เราไม่ต่ออุปกรณ์ภายนอก แม้กระทั่งวงจรกำเนิดสัญญาณนาฬิกา เพียงแค่ต่อไฟเลี้ยง +5V วงจรรีเซ็ต และต่อคริสตอลกำเนิดความถี่ในการทำงานไมโครคอนโทรลเลอร์ก็สามารถทำงานได้แล้ว

การนำไมโครโปรเซสเซอร์มาใช้งานจะต้องมีการเขียนโปรแกรมควบคุมการทำงานและมีหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลที่ได้จากการประมวลผล และต้องมีพอร์ทอินพุทเอาท์พุท สำหรับให้ระบบติดต่อกับอุปกรณ์ภายนอก ตัวอย่างเช่น ถ้านำไมโครโปรเซสเซอร์เบอร์ 8052 มาออกแบบเป็นระบบให้ทำงานหนึ่งจะต้องนำไอซีต่างๆ มาต่อเพิ่มเช่น การต่อ รอมและแรมภายนอก การใช้ ไอซี 8255 ต่อเป็นอินพุทเอาท์พุทพอร์ท แต่ถ้าหากใช้ไมโครคอนโทรลเลอร์จะมีหน่วยความจำและพอร์ทอยู่ภายในชิพ ที่เรียกว่า Single Chip Microcontroller ทำให้การใช้งานทำได้เพียงแต่ต่ออุปกรณ์ภายนอกที่จำเป็นเท่านั้น บางครั้งไมโครคอนโทรลเลอร์จะถูกเรียกว่าคอมพิวเตอร์ชิพเดี่ยว (Single Chip Microcontroller) ดังตัวอย่างรูปที่ 2.1 เป็นการออกแบบระบบไมโครคอนโทรลเลอร์ ซึ่งจะเห็นว่าใช้อุปกรณ์น้อยมาก



รูปที่ 2.1 ระบบไมโครคอนโทรลเลอร์

2.2 คุณสมบัติที่สำคัญของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายเบอร์ขึ้นกับโครงสร้างภายใน บางเบอร์จะมีหน่วยความจำภายในเป็นแบบรอม บางเบอร์เป็นแบบอีพรอม และปัจจุบันมีแบบแฟลชรอม

คุณสมบัติที่สำคัญของชิพไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีดังนี้

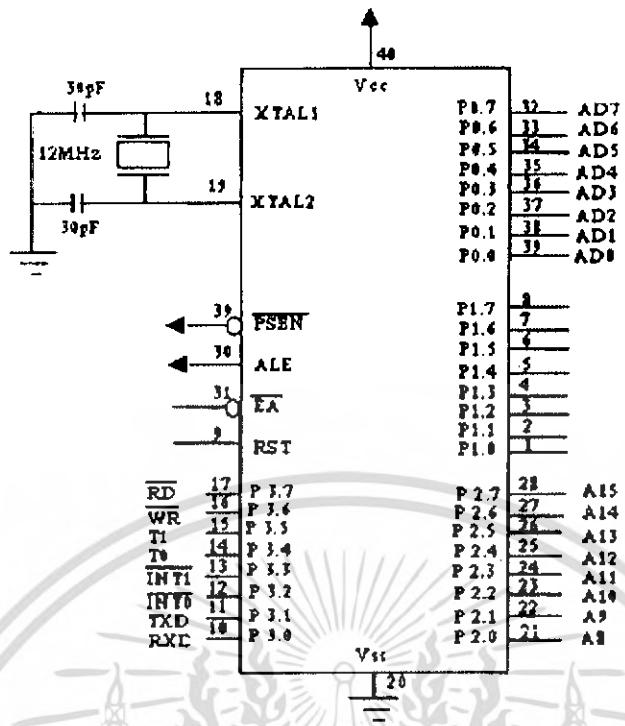
- ต้องการแหล่งจ่ายไฟ 5V เพียงชุดเดียว
- มีหน่วยความจำภายในชิพ จำนวน 128 ไบต์ (บาง CPU จะมี 256 ไบต์)
- มีพอร์ทอินพุทเอาท์พุทจำนวน 4 พอร์ท พอร์ทละ 8 บิต หรือสามารถใช้งานเป็นพอร์ท ขนาด 1 บิตแยกจากกันได้รวมทั้งสิ้น 32 บิต
- สามารถ Interrupts ได้จาก 5 แหล่ง
- มีพอร์ทอนุกรมที่สามารถรับส่งข้อมูลแบบฟูลดูเพล็กซ์ ด้วยอัตราเร็วในการรับส่งได้ตั้งแต่ 300 ถึง 375 กิโลบิตต่อวินาที
- สามารถใช้หน่วยความจำสำหรับโปรแกรมและข้อมูลที่อยู่ภายในชิพได้อย่างละ 64 กิโลไบต์
- คำสั่งส่วนใหญ่ใช้เวลาทำงานเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกะเฮิร์ต
- มีรีจิสเตอร์สำหรับใช้งานเป็นไทม์เมอร์หรือเคาท์เตอร์เพื่อนับสัญญาณนาฬิกาภายในชิพ หรือนับการเปลี่ยนสถานะของสัญญาณภายนอกขนาด 16 บิต จำนวน 2 ตัว
- หน่วยความจำภายในบางส่วนสามารถเข้าถึงข้อมูลได้ทั้งในระดับไบต์และระดับบิตเพื่อให้การออกแบบโปรแกรมและการควบคุมระบบทำได้ง่ายขึ้น

เบอร์	หน่วยความจำโปรแกรมบนชิพ	หน่วยความจำข้อมูลบนชิพ	TIMERS
8051	4 KB (ROM)	128 byte	2
8031	-	128 byte	2
8751	4 KB (EPROM)	128 byte	2
8052	4 KB (ROM)	256 byte	3
8032	-	256 byte	3
8752	8 KB (EPROM)	256 byte	3
89C51	4 KB (EPROM)	128 byte	2

ตารางที่ 2.1 แสดงไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ต่างๆ

2.3 โครงสร้างของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกันดังรูปที่ 2.3



รูปที่ 2.2 แสดงขาต่างๆ ของ MCS-51

2.3.1 ความหมายของขาต่างๆ มีดังนี้

พอร์ท 0 (Port 0) ได้แก่ ขาที่ 32-39 ของ MCS-51 สามารถใช้เป็นอินพุทเอาท์พุทสามารถใช้แบบบิตได้ นอกจากนี้ในการติดต่อกับหน่วยความจำภายนอกยังใช้เป็นขาบัสแอดเดรส และบัสข้อมูลอีกด้วย

พอร์ท 1 (Port 1) ได้แก่ ขาที่ 1-8 เป็นพอร์ทอินพุทเอาท์พุท 8 บิต สามารถใช้แบบบิตได้ คือ P1.0 – P1.7

พอร์ท 2 (Port 2) ได้แก่ ขาที่ 21-28 จะใช้งานสองหน้าที่คือ ใช้เป็นพอร์ทอินพุทเอาท์พุสามารถใช้แบบบิตได้และใช้เป็นขาแอดเดรส 8 บิต ในการอ้างอิงหน่วยความจำภายนอก

พอร์ท 3 (Port 3) ได้แก่ ขาที่ 10-17 จะใช้เป็นสองหน้าที่คือ เป็นพอร์ทอินพุทเอาท์พุทใช้แบบบิตได้ และใช้เป็นขาสัญญาณควบคุมต่างๆ ซึ่งมีหน้าที่ดังนี้

- ขา P3.0 ใช้รับข้อมูลจากภายนอกแบบอนุกรม
- ขา P3.1 ใช้ส่งข้อมูลไปภายนอกแบบอนุกรม
- ขา P3.2 ใช้เป็นอินพุทเพื่อรับสัญญาณอินเตอร์รัพต์ชนิดที่ 0
- ขา P3.3 ใช้เป็นอินพุทเพื่อรับสัญญาณอินเตอร์รัพต์ชนิดที่ 1
- ขา P3.4 สัญญาณอินพุทให้เคาท์เตอร์ของไทเมอร์ 0
- ขา P3.5 สัญญาณอินพุทให้เคาท์เตอร์ของไทเมอร์ 1

- ขา P3.6 ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิพ

- ขา P3. ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิพ

PSEN (Program Store Enable) เป็นขาที่ส่งสัญญาณออกที่ขา 29 ขานี้จะแอคทีฟเมื่อ MCS-51 ต้องการอ่าน โปรแกรมจากหน่วยความจำภายนอก โดยปกติถ้าหน่วยความจำภายนอกเป็นอีพรอม ขา PSEN จะต่อกับขาสัญญาณเปิดด้านเอาท์พุท (Output Enable: OE) ของอีพรอม

ALE/PROG (Address Latch Enable) เนื่องจากพอร์ท 0 สามารถใช้เป็นขาอ้างตำแหน่ง และขาข้อมูล MCS-51 จะมีขา ALE ได้แก่ขา 30 ขานี้จะใช้มัลติเพล็ก (Multiplex) สัญญาณบัสตำแหน่งของ พอร์ท 0 ในการใช้งานระบบ MCS-51 นั้น จะต้องมีอุปกรณ์มาต่อกับพอร์ท 0 ที่ทำหน้าที่ คงค่า (Latch) สัญญาณบัสตำแหน่ง เมื่อ MCS-51 ต้องการติดต่อกับหน่วยความจำภายนอก MCS-51 จะส่งสัญญาณบัสตำแหน่งออกมาก่อนทาง พอร์ท 0 ไว้เพื่อใช้ พอร์ท 0 เป็นบัสข้อมูล

EA/Vpp (External Access) ขา EA ได้แก่ขาที่ 31 ถ้าขานี้เป็นลอจิก "1" จะใช้กับเบอร์ 8051/8052 เพื่อบอกว่าให้อ่านโปรแกรมจากหน่วยความจำภายใน แต่ถ้าเป็นลอจิก "0" จะให้ MCS-51 ทำโปรแกรมโดยอ่านจากหน่วยความจำโปรแกรมภายนอก (ถ้าขา EA เป็น "0" ขา PSEN จะแอคทีฟ) ถ้าหากเป็นเบอร์ 8031 หรือ 8032 ขา EA จะเป็น "0" เสมอ เพราะว่าไม่มีโปรแกรมหน่วยความจำภายใน แต่ถ้าใช้เบอร์ 8051/8052 ซึ่งมีหน่วยความจำภายในและให้ขา EA เป็น "0" ซึ่งจะหยุดการทำงานของรอม ภายในและอ่านโปรแกรมจากอีพรอมภายนอกแทน

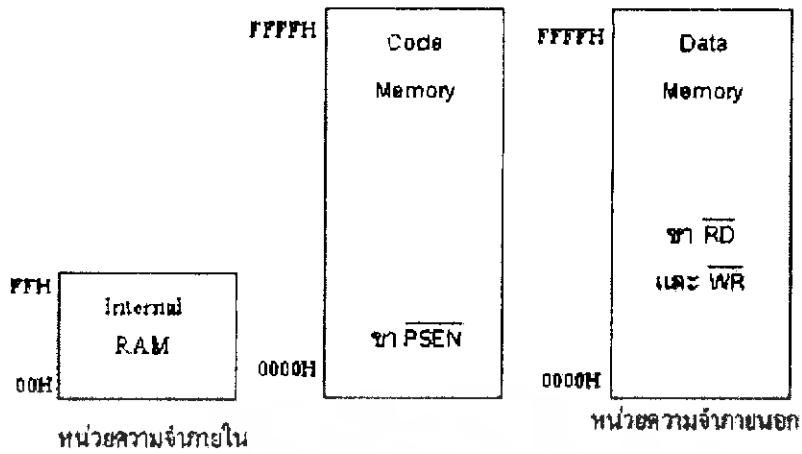
RST (Reset) ขา RST ได้แก่ขา 9 จะใช้ในการรีเซ็ต MCS-51 โดยจะให้ขานี้เป็นลอจิก "1" อย่างน้อย 2 คาบเวลา จึงจะรีเซ็ตระบบได้

XTAL1 ขาที่ 19 ใช้ต่อคริสตอลจากภายนอก โดยเป็นอินพุทเข้าสู่วงจรออสซิลเลตอร์

XTAL2 ขาที่ 19 ใช้ต่อคริสตอลจากภายนอก โดยเป็นเอาท์พุทต่อออกจากวงจรออสซิลเลตอร์

2.3.2 โครงสร้างหน่วยความจำ

หน่วยความจำสำหรับ MCS-51 จะมี 2 ชนิดคือ หน่วยความจำที่ใช้เก็บโปรแกรมรอม กับหน่วยความจำที่ใช้เก็บข้อมูลในการประมวลผล แรม MCS-51 บางเบอร์ 8051, 8052 จะมีหน่วยความจำในชิพ และMCS-51 ทุกเบอร์ยังสามารถอ้างหน่วยความจำโปรแกรมภายนอกได้มากที่สุด 64 กิโลไบต์ สำหรับหน่วยความจำแรมภายใน จะประกอบไปด้วยพื้นฐานที่ใช้งานทั่วไป, ชุดรีจิสเตอร์, พื้นที่ใช้งานระดับบิต และรีจิสเตอร์ฟังก์ชั่นพิเศษ เราอาจเขียนไดอะแกรมของหน่วยความจำของ 8031 ได้ดังรูปที่ 2.4 โดยในรูปจะบอกด้วยว่าขาใดจะแอคทีฟ



รูปที่ 2.3 การจัดหน่วยความจำของ MCS-51

2.3.3 Bit – addressable RAM

ใน MCS-51 จะมีหน่วยความจำที่สามารถอ้างข้อมูลในระดับบิตได้ตั้งแต่ตำแหน่ง 20H ถึง 2FH รวม 16 ไบต์ โดยสามารถ เซ็ต, เคลียร์, แอนท์, ออร์ ทางลอจิกได้ จำนวนบิตที่ใช้งานได้ทั้งหมดมีจำนวน 128 บิต (8 บิต X 16 ไบต์)

2.3.4 ชุดรีจิสเตอร์ (Register Banks)

หน่วยความจำข้อมูลภายในที่เป็นชุดรีจิสเตอร์ มีทั้งหมด 32 ตำแหน่ง โดยจะมี 4 ชุด แต่ละชุดจะมีรีจิสเตอร์ 8 ตัว คือ R0 ถึง R7 โดยชุดแรกจะอยู่ในตำแหน่ง 00H - 7H

2.3.5 รีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register)

ใน MCS-51 รีจิสเตอร์จะมีหน่วยความจำภายในชิพ โดยส่วนหนึ่งเป็นรีจิสเตอร์พิเศษ (Special Function Register: SFR) ซึ่งมีทั้งหมด 21 ตัว โดยรีจิสเตอร์พิเศษต่างๆจะเริ่มที่หน่วยความจำตั้งแต่ 80H ถึง FFH ซึ่งมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจิสเตอร์ฟังชั่นพิเศษ เพียง 21 ตำแหน่ง แต่ถ้าเป็น 8032/8051 จะใช้ 26 ตำแหน่งหรือมี SFR 26 ตัว

2.3.6 Program Staus Word

รีจิสเตอร์ตัวนี้เรียกย่อๆว่า PSW จะอยู่ที่ตำแหน่ง D0H ซึ่งสามารถเข้าถึงข้อมูลระดับบิตได้ โดยรีจิสเตอร์นี้จะเป็นตัวบอกสภาวะต่างๆของไมโครคอนโทรลเลอร์ ความหมายของแต่ละบิตแสดงได้ดังตารางที่ 2.5

บิต	ชื่อบิต	ตำแหน่ง	ความหมาย
PSW.7	CY	D7H	แฟลกซ์ตัวทด
PSW.6	AC	D6H	แฟลกซ์ตัวช่วยทด
PSW.5	F0	D5H	แฟลกซ์ 0
PSW.4	RS1	D4H	บิตสำหรับเลือกชุดรีจิสเตอร์ 1
PSW.3	RS0	D3H	บิตสำหรับเลือกชุดรีจิสเตอร์ 0
			00 = ชุด 0 ; ตำแหน่ง 00H – 07H 01 = ชุด 1 ; ตำแหน่ง 08H – 0FH 10 = ชุด 2 ; ตำแหน่ง 10H – 17H 11 = ชุด 3 ; ตำแหน่ง 18H – 1FH
PSW.2	OV	D2H	แฟลกซ์ค่าเกิน
PSW.1	-	D1H	รีเซิฟว์
PSW.0	P	D0H	อีเวนพารีตี้แฟลกซ์

ตารางที่ 2.2 แสดงบิตและหน้าที่ต่างๆใน PSW

1. แฟลกซ์ตัวทด (Carry Flag : CF) บิตนี้เป็นบิตที่ 7 ของ PSW บิตนี้จะมีความสำคัญหากมีการกระทำทางคณิตศาสตร์โดยบิตนี้จะเซ็ต เมื่อเกิดการทดของบิตที่ 7 ขณะที่ทำการบวกเลข หรือ เซ็ตเมื่อเกิดการยืมของบิตที่ 7 เมื่อเกิดการลบเลข
2. แฟลกซ์ตัวช่วยทด (Auxiliary Carry Flag) เมื่อมีการบวกแบบ Binary – Code – Decimal (BCD) บิต แฟลกซ์ตัวช่วยทด (AC) หรือบิตตัวช่วยทดจะถูกเซ็ต เมื่อมีการทดจากบิตที่ 3 ไปบิตที่ 4 หรือถ้าใน Lower Nibble มีค่าระหว่าง 0AE - 0FH เนื่องจากรหัส BCD นี้มีค่าได้มากที่สุดแค่ 9 ถ้าหากมีการบวกเลขแบบ BCD จะต้องตามด้วยคำสั่ง DAA (Decimal Adjust Accumulator) เพื่อปรับค่าที่มีค่าเกิน 9 โดยบวกเลข 6 เข้าไป จะทำให้เป็นรหัส BCD ที่แทนเลขฐานสิบได้
3. แฟลกซ์ศูนย์ (Flag 0) เป็นแฟลกซ์ที่สามารถใช้งานทั่วไปได้
4. บิตเลือกชุดรีจิสเตอร์ (Register Bank Select Bits) ตามที่ทราบมาแล้วว่าใน MCS-51จะมีชุดรีจิสเตอร์อยู่ 4 ชุด ถ้าจะเลือกให้ชุดใดแอคทีฟจะกำหนดได้ในบิต RS1 และ RS2 ของ PSW และจะเคลียร์ ตัวเองเมื่อระบบถูกรีเซ็ต ถ้าหากต้องการต่อกับชุดรีจิสเตอร์ 3 โดยย้ายข้อมูลจาก R7 (ตำแหน่ง 1FH) มาเก็บในแอคคิวมูลเลเตอร์
5. แฟลกซ์ค่าเกิน(Overflow Flag) แฟลกซ์ OV จะถูกเซ็ต หลังจากการกระทำทางคณิตศาสตร์แล้วเกิดค่าเกิน คือจำนวนที่เกิดจากการบวกหรือลบ มีค่าเกินกว่าจำนวนไบต์จะเป็นไปได้

คือ มากกว่า +128 หรือน้อยกว่า -127 ตัวอย่างเช่น ถ้าเกิดการบวกเลขสองจำนวนนี้จะเกิดการเซ็ตบิต OV ขึ้นใน PSW

6. บิตพาริตี้ (Parity Bit) พาริตี้บิต (P) เป็นบิตที่บอกค่าพาริตี้ของรีจิสเตอร์ แอคคิวมูลเลเตอร์ ซึ่งอาจเป็นตัวตรวจสอบความถูกต้องของข้อมูลได้ โดยจะเซ็ตหรือเคลียร์ขึ้นกับแอคคิวมูลเลเตอร์ เช่น ถ้าแอคคิวมูลเลเตอร์มีค่าเป็น 10101101B บิต P จะเป็น "1"

2.3.7 รีจิสเตอร์ B (B Register)

รีจิสเตอร์ B จะอยู่ตำแหน่ง F0H ของหน่วยความจำข้อมูลภายใน เป็นรีจิสเตอร์ที่สามารถใช้งานทั่วไปได้ โดนทั่วไปรีจิสเตอร์นี้จะใช้คูณ หรือหารกับรีจิสเตอร์แอคคิวมูลเลเตอร์ เช่น การทำคำสั่ง MUL AB ซึ่งเป็นการคูณแบบ 8 บิต โดยผลลัพธ์ที่ได้จะมีขนาด 16 บิต ซึ่งรีจิสเตอร์ A จะเก็บค่า 8 บิตต่ำ และรีจิสเตอร์ B จะเก็บค่า 8 บิตสูง สำหรับการหารโดยการทำคำสั่ง DIV AB โดยค่าใน A จะถูกหารด้วย B ผลลัพธ์ที่ได้จะเก็บในรีจิสเตอร์A และ B โดย B จะเก็บค่า 8 บิตต่ำ ส่วน A จะเก็บค่า 8 บิตสูง รีจิสเตอร์B นี้สามารถเข้าถึงข้อมูลระดับบิตได้ โดยตำแหน่งของบิตคือตำแหน่ง F0H ถึง F7H

2.3.8 ตัวชี้สแตก (Stack Pointer)

ตัวชี้สแตก (SP) เป็นรีจิสเตอร์ขนาด 8 บิตอยู่ที่ตำแหน่ง 81H การเขียนค่าเข้าไปในตำแหน่งที่ตัวชี้สแตคชี้อยู่นี้ เรียกว่า "pushing" สำหรับการอ่านค่าที่ SP ชี้อยู่ เรียกว่า "Popping" ค่า ของตัวชี้สแตกจะเพิ่มขึ้นหนึ่งก่อนที่จะเขียนข้อมูลลงไป และจะลดลงไปหนึ่งเมื่ออ่านข้อมูลออกมาแล้ว หากโปรแกรมทำคำสั่ง Call จะใช้รีจิสเตอร์สแตคนี้เก็บค่าตำแหน่งเดิมของโปรแกรม (PC) ก่อนที่จะกระทำโปรแกรมย่อย เมื่อทำโปรแกรมย่อยเสร็จแล้วจะคืนค่าในสแตคให้กับโปรแกรม ตามเดิม โดยปกติค่าโปรแกรมจะกำหนดให้อยู่ในแรมภายใน

2.3.9 รีจิสเตอร์ Data Pointer (DPTR)

รีจิสเตอร์นี้ใช้สำหรับชี้ตำแหน่งรหัสโปรแกรมหรือข้อมูลในหน่วยความจำโดยเป็นรีจิสเตอร์ขนาด 16 บิต ซึ่งประกอบด้วยรีจิสเตอร์ 2 ตัว คือ DPL ตำแหน่งที่ 82H โดยจะเก็บเป็น 8 บิตต่ำ และ DPH ตำแหน่งที่ 83H โดยจะเก็บค่า 8 บิตสูง รีจิสเตอร์ทั้งสองตัวนี้จะรวมกันกลายเป็นรีจิสเตอร์ขนาด 16 บิต ถ้าหากต้องหาเก็บค่า 55H ไปยังหน่วยความจำข้อมูลภายนอกตำแหน่งที่ 1000 H

2.3.10 รีจิสเตอร์พอร์ท (Port Register)

ใน MCS-51 ค่าของพอร์ทจะหมายถึงค่าของหน่วยความจำด้วย หากต้องการส่งข้อมูลจากพอร์ท ก็เพียงอ่านค่าจากตำแหน่งที่หน่วยความจำที่พอร์ทนั้นอยู่ใน MCS-51 พอร์ท 0 จะอยู่ที่ตำแหน่ง 80H, พอร์ท 1 จะอยู่ที่ตำแหน่ง 90H, พอร์ท 2 จะอยู่ที่ตำแหน่ง A0H และพอร์ท 3 จะอยู่ที่ตำแหน่ง B0H พอร์ท 0, 2 และ3 โดยทั่วไปแล้วจะไม่ใช้ถ้าหากมีการติดต่อกับหน่วยความจำ

ภายนอกหรือพอร์ทพิเศษ (เช่น อินเทอร์รัพท์, พอร์ทสื่อสารอนุกรม) โดยปกติและแล้วจะใช้ พอร์ท 1 ในการติดต่อกับอุปกรณ์ภายนอกพอร์ททุกพอร์ทสามารถอ้างถึงข้อมูลในระดับบิตได้

2.3.11 รีจิสเตอร์เวลา (Timer Register)

ใน MCS–51 เบอร์ 8051 จะมีรีจิสเตอร์ที่ใช้นับและจับเวลาขนาด 16 บิต 2 ตัว คือ Timer 0 อยู่ที่ตำแหน่ง 8AH และ 8CH โดยตำแหน่ง 8AH หมายถึง TL0 ซึ่งจะเป็น 8 บิตต่ำ และ 8CH หมายถึง 8 บิตสูง TH0 รีจิสเตอร์อีกตัวคือ ไทม์เมอร์ 1 โดยแบ่งเป็น TL1 อยู่ที่ตำแหน่ง 8BH เป็น 8 บิตต่ำ TH1 อยู่ที่ตำแหน่ง 8DH เป็น 8 บิตสูง การใช้งานไทม์เมอร์ จะต้องกำหนดการใช้งานใน รีจิสเตอร์ TMOD (Timer Mode Control Register) ซึ่งอยู่ที่ตำแหน่ง 88H เสียก่อน

2.3.12 รีจิสเตอร์พอร์ทอนุกรม (Serial Port Register)

MCS–51 จะมีพอร์ทสื่อสารอนุกรม (Serial Port) อยู่ภายในชิพ ซึ่งสามารถจะรับหรือส่งข้อมูลแบบอนุกรมให้เขียนข้อมูลไปที่รีจิสเตอร์ SBUF (Serial Data Buffer) ซึ่งอยู่ที่ตำแหน่ง 99H โดยถ้าต้องการส่งข้อมูลแบบอนุกรมให้เขียนข้อมูลไปที่รีจิสเตอร์นี้ตัวพอร์ทอนุกรมสามารถโปรแกรมให้ทำงานได้ 4 โหมด โดยโปรแกรมผ่านรีจิสเตอร์ SCON (Serial Port Control Register) ตำแหน่ง 99H

2.3.13 รีจิสเตอร์อินเทอร์รัพท์ (Interrupt Port Register)

MCS–51 สามรถอินเทอร์รัพท์ได้ 5 ตำแหน่ง โดยมี 2 – priority ตัวอินเทอร์รัพท์จะไม่ทำงานหลังจากระบบถูกรีเซ็ต และทำงานหลังจากที่เขียนข้อมูลไปที่รีจิสเตอร์ IE หรือตำแหน่ง A8H ลำดับความสำคัญสามารถเซ็ตที่รีจิสเตอร์ IP หรือตำแหน่ง B8H

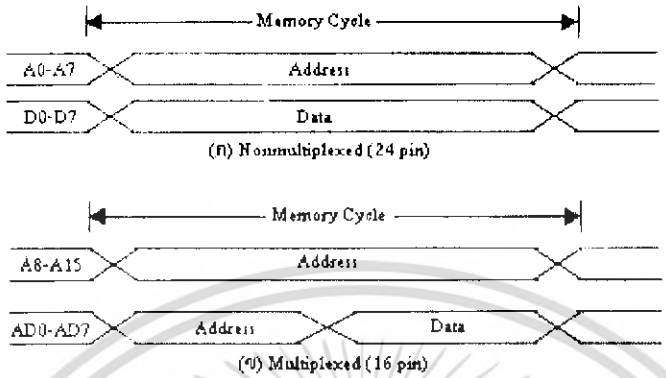
2.3.14 เพาเวอร์คอนโทรลรีจิสเตอร์ (PCON)

รีจิสเตอร์ PCON อยู่ที่ตำแหน่ง 87H ใช้หยุดการทำงานของ MCS–51 โดยจะหยุดจ่ายสัญญาณนาฬิกาให้ระบบ ทำให้ข้อมูลต่างๆ ภายใน MCS–51 ไม่มีการเปลี่ยนแปลง นอกจากนี้ลดพลังงานไฟฟ้าที่จ่ายให้ MCS–51 ลงด้วย

2.4 หน่วยความจำภายนอก (External Memory)

MCS–51 สามารถอ้างหน่วยความจำภายนอกได้ 64 กิโลไบต์ และอ้างหน่วยความจำโปรแกรมภายนอกได้ 64 กิโลไบต์ MCS–51 จะใช้พอร์ท 0 ในการอ้างตำแหน่งหน่วยความจำ 8 บิตล่าง และใช้พอร์ท 0 เป็นพอร์ทข้อมูล (Data) ด้วย โดยใช้ขา ALE มาคงค่าข้อมูลพอร์ท 0 และใช้พอร์ท 2 เป็นขาอ้างตำแหน่ง 8 บิตบน (รวมขาอ้างตำแหน่ง 16 เส้น ซึ่งอ้างได้ 64 กิโลไบต์)นอกจากพอร์ท 0 จะใช้งาน 2 หน้าที่ในการติดต่อกับหน่วยความจำจะใช้ วิธีมัลติเพล็กระหว่างตำแหน่งกับข้อมูล พิจารณาจากรูป ถ้าต้องการติดต่อกับหน่วยความจำที่เก็บข้อมูล 8 บิตและเก็บได้ 64 กิโลไบต์ จะต้องใช้สายสัญญาณ 24 เส้น คือเป็นขาตำแหน่ง 16 เส้น และขาข้อมูล 8 เส้น ดังรูป

แต่ถ้าใช้วิธีมัลติเพล็กคือใช้ขา A0-A7 เป็นขาข้อมูลด้วยคือ D0-D7 จะใช้สายสัญญาณเพียง 16 เส้น เท่านั้น จากรูปที่ 2.6 จะเห็นว่าเมื่อต้องการติดต่อกับหน่วยความจำจะส่งสัญญาณตำแหน่ง A0-A15 ออกมาก่อน 16 เส้น และเวลาต่อมาขา A0-A7 จะถูกเปลี่ยนเป็น D0-D7



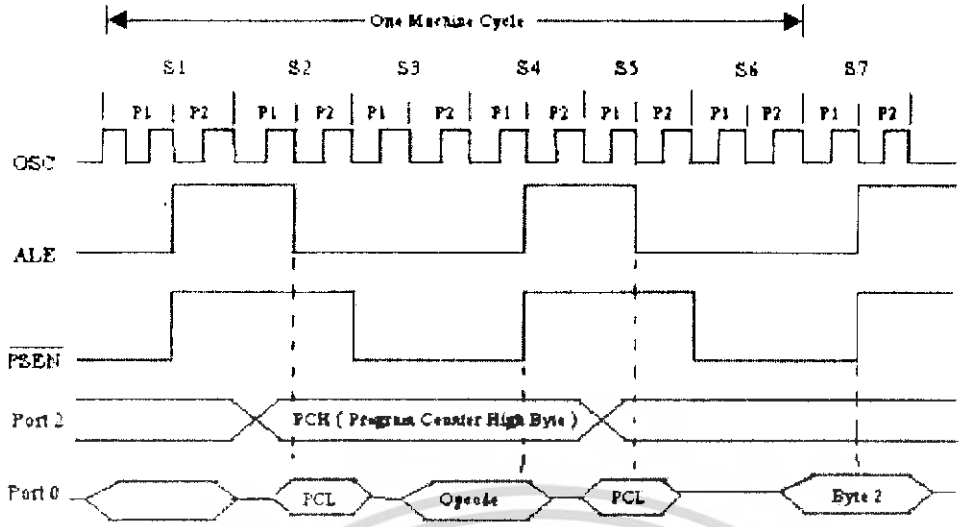
รูปที่ 2.4 ไดอะแกรมของกลุ่มสัญญาณที่ใช้อ่านข้อมูล

2.5 การติดต่อกับหน่วยความจำข้อมูลภายนอก

หน่วยความจำข้อมูลภายนอก MCS–51 สามารถอ่านและเขียนได้ การติดต่อกับหน่วยความจำข้อมูลภายนอก MCS-51 จะส่งขากำหนดตำแหน่งออกไปทางพอร์ท 0 และพอร์ท 2 จากนั้นจะส่งขา ALE เพื่อไปคงค่าตำแหน่ง 8 บิตต่ำ โดยการอ่านเขียนข้อมูลนั้นจะใช้ขา RD (P.7)และขา WR (P3.6) ตามลำดับ

2.6 การติดต่อกับหน่วยความจำโปรแกรมภายนอก

ในการอ่านข้อมูลจากหน่วยความจำโปแกรมภายนอก MCS–51 จะส่งค่าตำแหน่งของหน่วยความจำออกไปก่อน ซึ่งค่าตำแหน่งจะเก็บอยู่ใน PC โดยส่งออกไปทางพอร์ท 0 และพอร์ท 2 จากนั้นเวลาต่อมา ขจะส่งขา ALE ให้เป็นลอจิก “0” เพื่อคงค่าตำแหน่งของ 8 บิตต่ำ คือพอร์ท 0 จากนั้นจะส่งสัญญาณทางขา PSEN ให้เป็นลอจิก “0” เพื่ออ่านข้อมูลซึ่งจะได้ออปโค๊ด เข้าไปทางขาบัสข้อมูล คือพอร์ท 0 ไดอะแกรมเวลาการอ่านข้อมูลจากหน่วยความจำภายนอกแสดงได้ดังรูป 2.7



รูปที่ 2.5 ไดอะแกรมเวลาการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

2.7 กระบวนการรีเซ็ต (Reset Operation)

การรีเซ็ตหรือเริ่มต้นการทำงานใหม่ของ MCS-51 จะต้องใช้ลอจิก "1" ที่ขา RST เป็นเวลา 2 คาบเวลา (1 คาบเวลาเท่ากับ 12 สัญญาณนาฬิกา) จากนั้นให้กลับเป็นลอจิก "0"

บทที่ 3
การสื่อสารข้อมูล

การสื่อสารข้อมูล เป็นการส่งสารดิจิตอล (Digital Information) ซึ่งโดยมากอยู่ในเลขฐานสองจากแหล่งกำเนิดไปยังปลายทาง ข้อมูลจากแหล่งกำเนิดจะอยู่ในลักษณะสัญญาณอนาล็อกหรือสัญญาณดิจิตอลก็ตาม ข่าวสารจากแหล่งกำเนิด จะอยู่ในลักษณะสัญญาณดิจิตอล และข้อมูลที่รับได้ก็จะอยู่ในลักษณะดิจิตอลเช่นเดียวกัน ถึงแม้ว่าข้อมูลจะสามรถส่งได้ในลักษณะสัญญาณอนาล็อกหรือสัญญาณดิจิตอลก็ตาม ข่าวสารจากแหล่งกำเนิด อาจจะเป็นรหัสของตัวอักษร ตัวเลข หรือเครื่องหมายที่อยู่ในรูปแบบของเลขฐานสอง เช่น ASCII หรือรหัส EBCDIC ของไมโครโปรเซสเซอร์ (Microprocessor OP- Code), รหัสที่อยู่ของใช้ (User Address), โปรแกรมคอมพิวเตอร์, ข่าวสารข้อมูล(Data Base Information)

โครงข่ายการสื่อสารข้อมูล (Data Communication Network) แบบง่ายๆเช่น การเชื่อมโยงคอมพิวเตอร์ส่วนบุคคล หรือ คอมพิวเตอร์สองเครื่องในระยะทางใกล้ๆ ด้วยสายธรรมดาเข้าด้วยกันโดยตรง แต่ถ้า ระยะทางไกลๆอาจจะต้องใช้สายโทรศัพท์ หรือถ้าจะให้เป็นโครงข่ายการสื่อสารข้อมูลที่ยุ่งยาก ประกอบด้วยคอมพิวเตอร์เมนเฟรม (Mainframe Computer) 1 เครื่องหรือมากกว่ากับอุปกรณ์ปลายทางข้อมูลเป็นจำนวนร้อยๆเครื่อง

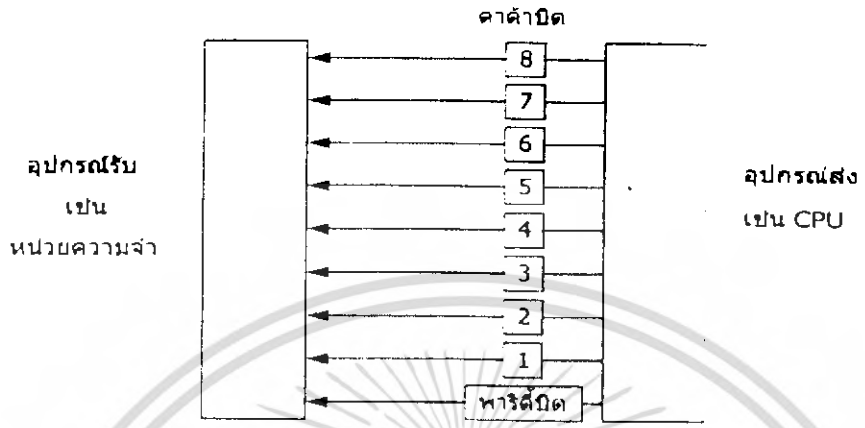
3.1 วิธีการถ่ายโอนข้อมูล มีอยู่ 2 ประเภท

1. การถ่ายโอนข้อมูลแบบขนาน ลักษณะการส่งข้อมูลแบบขนานทำได้โดยการส่งข้อมูลออกมาทีละ 1 ไบต์ จากอุปกรณ์ส่งไปยังอุปกรณ์รับตัวกลางระหว่าง 2 เครื่องจะต้องมีช่องทางให้ข้อมูลเดินอย่างน้อย 8 ช่องทาง โดยมากจะเป็นสายขนานให้กระแสไฟฟ้าวิ่งมากกว่าเป็นตัวกลางชนิดอื่น เนื่องจากมีการลดทอนของสัญญาณเนื่องจากความต้านทานของสาย ระยะทางระหว่าง 2 จุดจึงไม่ควรเกิน 100 ฟุต ปัญหาที่เกิดขึ้นหากระยะทางของสายมากกว่านี้ คือ ระดับของกราวด์ที่จุดรับผิดไปจากจุดส่งจะทำให้เกิดการผิดพลาดในการรับสัญญาณลอจิกทางฝ่ายรับ

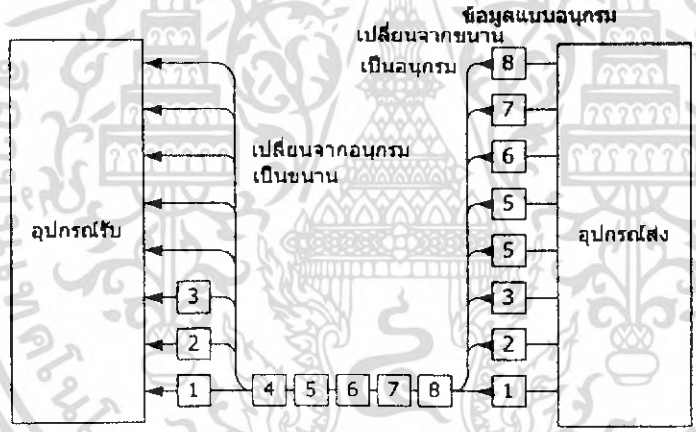
นอกจากสายที่เป็นทางเดินของข้อมูลแล้วอาจจะมีทางเดินของสัญญาณควบคุมอื่นๆอีก เป็นต้นว่า บิตที่บอก Parity ของสัญญาณเพื่อเป็นการตรวจสอบความผิดพลาดที่ปลายทาง หรือสายที่ควบคุมการโต้ตอบ (Hand Shank) จะเห็นว่าเป็นการส่งแบบขนานส่วนมากจะทำในลักษณะใกล้ๆ เนื่องจากมีช่องทางเดินของสัญญาณมากกว่า 8 สาย และอุปกรณ์ที่ต่อแบบขนานกับคอมพิวเตอร์จะเห็นได้แก่เครื่องพิมพ์เป็นต้น

2. การถ่ายโอนข้อมูลแบบอนุกรม ข้อมูลจะถูกส่งออกมาทีละบิตระหว่างจุดส่งและจุดรับ จะเห็นว่าการส่งข้อมูลแบบนี้จะช้ากว่าแบบขนานที่กล่าวมาแล้วอย่างแน่นอน แต่สาเหตุที่มีการนิยม

ใช้การส่งแบบนี้กันก็เพราะตัวกลางที่ใช้ในการสื่อสารแบบนี้ จะต้องการเพียงช่องทางเดียวหรือสายเพียงคู่เดียว ทำให้ค่าใช้จ่ายในเรื่องของสื่อกลางถูกกว่าแบบขนาน สำหรับการส่งระยะไกลๆ โดยเฉพาะเมื่อมีระบบสื่อสารทางโทรศัพท์ไว้ใช้งานอยู่แล้วย่อมจะเป็นการสะดวก ที่จะนำมาใช้เป็นการถ่ายโอนข้อมูลแบบอนุกรมได้



รูปที่ 3.1 การส่งข้อมูลแบบขนาน



รูปที่ 3.2 การส่งข้อมูลแบบอนุกรม

จากรูปที่ 3.2 แสดงให้เห็นว่าการส่งข้อมูลแบบอนุกรม ข้อมูลจากจุดส่งจะถูกเปลี่ยนให้เป็นอนุกรมเสียก่อนแล้วค่อยทยอยส่งออกไปทีละบิตไปยังจุดรับ ณ ที่จุดรับจะต้องมีกลไกในการส่งข้อมูลที่ส่งออกมาทีละบิต ให้เป็นสัญญาณแบบขนานซึ่งลงตัวพอดีนั่นคือ บิต 1 ลงที่ Data Bus เส้นที่ 1 พอดี การที่จะทำให้การแปลงสัญญาณจากอนุกรมทีละบิตให้ลงพอดีนั้น จำเป็นที่จะต้องมีกลไกที่เหมะสมเพื่อป้องกันการผิดพลาดในการรับ กลไกที่ว่าแบ่งออกเป็น 2 แบบ คือ

1) การสื่อสารแบบ Synchronous

2) การสื่อสารแบบ Asynchronous

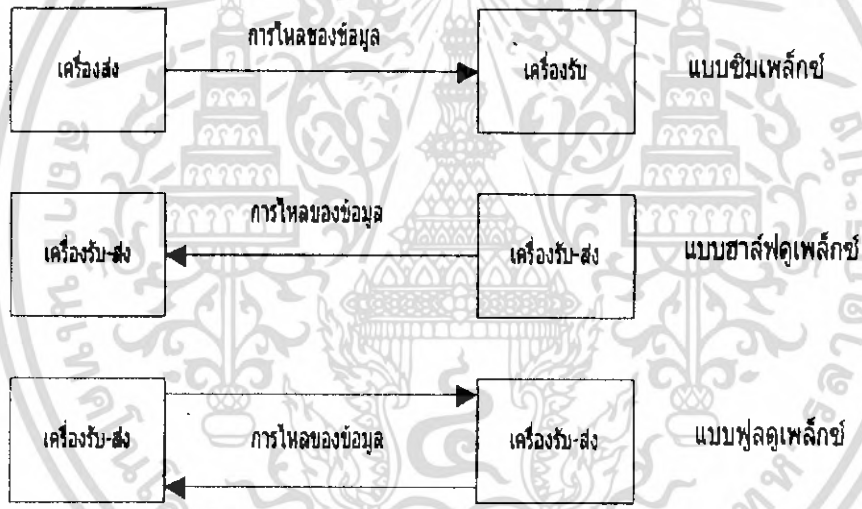
3.2 รูปแบบของการสื่อสารข้อมูลแบบอนุกรม

การติดต่อสื่อสารแบบอนุกรมอาจแบ่งตามรูปได้ 3 แบบ

1) แบบซิมเพล็ก (Simplex) ข้อมูลที่ถูกส่งได้ในทิศทางเดียวนั้นบางครั้งเรียกว่า การส่งทิศทางเดียว (Undirectional Data Bus) ในการสื่อสารแบบนี้อุปกรณ์ด้านหนึ่งจะส่งข้อมูลไปในช่องสัญญาณได้เท่านั้นและอุปกรณ์อีกด้านหนึ่งก็จะทำข้อมูลจากช่องสัญญาณเท่านั้น

2) แบบฮาล์ฟดูเพล็ก (Half duplex) หมายถึงการสื่อสารข้อมูลใน 2 ทิศทาง แต่ในช่วงเวลาหนึ่งจะได้เพียงทิศทางเดียวเท่านั้น อุปกรณ์สื่อสารทั้ง 2 ด้าน จะพลัดกันรับ - ส่ง การสื่อสารแบบนี้ส่วนใหญ่จะใช้ในระบบสาย 2 เส้น (2 Wire)

3) แบบฟูลดูเพล็ก (Full Duplex) หมายถึงการสื่อสารข้อมูลใน 2 ทิศทางพร้อมกัน การสื่อสารแบบนี้ใช้ได้ทั้งระบบสาย 2 เส้น (2 Wire) และสาย 4 เส้น (4 Wire) แต่ในระบบสาย 2 เส้น จะต้องอาศัยเทคนิคการแบ่งความถี่เข้าช่วย คือการส่งในความถี่ช่วงหนึ่งและจะรับในความถี่อีกช่วงหนึ่ง การสื่อสารข้อมูลแบบอนุกรมแสดงดังรูปที่ 3.3 แสดงรูปการสื่อสารข้อมูลแบบอนุกรม



รูปที่ 3.3 การสื่อสารแบบต่างๆ

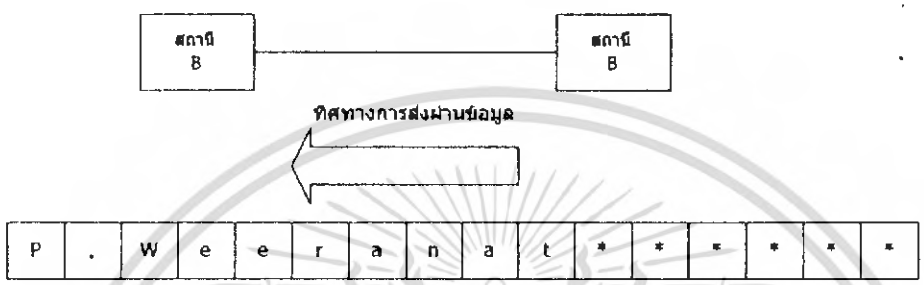
3.3 ความเร็วในการถ่ายโอนข้อมูลแบบอนุกรม

ความเร็วในการถ่ายโอนข้อมูลแบบอนุกรม หน่วยวัดข้อมูลเป็นบิตต่อวินาที (bps) และหน่วยที่บรรยายถึงความเปลี่ยนแปลงของสัญญาณใน 1 วินาที เรียกว่า บอดเรต (Baud Rate) หรืออัตราบอด ซึ่งแสดงการส่งข้อมูลอนุกรมมากกว่า 1 บิต เขียนในรูปสมการคณิตศาสตร์จะได้

อัตราบิต (Bit Rate) = อัตราบอด (Baud Rate) * บิตใน 1 บอด

3.4 การส่งแบบซิงโครนัส (Synchronous Transmission)

การส่งแบบซิงโครนัสข้อมูลจะถูกจัดเป็นกลุ่มๆ และทำการส่งข้อมูลทั้งกลุ่มไปทีเดียว เรียกกลุ่มของข้อมูลในกุ่มนี้ว่า "Block of Data" และในการส่งข้อมูลในแบบซิงโครนัสนี้ ช่วงเวลาของแต่ละบิตที่ทำการส่งใช้เวลาเดียวกัน และในการส่งผ่านทั้งตัวอักษร ตัวอักษรแรกและตัวถัดไปจะไม่มีอะไรมาคั่น (ภายในเบล็อกเดียวกัน) ดังนั้นช่วงเวลาระหว่างบิตสุดท้ายของตัวส่งผ่านจะต้องคิดในรูปแบบของระบบส่งผ่านข้อมูลที่อยู่ในรูปแบบของบิตที่แน่นอน



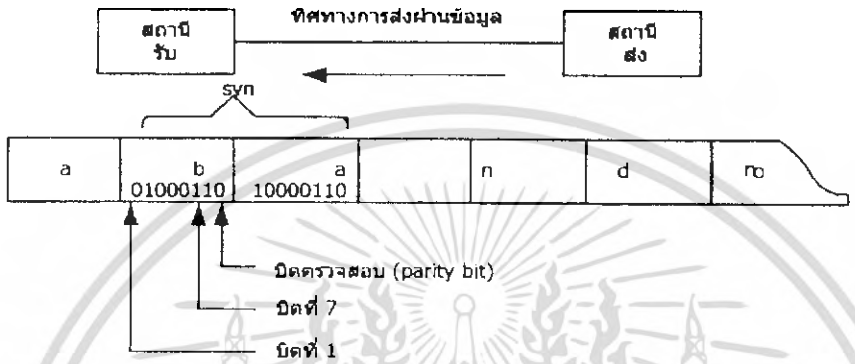
รูปที่ 3.4 การต่อเนื่องของข้อความที่ถูกส่งแบบซิงโครนัส

จากรูปที่ 3.4 แสดงให้เห็นการเอากลุ่มตัวอักษร (ข้อความที่ต้องการส่งผ่าน) มาเรียงต่อกันเพื่อเตรียมการส่งผ่านข้อมูลแบบซิงโครนัส จะเห็นได้ว่าตัวอักษรที่นำมาต่อกันเรียงชิดกัน โดยที่ไม่มีช่องว่างของตัวอักษรเลย หรือตามที่กล่าวมาแล้วว่าช่วงเวลาระหว่างบิตสุดท้ายของตัวอักษรตัวแรกกับบิตแรกของตัวอักษรตัวถัดไปเท่ากับศูนย์นั่นเอง ในเมื่อรูปแบบการส่งผ่านข้อมูลเป็นเช่นนี้ อุปกรณ์รับข้อมูลจะต้องทราบอะไรบ้าง จึงจะระบุลงไปว่าส่วนนั้นๆ เป็นกลุ่มบิตของตัวอักษรที่ถูกเข้ารหัสมา

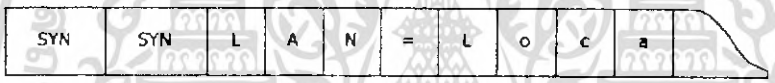
สิ่งที่อุปกรณ์ด้านรับจะต้องทราบก็คือบิตใดเป็นบิตแรกของตัวอักษรตัวแรกขนาดตัวอักษร (จำนวนบิตที่ใช้แทนในหนึ่งตัวอักษร) และอีกประการหนึ่งก็คือความเร็ว ในการส่งผ่านข้อมูลอุปกรณ์รับข้อมูลจึงจะจัดกลุ่มของบิตออกเป็นกลุ่มๆ เพื่อแทนค่ากลับเป็นตัวอักษรตัวต่างๆ ที่รับเข้ามา อย่างเช่นกรณีที่ข้อมูลถูกส่งผ่านมาอยู่ในรูปของรหัสแอสกี้ (ASCII) ตัวอักษรแต่ละตัวจะถูกเข้ารหัสในรูป 8 บิต แทนหนึ่งตัวอักษรโดยมีบิตแรกเป็นบิตตรวจสอบ ดังนั้น อุปกรณ์รับข้อมูลจะตัดบิตออกเป็นกลุ่มละ 8 บิต เพื่อนำมาตีความเป็นตัวอักษรแต่ละตัวนั่นเอง

สำหรับวิธีการที่ระบุไปได้ว่า บิตใดเป็นบิตแรกของตัวอักษรตัวแรกนั้นมีวิธีการดังนี้ ตามที่ได้กล่าวมาแล้วว่าข้อมูลที่ถูกส่งผ่านโดยวิธีการซิงโครนัสนั้นจะถูก จับมารวมกลุ่มกันเป็นกลุ่มของข้อมูล (Block of Data) และที่ส่วนต้นของบล็อกจะใส่ตัวอักขระซิง (SYN Character) ซึ่งเป็นอักขระพิเศษที่ใช้ในการควบคุมการส่งผ่านข้อมูลโดยที่อักขระซึ่งมีรูปแบบของบิตคือ 00010110

(มีบิตตรวจสอบแบบเลขคี่: Odd parity) และอุปกรณ์รับข้อมูลจะคอยตรวจสอบดูขบวนบิตที่รับเข้ามาว่ามีส่วนใดตรงกับอักขระ SYN บ้าง เมื่อพบแล้วอุปกรณ์รับข้อมูลจะทราบได้ทันทีว่าถึงจุดเริ่มต้นที่จะตัดกลุ่มของบิตกลุ่มละ 8 บิต เพื่อแทนตัวอักษรได้ และตัวอักษรหลายๆตัวที่ตีความได้คือ ข้อความที่ส่งมาในแต่ละบล็อก แต่การใช้อักขระ SYN เพียงตัวเดียวใส่ไว้ที่ส่วนต้นของบล็อกยังเป็นวิธีการที่ไม่ถูกต้อง เพราะในกรณีขบวนของบิตที่แทนตัวอักษรมีบางช่วงที่ไปตรงกับรูปแบบของบิตอักขระ SYN ได้



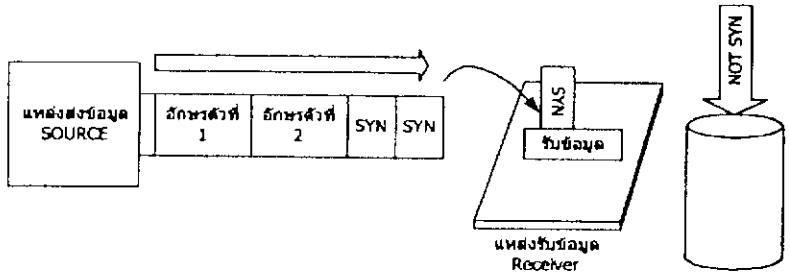
(a) กลุ่มที่เกิดขึ้นแล้วมีรูปแบบตรงกับอักขระ SYN



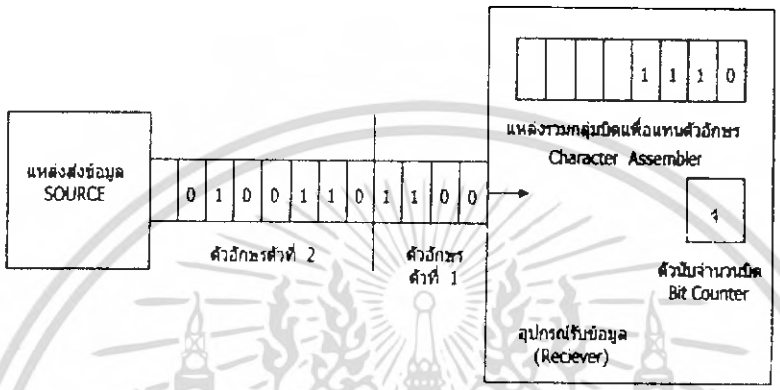
(b) การใช้อักขระ SYN 2 ตัวนำหน้าบล็อกของข้อมูล
ที่ส่งผ่านในระบบซิงโครนัส

รูปที่ 3.5 (a) และ (b) รูปแบบการใช้อักขระ SYN นำหน้ากลุ่มตัวอักษร

จากรูปที่ 3.5 a จะเห็นได้ว่าถ้าส่งข้อความที่มีตัวอักษร b และ a ติดกัน 4 บิตของตัวอักษร b ต่อกัน 4 บิตแรกของตัวอักษร a ตรงกับอักขระ SYN พอดีจะทำให้อุปกรณ์รับข้อมูลตีความผิดพลาดได้ ดังนั้นวิธีการการแก้ไขข้อผิดพลาดที่เกิดขึ้นได้จากกรณีเช่นนี้โดยการใช้อักขระ SYN 2 ตัว ใส่ไว้ที่ส่วนต้นของบล็อก ดังแสดงในรูปที่ 3.5 b และอุปกรณ์รับข้อมูลจะต้องทราบข้อตกลงนี้เป็นอย่างดี โดยที่ทันทีที่ตรวจพบอักขระ SYN จะดูอีก 8 บิตถัดไปว่าเป็นอักขระ SYN ด้วยหรือไม่ ถ้าใช่จะเริ่มต้นรับว่าทุกๆบิตที่ตามมาคือ ตัวอักษรแต่ละตัว กรณีที่ไม่ใช่ก็จะเริ่มตรวจหาอักขระ SYN ต่อไป หรือกล่าวได้ว่าเครื่องจะปรับตัวเข้าสู่ภาคการค้นหา SYN (Look for Syncmode) และเมื่อพบอักขระ SYN อย่างน้อย 2 ตัว ก็จะเริ่มเข้าสู่ขบวนการจัดกลุ่มบิตกลุ่มละ 8 บิต แทนตัวอักษรหรือกลุ่มข้อมูลที่ได้รับ



รูปที่ 3.6 การเปรียบเทียบอุปกรณ์รับข้อมูลตรวจหาอักขระในระบบซิงโครนัส



รูปที่ 3.7 การตัดแถวของบิตออกกลุ่มละ 8 บิต เพื่อแทนตัวอักษรของอุปกรณ์รับข้อมูลในระบบการส่งผ่านข้อมูลแบบซิงโครนัส

จากรูปที่ 3.6 และ 3.7 คือขบวนการรับข้อมูลที่ส่งผ่านมาในระบบซิงโครนัส ในบางระบบการใช้อักขระ SYN นำหน้ากลุ่มข้อมูลอาจใช้อักขระ SYN ถึง 3-4 ตัวก็ได้ เพื่อความแน่นอนในการส่งข้อมูลแบบซิงโครนัสที่สมบูรณ์แบบยิ่งขึ้น

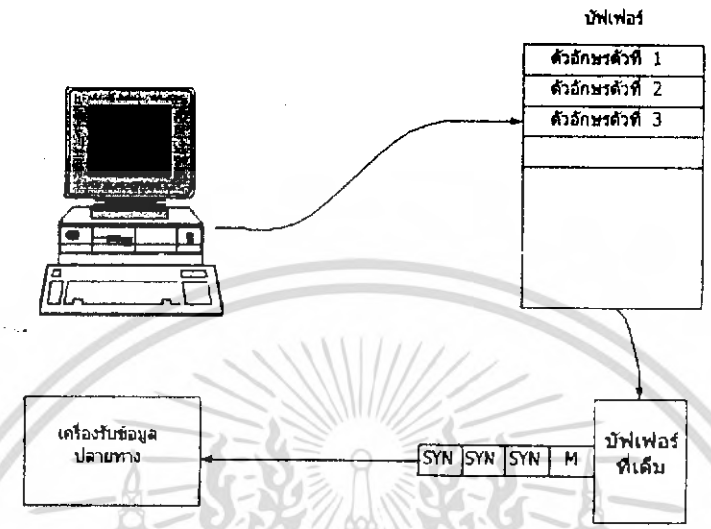
ก้ากล่าวถึงการส่งผ่านข้อมูลแบบซิงโครนัสจะถูกเรียกว่า แฟล็ก (Flag) ความหมายก็คือ มีสัญญาณหรือไม่มี เช่นเดียวกับการใช้สัญญาณการโบกธง แต่ในทุกกรณีจะเห็นได้ว่า หลักของการส่งสัญญาณแบบซิงโครนัสนั้น จะต้องใช้เครื่องรับข้อมูลที่ปลายทางเข้าใจถึงลักษณะของบิตพิเศษที่ส่งมาเพื่อให้รู้ว่านี่คือ จุดเริ่มต้นของกลุ่มตัวอักษร (Block) ที่กำลังส่งเรียงกันเข้ามา

บัฟเฟอร์เทอร์มินอล (Buffer Terminal) เมื่อเราทำการส่งตัวอักษรหลายๆตัวโดยระบบจัดกลุ่มเข้าเป็นบล็อกตามระบบของการซิงโครนัส ดังนั้น สภาวะของการส่งข้อมูลจึงเป็นเรื่องที่ต้องคำนึงถึงเป็นอย่างมาก เราทราบแล้วว่า ผู้ป้อนข้อมูลจะต้องพิมพ์ข้อมูลเข้าทางแป้นพิมพ์ จากนั้นข้อมูลจะถูกส่งออกไปตามสาย เพื่อไปยังสถานีปลายทางเหตุนี้จึงมีข้อควรพิจารณา 2 ข้อ คือ

1) ผู้พิมพ์ข้อมูลป้อนเข้าเครื่องไม่สามารถรักษาระดับความห่าง

2) ความเร็วในการส่งข้อมูลตามสาย จะมีความเร็วสูงกว่าที่ผู้ป้อนจะพิมพ์ข้อมูลมาก

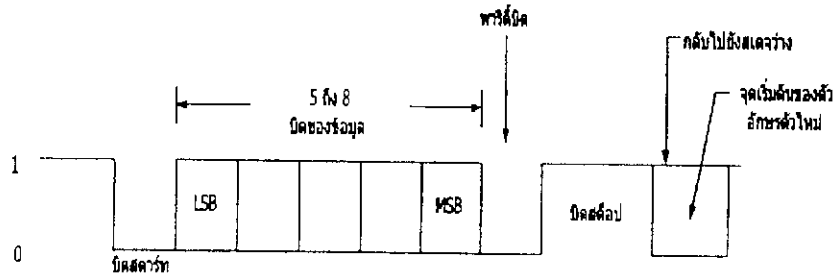
ด้วยเหตุผล 2 ข้อนี้จึงต้องสร้างบัฟเฟอร์เทอร์มินอลขึ้น ซึ่งหมายถึง เครื่องเทอร์มินอลจะต้องมีหน่วยความจำเพื่อเป็นที่รวบรวมอักษรขึ้น ดังรูปที่ 3.8 ซึ่งแสดงให้เห็นถึงเทอร์มินอลมีหน่วยความจำอยู่ด้วย หน่วยความจำประเภทนี้จะมีชื่อเรียกพิเศษว่า บัฟเฟอร์ (Buffer) เมื่อเป็นเช่นนี้ผู้ป้อนข้อมูลก็จะพิมพ์ตัวอักษรเข้าเครื่องเทอร์มินอลได้สบาย



รูปที่ 3.8 การใช้บัฟเฟอร์ช่วยในการส่งข้อมูลแบบซิงโครนัส

3.5 การส่งแบบอะซิงโครนัส

การส่งผ่านข้อมูลแบบซิงโครนัสนี้ พัฒนาจากการส่งโทรพิมพ์ในสมัยก่อน ลักษณะของสัญญาณแสดงในรูปที่ 3.9 เพื่อเพิ่มกลไกในการรับส่งอย่างถูกต้อง สัญญาณอะซิงโครนัสประกอบไปด้วยหรือบิตสตาร์ท (Start Bit) ขณะที่สถานะของการส่งเป็นแบบว่าง (Idel) คือยังไม่มีสัญญาณส่งออกมาจะมีสัญญาณหรือมีแรงดัน (หรือกระแส) ตลอดเวลา เพื่อความแน่ใจว่าฝ่ายรับยังติดต่ออยู่กับฝ่ายส่ง เมื่อเริ่มจะส่งสัญญาณของอะซิงโครนัสจะเป็น “0” หนึ่งช่วงสัญญาณนาฬิกาบิตนี้เรียกว่าสตาร์ทบิต ตามหลังของสัญญาณสตาร์ทบิตก็จะเป็นข้อมูลสำหรับ 1 ตัวอักษรซึ่งอาจจะมีขนาดตั้งแต่ 5 บิต ไปถึง 8 บิต โดยบิตที่ค่าน้อยที่สุด (LSB) จะถูกส่งออกมาก่อนใส่ไปจนถึงบิตที่มีค่ามากที่สุด (MSB)



รูปที่ 3.9 การเรียงบิตในแต่ละเฟรมของอะซิงโครนัส

สำหรับสัญญาณอะซิงโครนัส หลังจากบิตพาริตี้แล้วก็ต้องมีสต็อปบิตซึ่งในความกว้างของสต็อปบิตอาจจะเป็น 1, 1.5 หรือ 2 บิตแต่ผู้รับผู้ส่งตกลงกัน การเริ่มใช้พอร์ตอนุกรมจึงจำเป็นจะต้องตั้งค่าต่างๆ สำหรับการส่งแบบอนุกรม

1)ความเร็วในการส่ง

2)ความยาวรหัส 1 อักขระ

3)บิตตรวจสอบ

4)จำนวนสต็อปบิต

ในการส่งโทรพิมพ์หรือโทรเลข เมื่อก่อนนี้ใช้ความเร็วแค่ 70 บอด และ 110 บอด สำหรับคอมพิวเตอร์ความเร็วในการส่งมีให้เลือกตั้งแต่ 110, 200, 300, 2400, 4800, 9600 บอด และมากกว่านี้เนื่องจากมาจากไอซีหลายเบอร์ทำหน้าที่รับส่งแบบอะซิงโครนัสให้ใช้ การส่งอนุกรมจึงสบายสำหรับการออกแบบพอร์ตอนุกรม

3.6 การใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์

สำหรับไมโครคอนโทรลเลอร์แล้วการมีพอร์ตที่เป็น Digital I/O อย่างเดียวนั้นยังไม่พอ หรือยังไม่ตรงกับความต้องการของผู้ออกแบบระบบ Embedded System สักเท่าไรนักในไมโครคอนโทรลเลอร์ใหม่ๆนั้น มักจะมีพอร์ตอนุกรมเพิ่มเข้ามาอีก เพื่อทำให้การใช้งานในการติดต่อสื่อสารกันอุปกรณ์ภายนอกทำได้ง่ายขึ้น แต่การใช้งานก็ควรมีข้อควรระวังอยู่บ้างเช่นกัน

3.6.1 การสื่อสารแบบอนุกรมของไมโครคอนโทรลเลอร์

การสื่อสารแบบอนุกรมก็คือการที่อุปกรณ์หรือ Devices สองตัวทำการติดต่อกันโดยสัญญาณออกไปหรือรับเข้ามามีลักษณะของรอบสัญญาณเปลี่ยนแปลงสลับไปมาอย่างต่อเนื่องณ เวลาใดเวลาหนึ่ง โดยมีรูปแบบการส่งที่แน่นอนเช่นการส่งรหัสมอร์ส การสื่อสารผ่าน RS232, RS485 ฯลฯ หลายคนอาจคิดว่าเมื่อพูดถึงพอร์ตอนุกรมของไมโครคอนโทรลเลอร์มักจะเหมารวมว่ามันคือพอร์ต RS232 แต่จริงๆแล้วRS232 คือข้อกำหนดของสัญญาณที่มีค่า +/- 3V ถึง +/- 15V เพราะฉะนั้นถ้าต้องการให้ไมโครคอนโทรลเลอร์สามารถสื่อสารแบบ RS232 ได้จะต้องมีไอซี หรือ

วงจรที่ทำให้สัญญาณอนุกรมที่เป็น TTL เปลี่ยนเป็นสัญญาณมาตรฐาน RS232 ก่อน เช่น MAX232 , DS275 เป็นต้น แต่พอร์ตอนุกรมสามารถสื่อสารกับ ไมโครคอนโทรลเลอร์ได้โดยไม่ต้องมี IC หรือวงจรเข้ามาช่วย (ถ้าระยะทางไม่ไกลมาก)หรือกับอุปกรณ์อื่นๆที่มีการสื่อสารแบบเดียวกัน

3.6.2 ประโยชน์ของพอร์ตอนุกรม

พอร์ตอนุกรมนั้นนอกจากจะใช้สื่อสารกับอุปกรณ์ภายนอกด้วยกันแล้วยังใช้เป็นอุปกรณ์สำหรับการดีบักโปรแกรม เช่น ใช้ดูค่าตัวแปรของโปรแกรม ดูสถานะของพอร์ต ฯลฯ

3.6.3 การเพิ่มระยะทางของพอร์ตอนุกรม

ปกติแล้วสัญญาณอนุกรมที่ออกจากตัวไมโครคอนโทรลเลอร์จะ เป็นสัญญาณTTLที่มีขนาด 0-5V ถ้าการสื่อสารไม่ไกลกันเช่นอยู่บนบอร์ดเดียวกันก็ไม่ต้องใช้วงจรใดๆช่วย แต่ถ้าตัวบอร์ด หรือวงจรอยู่ไกลกันก็ใช้ ไอซีช่วยเช่น DS232 ซึ่งก็จะกลายเป็นการสื่อสารแบบ RS232 ทันที แต่ RS232 ก็มีข้อจำกัดด้านระยะทางเหมือนกันคือ ประมาณ 15-20 เมตร ที่บอร์ดเร็ต 9600 และบอร์ดเร็ตจะแปรผกผันกับระยะทางด้วยและยังขึ้นกับไอซี ไดร์เวอร์และชนิดของสายด้วยแต่ถ้าใช้สายยาวๆแล้วไม่แนะนำให้ใช้บอร์ดเร็ตที่มีค่าสูง ถึงแม้จะผ่านการทดลองแล้วว่าสามารถที่จะใช้งานได้จริงสถานที่ที่ทดลองอาจมีสภาพแวดล้อมที่ต่างกันกับที่ที่ใช้งานจริง

3.6.4 ลักษณะสัญญาณของพอร์ตอนุกรม

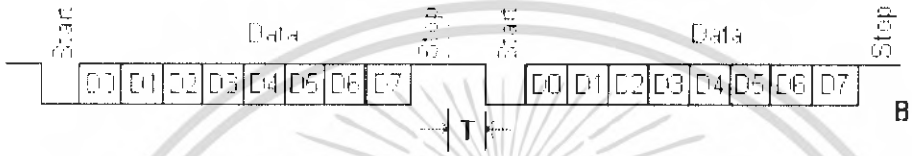
สัญญาณของพอร์ตอนุกรมที่เป็นที่ใช้งานโดยทั่วไปคือ 1 บิต START,ข้อมูล 8 บิต และ STOP 1 บิตไม่มี PARITY บิต รวมทั้งหมดก็ 10 บิต จะมีสัญญาณตามรูปข้างล่าง

รูปที่ 3.10 แสดงรูปสัญญาณการส่งข้อมูลแบบ Serial Port

โดยบิตแรกจะเป็น Start Bit ซึ่งจะ Active Low ถัดไปอีก 8 บิตจะเป็นข้อมูลศูนย์หรือหนึ่งและบิตสุดท้ายจะเป็น Stop Bit ซึ่งจะ Active High สำหรับความกว้างของแต่ละบิตก็คำนวณได้โดยอินเวอร์สค่าบอร์ตเร็ตหรือ 1/ บอร์ดเร็ต เช่นบอร์ดเร็ต 9600 จะมีความกว้างของสัญญาณ 1 บิตเท่ากับ 104 uS เพราะฉะนั้น 10 บิตก็เท่ากับ 1.04 mS

3.6.4 ปัญหาด้านรับไม่สามารถรับข้อมูลไม่ได้

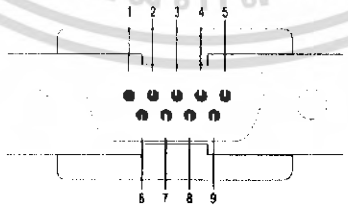
จากค่าเวลาอันนี้ทำให้เรารู้ว่าถ้าเราส่งข้อมูลติดๆกัน โดยไม่คำนึงถึงเวลาของแต่ละบิตจะทำให้ด้านรับซึ่งอาจเป็นไมโครคอนโทรลเลอร์ หรือ PC แยกข้อมูลไม่ออกเช่นหลังจากส่งบิต Stop ออกไป แล้วทำการส่งข้อมูลชุดใหม่ออกไปทันทีโดยที่เวลาของ Stop บิตยังไม่ครบทำให้ด้านรับเกิดอาการงงหรือรับข้อมูลได้มั่วๆ ดูรูปข้างล่างประกอบ



รูปที่ 3.11 A ข้อมูลที่รับ-ส่งผิดพลาด รูปที่ 3.11 B ข้อมูลที่รับ-ส่งถูกต้อง

รูปที่ 3.11 ข้อมูลที่รับ - ส่ง แบบถูกต้องและแบบผิดพลาด

จากรูปที่ 3.11 ถ้าการส่งข้อมูลเป็นแบบรูป A จะทำให้ด้านรับเกิดความผิดพลาด ได้เนื่องจากบิต Stop ของข้อมูลชุดแรกยังไม่ครบเวลาแล้วส่ง Start บิต ของข้อมูลชุดถัดไปทับ ซึ่งเหตุการณ์แบบนี้โดยไม่ตั้งใจมากกว่าโดยเฉพาะไมโครคอนโทรลเลอร์ ที่ทำงานด้วยความเร็วสูงๆ ยิ่งใช้ภาษาระดับต่ำๆ ด้วยแล้วมีโอกาสเกิดขึ้นได้มากเพราะส่วนใหญ่ใช้วิธีเช็ค Empty Tx bit หลังจาก Empty Tx bit ได้ Active แล้วเราก็เอาข้อมูลชุดใหม่ส่งทันทีโดยไม่หน่วงเวลาหลังจาก Empty Tx bit ได้ Active แล้วเป็นเวลามากกว่าหรือเท่ากับเวลาของแต่ละบิตที่ได้กล่าวไปแล้ว ส่วนในการใช้ภาษาระดับสูงอันนี้ไม่ค่อยเกิดให้เห็นเพราะว่าภาษาระดับสูงนั้น กว่าจะส่งข้อมูล แต่ละไบต์ หรือแต่ละชุดได้จะต้องมีการทำงานหลายอย่างเช่นกรณีของภาษาซี ในคำสั่ง Printf กว่าจะส่งข้อมูลออกได้คอมไพล์เลอร์จะต้องแปลง,ตรวจสอบชนิดของตัวแปร,จับใส่ลง Buffer ซึ่งตรงนี้ใช้เวลามากพอสมควร แต่ก็ขึ้นกับเวอร์ชั่นหรือยี่ห้อของคอมไพล์เลอร์ ด้วยบางตัวก็เก่งสามารถทำได้เร็ว



รูปที่ 3.12 แสดงขาพอร์ตอนุกรมแบบ 9 PIN

Signal Name	DB-9PIN
FG(Frame Ground)	-
TD(Transmit Data)	3
RD(Receive Data)	2
RTS(Request To Send)	7
CTS(Clear To Send)	8
SG(Signal Ground)	5
DSR(Data Set Ready)	6
CD(Carrier Detect)	1
RI(Ring Indicator)	9
DTR(Data Terminal Ready)	4

ตารางที่ 3.1 แสดง หมายเลขประจำขาของพอร์ตอนุกรมแบบ 9 PIN

FG (Frame Ground) = สัญญาณจากโมเด็มให้ PC เตรียมพร้อม

TD (Transmit Data) = ส่งข้อมูลทีละบิตโดยเป็นลำดับไปที่โมเด็ม

RD (Receive Data) = รับข้อมูลทีละบิตโดยเป็นลำดับมาจากโมเด็ม

RST (Request to Send) = สัญญาณจากโมเด็มให้ PC พร้อมที่จะส่งข้อมูล

CTS (Clear to Send) = ตรวจสอบสัญญาณโมเด็มว่าพร้อมจะรับข้อมูลจาก PC

SG (Signal Ground) = ขากราวน์

DSR (Data Set Ready) = สัญญาณบอกว่าโมเด็มพร้อมทำงานแล้ว

CD (Carrier Detect) = เมื่อใดที่ตรวจสัญญาณเจอที่ปลายทางของสัญญาณจะทำให้สัญญาณ Active

RI (Ring Indicator) = ตรวจจับสัญญาณของโมเด็ม

*** ขาที่ 3 , 4 และ 7 เป็น Output, ขา 5 เป็นกราวน์ ส่วนขาที่เหลือเป็น Input ***

3.6.5 การ Test ว่าพอร์ตอนุกรมของ PC เสียหรือไม่

การ Test อันนี้ทำได้ง่ายมาก ทำได้โดยช็อตขา 2 กับขา 3 เข้าด้วยกันแล้วเปิดโปรแกรม Hyper terminal จากนี้ลองพิมพ์อะไรลงไปก็ได้ตัวที่พิมพ์จะถูกส่งไปที่ขา Td แล้วรับเข้ามาที่ขา Rd ปรากฏให้เราเห็นที่หน้าจอนั้นเอง แต่กรณีที่ต่อสายออกจาก DB-9 แล้วก็ช็อตที่ปลายสายก็ได้ ถ้าช็อตปลายสายแล้วตัวอักษรที่พิมพ์ไม่ปรากฏก็แสดงว่าต่อสายผิดต้อง ถ้าพิมพ์แล้วมีข้อความออกมาก็ไม่ได้หมายความว่าต่อสายถูก อาจจะสลับสาย Rd กับTd ก็ได้

* Rd กับ Td จะเป็นชื่อที่ใช้เรียกขารับและส่งข้อมูลของด้าน PC ส่วน Rx และ Tx หมายถึงขารับและส่งด้านของอุปกรณ์ภายนอกที่นำมาต่อกับ PC ซึ่ง x อาจจะหมายถึงอะไรก็ได้ ในที่นี้หมายถึงไมโครคอนโทรลเลอร์

3.6.6 การใช้งานไมโครคอนโทรลเลอร์ที่มีฮาร์ดแวร์ของพอร์ตอนุกรมในตัว (Hardware Serial Port)

ในการใช้งานชุดโมดุลของ Hardware Serial Port นั้นตัวไมโครคอนโทรลเลอร์จะต้องมีส่วนที่เรียกว่า UART หรือ USART ซึ่งก็แล้วแต่จะเรียกกัน

3.7 การเขียนโปรแกรมติดต่อและควบคุมพอร์ตอนุกรม (Serial Port)

3.7.1 รู้จักกับมาตรฐาน RS-232C

มาตรฐาน RS-232C เป็นมาตรฐานที่ได้รับการออกแบบมาเพื่อที่จะทำให้อุปกรณ์ต่อพ่วงจากผู้ผลิตที่ต่างกันสามารถทำงานร่วมกันได้ มาตรฐานหลายชนิดได้รับการออกแบบขึ้นมา แต่มาตรฐานที่ได้รับความนิยมและใช้กันกว้างขวางมากที่สุดคือ มาตรฐาน RS-232C ซึ่งถูกประกาศใช้ในปี 1969 โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association: EIA) ในยุคแรกๆการอินเตอร์เฟสแบบ RS-232C ถูกออกแบบสำหรับเชื่อมต่อเทอร์มินอล (DTE: Data Terminal Equipment) กับโมเด็ม (DCE: Data Communication Equipment) ทั้งนี้ก็เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลบนสายเส้นเดียวกัน

มาตรฐาน RS-232C ได้แบ่งอุปกรณ์ออกเป็น 2 ประเภท ซึ่งอุปกรณ์ทั้งสองประเภทนี้คือ

1. อุปกรณ์ DTE (Data Terminal Equipment) เป็นอุปกรณ์สำหรับส่งข้อมูล (Output)

2. อุปกรณ์ DCE (Data Communication Equipment) เป็นอุปกรณ์สำหรับรับข้อมูล (Input)

ตามมาตรฐาน RS-232C แล้วคอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งคอนเน็กเตอร์ที่นิยมใช้กันอยู่จะเป็นชนิด D-Type แบบ 9 ขา และแบบ 25 ขา โดยจะติดตั้งอยู่หลังเครื่องคอมพิวเตอร์ ระดับแรงดันจะมีค่าระหว่าง -3 V ถึง -15 V สำหรับลอจิก High และลอจิก Low จะมีแรงดันระหว่าง −3 V ถึง +15 V สามารถรับส่งข้อมูลได้ที่ความยาวของสัญญาณสูงสุด 50 ฟุต หรือ 15 เมตร แต่ถ้าเราต้องการสื่อสารกับอุปกรณ์อื่นที่อยู่ห่างกันมากๆ เราจำเป็นต้องใช้อุปกรณ์อื่นๆเข้าช่วย เช่น การใช้โมเด็ม เป็นต้น

3.7.2 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรมที่นิยมใช้กับคอมพิวเตอร์นั้น เป็นการสื่อสารข้อมูลแบบอะซิงโครนัสนั่นคือ ต้องใช้สายสัญญาณเส้นเดียวทำหน้าที่ทั้งส่งส่วนที่เป็นข้อมูล และส่วนที่ใช้เป็นส่วนการควบคุมการส่งข้อมูล ดังนั้นข้อมูลที่อ่านได้แต่ละบิตจากการส่งแบบอนุกรม จึงต้องถูกแยกแยะว่าใช้สำหรับวัตถุประสงค์ใด โดยเราสามารถแบ่งได้เป็น 4 ส่วนคือ

1. Start Bit	ขนาด 1 บิต
2. บิตข้อมูล (Data Character)	ขนาด 7 หรือ 8 บิต
3. Parity Bit	ขนาด 1 บิต
4. Stop Bit	ขนาด 1 หรือ 2 บิต

แต่ละตัวอักษรที่ถูกส่งออกไปเป็นกลุ่มจะประกอบไปด้วยบิตเริ่มต้น บิตข้อมูล บิตพาริตี้ (จะมีหรือไม่มีก็ได้) และบิตจบ โดยเราพอจะสรุปหน้าที่ของแต่ละส่วนได้ดังนี้

- **Start Bit** บิตเริ่มต้น จะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง

- **Data Character** บิตข้อมูล การส่งบิตข้อมูลจะส่งเป็นกลุ่มๆ โดยทั่วไปจะส่งเป็น 7 หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง ASCII Word

- **Parity Bit** บิตพาริตี้ ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง เราจะใส่บิตพาริตี้เข้าไป แต่ทั้งตัวรับและตัวส่งจะต้องรู้กันว่าใช้พาริตี้แบบใดในการส่งข้อมูล

- **Stop Bit** บิตจบ เป็นบิตที่ส่งมาปิดท้ายข้อมูล

3.7.3 อัตราเร็วในการส่งข้อมูลแบบอนุกรม

การที่อุปกรณ์ 2 อย่างจะติดต่อสื่อสารกันได้นั้น จะต้องกำหนดอัตราเร็วให้เท่ากัน ซึ่งอัตราเร็วในการสื่อสารแบบอะซิงโครนัสคือ ค่าบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที ซึ่งค่าอัตราเร็วในการสื่อสารแบบอนุกรมสำหรับมาตรฐาน RS-232C นั้นมีใช้ดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

3.8 การสื่อสารไร้สายโดยการมอดูเลต

ในขบวนการมอดูเลต เราใช้คลื่นรูปซายน์ที่มีความถี่สูงเป็นพาหะ แล้วเปลี่ยนแปลงคุณสมบัติบางอย่างของพาหะด้วยสัญญาณข่าวสาร โดยทั่วไปสัญญาณข่าวสารได้แก่ สัญญาณออดิโอ(หรือเสียงพูด) สัญญาณภาพ หรือข่าวสารอื่นๆ การเปลี่ยนแปลงคุณสมบัติของคลื่นพาหะนี้เรียกว่า การมอดูเลต

การมอดูเลตให้กับคลื่นพาหะแบ่งออกได้เป็น 3 แบบ

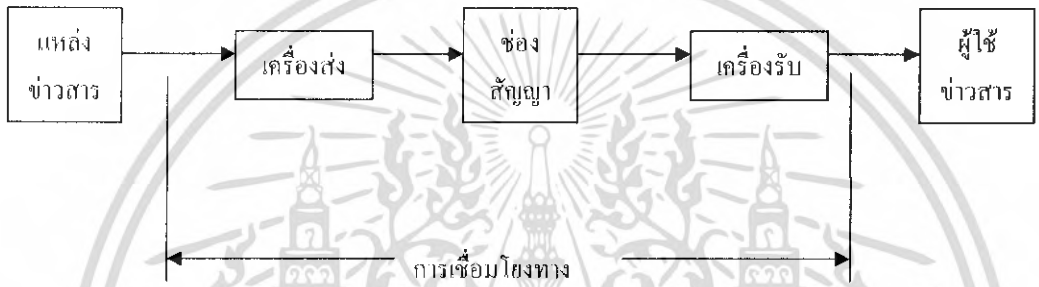
1. มอดูเลตทางแอมพลิจูด (amplitude modulation หรือ AM)
2. มอดูเลตทางความถี่ (frequency modulation หรือ FM)
3. มอดูเลตทางเฟส (phase modulation หรือ PM)

ในทางปฏิบัติสัญญาณ FM กับสัญญาณ PM จะคล้ายคลึงกันมาก บางทีเราเรียกรวมๆทั้ง FM และ PM ว่า การมอดูเลตทางมุม (angle modulation) กล่าวโดยสรุป การมอดูเลตแบ่งออกเป็น 2 แบบใหญ่ๆ คือ AM และ FM (หรือ PM)

3.8.1 หลักการสื่อสารไร้สายโดยการมอดูเลต

สามารถอธิบายการทำงานได้ตามรูปที่ 3.13 ดังนี้ โดยข่าวสารจะถูกแปลงให้อยู่ในรูปของสัญญาณไฟฟ้าที่เรียกว่า สัญญาณมอดูเลตติ้ง (molulating signal) สัญญาณนี้จะแปลงให้อยู่ในรูปของรหัสตามวิธีการสื่อสารแบบดิจิตอลหรือส่งตรงเข้าเครื่องส่งตามวิธีการสื่อสารแบบอนาล็อกก็

ได้ ในเครื่องส่งจะมีตัวมอดูเลตที่ทำหน้าที่รวมสัญญาณมอดูเลตติ้งกับตัวพาห์ (carrier) ตัวพาห์นี้มีกำลังส่งสูงพอที่จะพาสัญญาณมอดูเลตติ้งไปที่ไกลๆได้ด้วยความถี่สูงตามกระบวนการมอดูเลชั่น (modulation) จานั้นสัญญาณจะถูกคัปปลิ้งออกอากาศหรือส่งตามสายส่งก็ได้สัญญาณที่ผ่านช่องสัญญาณ (channel) ซึ่งไม่ว่าจะสายส่งหรืออากาศก็ตามจะถูกรบกวนจากสัญญาณรบกวนภายนอก (noise) เมื่อสัญญาณไปถึงเครื่องรับของผู้ใช้ปลายทาง เสาอากาศของเครื่องรับจะแปลงสัญญาณที่เป็นคลื่นแม่เหล็กไฟฟ้า แต่ถ้าส่งตามสายเครื่องรับจะรับสัญญาณในรูปของกระแสไฟฟ้าหรือแรงดันไฟฟ้าได้ทันที จากนั้นตัวดีมอดูเลเตอร์ (demodulator) ในเครื่องรับจะแปลงสัญญาณที่มีความถี่สูงให้มีความถี่ต่ำลงและแยกสัญญาณมอดูเลตติ้งออกจากตัวพาห์ตามกระบวนการดีมอดูเลชั่น (demodulation) และถูกถอดรหัส (decode) กลับเป็นสัญญาณอนาล็อกเหมือนเดิม ตามวิธีการสื่อสารแบบดิจิตอล



รูปที่ 3.13 แผนภาพของระบบสื่อสาร

3.8.2 ทำไมต้องมีการมอดูเลชั่น

1. ทำให้สัญญาณมีกำลังสูงสามารถเดินไปที่ไกลๆได้

2 .ทำให้สัญญาณมีความถี่สูงขึ้น ซึ่งเหมาะสำหรับการรับส่งสัญญาณมากขึ้นเพราะใช้เสาอากาศที่สั้นลงได้

3. สามารถแบ่งความถี่ให้หลายๆสัญญาณส่งพร้อมกันภายใต้ตัวพาห์ตัวเดียวกันได้ เรียกว่าการมัลติเพล็กซิง

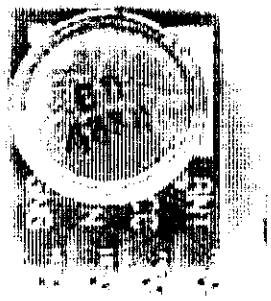
4. ทำให้สัญญาณมีความต้านทานการรบกวนของสัญญาณรบกวนได้ดีขึ้น

3.9 RF Transmission

ในการติดต่อกันระหว่างคอมพิวเตอร์กับหุ่นยนต์นั้นจะใช้ RF เบอร์ TLP315 และRLP315 ซึ่งเป็นโมดุลในการรับ-ส่ง RF โดย TLP315 เป็นโมดุลตัวส่งที่มี 4 ขา ประกอบไปด้วย

1. GND
2. Data In
3. Vcc
4. Antenna (RF Output)

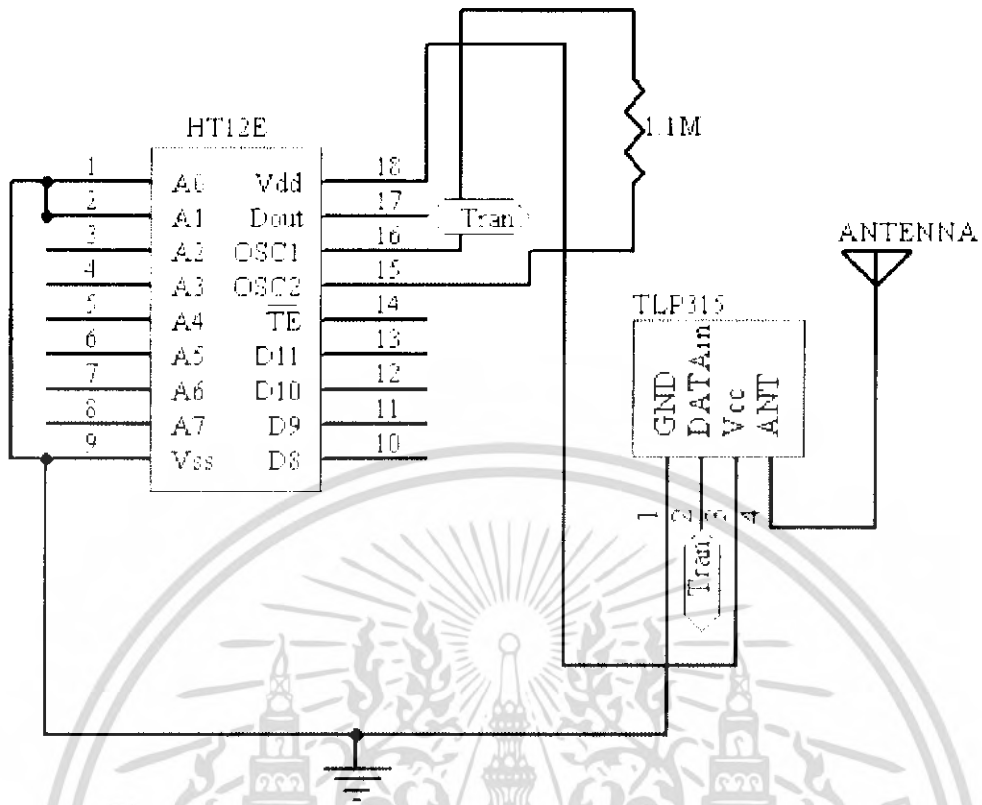
ดังรูปที่ 3.14 แสดงขา 1 , 2 , 3 และขา 4 เรียงจากซ้ายไปขวาตามลำดับ



รูปที่ 3.14 แสดง TLP433ภาคส่ง RF

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating Supply Voltage	-	2.0	-	12.0	V
Icc 1	Peak Current (2V)	-	-	-	1.64	mA
Icc2	Peak Current (2V)	-	-	-	19.4	mA
Vh	Input High Voltage	Idata = 100μA(High)	Vcc-0.5	Vcc	Vcc+0.5	V
Vl	Input Low Voltage	Idata = 0μA(Low)	-	-	0.3	V
FO	Absolute Frequency	433 MHz Module	432.8	433	433.29	MHz
PO	RF Output Power = 50 Ω	Vcc = 9-12V	-	16	-	dBm
DR	Data Rate	External Encoding	512	4.8K	200K	bps

ตารางที่ 3.2 แสดงข้อมูลของ TLP433



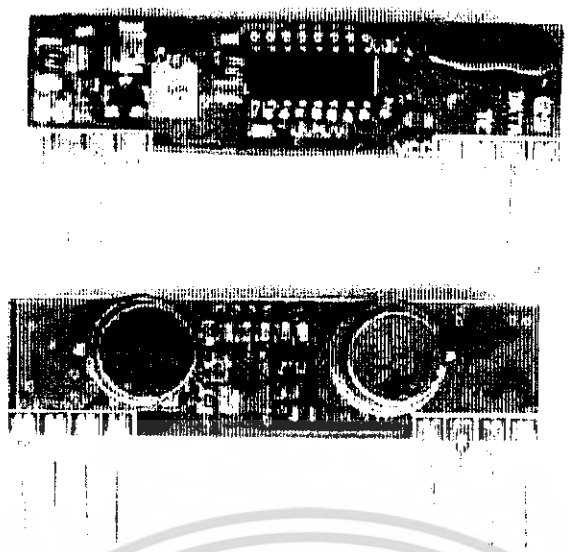
รูปที่ 3.15 แสดงการประยุกต์ใช้งาน TLP315

เมื่อทำการ Set ค่า A_0 และ A_1 แล้วป้อนข้อมูลเข้าทาง D_8–D_{11} และทำการ Set ขา TE ของ HT12E ให้เป็น 0 แล้วข้อมูลจะถูก Encoder มาที่ขา D_{OUT} ของ HT12E และเมื่อเรานำขานี้ไปต่อกับขา Data In ของ TPL315 ก็จะกลายเป็นการส่งข้อมูลออกไปทาง Antenna ของ TLP315

และ RLP315 ซึ่งเป็นโมดูลที่มี 8ขา ประกอบด้วย

1. GND
2. Digital Data Output
3. Linear Output/Test
4. Vcc
5. Vcc
6. GND
7. GND
8. Antenna

ดังรูปที่ 3.16 เรียงตามขาจากซ้ายไปขวา 1, 2, 3, 4, 5, 6, 7 และ 8 ตามลำดับ

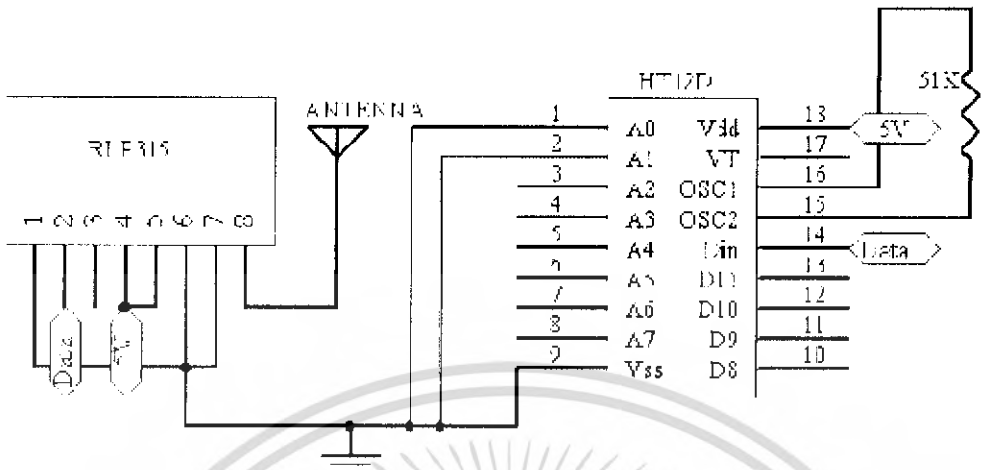


รูปที่ 3.16 แสดง RLP433 ภาครับ RF

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating Data Voltage	-	3.3	5.0	6.0	V
Itot	Operating Current	-	-	4.5	-	mA
Vdata	Data Out	Idata = +200 µA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 µA (Low)	-	-	0.3	V
Electrical Characteristics						
Characteristics		**SYM**	**Min**	**Typ**	**Max**	**Unit**
Operation Radio Frequency		FC	315,418 and 433.29			MHz
Sensitivity		Perf	-	-110	-	dBm
Channel Width		-	-	+/- 500	-	kHz
Noise Equivalent BW		-	-	4	-	kHz
Receiver Turn On Time		-	-	5	-	Ms
Operation Temperature		TOP	-20	-	80	C
Baseboard Data Rate		-	-	4.8	-	Hz

ตารางที่ 3.3 แสดงข้อมูลของ RLP315

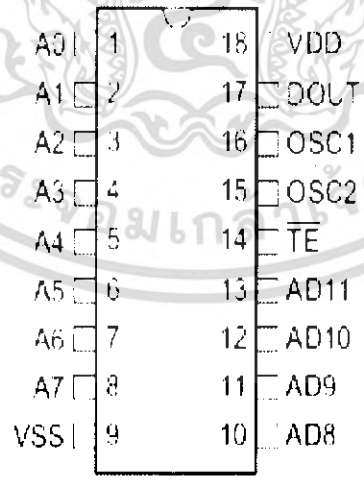
การประยุกต์ใช้งานเราจะใช้งานร่วมกับ IC เบอร์ HT12E และ HT12D ซึ่งเป็นตัว Encoder และDecoder เพื่อที่จะทำให้สามารถเพิ่มช่องสัญญาณในการรับ-ส่งได้



รูปที่ 3.17 แสดงการประยุกต์ใช้งาน RLP315

เมื่อข้อมูลไปถึงภาครับ โดยรับจาก Antenna ของ RLP315 ซึ่งข้อมูลนี้ก็คือข้อมูลที่ถูก Encoder มาแล้วเราก็นำข้อมูลนี้ผ่านเข้าไปที่ HT12D (ที่ถูก Set ขา A_0 และ A_1 ให้ตรงกับตัวส่งแล้ว) ทางขา Data In ของ HT12D ข้อมูลที่ถูก Decode แล้วก็จะปรากฏอยู่ที่ D_8–D_{11} ของ HT12D ในขณะที่ขา VT ของ HT12D ก็จะเป็น Logic 1 เพื่อแสดงว่า HT12D สามารถ Decode สัญญาณได้

3.9.1 HT12E IC Encoder ของภาคส่ง



รูปที่ 3.18 แสดงภาพ IC HT12E

Pin Name	I/O	Internal Connection	Description
A0-A7	I	NMOS Transmission Gate Protection Diode	Input pins address A0-A7 setting. These pins can be externally set to VSS or left open
AD8-AD11	I	NMOS Transmission Gate Protection Diode	Input pins for address/data AD8-AD11 setting. These pins can be externally set to VSS or left open
DOUT	O	CMOS OUT	Encoder data serial transmission output
TE	I	CMOS IN Pull – high	Transmission enable, active low
OSC 1	I	OSCILLATOR 1	Oscillator input pin
OSC 2	O	OSCILLATOR 2	Oscillator input pin
VSS	I	-	Negative power supply, grounds
VDD	I	-	Positive power supply

ตารางที่ 3.4 แสดงคำบรรยายขาแต่ขาของ HT12E

3.9.2 การทำงานของ HT12E

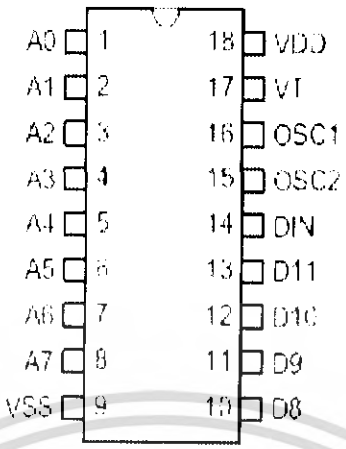
สถานะของแต่ละขาของ address/data สามารถกำหนดค่าจำเพาะได้โดยให้ preset เป็นลอจิก Low หรือ High ถ้าสัญญาณ transmission-enable นำมาใช่ encoder สแกนและส่งผ่านสถานะของ 12 บิต ของ address/data แบบอนุกรมผ่าน A0 ถึง AD11 สำหรับ HT12E encoder

เนื่องจากการส่งผ่านข้อมูล บิตเหล่านี้จะถูกส่งด้วยกระบวนการซิงโครนัส ถ้า trigger signal ไม่ทำงาน IC จะเข้าสู่โหมด standby คือไม่ทำงานและกระแสจะลดลงน้อยกว่า 1uA สำหรับแหล่งจ่ายแรงดัน 5 โวลต์

โดยปรกติการประยุกต์ใช้ preset ขา address ด้วยรหัสจำเพาะโดยใช้ DIP หรือ PCB ขณะที่ข้อมูลถูกเลือกโดยการกดสวิตซ์

Transmission enable สำหรับ Encoder HT12E การส่งผ่านจะถูกกำหนดโดย ใช้สัญญาณ Low ให้กับขา TE

3.9.3 HT12D IC Decoder ของภาครับ



รูปที่ 3.19 แสดงภาพ IC HT12D

Pin Name	I/O	Internal Connection	Description
A0-A7	I	NMOS Transmission Gate	Input pins for address A0-A7 setting. These pins can be externally set to VSS or VDD
AD8-AD11	O	CMOS OUT	Output data pins
DIN	I	CMOS IN	Serial data pin
VT	O	CMOS OUT	Valid Transmission, active high
OSC 1	I	OSCILLATOR 1	Oscillator input pin
OSC 2	O	OSCILLATOR 2	Oscillator input pin
VSS	I	-	Negative power supply, grounds
VDD	I	-	Positive power supply

ตารางที่ 3.5 แสดงคำบรรยายขาแต่ละขาของ HT12D

3.9.4 การทำงานของ HT12D

12 บิต อนุกรมของ decoder สามารถแบ่งได้หลายกรณีโดยการรวม ขา Address และขา Data โดยการจับคู่กับ 12 บิต อนุกรมของ encoder

Decoder รับข้อมูลที่ส่งมาจาก encoder แปล N บิตแรกของรหัสเป็น Address และ 12-N บิตสุดท้ายเป็น Data เมื่อ N เป็นรหัส Address สัญญาณที่ขา DIN active จึงจะทำการ decode ข้อมูล Decoder จะ check ข้อมูลจาก Address ที่ได้รับมาอย่างต่อเนื่อง 3 ครั้ง ถ้า Address Code ตรงกับ Address ของ decoder 12-N บิตของข้อมูลจะถูก decode ไปที่ขา Output และขา VT จะเป็น High เพื่อบอกว่าข้อมูลถูกต้อง

ดังนั้นขา VT จะเป็น High เมื่อข้อมูลถูกต้องเท่านั้น โดยปรกติจะเป็น Low

บทที่ 4
เซอร์โวมอเตอร์ และวงจรจ่ายแรงดันไฟตรง

4.1 หลักการทำงานของเซอร์โวมอเตอร์

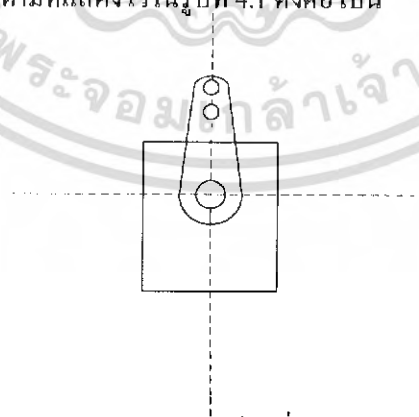
เซอร์โวมอเตอร์ ประกอบด้วย มอเตอร์ความเร็วสูง ภายในมีเฟืองทดรอบ ให้หมุนช้าลง เพื่อจะได้มีกำลังแรงบิดที่สูงขึ้น นอกจากนี้ยังมีวงจรควบคุมมอเตอร์ ซึ่งวงจรนี้จะนำค่าแรงดันเฉลี่ยของพัลส์รูปที่เหลี่ยม เข้าไปเปรียบเทียบกับค่าแรงดันค่าหนึ่งที่มีอยู่ในวงจร ถ้าค่าต่างกัน วงจรควบคุมจะสั่งให้มอเตอร์หมุนไปตามทิศทาง ซึ่งขึ้นอยู่กับขนาดความกว้างพัลส์ โดยที่แกนเฟืองทดรอบจะถูกพ่วงไปขับแกนของ VR (ตัวต้านทานปรับค่าได้) ซึ่งอยู่ในวงจรควบคุมมอเตอร์ ในขณะที่มอเตอร์หมุน VR จะถูกปรับค่าทำให้ค่าแรงดันเปรียบเทียบของวงจรควบคุมมอเตอร์เปลี่ยนไปด้วย จนกระทั่งค่าแฉลี่ยของพัลส์ในวงจรควบคุมมอเตอร์ เท่ากับค่าเฉลี่ยของพัลส์ที่เข้ามา จึงทำให้มอเตอร์หยุดหมุนได้

เซอร์โวมอเตอร์จะมีสายไฟ 3 เส้นคือ สายไฟเลี้ยง สายกราวด์ และสายสัญญาณพัลส์ควบคุม ซึ่งลักษณะของสัญญาณ พัลส์ที่ใช้ควบคุมตำแหน่งของเซอร์โวมอเตอร์ จะเป็นการส่งพัลส์ที่มีความกว้างต่างกัน เพื่อควบคุมให้เซอร์โวมอเตอร์ หมุนไปยังตำแหน่งที่ต้องการ โดยที่มีความกว้างของพัลส์ จะเป็นตัวกำหนดขนาด และทิศทางของการหมุนแกนเซอร์โวมอเตอร์สำหรับคาบเวลา หรือระยะห่างระหว่างพัลส์แต่ละลูก จะเป็นตัวกำหนดแรงบิดของมอเตอร์

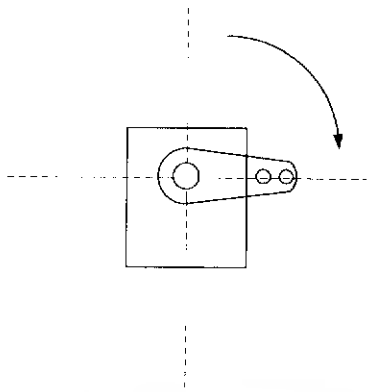
ถ้ากำหนดให้ในสภาวะปกติ เมื่อป้อนพัลส์ สี่เหลี่ยม ที่มีความกว้างขนาด 1.5 ms ให้กับเซอร์โวมอเตอร์ แขนของเซอร์โวมอเตอร์จะอยู่ตำแหน่งกลาง

เมื่อป้อนพัลส์ สี่เหลี่ยม ที่มีความกว้างขนาด 2 ms ให้กับเซอร์โวมอเตอร์ แขนของเซอร์โวมอเตอร์จะหมุนตามเข็มนาฬิกา

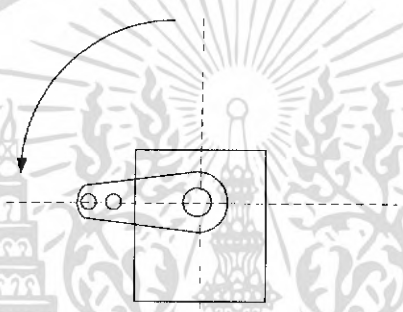
เมื่อป้อนพัลส์ สี่เหลี่ยม ที่มีความกว้างขนาด 1 ms ให้กับเซอร์โวมอเตอร์ แขนของเซอร์โวมอเตอร์จะหมุนทวนเข็มนาฬิกา ตามที่แสดงไว้ในรูปที่ 4.1 ดังต่อไปนี้



ก. แสดงการตอบสนองของเซอร์โวเมื่อจ่ายพัลส์ขนาด 1.5 ms



ข. แสดงการตอบสนองของเซอร์โวเมื่อจ่ายพัลส์ขนาด 2 ms



ค. แสดงการตอบสนองของเซอร์โวเมื่อจ่ายพัลส์ขนาด 1 ms

รูปที่ 4.1 แสดงการตอบสนองของเซอร์โวมอเตอร์ต่อสัญญาณพัลส์ในความถี่ที่ต่างกัน

ดังนั้นถ้าจ่ายสัญญาณพัลส์ที่มีความกว้างมากหรือน้อยกว่าความกว้างของพัลส์ 1.5 ms ก็จะทำให้เซอร์โวมอเตอร์หมุนต่างทิศกัน ทั้งตามเข็มและทวนเข็มนาฬิกา โดยตำแหน่งของแขนหมุนเซอร์โวมอเตอร์จะเบี่ยงเบนออกจากจุดกึ่งกลางเป็นสัดส่วนกับความกว้างของพัลส์ที่จ่ายให้

4.2 ภาคการทำงานของเซอร์โวมอเตอร์

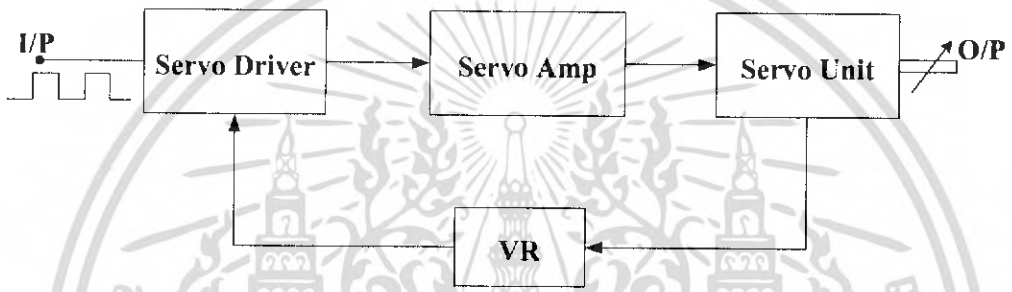
ในเซอร์โวมอเตอร์หนึ่งตัวจะประกอบไปด้วย 3 ภาคการทำงานแต่ละภาคมีหน้าที่และการทำงานดังนี้คือ

ภาคขับเซอร์โว ประกอบด้วย วงจรสร้างสัญญาณพัลส์ และวงจรเปรียบเทียบสัญญาณพัลส์ที่สร้างขึ้น กับสัญญาณพัลส์ I/P ที่รับเข้ามา

ภาคขยายเซอร์โว ประกอบด้วย วงจร RC Network ที่ช่วยหน่วงสัญญาณให้เซอร์โวสามารถทำงานได้ตลอดช่วงคาบเวลาจนกระทั่งมีสัญญาณลูกต่อไปมารวมถึงวงจรกลับขั้วแรงดันไฟฟ้าควบคุมทิศทางการหมุนของมอเตอร์

หน่วยเซอร์โว ประกอบด้วย มอเตอร์ความเร็วสูง เฟืองทดรอบ แกนหมุน อุปกรณ์ต่าง ๆ และ VR (ตัวต้านทานปรับค่าได้) ทำหน้าที่ป้อนกลับตำแหน่ง (Position Feedback)

ซึ่งในขณะที่มอเตอร์หมุน VR จะถูกปรับค่า Feedback กลับมาปรับและเปรียบเทียบค่าความกว้างของพัลส์ที่ภาคขับเซอร์โว เมื่อขนาดความกว้างของพัลส์ มีค่าแรงดันเฉลี่ยเท่ากันมอเตอร์จะหยุดหมุนทันที ซึ่งรูปที่ 4..2 ได้แสดงภาคการทำงานของเซอร์โวมอเตอร์ตามที่ได้กล่าวไว้ในข้างต้น และได้แสดงไว้ดังรูปต่อไปนี้



รูปที่ 4.2 แสดงภาคการทำงานของเซอร์โวมอเตอร์

รูปที่ 4.3 ส่วนประกอบต่างๆของเซอร์โวมอเตอร์

4.3 วงจรจ่ายแรงดันไฟตรง (DC supply)

ส่วนประกอบที่สำคัญ

วงจรจ่ายไฟตรงนั้นโดยทั่วไปประกอบด้วยส่วนประกอบที่สำคัญ 4 ส่วนใหญ่ๆ ดังนี้

4.3.1 หม้อแปลงไฟฟ้า (Transformer)

หม้อแปลงไฟฟ้าเป็นส่วนที่ทำหน้าที่เปลี่ยนค่าของระดับแรงดันไฟฟ้ากระแสสลับให้มีค่าเป็นไปตามการออกแบบหรือที่คำนวณไว้ ซึ่งหม้อแปลงมีทั้งแบบแปลงขึ้น (Step Up) และแบบแปลงลง (Step Down) โดยแรงดันไฟฟ้าสลับที่ผ่านออกมาจากหม้อแปลงนั้นจะถูกเปลี่ยนระดับเท่านั้น ส่วนรูปร่างสัญญาณ และค่าความถี่ยังคงเหมือนเดิม

4.3.2 วงจรเรียงกระแส (Rectifier)

วงจรเรียงกระแสจะทำหน้าที่เปลี่ยนไฟฟ้ากระแสสลับให้เป็นไฟฟ้ากระแสตรง โดยจะได้ไฟฟ้ากระแสตรงในลักษณะที่คล้ายสัญญาณ Pulse คือมีการกระเพื่อมสูงมาก สำหรับการหาค่าแรงดันไฟตรงทำโดยหาค่าเฉลี่ยของรูปสัญญาณ และมีคุณสมบัติต่างๆ ที่สำคัญดังตารางที่ 1.1

โดย V_P คือแรงดันไฟฟ้าเอาท์พุทของวงจรเรียงกระแส (V_P)

V_{AC} คือค่าแรงดันไฟสลับที่จ่ายให้กับวงจรเรียงกระแส (V_{RMS})

F_R คือค่าความถี่ของการกระเพื่อมของแรงดันเอาท์พุทของวงจรเรียงกระแส (H_Z)

f_{line} คือความถี่ของไฟสลับ (H_Z)

$I_{D\,(DC)}$ คือกระแสเฉลี่ยที่ไหลผ่านไดโอดของวงจรเรียงกระแส (A)

$I_{O\,(DC)}$ คือกระแสไฟตรงเอาท์พุทของวงจรเรียงกระแส (A)

คุณสมบัติ	HW	FWCT	FWB
V_P (V_P)	$V_P = \sqrt{2}\,V_{AC} - 0.7$	$V_P = \sqrt{2}\,V_{AC} - 0.7$	$V_P = \sqrt{2}\,V_{AC} - 1.4$
F_R (H_Z)	f_{line}	$2f_{line}$	$2f_{line}$
I_D (A)	$I_{O\,(DC)}$	$I_{O\,(DC)}/2$	$I_{O\,(DC)}/2$

ตารางที่ 4.1 สรุปคุณสมบัติที่สำคัญของวงจรเรียงกระแสแบบต่างๆ

4.3.3 วงจรกรองแรงดันกระเพื่อม (Ripple Filter)

วงจรกรองลดแรงดันกระเพื่อม จะทำหน้าที่กรองแรงดันกระเพื่อมให้ได้ขนาดของแรงดันกระเพื่อมน้อยลง ซึ่งโดยทั่วไปเปอร์เซ็นต์ริปเปิลน้อยกว่า 10 เปอร์เซ็นต์

วงจรกรองแรงดันสัญญาณสามารถสร้างขึ้นมาได้จากอุปกรณ์ที่เป็นอุปกรณ์รีแอคทีฟ (Reactive) ได้แก่ ตัวเก็บประจุ และตัวเหนี่ยวนำ ซึ่งจะต่อในลักษณะความถี่ต่ำผ่าน (Low Pass Filter) ซึ่งในที่นี้จะใช้ตัวเก็บประจุมาต่อขนานกับเอาท์พุทของวงจรเรียงกระแส โดยค่าตัวเก็บประจุและแรงดันไฟตรงที่หาได้ หาได้จากสูตร

$$C = / (F_R . \Delta V_{DC})$$

และสูตร

$$V_{O(DC)} = \sqrt{2}\, V_{AC} - V_D - 0.5 \Delta V_{DC}$$

โดยที่ C คือค่าความจุไฟฟ้าของตัวเก็บประจุ (F)

I_O คือกระแสไฟตรงที่ทางออก (A)

ΔV_{DC} คือค่าขนาดของแรงดันกระเพื่อม (V_{P-P})

$V_{O(DC)}$ คือค่าแรงดันไฟตรงที่ทางออก (V)

V_{AC} คือค่าแรงดันไฟสลับที่จ่ายให้กับวงจรเรียงกระแส (V_{RMS})

V_D คือค่าแรงดันที่ตกคร่อมไดโอด (V) ซึ่งสำหรับแบบ HW, FWCT มีค่า 0.7 V และสำหรับแบบ FWB มีค่า 1.4 V

F_R คือค่าความถี่ของการกระเพื่อม (H_Z) ซึ่งค่า $F_R = f_{line}$ สำหรับแบบ HW และมีค่า $F_R = 2f_{line}$ สำหรับแบบ FW, FWCT

f_{line} คือค่าความถี่ของไฟสลับ (H_Z)

ไฟตรงที่ได้หลังจากวงจรกรองลดแรงดันกระเพื่อมแล้ว ยังคงมีการกระเพื่อมอยู่บ้าง เราใช้ค่าตัวประกอบของการกระเพื่อม (Ripple Factor: R) หรือร้อยละของการกระเพื่อม (Percent Ripple: %R) เป็นเครื่องบ่งชี้ว่าไฟตรงนั้นมีคุณภาพอย่างไร ค่าของ R และ %R นิยมโดย

R = ค่า RMS ของแรงดันกระเพื่อม / ค่าของแรงดันไฟตรง

$\%R = 100R$

โดยการประมาณว่าแรงดันกระเพื่อมมีรูปคลื่นเป็นสามเหลี่ยม ค่าของ R และ %R จะเป็น

$$R = \Delta V_{DC} / (2\sqrt{3}\, V_{O(DC)})$$

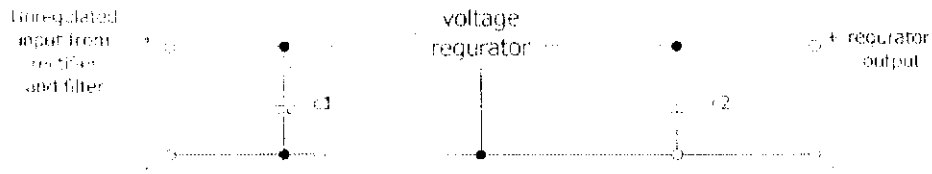
$$\%R = 100 \Delta V_{DC} / (2\sqrt{3}\, V_{O(DC)})$$

4.3.4 วงจรแรงดันไฟตรงคงที่ (Voltage Regulator)

วงจรแรงดันไฟตรงคงที่ใช้สำหรับวงจรอิเล็กทรอนิกส์ที่ต้องการไฟตรงที่เรียบมากๆ วงจรแรงดันไฟตรงคงที่จะทำหน้าที่กำจัดแรงดันกระเพื่อมที่ยังหลงเหลืออยู่เพียงเล็กน้อยให้หมดไป แรงดันไฟตรงคงที่จะมีคุณภาพใกล้เคียงกับแบตเตอรี่มากสามารถสร้างได้จากอุปกรณ์อิเล็กทรอนิกส์พื้นฐานหรือใช้วงจรรวม (IC) ได้

วงจรเรกกูเลเตอร์โดยใช้ IC 3 ขา

วิธีการต่อ IC 3 ขา โดยทั่วไปแสดงในรูปที่ 4.4 ทุกตัวจะมีวงจรป้องกันการลัดวงจร วงจรการทำงานอัตโนมัติและโดยทั่วไปไม่จำเป็นต้องต่ออุปกรณ์ภายนอกเพิ่ม



รูปที่ 4.4 วงจรเรกกูเลเตอร์โดยใช้ IC 3 ขา โดยทั่วไป

ตัวเก็บประจุ C1 ควรจะมีค่าประมาณ 100 – 200 nF สำหรับแบบ Ceramic หรือ 2 μF สำหรับแบบ Tantalum เพื่อทำให้วงจรทางด้านอินพุทของ IC มีเสถียรภาพมากขึ้น

ตัวเก็บประจุ C2 มีค่าประมาณ 1 μF สำหรับแบบ Tantalum เพื่อทำให้การ Regulation มี noise rejection ที่ดีและป้องกันการเกิด Transient

สิ่งแรกที่เราจะต้องพิจารณาคือ การทนกำลังงานสูงสุดของ IC และความร้อน เพราะถ้า IC มีอุณหภูมิสูงเกินไป วงจรปิดการทำงานอัตโนมัติจะทำงาน ดังนั้นการระบายความร้อนที่ดีจึงเป็นเรื่องจำเป็นต่อไปที่เราพิจารณาคือ Drop out voltage ซึ่งคือผลต่างของความต่างศักย์ระหว่างอินพุทและเอาท์พุท ซึ่งขึ้นอยู่กับกระแสโหลด และอุณหภูมิด้วย เพื่อทำให้วงจรเพียงพอที่จะทำงานได้อย่างปลอดภัย โดยทั่วไปจะมีค่าประมาณ 1.5 – 2 V

อีกสิ่งหนึ่งที่สำคัญคือการพิจารณา Ripple ก่อนเข้าอินพุทว่ายังจะสามารถทำให้วงจรทำงาน ได้อยู่ตลอดช่วงกว้างของอินพุทที่เปลี่ยนแปลงได้หรือไม่ด้วย

บทที่ 5
การประมวลผลภาพเชิงตัวเลข (Digital image Processing)

การเกิดภาพโดยใช้เลนส์เป็นวิธีการพื้นฐานและพบทั่วไปในการถ่ายภาพ แสงที่กระทบวัตถุและสะท้อนกลับมาที่เลนส์จะถูกรวมและนำมาแสดงยังจุดที่สอดคล้องกันกับวัตถุ ดังนั้นกระบวนการถ่ายภาพจึงเป็นการเปลี่ยนข้อมูลภาพของวัตถุจาก 3 มิติ มาเป็นข้อมูล 2 มิติโดยที่การบันทึกหรือประมวลผลรูปแบบของแสงที่สะท้อนมาจากวัตถุทำได้โดยการใส่เซ็นเซอร์ที่ให้สัญญาณทางไฟฟ้าออกมาในรูปแบบที่ต่อเนื่อง ในขั้นตอนนี้เซ็นเซอร์จะทำการสแกนหรือทำการวัดผลรวมความเข้มของแสงที่จุดเล็กๆที่ละจุดไปเรื่อยๆตามแนวทางที่กำหนดไว้ หรือตามแนวราสเตอร์(Raster scan) ซึ่งจากปกติจะไล่จากซ้ายไปขวาและจากบนลงล่าง ค่าที่เซ็นเซอร์วัดได้นี้ถือว่าเป็นค่าความเข้มภาพ f(x,y) ที่พิกัด (x,y) และก็มีความต่อเนื่องด้วย ดังนั้นในการที่จะนำภาพมาประมวลผลโดยคอมพิวเตอร์จำเป็นต้องทำให้ภาพที่มีลักษณะต่อเนื่องนี้กลายเป็นภาพดิจิตอลหรือภาพเชิงตัวเลขเสียก่อน โดยทำการดิจิไทซ์(digitization) ค่าความเข้ม f(x,y) ซึ่งเป็นฟังก์ชันต่อเนื่องไปเป็นฟังก์ชันไม่ต่อเนื่อง g(x,y) โดยการแบ่ง f(x,y) ออกเป็นช่วงๆนี้สามารถแทนด้วยค่าตัวเลขใดค่าหนึ่งจาก L ระดับ ซึ่งโดยทั่วไปจุดภาพแต่ละจุด (pixel) จะเป็นสมาชิกของเมตริกซ์ของจุดภาพที่มีขนาด M แถว N หลัก เพราะฉะนั้น x,y จะมีค่าอยู่ในช่วง ($1 \leq y \leq M$, $1 \leq x \leq N$) และจำนวนช่วงระดับความเข้มของช่วงจุดภาพ L ระดับนี้จะบ่งบอกถึงความละเอียด (resolution) ของภาพเชิงตัวเลขด้วย ปกติแล้วนิยมกำหนดให้ L มีค่าเท่ากับ 256 ระดับ ซึ่งจะทำให้ระดับความเข้มของภาพอยู่ในช่วง (0-255) โดยใช้ 8 บิตของเลขฐานสองในการเก็บข้อมูลภาพแต่ละจุดนั่นเอง แต่ก็ยังมีการเก็บข้อมูลที่มากกว่านี้ได้ คือ อาจจะเป็น 16 หรือ 24 บิตซึ่งจะทำให้ระดับความเข้มของจุดภาพเป็นไปได้ถึง 2^{16} หรือ 2^{24} ตามลำดับ

รูปที่ 5.1 การแทนสัญญาณอนาลอกด้วยสัญญาณเชิงตัวเลข

จากการที่นำภาพเชิงตัวเลขไปใช้ในการประมวลผลในรูปแบบต่างๆมากมายนั้น สามารถที่จะแบ่งรูปแบบของการประมวลผลภาพเหล่านั้นออกเป็น 2 ระดับด้วยกัน คือ การประมวลผลภาพในระดับต่ำ (Low–level Image Processing) และการประมวลผลภาพในระดับสูง (High–level Image Processing)

การประมวลผลภาพในระดับต่ำนั้นเป็นการประมวลผลภาพเชิงตัวเลขเกือบทั้งหมดเพื่อหาตัวแปรต่างๆมาอธิบายข้อมูลภาพ และมีจุดประสงค์ที่จะนำตัวแปรเหล่านี้ไปใช้ในการประมวลผลภาพในระดับสูงต่อไป แต่การประมวลผลภาพในระดับสูงคือการทำให้คอมพิวเตอร์รู้จักและเข้าใจภาพได้ เช่น การจดจำรูปแบบของตัวเลข ตัวอักษร เป็นต้น ในขณะที่การประมวลผลภาพในระดับต่ำ โดยทั่วไปจะประกอบด้วย การกำจัดสัญญาณรบกวน การทำให้ภาพคมชัดขึ้น การหาขอบเขตของภาพ การทำเซกเมนต์ภาพหรือการแบ่งแยกวัตถุภายในภาพ การสร้างภาพไบนารี เป็นต้น

ความแตกต่างที่สำคัญอีกข้อหนึ่งของการประมวลผลภาพใน 2 มิติ คือ ข้อมูลที่จะนำมาใช้ประมวลผลภาพ ซึ่งการประมวลผลภาพในระดับต่ำจะใช้ค่าความสว่างหรือระดับความเข้มของจุดภาพโดยตรง ส่วนการประมวลผลภาพในระดับสูงข้อมูลที่จะถูกนำมาประมวลผลจะถูกแสดงในรูปของสัญลักษณ์ โดยสัญลักษณ์เหล่านี้จะแสดงถึงสิ่งต่างๆที่อยู่บนภาพ และใช้ตัวแปรที่ได้จากการประมวลผลในระดับต่ำมาอธิบายถึงสัญลักษณ์เหล่านั้น ซึ่งจะเห็นได้ว่าการประมวลผลภาพในระดับต่ำนี้มีความสำคัญมากสำหรับที่จะให้คอมพิวเตอร์สามารถเข้าใจและรู้จักภาพได้

5.1 การทำเทรชโชลด์ (Thresholding technique)

การทำเทรชโชลด์ถือว่าเป็นเทคนิคที่สำคัญในการประมวลผลภาพในส่วนของการทำเซกเมนต์ภาพ ซึ่งประสงค์ของการทำเซกเมนต์ภาพ คือ การแยกองค์ประกอบของภาพไปเป็นส่วนประกอบย่อยๆ ที่มีความสัมพันธ์กันทางกายภาพนั้น และส่วนประกอบที่ถูกแยกออกมานั้นอาจถูกนำไปประมวลผลภาพในส่วนอื่นได้ต่อไป ซึ่งการทำเซกเมนต์ภาพจะมีหลักการทำงานในแนวเดียวกันกับสายตาของคน คือ สามารถแยกลักษณะเด่นออกมาจากภาพที่มองเห็นและเทคนิคการทำการทำเทรชโชลด์ซึ่งถือว่าเป็นเทคนิคในการแยกองค์ประกอบของภาพที่ง่ายเทคนิคหนึ่ง มีคุณสมบัติอยู่ในบางช่วงใดๆ จะถูกจัดเป็นกลุ่มได้โดยที่ระดับความเข้มหนึ่งนั้นสามารถที่จะแบ่งแยกกลุ่มของจุดภาพออกเป็น 2 กลุ่มได้อย่างชัดเจน คือ กลุ่มของวัตถุ(Object) ซึ่งจะมีระดับความเข้มของภาพ g(x,y) ค่อนข้างต่ำ (มืด) กับกลุ่มของส่วนที่เป็นพื้นหลัง (Background) ที่จะมีระดับความเข้มของภาพ g(x,y) ค่อนข้างสูง (สว่าง) ดังเช่นรูปที่ ซึ่งแสดง ฮิสโตแกรมของระดับความเข้มของภาพที่ถูกแบ่งออกเป็น 256 ระดับ จะเห็นได้ว่าการที่จะแยกกลุ่มข้อมูลออกเป็น 2 กลุ่มอย่างชัดเจนย่อมสามารถทำได้โดยการเลือกค่าเทรชโชลด์ที่มีค่าความเข้มอยู่ระหว่างกลุ่มทั้งสองบนฮิสโตแกรมระดับความเข้มของภาพ แล้วทำการตรวจสอบแต่ละจุดภาพว่าถ้ามีค่า g(x,y) น้อยกว่าค่าเทรชโชลด์ถือว่าเป็นจุดภาพของวัตถุที่แสดงได้ด้วยจุดดำ แต่ถ้าหากว่าจุด g(x,y) นั้นมีค่ามากกว่าหรือเท่ากับค่าเทรชโชลด์ก็ถือว่าเป็นจุดภาพในส่วนพื้นหลังได้ด้วยจุดขาว ดังนั้นข้อมูลภาพ $g_{thr}(x,y)$ ที่ผ่านการทำเทรชโชลด์สามารถนิยามได้ดังนี้

$$g_{thr}(x,y) = 0 \text{ if } g(x,y) < T$$

$$g_{thr}(x,y) = 1 \text{ if } g(x,y) \geq T \qquad \text{.....(1)}$$

โดยที่

$g_{thr}(x,y)$	คือ	ข้อมูลภาพผลลัพธ์เป็นไบนารี
$g(x,y)$	คือ	ข้อมูลภาพอินพุทที่มีระดับความเข้ม 0 ถึง L ระดับ
T	คือ	ค่าเทรชโฮลด์เป็นค่าคงที่ที่มีค่าระหว่าง 0 ถึง L
0	คือ	จุดดำ (ส่วนที่เป็นวัตถุ)
1	คือ	จุดขาว (ส่วนที่เป็นพื้นหลัง)

รูปที่ 5.2 ฮิสโตแกรมระดับความเข้มข้นของภาพที่มีความเหมาะสม
สำหรับการทำเทรชโฮลด์แบบครอบคลุม

รูปแบบในการทำเทรชโฮลด์

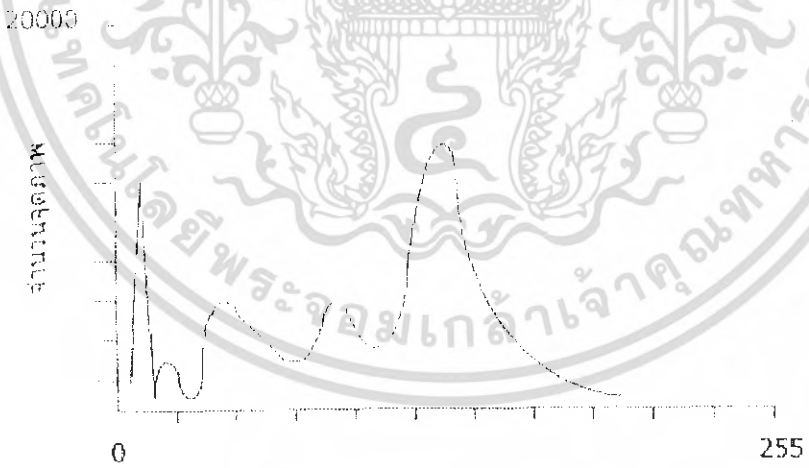
ภาพที่มีระดับความเข้มของจุดภาพของส่วนที่เป็นวัตถุและจุดภาพของส่วนที่เป็นพื้นหลังแตกต่างกันอย่างชัดเจนและมีความสม่ำเสมอตลอดทั้งภาพ สามารถใช้ค่าเทรชโฮลด์เพียงค่าเดียวในการทำเซกเมนต์กับแต่ละจุดภาพทั่วทั้งภาพได้ เรียกการทำเทรชโฮลด์แบบนี้ว่า การทำเทรชโฮลด์แบบครอบคลุม (Global Thresholding) แต่ถ้าภาพนั้นมีระดับความเข้มไม่สม่ำเสมอเกิดขึ้นในส่วนของภาพวัตถุหรือพื้นหลัง หรือในทั้งสองส่วน การใช้ค่าเทรชโฮลด์เพียงค่าเดียวตลอดทั้งภาพย่อมไม่เหมาะสมกับภาพนั้น ในกรณีนี้ค่าเทรชโฮลด์ที่ดีควรมีการปรับเปลี่ยนค่าไปตามตำแหน่งของจุดภาพนั้นได้ คือ การใช้ค่าเทรชโฮลด์ที่ต่างกันสำหรับจุดภาพที่ตำแหน่งต่างกัน และเรียกการทำเทรชโฮลด์ในลักษณะดังกล่าวนี้ว่า การทำเทรชโฮลด์แบบปรับค่า (Adaptive Thresholding)

5.1.1 การทำเทรชโชลด์แบบครอบคลุม (Global Thresholding)

สำหรับขั้นตอนการหาค่าเทรชโชลด์ที่ครอบคลุมตลอดทั้งภาพโดยอัตโนมัติ ปกติจะมีพื้นฐานของการดำเนินการอยู่บนฮิสโตแกรมของระดับความเข้มของจุดภาพ ซึ่งฮิสโตแกรมระดับความเข้มนี้สามารถสร้างได้จากการนับจำนวนของจุดภาพที่มีระดับความเข้มเท่ากับค่าความเข้มที่จุดนั้นทั้งหมดทั่วภาพนั่นเอง จากนั้นจึงทำการหาค่าเทรชโชลด์ในรูปแบบต่างๆ ที่สามารถแบ่งฮิสโตแกรมนี้ออกเป็น 2 ส่วน (ส่วนที่เป็นระดับความเข้มของวัตถุ กับส่วนที่เป็นระดับความเข้มของพื้นหลัง) ได้อย่างถูกต้องตรงตามความต้องการ โดยเฉพาะอย่างยิ่ง ในกรณีของภาพที่มีอัตราความแตกต่างของระดับความเข้มระหว่างส่วนที่เป็นวัตถุกับส่วนที่เป็นพื้นหลังที่มีค่าสูง (แตกต่างกันมาก) และยังมีระดับความเข้มที่เกิดขึ้นในแต่ละส่วนมีความสม่ำเสมอ ย่อมเหมาะสมที่จะใช้ระดับความเข้มที่มีจำนวนของจุดภาพที่ต่ำสุดซึ่งอยู่ระหว่างกลุ่มระดับความเข้มที่มีค่าสูงสุด ทั้งสองกลุ่มบนฮิสโตแกรมเป็น "ค่าเทรชโชลด์" หรือในกรณีทั่วๆไปค่าเทรชโชลด์อาจจะพิจารณาจากค่าระดับความเข้มที่สามารถแบ่งฮิสโตแกรมออกเป็น 2 กลุ่มแล้วทำให้ความแปรปรวนที่เกิดขึ้นระหว่างกลุ่มมีค่ามากที่สุด แต่ความแปรปรวนที่เกิดขึ้นระหว่างกลุ่มที่มีค่าต่ำสุด หลังจากนั้นนำค่าเทรชโชลด์ที่คำนวณได้ไปทำเทรชโชลด์กับแต่ละจุดภาพทั่วทั้งภาพเพื่อให้ได้ภาพผลลัพธ์ที่เป็นไบนารีในที่สุด

5.1.2 การทำเทรชโชลด์แบบปรับค่า (Adaptive Thresholding)

ในกรณีที่ข้อมูลภาพมีความไม่สม่ำเสมอเกิดขึ้นในส่วนของวัตถุ หรือส่วนของพื้นหลัง หรือในทั้งสองส่วน ซึ่งภาพในลักษณะเช่นนี้ฮิสโตแกรมระดับความเข้มของภาพที่เกิดขึ้นอาจมีลักษณะดังรูปที่ 5.3



รูปที่ 5.3 ฮิสโตแกรมระดับความเข้มของภาพที่เหมาะสม
สำหรับการทำเทรชโชลด์แบบปรับค่า

การใช้ค่าเทรชโชลด์แบบครอบคลุมเพียงค่าเดียวกับภาพทั้งภาพนี้ อาจทำให้ผลลัพธ์ที่ได้ไม่ถูกต้อง จากปัญหาที่เกิดขึ้นนี้สามารถแก้ไขได้โดยการแบ่งข้อมูลของภาพทั้งภาพออกเป็นภาพย่อยๆ ที่แสดงได้ดังรูปที่ 5.4 ซึ่งแต่ละภาพย่อยก็จะมีการหาค่าเทรชโชลด์ในรูปแบบที่กำหนดเพื่อหาค่าเทรชโชลด์ที่เหมาะสมสำหรับภาพย่อยนั้น และใช้หาค่าเทรชโชลด์ที่ได้จากการทำเซกเมนต์กับแต่ละภาพย่อยนั้น ขั้นตอนสุดท้ายคือนำแต่ละภาพย่อยที่ผ่านการทำเซกเมนต์แล้วมารวมกันตามพิกัดเดิม

รูปที่ 5.4 การแบ่งภาพออกเป็นภาพย่อยๆและหาค่าเทรชโชลด์ในแต่ละภาพย่อย

5.2วิธีการหาค่าเทรชโชลด์

ขั้นตอนในการทำเซกเมนต์ภาพโดยใช้เทคนิคเทรชโชลด์เพื่อให้ได้ผลลัพธ์ที่เหมาะสมและคมชัดนั้น สิ่งสำคัญที่สุดคือการหาค่าเทรชโชลด์ เนื่องจากถ้าเลือกค่าเทรชโชลด์ที่ไม่เหมาะสม (ค่าเทรชโชลด์มีค่ามากหรือน้อยเกินไป) หรือภาพที่ได้มีสิ่งรบกวน (noise) เกิดขึ้น แล้วย่อมเป็นผลทำให้ภาพผลลัพธ์ที่ได้ไม่ตรงตามความต้องการ ดังนั้นปัญหาของการสร้างภาพไบนารีโดยวิธีเทรชโชลด์นี้ก็คือ ทำอย่างไรจึงจะสามารถคำนวณหาค่าเทรชโชลด์ที่เหมาะสมสำหรับแต่ละภาพที่จะนำมาทำเซกเมนต์ ซึ่งได้มีผู้เสนอวิธีการในการคำนวณหาค่าเทรชโชลด์นี้ไว้หลายวิธี โดยแต่ละวิธีก็เหมาะกับลักษณะการทำงานที่แตกต่างกันไป เช่นการหาค่าเทรชโชลด์โดยการกำหนดค่าล่วงหน้า (Preassigned threshold value), การหาค่าเทรชโชลด์จากค่ากลาง (Mid-range threshold value), และการหาค่าเทรชโชลด์โดยพิจารณาจากฮิสโตแกรม (Histogram threshold value) โดยแต่ละวิธีสามารถอธิบายได้ดังนี้

5.2.1การหาค่าเทรชโชลด์โดยการกำหนดค่าล่วงหน้า (Preassigned threshold value)

การหาค่าเทรชโชลด์ด้วยวิธีนี้เป็นวิธีที่ง่ายที่สุด เพราะสามารถหาค่าเทรชโชลด์ได้จากการกำหนดค่าเองจากผู้ใช้ (User) ซึ่งการกำหนดนี้ขึ้นอยู่กับประสบการณ์ของผู้ใช้นั้นๆ โดยการเลือกค่าคงที่ค่าหนึ่งที่อยู่ระหว่างค่าต่ำสุดและค่าสูงสุดของระดับความเข้มของข้อมูลภาพอินพุท เช่น ข้อมูลภาพอินพุทที่มีระดับเทา 256 ระดับ (0-255) ค่าเทรชโชลด์ที่สามารถเลือกได้ก็คือค่าที่อยู่

ระหว่างค่า 0 ถึง 255 เมื่อเลือกค่าเทรชโชลด์ที่ได้แล้วก็สามารถทำเซกเมนต์ได้โดยใช้สมการ (1) ทั้งนี้การกำหนดค่าเทรชโชลด์ขึ้นมานี้ เช่น 50, 100, 200 อาจนำมาทดลองทำการเซกเมนต์ภาพก่อน แล้วดูผลลัพธ์ที่ได้ จากนั้นจึงเลือกใช้ค่าที่เหมาะสมที่สุดมาใช้งาน

5.2.2 การหาค่าเทรชโชลด์จากค่ากลาง (Mid-range threshold value)

การหาค่าเทรชโชลด์โดยพิจารณาจากค่ากลาง เป็นการหาค่าเทรชโชลด์ที่แตกต่างจากวิธีแรกเนื่องจากเป็นการหาค่าเทรชโชลด์โดยอัตโนมัติโดไม่ต้องให้ผู้ใช้เป็นผู้กำหนด โดยการหาค่าเทรชโชลด์วิธีนี้ได้อาศัยการคำนวณพื้นฐานทางสถิติในเรื่องของการหาค่ากลางแบบที่เป็นค่าเฉลี่ย (Mean) มาประยุกต์ใช้ค่าเทรชโชลด์ที่คำนวณได้จะเป็นค่าที่ได้จากค่ากลางที่อยู่ระหว่างค่าระดับความเข้มสูงสุด (Maximum level) และการหาค่าระดับความเข้มต่ำสุด (Minimum level) ของข้อมูลภาพอินพุท สำหรับการคำนวณค่ากึ่งกลางนี้สามารถคำนวณได้จากสมการ

$$T = \{ Max[g(x,y)] + Min[g(x,y)] \} /2 \qquad \text{.....(2)}$$

โดยที่

T คือ ค่าเทรชโชลด์

G(x,y) คือข้อมูลภาพอินพุทที่มีระดับความเข้ม 0 ถึง L ระดับ

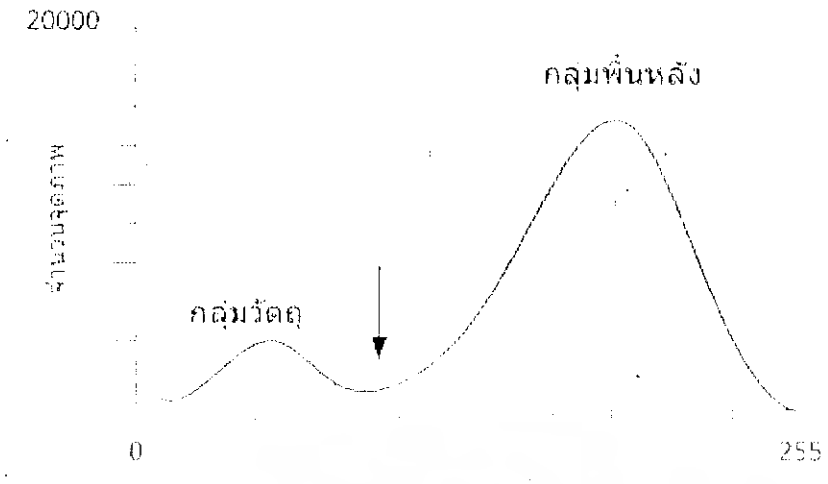
Max[g(x,y)] คือค่าสูงสุดของระดับเทาของข้อมูลอินพุท

Min[g(x,y)] คือค่าต่ำสุดของระดับเทาของข้อมูลอินพุท

เมื่อคำนวณค่าเทรชโชลด์ได้แล้ว ก็สามรถทำการเซกเมนต์ภาพได้โดยนำค่าเทรชโชลด์ที่ได้มาแทนค่าในสมการที่ (1)

5.2.3 การหาค่าเทรชโชลด์โดยพิจารณาจากฮิสโตแกรม (Histogram threshold value)

การหาค่าเทรชโชลด์โดยพิจารณาจากฮิสโตแกรมระดับเทาของข้อมูลอินพุท โดยที่การหาค่าเทรชโชลด์วิธีนี้ ข้อมูลภาพที่เหมาะสมที่เหมาะสมต้องมีลักษณะที่สามารถแบ่งแยกเป็นกลุ่มได้อย่างชัดเจน คือ กลุ่มหนึ่งจะเป็นกลุ่มของวัตถุ และอีกกลุ่มหนึ่ง เป็นพื้นหลัง ซึ่งแนวความคิดในการคำนวณค่าเทรชโชลด์โดยวิธีนี้สามารถแสดงได้ดังรูปที่ 5.5



รูปที่ 5.5 แนวความคิดในการคำนวณค่าเทรชโชลด์โดยวิธีพิจารณาฮิสโตแกรม

สำหรับการคำนวณหาค่าเทรชโชลด์โดยวิธีพิจารณาจากฮิสโตแกรม สามารถสรุปเป็นขั้นตอนได้ดังนี้

1. ข้อมูลภาพอินพุทมีระดับความเข้ม 0 ถึง L ระดับ
2. คำนวณหาฮิสโตแกรมระดับความเข้มของภาพหน้าเอกสาร โดยการนับจำนวนจุดภาพที่ระดับความเข้มแต่ละระดับ
3. จากฮิสโตแกรมจะพบว่าจะเกิดกลุ่มของระดับความเข้มสูงสุด (Peak) 2 กลุ่ม กลุ่มหนึ่งคือกลุ่มของวัตถุ และอีกกลุ่มหนึ่งคือกลุ่มของพื้นหลัง
4. เลือกค่าที่ต่ำที่สุดที่อยู่ระหว่างสองกลุ่มนั้น กำหนดค่านี้เป็นค่าเทรชโชลด์
5. ทำการเซกเมนต์ภาพโดยนำค่าเทรชโชลด์ที่ได้มาแทนค่าในสมการที่ (1)

จากขั้นตอนการทำงานข้างต้น ถ้าได้ทดลองทำกับภาพ 256 ระดับเทาที่เป็นภาพตัวอักษรบนพื้นกระดาษที่มีสีอ่อนกว่า ซึ่งเมื่อทำการสร้างฮิสโตแกรมระดับเทาของภาพแล้วจะปรากฏกลุ่มของข้อมูลสูงสุด2 กลุ่มคือ กลุ่มของข้อความ และกลุ่มของพื้นหลัง จากนั้นทำการเลือกค่าต่ำสุดระหว่าง 2 กลุ่มนั้นเป็นค่าเทรชโชลด์ซึ่งค่าที่ได้จากวิธีนี้จะมีค่าที่เที่ยงตรงที่สุด แต่อย่างไรก็ดีวิธีนี้ไม่เหมาะสมกับภาพที่ไม่สามารถแยกกลุ่มของสิ่งที่อยู่ในภาพได้อย่างชัดเจน ระหว่างกลุ่มของวัตถุและกลุ่มของพื้นหลัง เนื่องจากว่าถ้าหากภาพอินพุทไม่สามารถแยกแยะได้แล้วจะทำให้ค่าเทรชโชลด์ที่คำนวณได้ผิดไปจากความเป็นจริง คืออาจจะมากหรือน้อยเกินไป อันเป็นผลทำให้ได้ภาพที่ไม่เหมาะสม รายละเอียดบางส่วนขาดหายไปได้

5.3 การกำจัดสัญญาณรบกวน (Noise reduction)

การกำจัดสัญญาณรบกวนในภาพ เป็นกระบวนการส่วนแรกก่อนที่จะทำการประมวลผลภาพจริง เพื่อกำจัดภาพที่ไม่พึงประสงค์ออกจากภาพนั้น สัญญาณรบกวนบนภาพอาจเกิดจากขั้นตอนการเก็บภาพ หรือขั้นตอนการส่งข้อมูลภาพจากที่หนึ่งไปยังอีกที่หนึ่ง เป็นต้น

การกำจัดสัญญาณรบกวนนั้น เป็นการกรองสัญญาณภาพแบบความถี่ต่ำผ่าน(Low Pass Filter) ซึ่งภาพผลลัพธ์ที่ได้จะมีองค์ประกอบที่ราบเรียบ (Smoothing) มากขึ้น ซึ่งตรงกับจุดประสงค์ของการลดสัญญาณรบกวนนั่นเอง โดยการกรองภาพใดๆนั้นเราจะอาศัยตัวดำเนินการซึ่งมีลักษณะเป็นตารางสี่เหลี่ยมจัตุรัสขนาดจำกัด หรือบางครั้งเรียกว่าหน้าต่าง หรือหน้ากาก เคลื่อนที่ไปกระทำบางอย่างกับทุกจุดภาพในภาพที่ต้องการลดสัญญาณรบกวนนั้นเพื่อคำนวณหาค่าใหม่ของจุดภาพในตำแหน่งที่อยู่กึ่งกลางของหน้าต่างในการกระทำกับแต่ละจุดภาพนั้น สำหรับหน้าต่างที่นิยมใช้คือ ขนาด 3X3 แต่ทั้งนี้ก็สามารถใช้ขนาดอื่นๆได้อีก เช่น 5X5 , 7X7 หรือ 15X15 ก็ได้

การดำเนินการกรองภาพด้วยหน้าต่างนั้นสามารถแบ่งได้ 2 รูปแบบ ตามลักษณะของหน้าต่าง กล่าวคือ ถ้าการกรองนั้นใช้หน้าต่างที่บรรจุค่าคงที่ไว้แล้ว ผลลัพธ์ที่ได้จากการกรองนี้จะเกิดจากการคำนวณร่วมกันระหว่าง ค่าระดับความเข้มของจุดภาพที่อยู่ใต้หน้าต่างและค่าคงที่ของหน้าต่างนั้นทุกค่า แล้วจะเรียกว่า การกรองแบบเป็นเชิงเส้น (Linear Filter) แต่ถ้าการกรองนั้นใช้หน้าต่างที่ไม่ได้บรรจุค่าใดๆ ไว้โดยผลลัพธ์ที่ได้จากการกรองจะเกิดการคำนวณค่าของจุดภาพทุกจุดที่อยู่ภายในหน้าต่างเท่านั้นแล้วจะเรียกว่า การกรองแบบไม่เชิงเส้น (Nonlinear Filter)

5.3.1การกรองแบบหาค่าเฉลี่ย (Average Filter)

การกรองภาพด้วยวิธีหาค่าเฉลี่ยนี้เป็นการกรองแบบเป็นเชิงเส้นซึ่งมีวิธีการคำนวณหาค่าความเข้มใหม่ (R) ของจุดภาพที่ตำแหน่งกึ่งกลางภายใต้หน้าต่างขนาด 3X3 ที่แสดงเป็นกรณีทั่วไปดังรูปที่ ตามสมการ

$$R = W_1Z_1 + W_2Z_2 + \ldots + W_9Z_9 \qquad \ldots\ldots\ldots\ldots(3)$$

โดย $Z_1, Z_2, \ldots Z_9$ คือ ค่าระดับความเข้มของจุดภาพภายใต้หน้าต่างและมีตำแหน่งที่ตรงกับหน้าต่างนี้

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

รูปที่ 5.6 หน้าต่างขนาด 3X3 ที่บรรจุค่าคงที่ไว้

ในกรณีที่เป็นการกรองแบบหาค่าเฉลี่ยจะต้องมีหลักการเพิ่มเติมเฉพาะคือ หน้าต่างที่ใช้ต้องบรรจุค่าที่เป็นบวก และผลรวมของค่าสมาชิกทุกตัวในแต่ละหน้าต่างจะต้องเท่ากับ 1 เช่น W ทุกตัวที่อยู่ในรูปที่ จะมีค่าเป็น 1/9 เป็นต้น แล้วค่าความเข้มใหม่ (R) ที่คำนวณได้จากสมการที่ (3) ก็จะเป็นผลลัพธ์ที่ได้จากการกรองแบบหาค่าเฉลี่ยของแต่ละจุดภาพนั่นเอง

5.3.2การกรองแบบมัธยฐาน (Median Filter)

การกรองแบบมัธยฐานเป็นการกรองแบบไม่เชิงเส้น โดยมีหลักการสำคัญคือ การครอบหน้าต่างที่มิได้บรรจุค่าใดๆ ไว้ลงบนข้อมูลภาพ แล้วนำข้อมูลเหล่านั้นมาเรียงลำดับจากค่าต่ำสุดไปหาค่าสูงสุด จากนั้นก็เลือกค่าที่อยู่ในตำแหน่งกึ่งกลางข้อมูลที่เรียงลำดับนั้นมาเป็นค่าความเข้มใหม่ (R) ของจุดภาพที่ตำแหน่งกึ่งกลางภายใต้หน้าต่างนั้น เช่น ใช้หน้าต่างขนาด 3X3 กับข้อมูลภาพดังรูปที่ 5.7 (a) จากนั้นนำค่าความเข้มภายในตารางมาเรียงลำดับเป็น 1, 1, 1, 1, 1, 1, 1, 1, 9 แล้วเลือกเอาค่าลำดับที่ 5 ซึ่งในกรณีนี้เป็นค่า 1 เป็นค่าความเข้มใหม่ของจุดภาพที่ตำแหน่งกึ่งกลางหน้าต่างขณะนี้ ทำเช่นนี้ไปจนตลอดทั้งข้อมูลภาพจะได้รูปที่ 5.7 (b) เป็นข้อมูลที่ผ่านการกรองแล้ว จะเห็นได้ว่าค่าที่ได้จากการกรองแบบนี้ อาจจะได้ข้อมูลเดิมหรือค่าใหม่ก็ได้ ถ้าหากข้อมูลตัวใดที่มีค่าแตกต่างไปจากค่าของจุดภาพรอบข้างมาก ก็ถือว่าเป็นสัญญาณรบกวนและจะถูกกำจัดออกไป วิธีนี้มีข้อดีกว่าวิธีกรองแบบหาค่าเฉลี่ย คือ สามารถลดสัญญาณรบกวนได้ดีและยังคงรักษาความคมชัดของขอบภาพไว้ได้ด้วย

1	1	1	1	1
1	9	1	1	6
1	9	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

(a) ข้อมูลภาพ

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

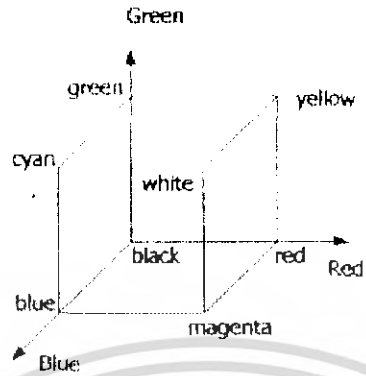
(b) ข้อมูลภาพที่ผ่านการกรองแล้ว

รูปที่ 5.7 การกรองภาพแบบมัธยฐาน

5.4 โมเดลสีแบบ RGB

โมเดลสีนี้ประกอบด้วยการรวมกันของแม่สีหลักซึ่งได้แก่ แดง(Red: R), เขียว (Green: G) และน้ำเงิน (Blue: B) ซึ่งสีค่าต่างๆในแถบสเปคตรัมของสีจะได้มาจากการผสมกันในอัตราส่วนที่แตกต่างกันของแม่สีทั้งสามโมเดลสี RGB นี้จะแสดงด้วยแกนของลูกบาศก์สามแกนในระนาบ 3 มิติ ซึ่งค่าสีแดง เขียว และน้ำเงินจะอยู่ที่มุมทั้งสามของแต่ละแกนดังแสดงด้วยรูปที่ ซึ่งจะเห็นว่าค่า

สีดำจะอยู่ที่จุดกำเนิด (Origin) สีขาวจะอยู่ที่มุมตรงข้ามกับสีดำ ค่าของสีในช่วงระดับเทาจะอยู่ตามเส้นที่เชื่อมระหว่างค่าสีดำและค่าสีขาว จากรูปถ้าเป็นในระบบการแสดงสีแบบ 24 บิต (แบ่งออกเป็น 8 บิตต่อแม่สีหนึ่งสี) นั้น ค่าสีแดงจะถูกแทนด้วยค่า (255, 0, 0) เป็นต้น



รูปที่ 5.8 แสดงรูปลูกบาศก์สีของโมเดลสีแบบ RGB

โมเดลสี RGB ง่ายต่อการออกแบบและใช้งานในระบบคอมพิวเตอร์กราฟฟิก แต่ไม่เหมาะสำหรับประยุกต์ใช้งานอื่นๆ ค่าของสีแดง สีเขียว และสีน้ำเงิน จะมีความสัมพันธ์กันอย่างมากซึ่งจะเป็นการยากที่จะนำไปประมวลผลเกี่ยวกับภาพ ดังนั้นหลายครั้งที่มีความจำเป็นที่จะต้องทำการแปลงภาพจากโมเดลสี RGB ให้อยู่ในรูปแบบของระดับเทา (Gray scale image) เพื่อความสะดวกดังกล่าว

ในการแปลงภาพจากโมเดลสี RGB ให้อยู่ในรูปแบบภาพระดับเทานั้น สามารถทำได้โดยการใช้สมการแปลงดังนี้

$$\text{Gray scale intensity} = 0.29R + 0.587G + 0.11B \quad \ldots\ldots\ldots(4)$$

ซึ่งเป็นสมการที่ใช้สำหรับการแปลงภาพมาตรฐาน NTSC

$$\text{Gray scale intensity} = 0.333R + 0.333G + 0.333B \quad \ldots\ldots\ldots(5)$$

ซึ่งเป็นสมการที่ใช้ในการแปลงจาก RGB ไปเป็น HIS

ในระบบดิจิตอลวีดีโอนั้น เรากล่าวถึงจำนวนบิตที่ใช้การสุ่มตัวอย่างสัญญาณแต่ละครั้งเป็นตัววัดความละเอียดของภาพที่ได้เป็นจำนวนบิตต่อพิคเซล (Bit per pixel: bpp) เนื่องจากในการสุ่มตัวอย่างสัญญาณแต่ละครั้งนั้นจะได้เป็นหนึ่งพิคเซล โดยในระบบ monochrome ที่มีคุณภาพสูงนั้นจะใช้ 8 bpp ซึ่งหมายถึงจำนวนระดับเทา (gray level) ของภาพเท่ากับ 256 ระดับ แต่ในระบบการแสดงผลแบบสีนั้นเราต้องการหนึ่งช่องสัญญาณแบบ monochrome ต่อแม่สีแต่ละสี (แดง, เขียว และสีน้ำเงิน) ซึ่งจะต้องใช้จำนวนบิตทั้งหมดเป็น 24 บิต ซึ่งจะได้ค่าระดับสีที่เป็นไปได้คือ 16, 777,216 สีในระบบดิจิตอลวีดีโอบางระบบซึ่งจะใช้ 24 บิตต่อพิคเซล

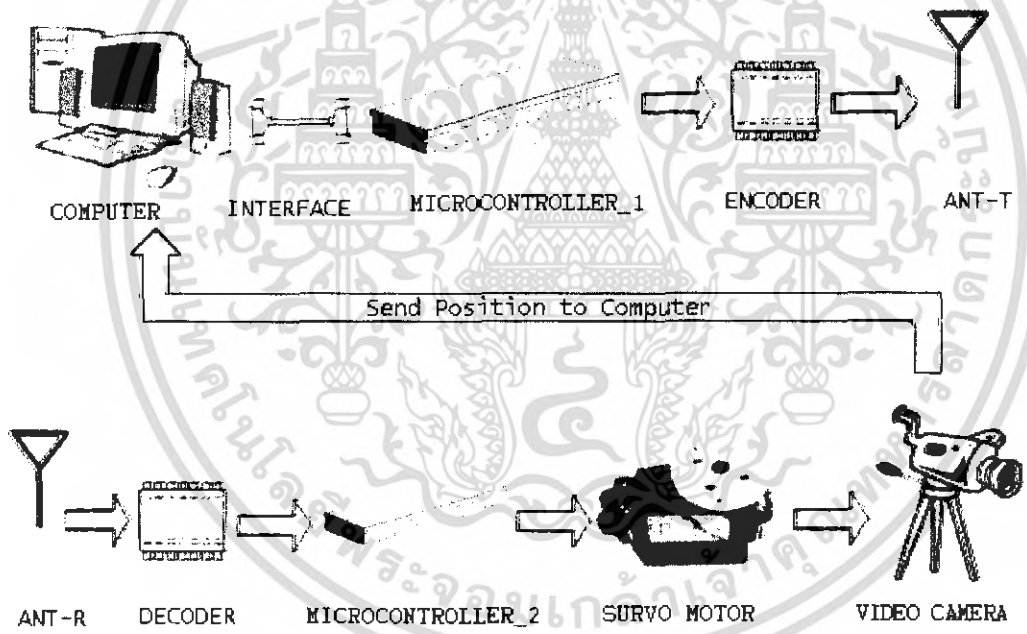
บทที่ 6
การออกแบบและการทำงานของวงจร

6.1 หลักการออกแบบของวงจร

การทำงานของวงจรคือการที่นำเอาสัญญาณจากกล้องวีดีโอ มาทำการประมวลผลภาพวัตถุ โดยเริ่มจากรับภาพเป้าหมายที่ต้องการ ผ่านมาทางกล้องวีดีโอ ส่งให้คอมพิวเตอร์ทำการเก็บค่าที่ได้แล้วทำการประมวลผลค่าภาพนั้นเพื่อหาตำแหน่งของเป้าหมาย จากนั้นคอมพิวเตอร์จะทำการส่งค่าข้อมูลที่ได้ สั่งให้ไมโครคอนโทรลเลอร์ทำการควบคุมกล้องซึ่งติดตั้งอยู่กับเซอร์โวมอเตอร์เพื่อให้หมุนติดตามเป้าหมายนั้นไป

6.2 ไดอะแกรมการทำงานของวงจร

การทำงานของวงจรกล้องติดตามเป้าหมายมีบล็อกไดอะแกรมการทำงานดังรูปที่ 6.1



รูปที่ 6.1 ไดอะแกรมการทำงานของวงจรกล้องติดตามเป้าหมาย

ความหมายของแต่ละบล็อกการทำงาน

1. กล้องวีดีโอทำหน้าที่จับภาพเป้าหมายที่ต้องการ โดยสัญญาณภาพที่ได้จะถูกส่งเข้าสู่การ์ดวีดีโอของคอมพิวเตอร์ เพื่อทำการประมวลผลภาพที่ได้ต่อไป

2. คอมพิวเตอร์ทำหน้าที่ประมวลผลภาพที่ได้จากกล้องวีดีโอ โดยใช้โปรแกรม Delph7 ช่วยในการประมวลผล ทำการวิเคราะห์ภาพเพื่อหาตำแหน่งของวัตถุ แล้วทำการส่งค่าที่ได้ไปยังส่วนควบคุมตำแหน่งต่อไป

3. ไมโครคอนโทรลเลอร์ตัวที่ 1 ทำหน้าที่รับสัญญาณ Serial ส่วนควบคุมตำแหน่งจากคอมพิวเตอร์ผ่านคอนเน็กเตอร์ DB9 หลังจากนั้นทำการเปลี่ยนสัญญาณที่ได้ให้เป็นสัญญาณ BCD ส่งค่าที่ได้ส่งออกทางพอร์ต P1 เพื่อทำการเข้ารหัสต่อไป

4. IC Encoder ทำหน้าที่นำสัญญาณที่ออกมาจากพอร์ต P1 ของไมโครคอนโทรลเลอร์ตัวที่ 1 มาเข้ารหัสก่อนแล้วส่งสัญญาณที่ได้เข้าสู่โมดูลภาคส่ง ส่งสัญญาณต่อไป

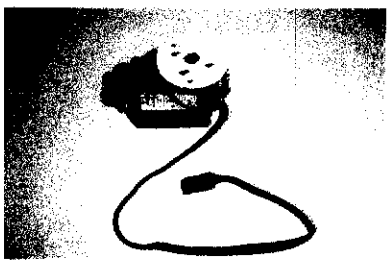
5. IC Decoder ทำหน้าที่รับสัญญาณที่ได้จากโมดูลภาครับ มาถอดรหัสสัญญาณให้เป็นสัญญาณ BCD แล้วส่งสัญญาณที่ได้ไปสู่ไมโครคอนโทรลเลอร์ตัวที่ 2 ต่อไป

6. ไมโครคอนโทรลเลอร์ตัวที่ 2 ทำหน้าที่นำสัญญาณ BCD ที่ได้จาก IC Decoder มาทำการควบคุมการทำงานของเซอร์โวมอเตอร์ทั้งสองตัวเพื่อควบคุมให้กล้องวีดีโอทำการหมุนติดตามเป้าหมายนั้นไป โดยภาพที่ได้จากกล้องวีดีโอเป้าหมายจะอยู่กึ่งกลางเสมอ

6.3 อุปกรณ์และการประกอบติดตั้ง

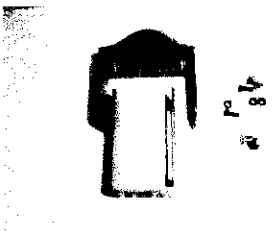
อุปกรณ์หลักๆมีดังนี้คือ

1. เซอร์โวมอเตอร์จำนวน 2 ตัว
2. กล้องวีดีโอไร้สาย
3. ตัวยึดระหว่างกล้องวีดีโอกับเซอร์โวมอเตอร์
4. เครื่องคอมพิวเตอร์และการ์ดวีดีโอ
5. สายนำสัญญาณและขั้วต่อ DB9
6. ไอซีอินเตอร์เฟส MAX 232
7. โมดูล TLP315 และ RLP315
8. ไอซีเข้ารหัส HT12E และถอดรหัส HT12D
9. ไมโครคอนโทรลเลอร์ AT89C52 และ AT89C2051

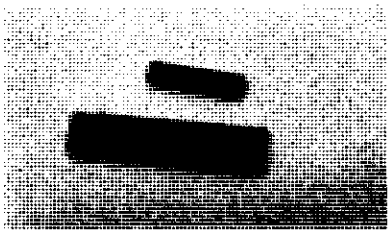


เซอร์โวมอเตอร์

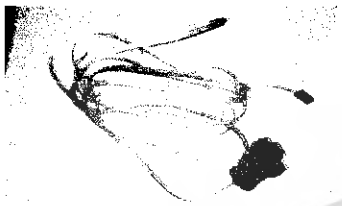
การ์ดวีดีโอ



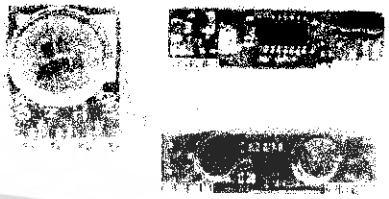
ตัวยึดเซอร์โวมอเตอร์



ไมโครคอนโทรลเลอร์



สายนำสัญญาณและขั้วต่อ DB9



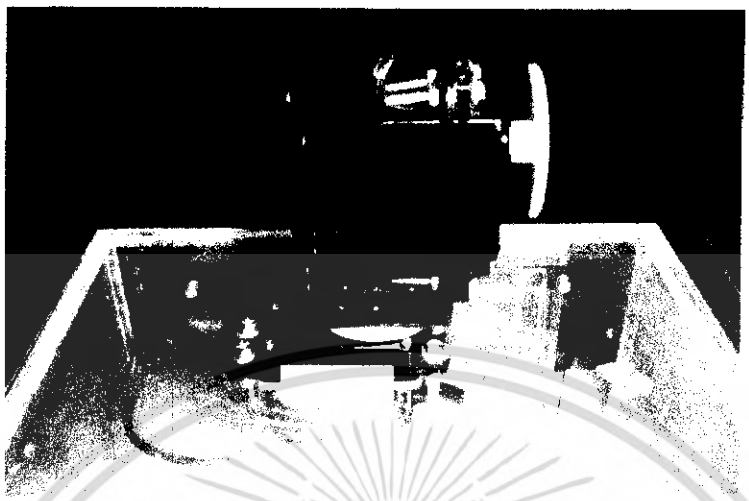
โมดูล TLP315 และ RLP315

รูปที่ 6.2 อุปกรณ์ต่างๆในการประกอบ

หลักการประกอบเริ่มโดยการนำเซอร์โวมอเตอร์ตัวที่ 1 ติดตั้งกับฐานรองที่มีขนาดพอสมควรเพื่อความมั่นคง ซึ่งเซอร์โวมอเตอร์ตัวแรกนี้จะทำหน้าที่ควบคุมการหมุนของกล้องวีดีโอในแนวแกน X-Y

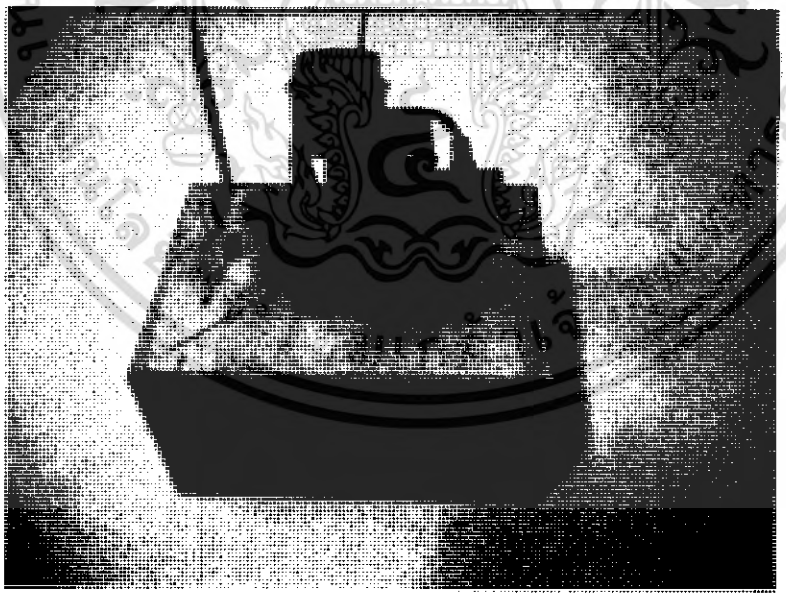
รูปที่ 6.3 เซอร์โวมอเตอร์ควบคุมกล้องในแนวแกน X-Y

ต่อจากนั้นทำการติดตั้งเซอร์โวมอเตอร์ตัวที่ 2 บนเฟืองของเซอร์โวมอเตอร์ตัวที่ 1 โดยเซอร์โวมอเตอร์ตัวที่ 2 จะทำหน้าที่ควบคุมกล้องในแนวแกน Z



รูปที่ 6.4 เซอร์โวมอเตอร์ควบคุมกล้องในแนวแกน Z

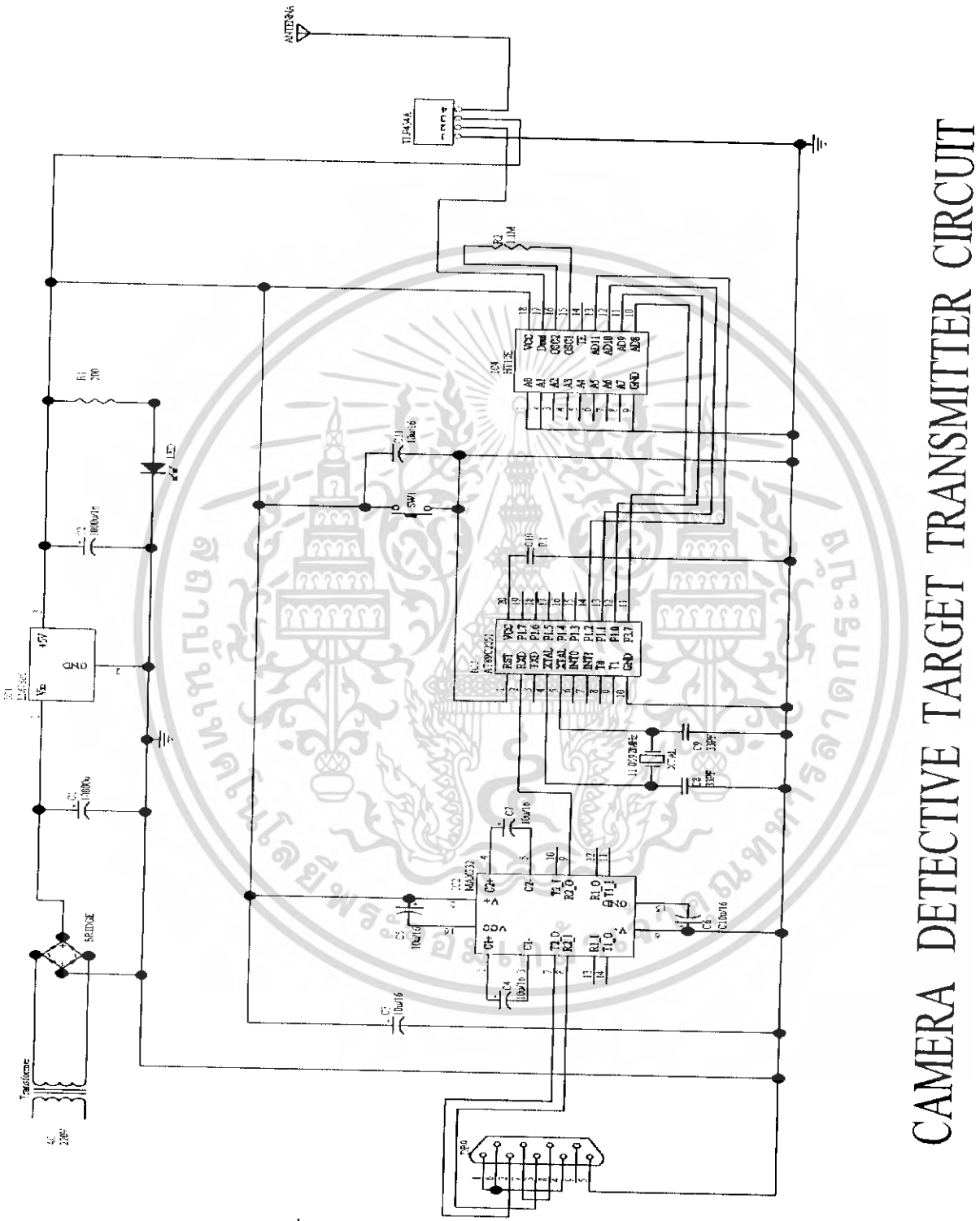
จะเห็นได้ว่าที่ยึดระหว่างกล้องวีดีโอกับเซอร์โวมอเตอร์มีขนาดเล็กเพื่อลดการสั่นสะเทือนขณะหมุนควบคุมกล้องจะทำให้ภาพที่ได้จากกล้องเป็นภาพที่สมบูรณ์ หลังจากนั้นทำการติดตั้งกล้องวีดีโอไว้บนสุดดังรูปที่ 6. 5 เมื่อติดตั้งเสร็จเห็นได้ว่ากล้องสามารถหมุนได้ครบทั้งครึ่งทรงกลมบน



รูปที่ 6.5 การติดตั้งกล้องวีดีโอ

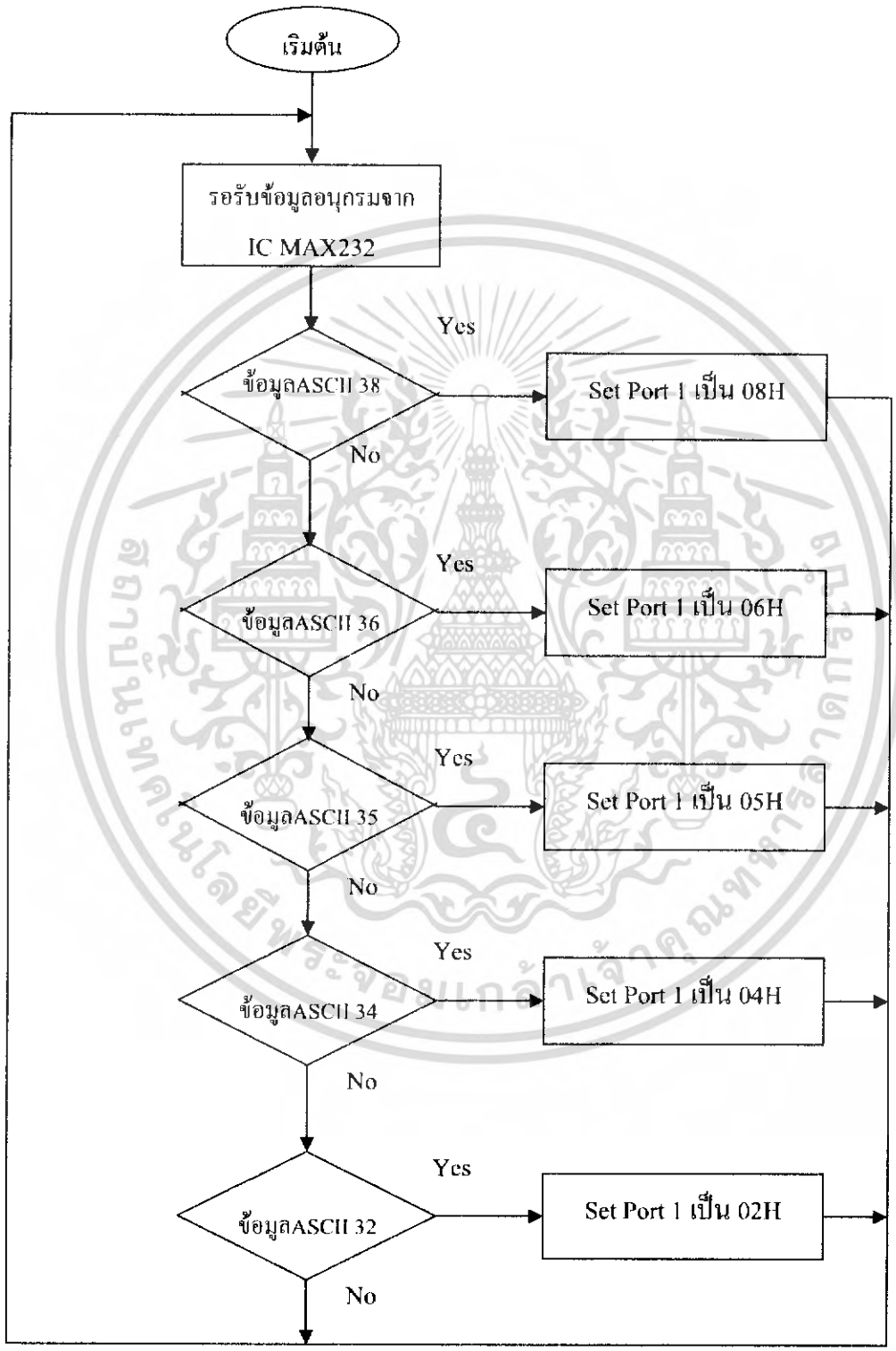
6.4 การออกแบบวงจรส่วนควบคุมตำแหน่งของกล้องวีดีโอ

เมื่อเราทำการประกอบอุปกรณ์ดังที่กล่าวมาเสร็จแล้ว เราก็มาจัดทำในส่วนควบคุมตำแหน่งของกล้องวีดีโอ โดยเริ่มจากการติดต่อระหว่างไมโครคอนโทรลเลอร์ภาคส่ง กับเครื่องคอมพิวเตอร์ โดยประกอบวงจรตามรูปที่ 6.6



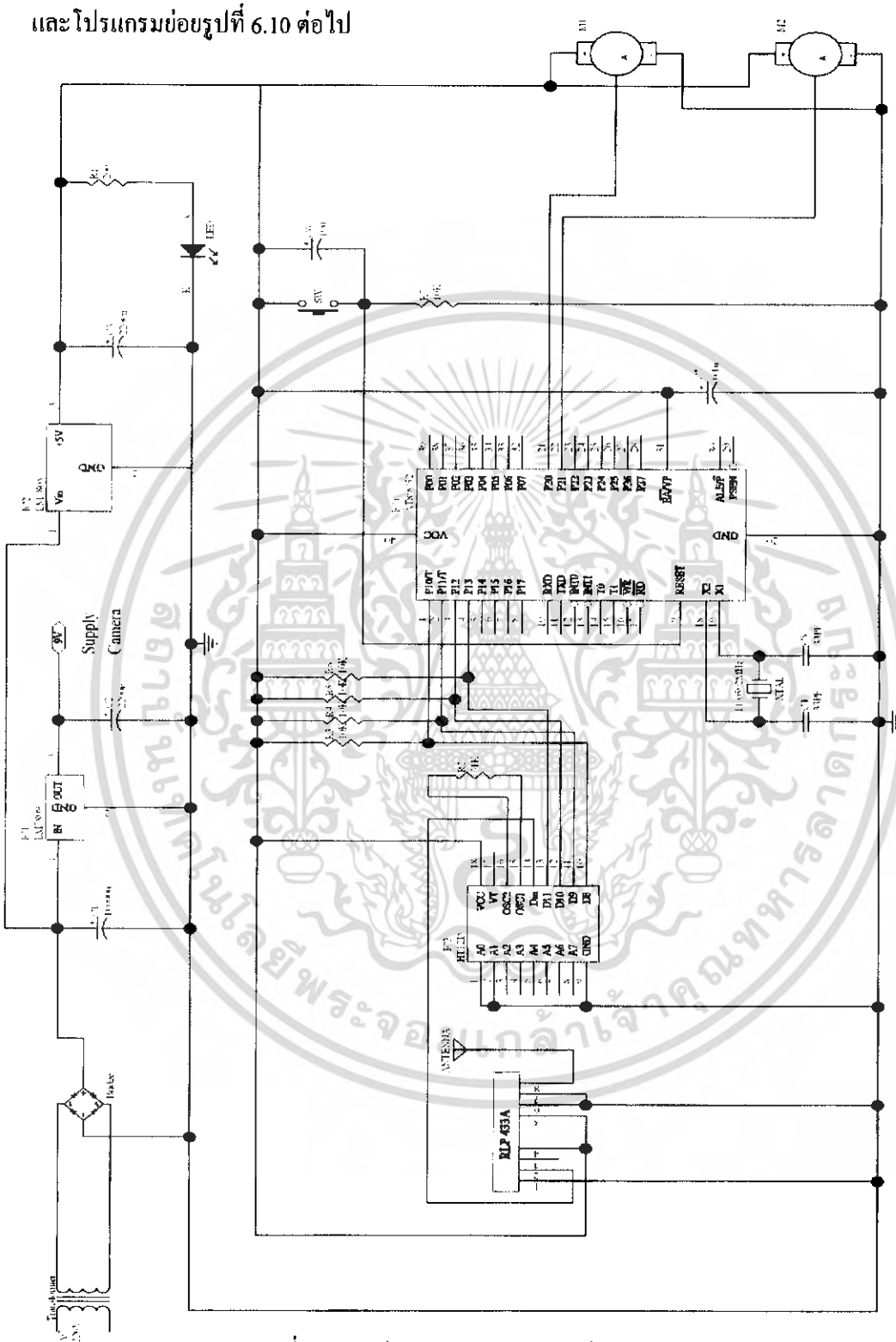
รูปที่ 6.6 วงจรไมโครคอนโทรลเลอร์ภาคส่ง

จากรูปจะเห็นได้ว่าในวงจรไมโครคอนโทรลเลอร์ภาคส่งจะประกอบด้วย ย่อยๆอีกหลายวงจรคือ วงจรภาคจ่ายไฟ 5V วงจรอินเตอร์เฟส วงจรไมโครคอนโทรลเลอร์ วงจรเข้ารหัส และโมดูลภาคส่งสัญญาณ RF หลังจากนั้นทำการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ภาคส่ง โดยมีการทำงานดัง Flow Chart รูปที่ 6.7

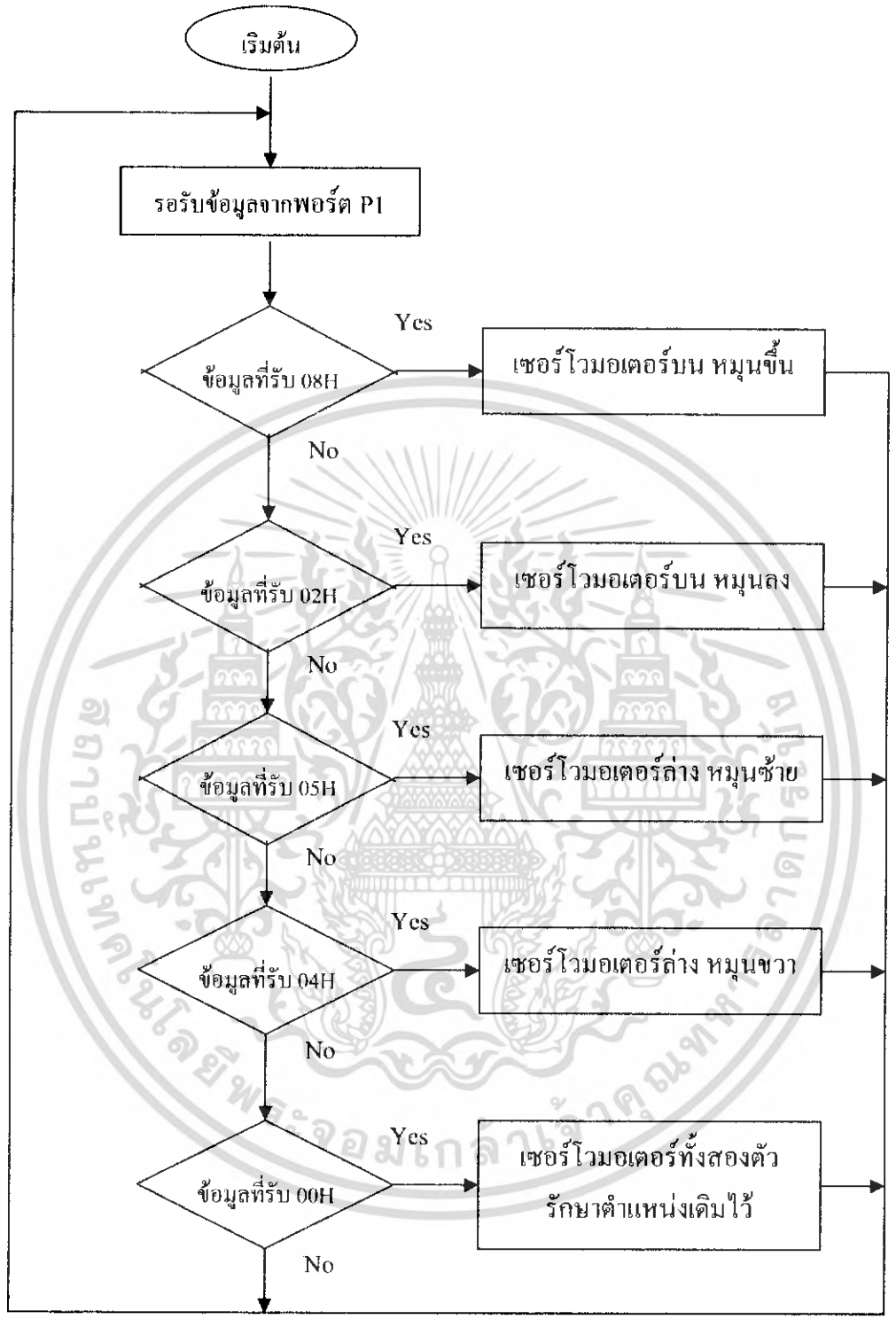


รูปที่ 6.7 โปรมแกรมการทำงานของไมโครคอนโทรลเลอร์ ภาคส่ง

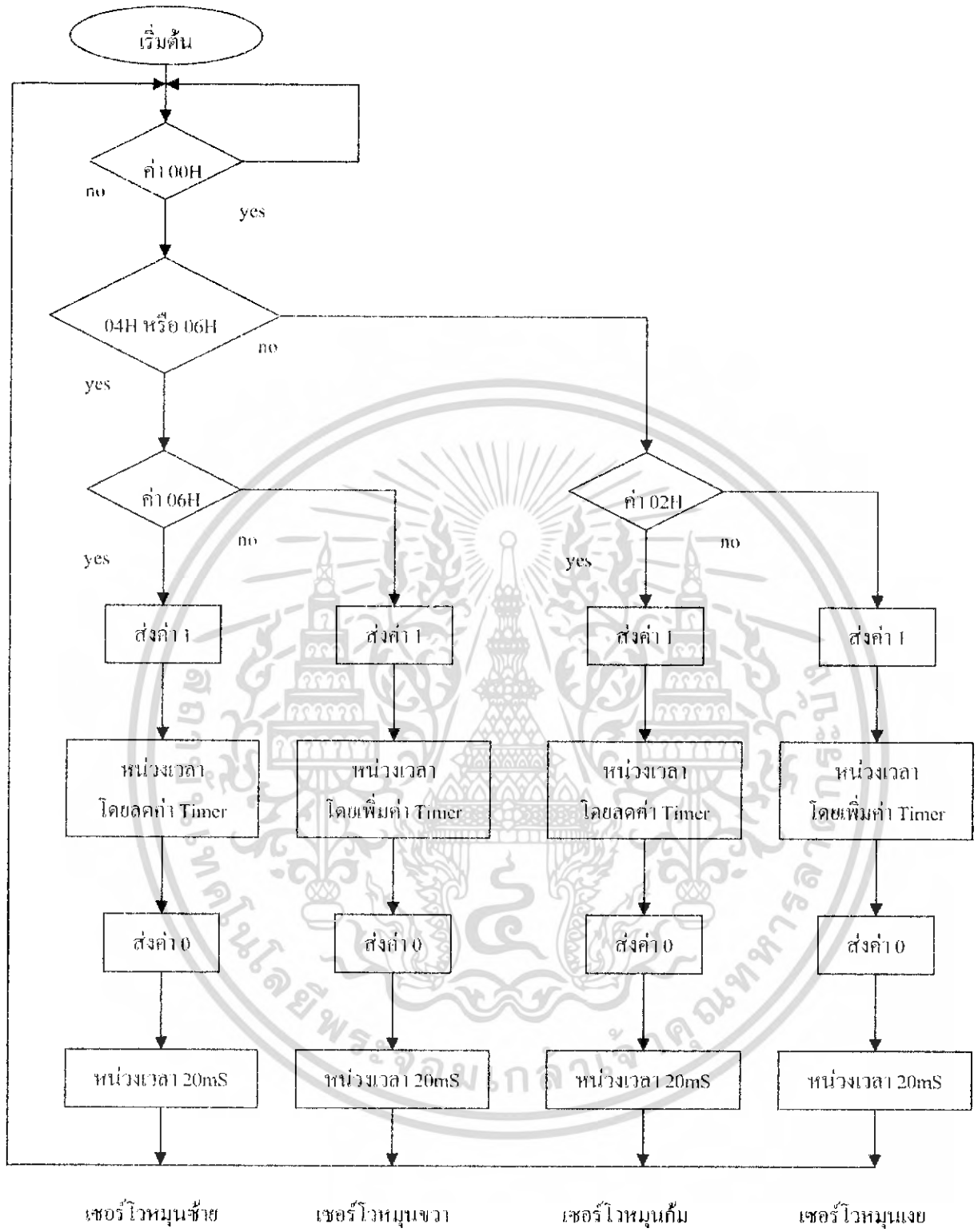
จากนั้นทำการประกอบวงจรไมโครคอนโทรลเลอร์ภาครับตามรูปที่ 6.8 และเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ภาครับไมโครคอนโทรลเลอร์ภาครับจะทำหน้าที่ควบคุมการทำงานของเซอร์โวมอเตอร์ทั้งสองตัวโดยเขียนโปแกรมควบคุมหลักดังรูปที่ 6.9 และโปรแกรมย่อยรูปที่ 6.10 ต่อไป



รูปที่ 6.8 วงจรไมโครคอนโทรลเลอร์ภาครับ



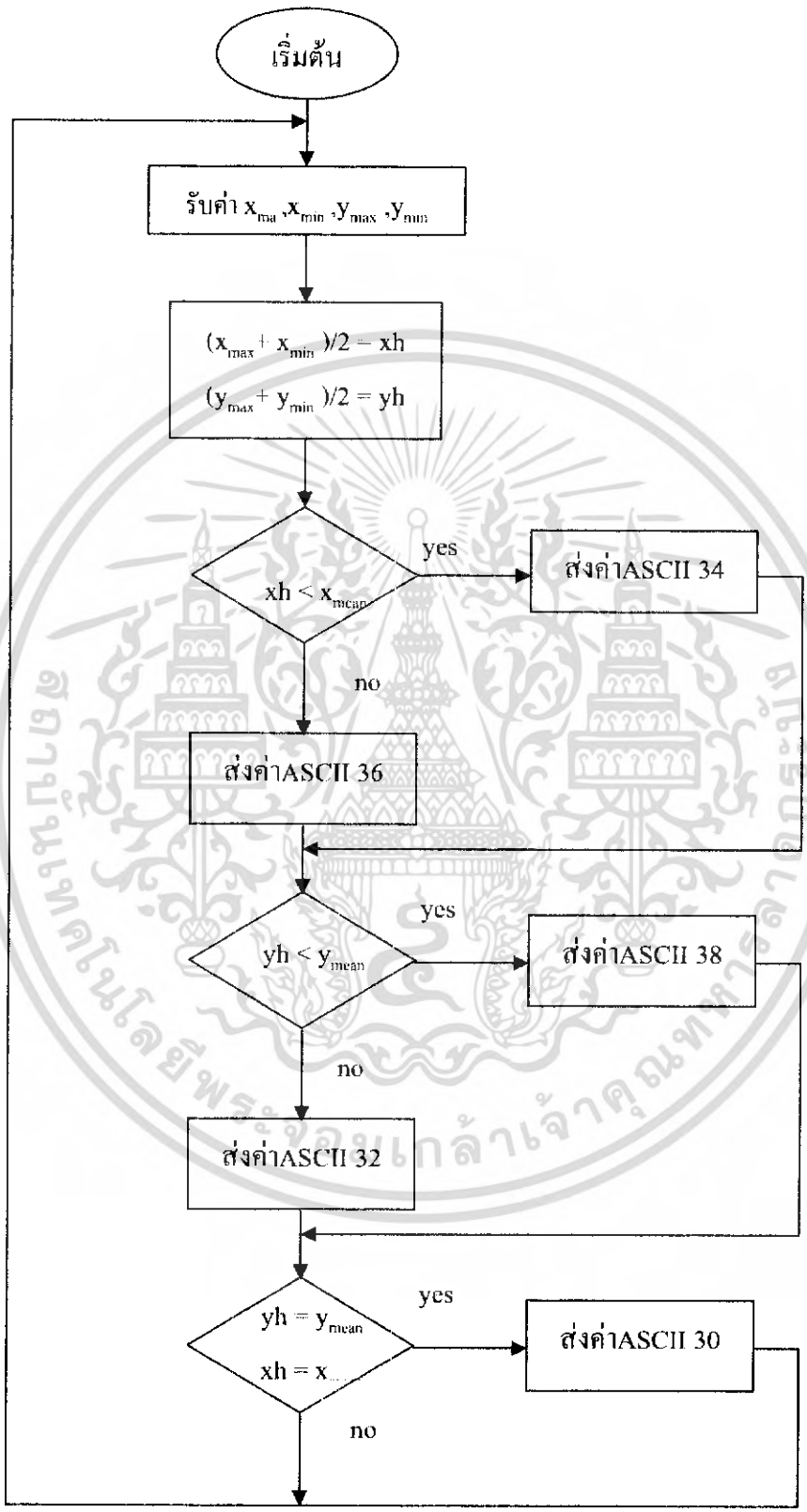
รูปที่ 6.9 โปรมแกรมการทำงานหลักของไมโครคอนโทรลเลอร์ ภาครับ



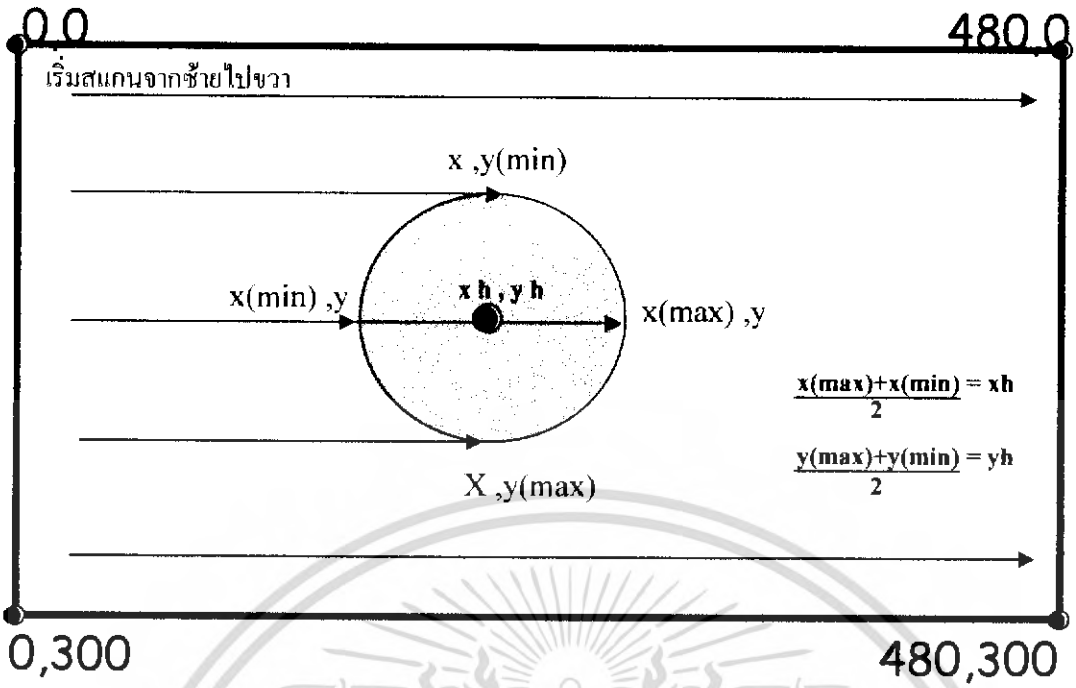
รูปที่ 6.10 โปรแกรมย่อยควบคุมการหมุนของเซอร์โวมอเตอร์

6.5 การออกแบบการเขียนโปรแกรมประมวลผลภาพ

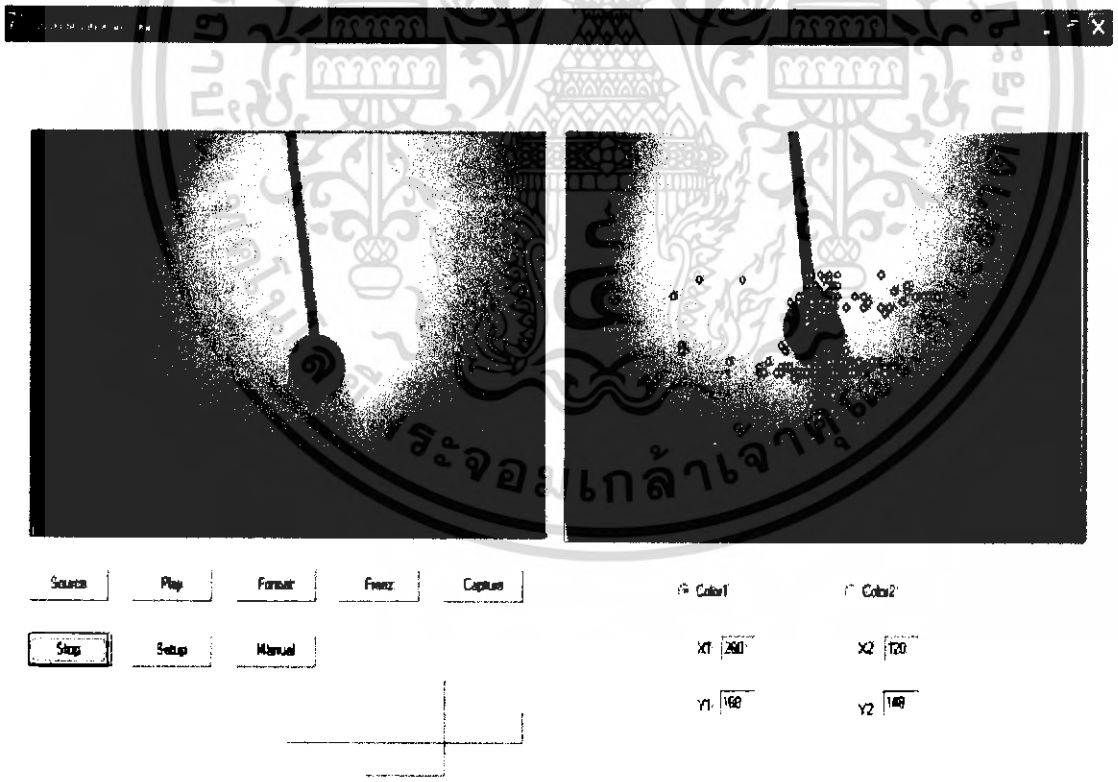
สามารถเขียนโปแกรมการประมวลผลภาพได้ดังรูปโปรแกรมด้านล่าง



รูปที่ 6.11 รูปโปรแกรมการประมวลผลภาพ



รูปที่ 6.12 ภาพหลักการสแกนหาตำแหน่งกึ่งกลางของวัตถุ



รูปที่ 6.13 รูปหน้าต่างของโปรแกรมประมวลผลภาพ

บทที่ 7

การทดลอง

7.1 การทดลองวัดสัญญาณที่เข้ารหัสและถอดรหัสเพื่อควบคุมการทำงาน

7.1.1 ทำการวัดสัญญาณที่เข้ามาที่ไมโครคอนโทรลเลอร์ตัวที่1

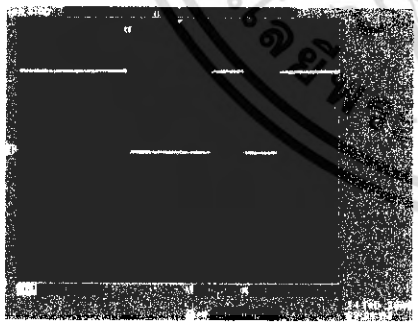
7.1.2 ทำการวัดสัญญาณที่ออกจาก Encoder

7.1.3 ทำการวัดสัญญาณที่ออกจาก Decoder

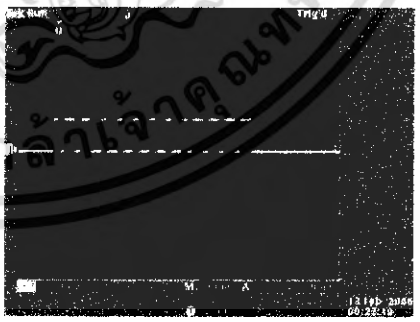
7.1.4 บอกสถานการณ์ทำงานของเซอร์โวมอเตอร์

ขาที่ 2 ของ AT89C2051 (ASCII)	ขาที่ 17 ของ HT12E (Binary)	ขาที่ 10 – 13 ของ HT12D (Binary)	สถานการทำงาน
38	1001111110001	1 0 0 0	เซอร์โวมอเตอร์ตัวบนหมุนขึ้น
36	1001111110110	0 1 1 0	เซอร์โวมอเตอร์ตัวบนล่างหมุนขวา
34	1001111110010	0 1 0 0	เซอร์โวมอเตอร์ตัวบนล่างหมุนซ้าย
32	1001111110100	0 0 1 0	เซอร์โวมอเตอร์ตัวบนหมุนลง
30	1001111110000	0 0 0 0	เซอร์โวมอเตอร์ทั้งสองตัวหยุด

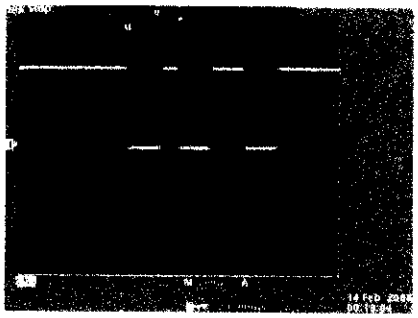
ตารางที่ 7.1 สัญญาณและสถานการทำงาน



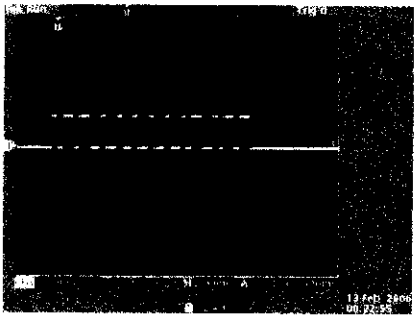
ASCII 30



Binary 1001111110000

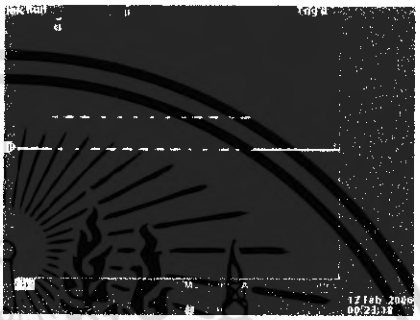


ASCII 32



Binary 1001111110100

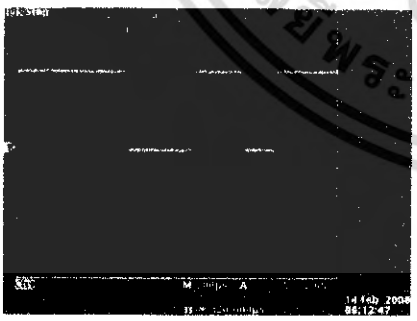
ASCII 34



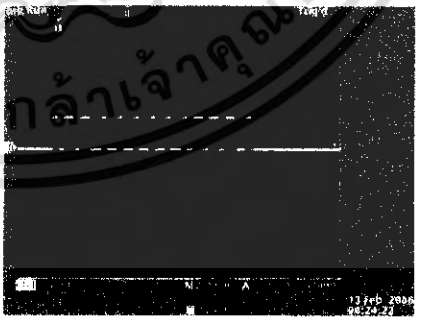
Binary 1001111110010

ASCII 36

Binary 1001111110110



ASCII 38



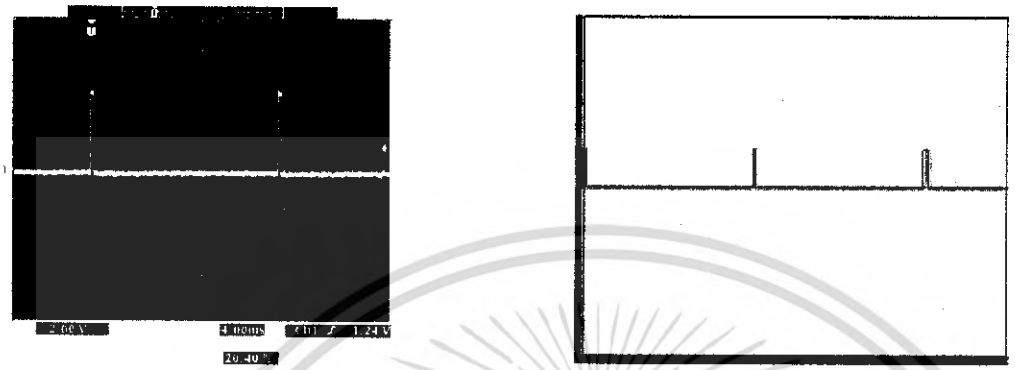
Binary 1001111110001

รูปที่ 7.1 รูปสัญญาณการเข้ารหัสของวงจร

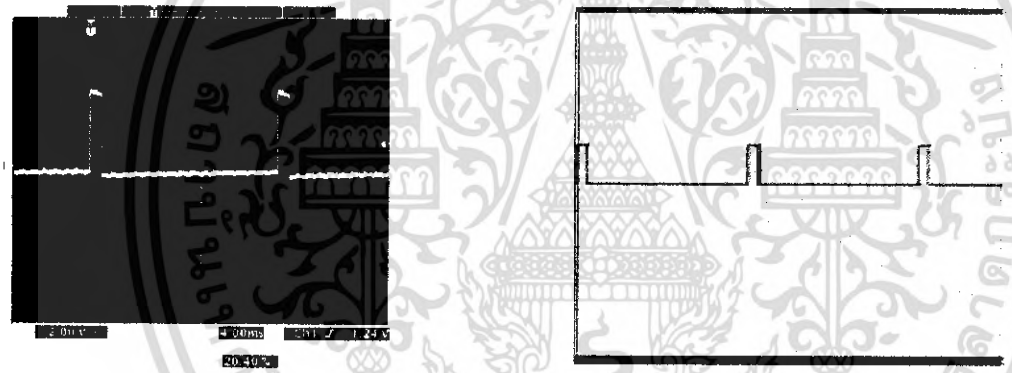
7.2 การทดลองวัดสัญญาณที่ใช้ในการควบคุมเซอร์โวมอเตอร์

7.2.1 ทำการวัดสัญญาณที่ขา P2.1 และ P2.2 ของไมโครคอนโทรลเลอร์ตัวที่ 2

7.2.2 ผลการทดลองรูปสัญญาณที่ได้จากการทดลองในส่วนควบคุมตำแหน่ง กล้องวีดีโอเทียบกับสัญญาณที่ได้จากการ Simulated



ก. แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งซ้ายสุด

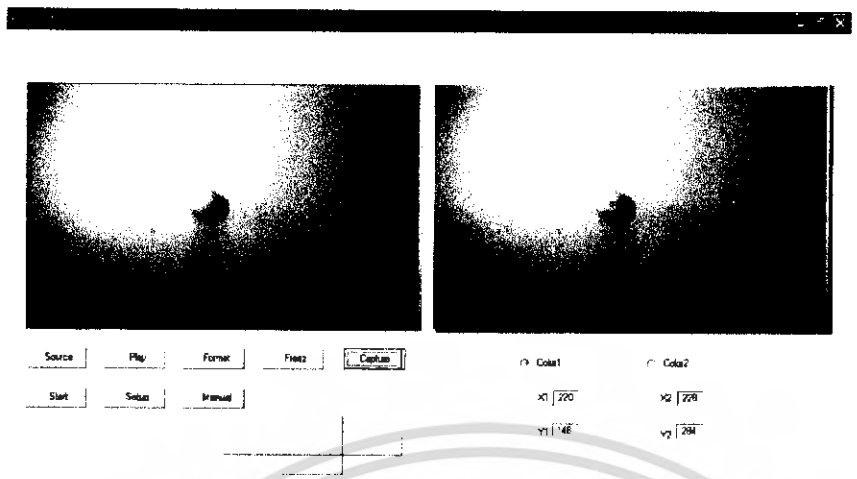


ข. แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งตรงกลาง

ค. แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งขวาสุด

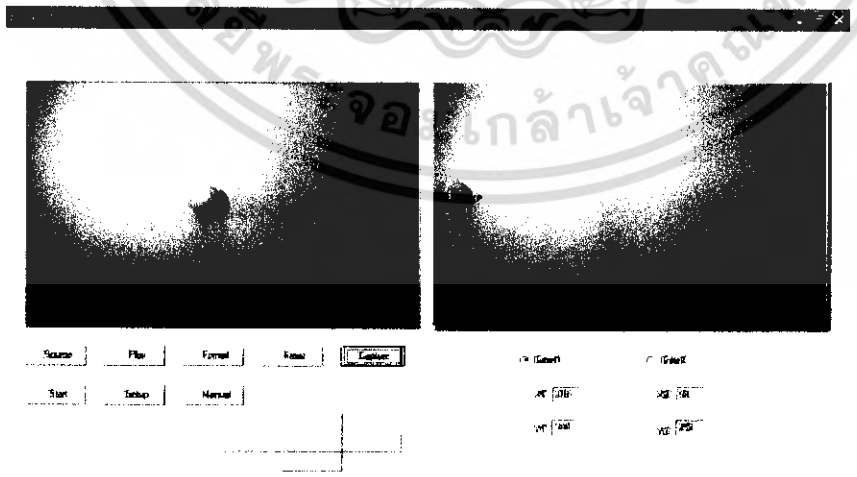
รูปที่ 7.2 แสดงพัลส์ที่ทำให้มอเตอร์หมุนไปตำแหน่งต่างๆ

7.3 การทดลองจับเวลาในการเคลื่อนที่ของกล้องตามวัตถุ



ก. กล้องที่ตำแหน่งเริ่มต้น

ข. กล้องเริ่มเคลื่อนที่ตามวัตถุ



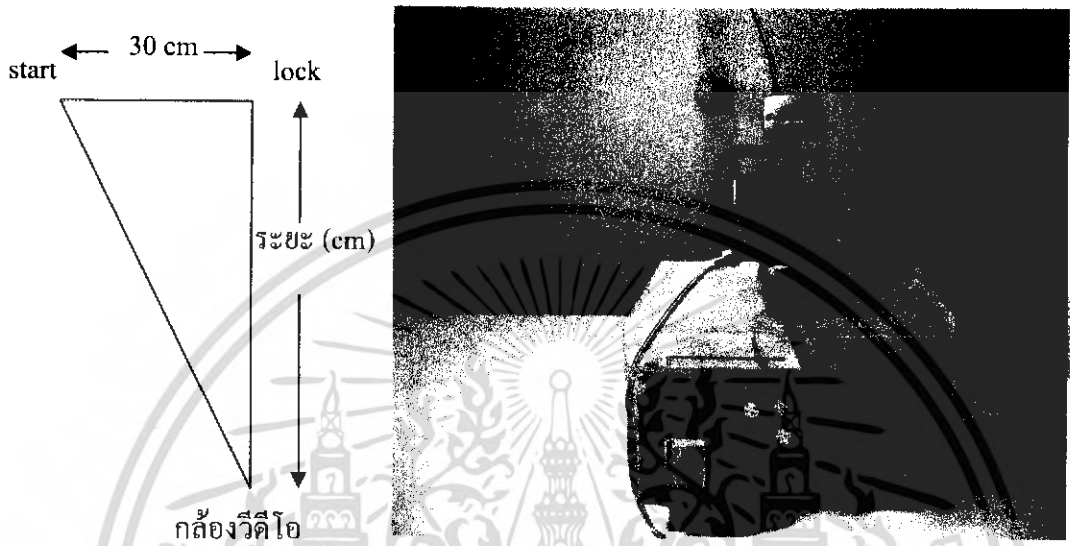
ค. กล้องเคลื่อนที่จนถึงตำแหน่ง lock

รูปที่ 7.3 การทดลองจับเวลาในการเคลื่อนที่ของกล้องตามวัตถุ

7.3.1 ทำการจับเวลาการเคลื่อนที่ของกล้องในขณะที่กล้องเห็นวัตถุจากตำแหน่งเริ่มต้นจนถึงตำแหน่ง Lock

7.3.2 หาค่าเฉลี่ยของเวลาที่กล้องใช้ในการตามวัตถุ

7.3.3 ทำการวัดระยะห่างของวัตถุ กล้อง และฉากหลัง



รูปที่ 7.4 แสดงระยะห่างของกล้อง และวัตถุ

จำนวนครั้ง	1	2	3	4	5	6	7	8	9	10	ค่าเฉลี่ย
เวลา (วินาที)	1.36	1.58	1.57	1.50	1.30	1.45	1.44	1.72	1.32	1.45	1.46

ก. ระยะห่างของกล้องกับวัตถุ 40 cm.

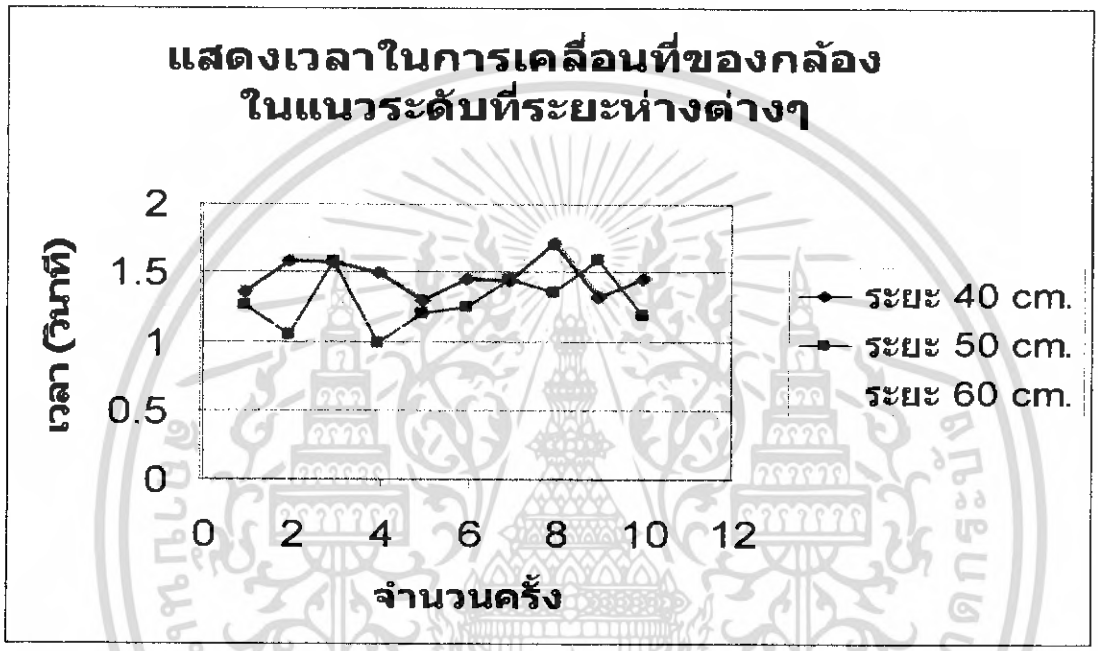
จำนวนครั้ง	1	2	3	4	5	6	7	8	9	10	ค่าเฉลี่ย
เวลา (วินาที)	1.27	1.05	1.58	0.99	1.21	1.25	1.45	1.36	1.60	1.20	1.31

ข. ระยะห่างของกล้องกับวัตถุ 50 cm.

จำนวนครั้ง	1	2	3	4	5	6	7	8	9	10	ค่าเฉลี่ย
เวลา (วินาที)	0.97	1.19	0.90	1.23	1.00	0.95	0.80	0.99	1.13	0.99	1.01

ค. ระยะห่างของกล้องกับวัตถุ 60 cm.

ตารางที่ 7.2 แสดงค่าเวลาที่กล้องเคลื่อนที่ในแนวระดับที่ระยะต่างๆ



รูปที่ 7.5 กราฟแสดงค่าเวลาที่กล้องเคลื่อนที่ในแนวระดับที่ระยะต่างๆ

ในการทดลองการจับเวลาการเคลื่อนที่ของกล้องจากตำแหน่งเริ่มต้นที่กล้องเริ่มเห็นวัตถุจนถึงตำแหน่งที่วัตถุอยู่กึ่งกลางจอภาพโดยทำกำหนดระยะในการเคลื่อนที่ให้มีค่าเท่ากับ 30 cm. ในการทดลองที่ระยะของกล้องห่างจากวัตถุ 40 cm. เวลาที่ใช้ในการเคลื่อนที่เฉลี่ยเท่ากับ 1.46 วินาที ที่ระยะของกล้องห่างจากวัตถุ 50 cm. เวลาที่ใช้ในการเคลื่อนที่เฉลี่ยเท่ากับ 1.31 วินาที ส่วนที่ระยะของกล้องห่างจากวัตถุ 60 cm. เวลาที่ใช้ในการเคลื่อนที่เฉลี่ยเท่ากับ 1.01 วินาที

บทที่ 8
สรุปและวิจารณ์

8.1 สิ่งที่ได้ดำเนินการในภาคการศึกษาที่ 1

1. ส่วนของโปรแกรม

เขียนโปรแกรมไมโครคอนโทรลเลอร์ควบคุมการเคลื่อนที่ของเซอร์โวมอเตอร์ โดยการบังคับการควบคุมด้วยมือผ่านทางคอมพิวเตอร์

2. ส่วนของวงจร

ประกอบวงจรไมโครคอนโทรลเลอร์รับค่าจากคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมมาทำการควบคุมเซอร์โวมอเตอร์

3. ส่วนของตัวกล้อง

ทำฐานยึดกล้องในแนวแกน Z และในแนวระนาบ XY

8.2 สิ่งที่ได้ดำเนินการในภาคการศึกษาที่ 2

1. ส่วนของโปรแกรม

- เขียนโปรแกรมในส่วนการประมวลผลภาพ โดยการกำหนดวัตถุเป้าหมาย เมื่อวัตถุเป้าหมายเคลื่อนที่ มอเตอร์เซอร์โวจะหมุนตามวัตถุเป้าหมายที่เคลื่อนที่โดยอัตโนมัติ ซึ่งส่วนของวัตถุเป้าหมายจะต้องมีสีที่แตกต่างจากพื้นหลังอย่างเห็นได้ชัด
- เขียนโปรแกรมไมโครคอนโทรลเลอร์เพิ่มเติมอีกหนึ่งตัวเพื่อทำการเข้ารหัสสัญญาณ แก้ไขโปรแกรมเดิมบางส่วนจากการรับค่าข้อมูลอนุกรมมาเป็นการรับค่าข้อมูลแบบขนานในการควบคุมเซอร์โวมอเตอร์
- เพิ่มความละเอียดของมุมในการเคลื่อนที่ของเซอร์โวมอเตอร์ให้มากขึ้น

2. ส่วนของวงจร

- ทำการเปลี่ยนวงจรเป็นการสื่อสารข้อมูล จากคอมพิวเตอร์ไปยังส่วนควบคุมการเคลื่อนที่ของเซอร์โวมอเตอร์โดยผ่านทางความถี่เพื่อความสะดวกและเพิ่มระยะทาง

3. ส่วนของตัวกล้อง

- เปลี่ยนตัวกล้องวีดีโอธรรมดาเป็นกล้องวีดีโอแบบไร้สายเพื่อความสะดวกและเพิ่มระยะทาง
- ฐานของตัวกล้องมีความแข็งแรงมากกว่าเทอมที่ 1

8.3 ปัญหาที่เกิดจากการทดลอง และ แนวทางแก้ไข

จากการทดลอง แบ่งการทำงานหลักๆออกเป็น 2 ส่วน คือ

1. การทำงานในส่วนของ เซอร์โวมอเตอร์ ซึ่งในการทำงาน ของส่วนนี้ปัญหาที่เกิดคือ การควบคุมหรือการกำหนดค่าความเร็วให้สอดคล้องกับการทำงานในส่วนของ โปรแกรมคอมพิวเตอร์ ซึ่งในการรับส่งค่าการทำงาน จะทำการส่งค่าโดย ส่งผ่านการทำงานทางความถี่ ซึ่งทำงานสอดคล้องกับไมโครคอนโทรลเลอร์

ในการทดลองนี้ใช้ไมโครคอนโทรลเลอร์ 2 ตัว ในการรับและส่งข้อมูล ปัญหาที่เกิดคือ ในการส่งผ่านความถี่นั้น เกิดค่าความผิดพลาดในการรับส่งข้อมูล ทำให้การรับส่งเกิดข้อผิดพลาด ในส่วนนี้ได้ทำแก้ไขโดยอาศัยการรับส่งในระยะทางที่ใกล้ๆ และปราศจากสิ่งกีดขวาง และใช้เสาอากาศในการรับส่งที่มีประสิทธิภาพมากขึ้น ซึ่งเมื่อทำการแก้ไข ทำให้การรับส่งสามารถทำได้ดีขึ้น และเกิดข้อผิดพลาดน้อยลงในการรับส่งข้อมูลผ่านทางความถี่ ซึ่งความถี่ที่ใช้คือ 433 MHz และเมื่อนำส่วนนี้ไปเชื่อมโยงกับการทำงานของคอมพิวเตอร์ ทำให้ระบบการทำงานของกล้องวีดีโอในการติดตามวัตถุมีเสถียรภาพมากขึ้น

2.การทำงานในส่วนของโปรแกรมคอมพิวเตอร์ ซึ่งในส่วนนี้ เราได้ใช้ โปรแกรม Delphi ในการประมวลผลซึ่งในส่วนนี้ปัญหาที่เกิดคือ การประมวลผลภาพเกิดข้อผิดพลาด ซึ่งเกิดจากแสงที่ตกกระทบ วัตถุ มีความเข้มแสงที่ไม่เท่ากัน ทำให้ในขณะที่ทำการติดตามวัตถุอยู่นั้น และกล้องวีดีโอที่ส่งภาพมาประมวลผลได้ภาพที่ต่างกันอยู่ตลอดเวลาทำให้เกิดความผิดพลาดได้ง่ายในการประมวลผลภาพ แนวทางในการแก้ไขในส่วนนี้คือ ทำการส่องแสงไฟให้สว่างขึ้น และให้แสงที่คงที่อยู่ตลอดเวลา ทำให้ลดข้อผิดพลาดในการประมวลผลภาพลงได้ และอีกส่วนคือการที่กล้องวีดีโอทำการส่งค่าได้ช้ากว่าการประมวลผล เพราะกล้องวีดีโอต้องส่งค่าผ่านการทำงานของการ์ดวีดีโอทำให้กล้องวีดีโอติดตามวัตถุได้ช้าลง แนวทางแก้ไขในส่วนนี้คือ เพิ่มองศาของเซอร์โวมอเตอร์เพื่อ ให้การหมุนไปข้างหน้าเร็วขึ้นและทำการติดตามวัตถุได้ทัน และลดค่าการสแกนจุดในโปรแกรมคอมพิวเตอร์ให้มีค่าการสแกนที่น้อยลงทำให้ในการประมวลผลภาพมีความเร็วมากขึ้น และติดตามวัตถุได้ทันในการติดตามวัตถุที่ความเร็วไม่สูงมาก

หนังสืออ้างอิง

1. วรพจน์ กรแก้ววัฒนกุล , ชัยวัฒน์ ลิ้มพรจิตรวิไล , " เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 " , บริษัทอินโนเวตีฟ เอ็กเพอร์ริเมนต์ จำกัด
2. วัชรินทร์ เคารพ , " คู่มือการใช้งานเซอร์โวมอเตอร์ " , บริษัท ETT จำกัด,2546
3. อรรถพล บุญยะโภคา , วรพจน์ กรแก้ววัฒนกุล , ชัยวัฒน์ ลิ้มพรจิตรวิไล , "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม", บริษัทอินโนเวตีฟ เอ็กเพอร์ริเมนต์ จำกัด

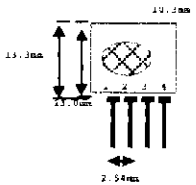
ภาคผนวก

ก. รายละเอียดของไอซีที่ใช้งาน

TLP434A & RLP434A RF ASK Hybrid Modules for Radio Control (New Version)

TLP434A Ultra Small Transmitter

Easy-Link Wireless



pin 1 : GND
pin 2 : Data In
pin 3 : Vcc
pin 4 : Antenna (RF output)

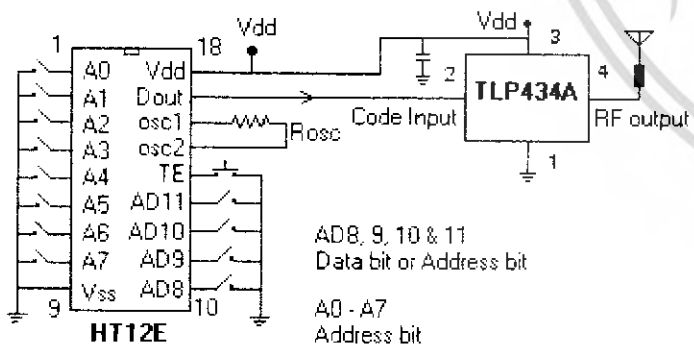
Frequency 315, 418 and 433.92 Mhz

Modulation : ASK
Operation Voltage : 2 - 12 VDC

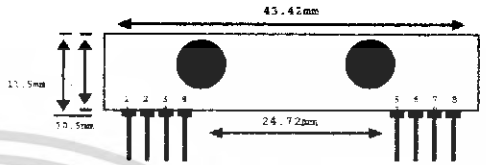
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating supply voltage		2.0	-	12.0	V
Icc 1	Peak Current (2V)		-	-	1.64	mA
Icc 2	Peak Current (12V)		-	-	19.4	mA
Vh	Input High Voltage	Idata= 100uA (High)	Vcc-0.5	Vcc	Vcc+0.5	V
Vl	Input Low Voltage	Idata= 0 uA (Low)	-	-	0.3	V
FO	Absolute Frequency	315Mhz module	314.8	315	315.2	MHz
PO	RF Output Power- 50ohm	Vcc = 9V-12V	-	16	-	dBm
		Vcc = 5V-6V	-	14	-	dBm
DR	Data Rate	External Encoding	512	4.8K	200K	bps

Notes : (Case Temperature = 25°C +- 2°C , Test Load Impedance = 50 ohm)

Application Circuit :
Typical Key-chain Transmitter using HT12E-18DIP, a Binary 12 bit Encoder from Holtek Semiconductor Inc.



RLP434A SAW Based Receiver



pin 1 : Gnd
pin 2 : Digital Data Output
pin 3 : Linear Output /Test
pin 4 : Vcc
pin 5 : Vcc
pin 6 : Gnd
pin 7 : Gnd
pin 8 : Antenna

Frequency 315, 418 and 433.92 Mhz

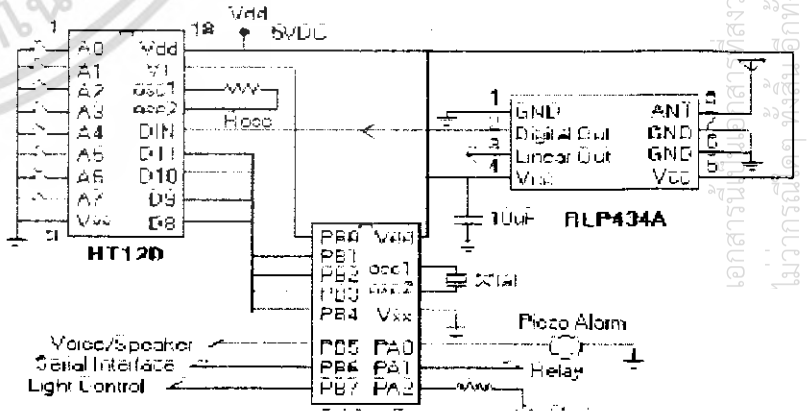
Modulation : ASK
Supply Voltage : 3.3 - 6.0 VDC
Output : Digital & Linear

Symbol	Parameter	Conditions	Min	Typ	Max	
Vcc	Operating supply voltage		3.3	5.0V	6.0	V
Itot	Operating Current		-	4.5		mA
Vdata	Data Out	Idata = -200 uA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 uA (Low)	-	-	0.3	V

Electrical Characteristics

Characteristics	SYM	Min	Typ	Max	Unit
Operation Radio Frequency	FC		315, 418 and 433.92		MHz
Sensitivity	Pref		-110		dBm
Channel Width			+-500		Khz
Noise Equivalent BW			4		Khz
Receiver Turn On Time			5		ms
Operation Temperature	Top	-20	-	80	C
Baseboard Data Rate			4.8		KHz

Application Circuit :
Typical RF Receiver using HT12D-18DIP, a Binary 12 bit Decoder with 8 bit uC HT48RXX from Holtek Semiconductor Inc.



HOLTEK

HT12A/HT12E
2^{12} Series of Encoders

Features

- Operating voltage
  - 2.4V~5V for the HT12A
  - 2.4V~12V for the HT12E
- Low power and high noise immunity CMOS technology
- Low standby current: 0.1μA (typ.) at V_{DD}=5V
- HT12A with a 38kHz carrier for infrared transmission medium
- Minimum transmission word
  - Four words for the HT12E
  - One word for the HT12A
- Built-in oscillator needs only 5% resistor
- Data code has positive polarity
- Minimal external components
- HT12A/E: 18-pin DIP/20-pin SOP package

Applications

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers
- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

General Description

The 2^{12} encoders are a series of CMOS LSIs for remote control system applications. They are capable of encoding information which consists of N address bits and 12–N data bits. Each address/data input can be set to one of the two logic states. The programmed addresses/data are transmitted together with the header bits via an RF or an infrared transmission medium upon receipt of a trigger signal. The capability to select a $\overline{TE}$ trigger on the HT12E or a DATA trigger on the HT12A further enhances the application flexibility of the 2^{12} series of encoders. The HT12A additionally provides a 38kHz carrier for infrared systems.

Selection Table

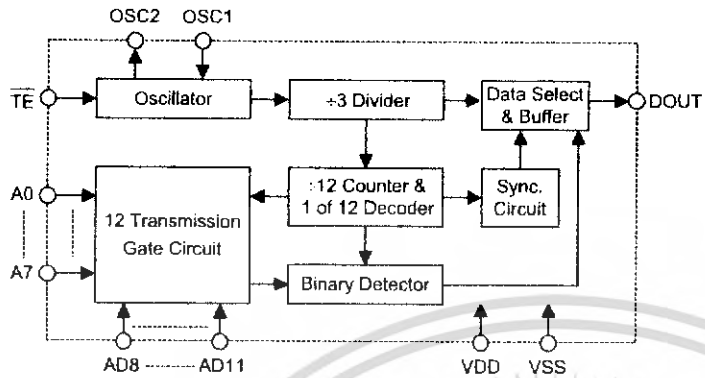
Function / Part No.	Address No.	Address/ Data No.	Data No.	Oscillator	Trigger	Package	Carrier Output	Negative Polarity
HT12A	8	0	4	455kHz resonator	D8~D11	18 DIP 20 SOP	38kHz	No
HT12E	8	4	0	RC oscillator	$\overline{TE}$	18 DIP 20 SOP	No	No

Note: Address/Data represents pins that can be address or data according to the decoder requirement.

Block Diagram

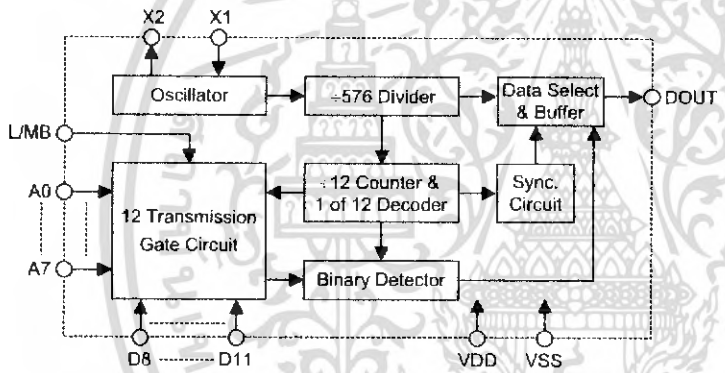
$\overline{TE}$ trigger

HT12E



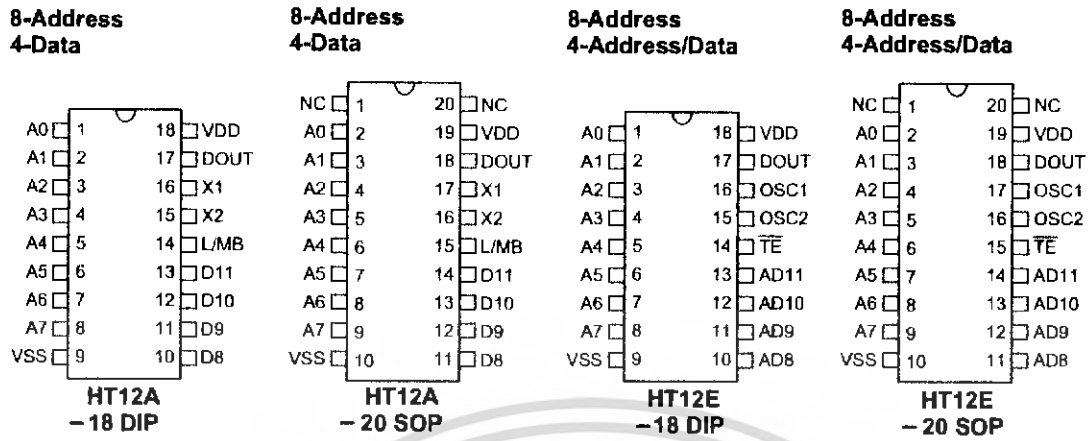
DATA trigger

HT12A



Note: The address data pins are available in various combinations (refer to the address/data table).

Pin Assignment



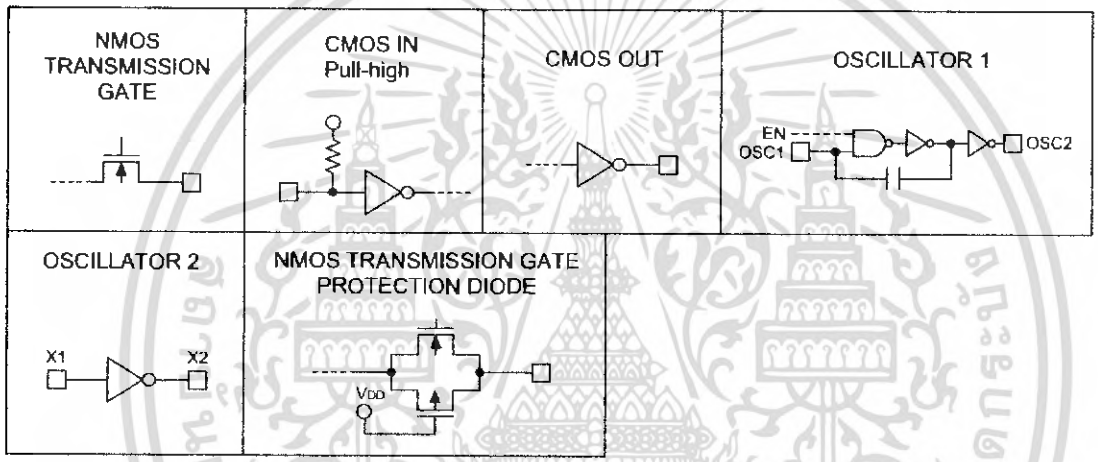
Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A7	I	CMOS IN Pull-high (HT12A) NMOS TRANSMISSION GATE PROTECTION DIODE (HT12E)	Input pins for address A0~A7 setting These pins can be externally set to VSS or left open
AD8~AD11	I	NMOS TRANSMISSION GATE PROTECTION DIODE (HT12E)	Input pins for address/data AD8~AD11 setting These pins can he externally set to VSS or left open
D8~D11	I	CMOS IN Pull-high	Input pins for data D8~D11 setting and transmission enable, active low These pins should be externally set to VSS or left open (see Note)
DOUT	O	CMOS OUT	Encoder data serial transmission output
L/MB	I	CMOS IN Pull-high	Latch/Momentary transmission format selection pin: Latch: Floating or VDD Momentary: VSS

Pin Name	I/O	Internal Connection	Description
$\overline{TE}$	I	CMOS IN Pull-high	Transmission enable, active low (see Note)
OSC1	I	OSCILLATOR 1	Oscillator input pin
OSC2	O	OSCILLATOR 1	Oscillator output pin
X1	I	OSCILLATOR 2	455kHz resonator oscillator input
X2	O	OSCILLATOR 2	455kHz resonator oscillator output
VSS	I	—	Negative power supply, grounds
VDD	I	—	Positive power supply

Note: D8~D11 are all data input and transmission enable pins of the HT12A.
$\overline{TE}$ is a transmission enable pin of the HT12E.

Approximate internal connections



Absolute Maximum Ratings

Supply Voltage (HT12A) .....–0.3V to 5.5V
Input Voltage.....V_{SS}–0.3 to V_{DD}+0.3V
Operating Temperature.....–20°C to 75°C
Supply Voltage (HT12E) .....–0.3V to 13V
Storage Temperature.....–50°C to 125°C

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

Electrical Characteristics

HT12A Ta=25°C

Symbol	Parameter	Test Conditions V_{DD}	Test Conditions Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage	—	—	2.4	3	5	V
I_{STB}	Standby Current	3V	Oscillator stops	—	0.1	1	μA
		5V		—	0.1	1	μA
I_{DD}	Operating Current	3V	No load f_{OSC}=455kHz	—	200	400	μA
		5V		—	400	800	μA
I_{DOUT}	Output Drive Current	5V	V_{OH}=0.9V_{DD} (Source)	−1	−1.6	—	mA
			V_{OL}=0.1V_{DD} (Sink)	2	3.2	—	mA
V_{IH}	"H" Input Voltage	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL}	"L" Input Voltage	—	—	0	—	0.2V_{DD}	V
R_{DATA}	D8~D11 Pull-high Resistance	5V	V_{DATA}=0V	—	150	300	kΩ

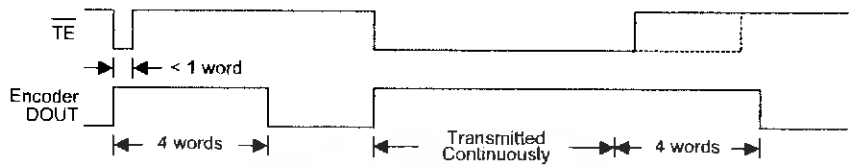
HT12E Ta=25°C

Symbol	Parameter	Test Conditions V_{DD}	Test Conditions Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage	—	—	2.4	5	12	V
I_{STB}	Standby Current	3V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	3V	No load f_{OSC}=3kHz	—	40	80	μA
		12V		—	150	300	μA
I_{DOUT}	Output Drive Current	5V	V_{OH}=0.9V_{DD} (Source)	−1	−1.6	—	mA
			V_{OL}=0.1V_{DD} (Sink)	1	1.6	—	mA
V_{IH}	"H" Input Voltage	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL}	"L" Input Voltage	—	—	0	—	0.2V_{DD}	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}=1.1MΩ	—	3	—	kHz
$R_{\overline{TE}}$	$\overline{TE}$ Pull-high Resistance	5V	$V_{\overline{TE}}$=0V	—	1.5	3	MΩ

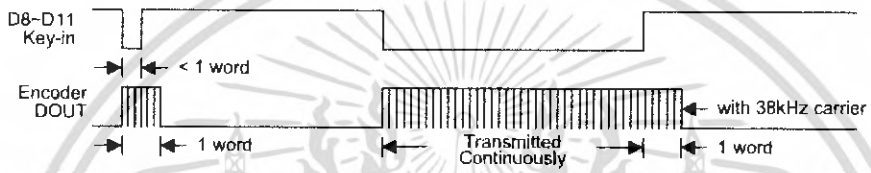
Functional Description

Operation

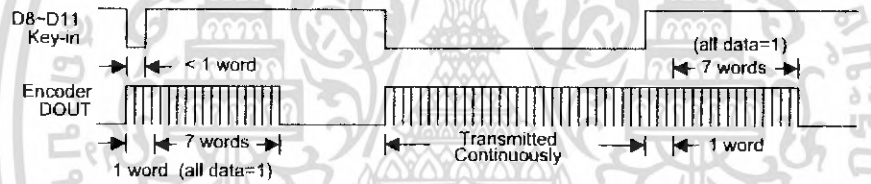
The 2^{12} series of encoders begin a 4-word transmission cycle upon receipt of a transmission enable ($\overline{TE}$ for the HT12E or D8~D11 for the HT12A, active low). This cycle will repeat itself as long as the transmission enable ($\overline{TE}$ or D8~D11) is held low. Once the transmission enable returns high the encoder output completes its final cycle and then stops as shown below.



Transmission timing for the HT12E



Transmission timing for the HT12A (L/MB=Floating or VDD)

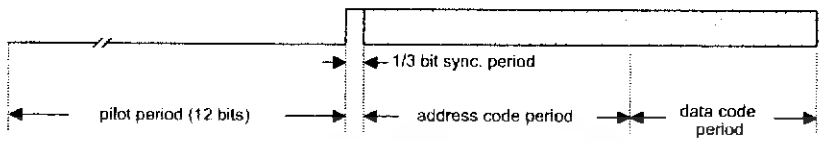


Transmission timing for the HT12A (L/MB=VSS)

Information word

If L/MB=1 the device is in the latch mode (for use with the latch type of data decoders). When the transmission enable is removed during a transmission, the DOUT pin outputs a complete word and then stops. On the other hand, if L/MB=0 the device is in the momentary mode (for use with the momentary type of data decoders). When the transmission enable is removed during a transmission, the DOUT outputs a complete word and then adds 7 words all with the "1" data code.

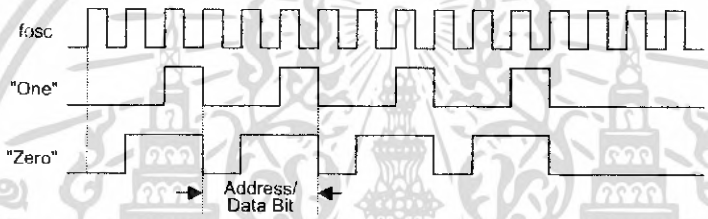
An information word consists of 4 periods as illustrated below.



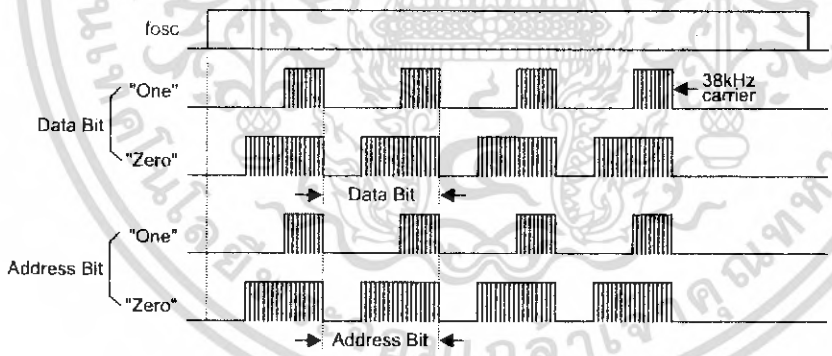
Composition of information

Address/data waveform

Each programmable address/data pin can be externally set to one of the following two logic states as shown below.



Address/Data bit waveform for the HT12E



Address/Data bit waveform for the HT12A

The address/data bits of the HT12A are transmitted with a 38kHz carrier for infrared remote controller flexibility.

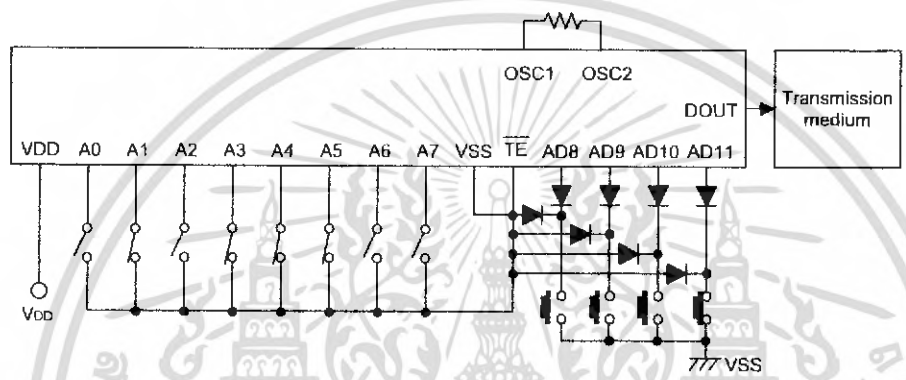
Address/data programming (preset)

The status of each address/data pin can be individually pre-set to logic "high" or "low". If a transmission-enable signal is applied, the encoder scans and transmits the status of the 12 bits of address/data serially in the order A0 to AD11 for the HT12E encoder and A0 to D11 for the HT12A encoder.

During information transmission these bits are transmitted with a preceding synchronization bit. If the trigger signal is not applied, the chip enters the standhy mode and consumes a reduced current of less than 1μA for a supply voltage of 5V.

Usual applications preset the address pins with individual security codes using DIP switches or PCB wiring, while the data is selected by push buttons or electronic switches.

The following figure shows an application using the HT12E:



The transmitted information is as shown:

Pilot & Sync.	A0	A1	A2	A3	A4	A5	A6	A7	AD8	AD9	AD10	AD11
	1	0	1	0	0	0	1	1	1	1	1	0

Address/Data sequence

The following provides the address/data sequence table for various models of the 2^{12} series of encoders. The correct device should be selected according to the individual address and data requirements.

Part No.	Address/Data Bits											
	0	1	2	3	4	5	6	7	8	9	10	11
HT12A	A0	A1	A2	A3	A4	A5	A6	A7	D8	D9	D10	D11
HT12E	A0	A1	A2	A3	A4	A5	A6	A7	AD8	AD9	AD10	AD11

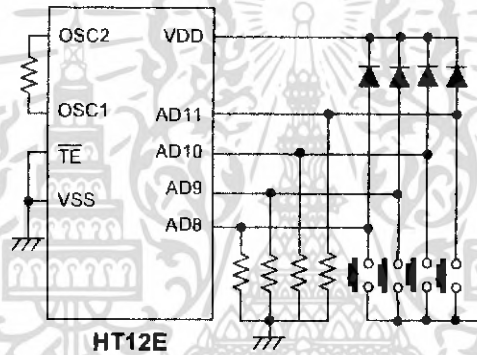
Transmission enable

For the HT12E encoders, transmission is enabled by applying a low signal to the $\overline{TE}$ pin. For the HT12A encoders, transmission is enabled by applying a low signal to one of the data pins D8~D11.

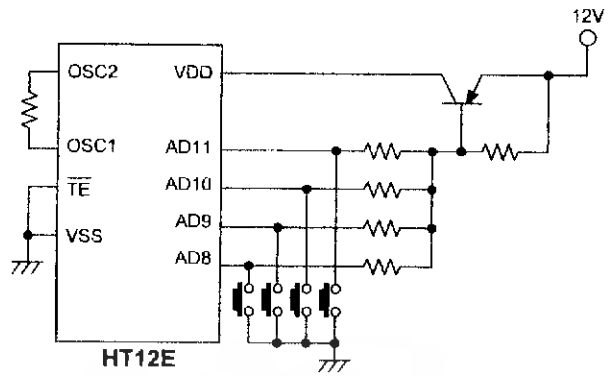
Two erroneous HT12E application circuits

The HT12E must follow closely the application circuits provided by Holtek (see the "Application circuits").

- Error: AD8~AD11 pins input voltage > V_{DD}+0.3V

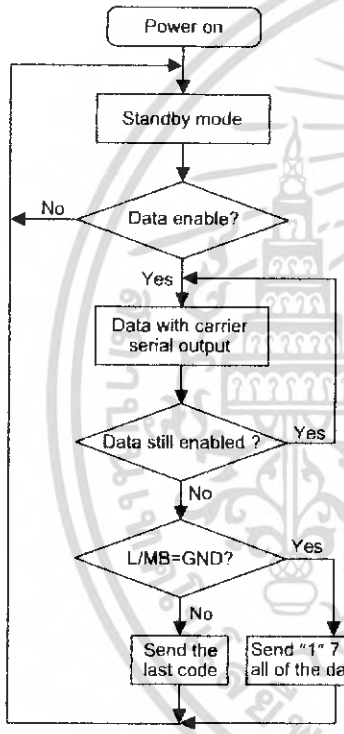


• Error: The IC's power source is activated by pins AD8~AD11

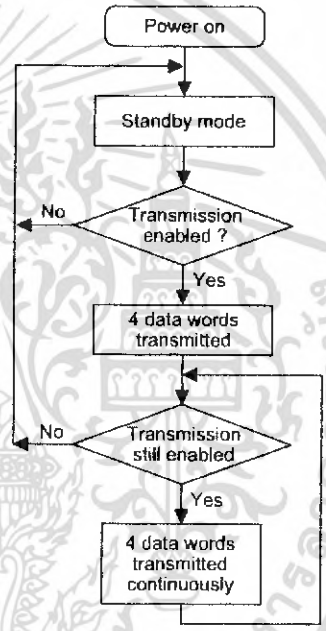


Flowchart

• HT12A

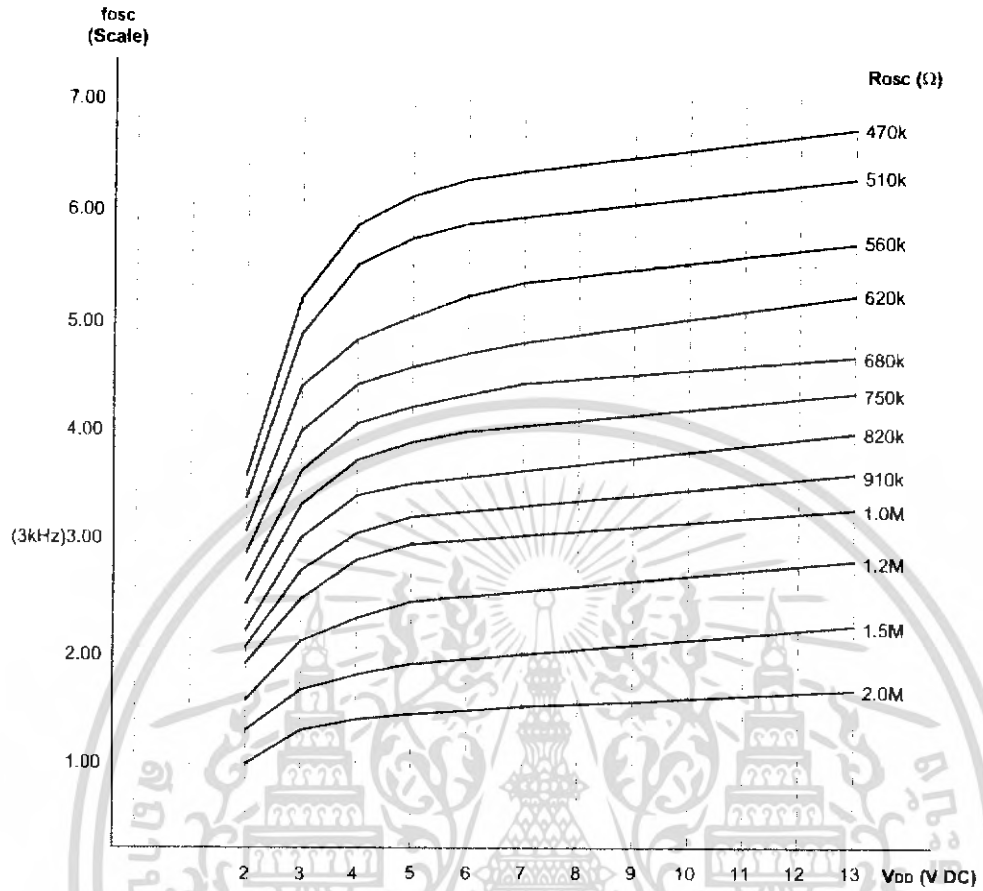


• HT12E



Note: D8~D11 are transmission enables of the HT12A.
$\overline{TE}$ is the transmission enable of the HT12E.

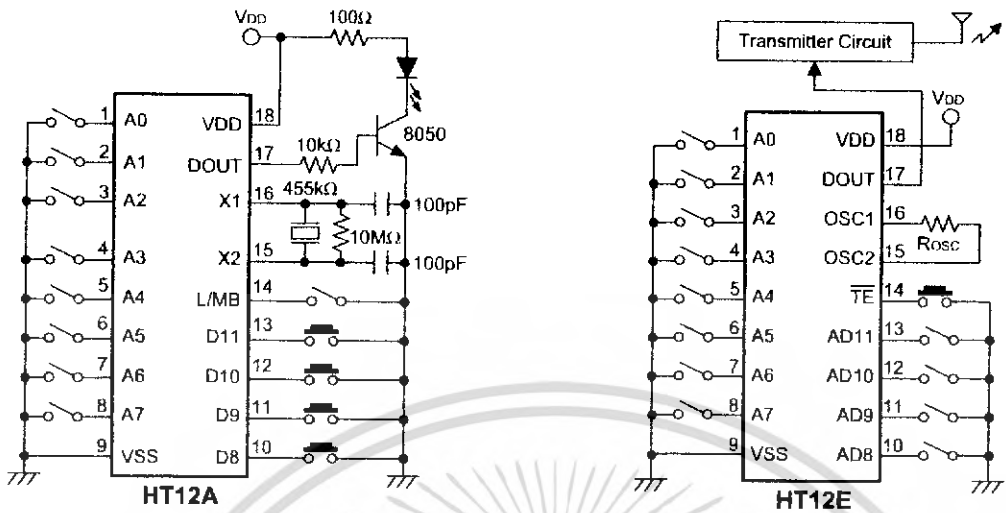
Oscillator frequency vs supply voltage



The recommended oscillator frequency is f_{OSCD} (decoder) $\cong 50\ f_{OSCE}$ (HT12E encoder)

$\cong \frac{1}{3} f_{OSCE}$ (HT12A encoder)

Application Circuits



Note: Typical infrared diode: EL-1L2 (KODENSHI CORP.)
Typical RF transmitter: JR-220 (JUWA CORP.)

Holtek Semiconductor Inc. (Headquarters)
No.3 Creation Rd. II, Science-based Industrial Park, Hsinchu, Taiwan, R.O.C.
Tel: 886-3-563-1999
Fax: 886-3-563-1189

Holtek Semiconductor Inc. (Taipei Office)
5F, No.576, Sec.7 Chung Hsiao E. Rd., Taipei, Taiwan, R.O.C.
Tel: 886-2-2782-9635
Fax: 886-2-2782-9636
Fax: 886-2-2782-7128 (International sales hotline)

Holtek Semiconductor (Hong Kong) Ltd.
RM.711, Tower 2, Cheung Sha Wan Plaza, 833 Cheung Sha Wan Rd., Kowloon, Hong Kong
Tel: 852-2-745-8288
Fax: 852-2-742-8657

HT12D/HT12F
2^{12} Series of Decoders

Features

- Operating voltage: 2.4V~12V
- Low power and high noise immunity CMOS technology
- Low standby current
- Capable of decoding 12 bits of information
- Binary address setting
- Received codes are checked 3 times
- Address/Data number combination
  - HT12D: 8 address bits and 4 data bits
  - HT12F: 12 address bits only
- Built-in oscillator needs only 5% resistor
- Valid transmission indicator
- Easy interface with an RF or an infrared transmission medium
- Minimal external components
- Pair with Holtek's 2^{12} series of encoders
- 18-pin DIP, 20-pin SOP package

Applications

- Burglar alarm system
- Smoke and fire alarm system
- Garage door controllers
- Car door controllers
- Car alarm system
- Security system
- Cordless telephones
- Other remote control systems

General Description

The 2^{12} decoders are a series of CMOS LSIs for remote control system applications. They are paired with Holtek's 2^{12} series of encoders (refer to the encoder/decoder cross reference table). For proper operation, a pair of encoder/decoder with the same number of addresses and data format should be chosen.

The decoders receive serial addresses and data from a programmed 2^{12} series of encoders that are transmitted by a carrier using an RF or an IR transmission medium. They compare the serial input data three times continuously with their local addresses. If no error or unmatched codes are found, the input data codes are decoded and then transferred to the output pins. The VT pin also goes high to indicate a valid transmission.

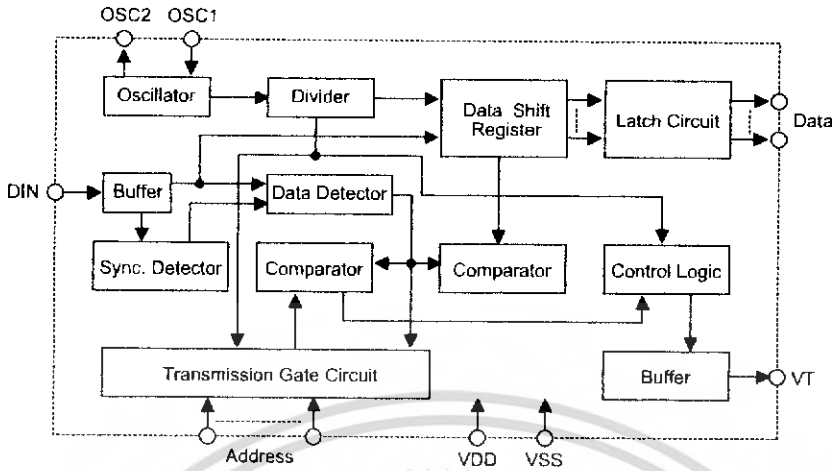
The 2^{12} series of decoders are capable of decoding informations that consist of N bits of address and 12-N bits of data. Of this series, the HT12D is arranged to provide 8 address bits and 4 data bits, and HT12F is used to decode 12 bits of address information.

Selection Table

Function / Part No.	Address No.	Data No.	Data Type	VT	Oscillator	Trigger	Package
HT12D	8	4	L	√	RC oscillator	DIN active "Hi"	18DIP, 20SOP
HT12F	12	0	—	√	RC oscillator	DIN active "Hi"	18DIP, 20SOP

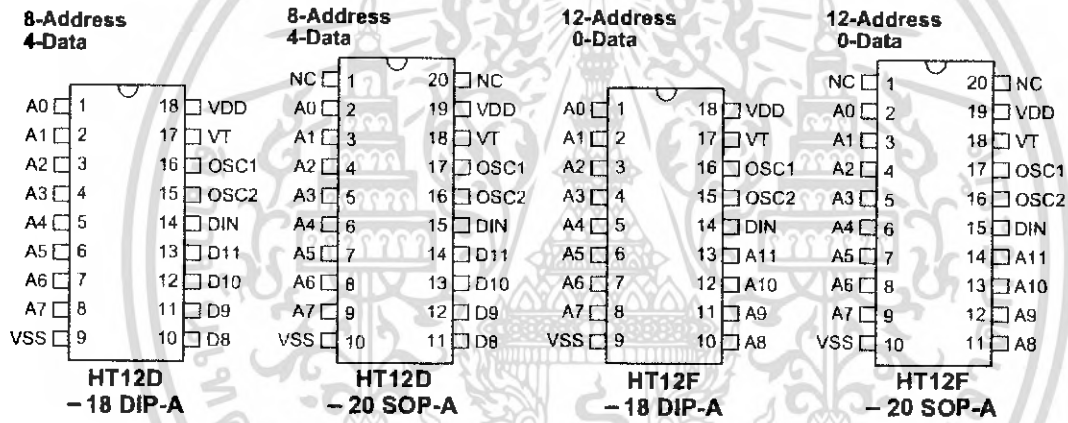
Notes: Data type: L stands for latch type data output.
VT can be used as a momentary data output.

Block Diagram



Note: The address/data pins are available in various combinations (see the address/data table).

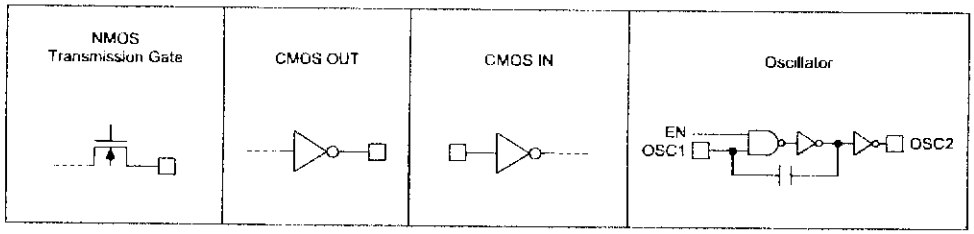
Pin Assignment



Pin Description

Pin Name	I/O	Internal Connection	Description
A0~A11 (HT12F)	I	NMOS Transmission Gate	Input pins for address A0~A11 setting These pins can be externally set to VSS or left open.
A0~A7 (HT12D)			Input pins for address A0~A7 setting These pins can be externally set to VSS or left open.
D8~D11 (HT12D)	O	CMOS OUT	Output data pins, power-on state is low.
DIN	I	CMOS IN	Serial data input pin
VT	O	CMOS OUT	Valid transmission, active high
OSC1	I	Oscillator	Oscillator input pin
OSC2	O	Oscillator	Oscillator output pin
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply

Approximate internal connection circuits



Absolute Maximum Ratings

Supply Voltage ..... −0.3V to 13V

Input Voltage ..... V_{SS}−0.3 to V_{DD}+0.3V

Storage Temperature ..... −50°C to 125°C

Operating Temperature ..... −20°C to 75°C

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	5	12	V
I_{STB}	Standby Current	5V	Oscillator stops	—	0.1	1	μA
		12V		—	2	4	μA
I_{DD}	Operating Current	5V	No load, f_{OSC}=150kHz	—	200	400	μA
I_O	Data Output Source Current (D8~D11)	5V	V_{OH}=4.5V	-1	-1.6	—	mA
	Data Output Sink Current (D8~D11)	5V	V_{OL}=0.5V	1	1.6	—	mA
I_{VT}	VT Output Source Current	5V	V_{OH}=4.5V	-1	-1.6	—	mA
	VT Output Sink Current		V_{OL}=0.5V	1	1.6	—	mA
V_{IH}	"H" Input Voltage	5V	—	3.5	—	5	V
V_{IL}	"L" Input Voltage	5V	—	0	—	1	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}=51kΩ	—	150	—	kHz

Functional Description

Operation

The 2^{12} series of decoders provides various combinations of addresses and data pins in different packages so as to pair with the 2^{12} series of encoders.

The decoders receive data that are transmitted by an encoder and interpret the first N bits of code period as addresses and the last 12–N bits as data, where N is the address code number. A signal on the DIN pin activates the oscillator which in turn decodes the incoming address and data. The decoders will then check the received address three times continuously. If the received address codes all match the contents of the decoder's local address, the 12-N bits of data are decoded to activate the output pins and the VT pin is set high to indicate a valid transmission. This will last unless the address code is incorrect or no signal is received.

The output of the VT pin is high only when the transmission is valid. Otherwise it is always low.

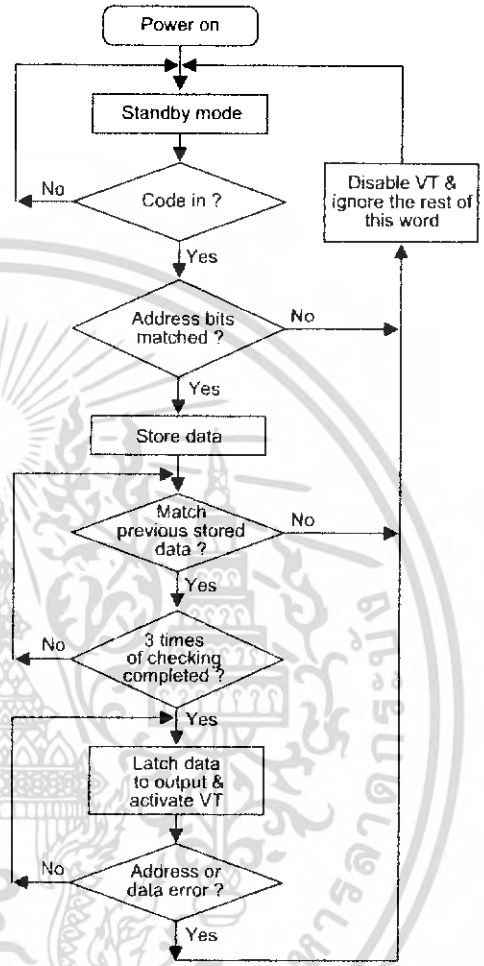
Output type

Of the 2^{12} series of decoders, the HT12F has no data output pin but its VT pin can be used as a momentary data output. The HT12D, on the other hand, provides 4 latch type data pins whose data remain unchanged until new data are received.

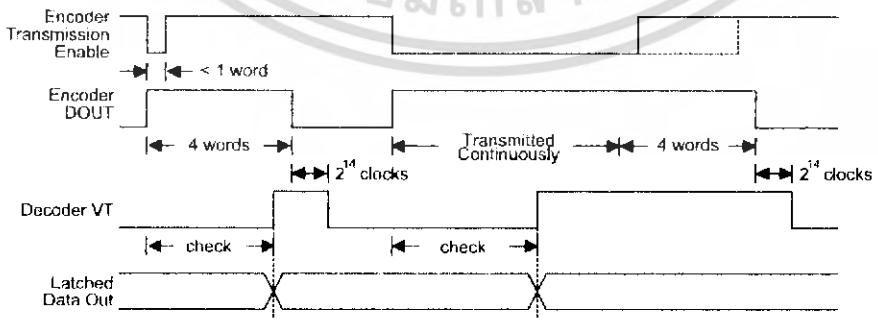
Part No.	Data Pins	Address Pins	Output Type	Operating Voltage
HT12D	4	8	Latch	2.4V~12V
HT12F	0	12	—	2.4V~12V

Flowchart

The oscillator is disabled in the standby state and activated when a logic "high" signal applies to the DIN pin. That is to say, the DIN should be kept low if there is no signal input.



Decoder timing



Encoder/Decoder cross reference table

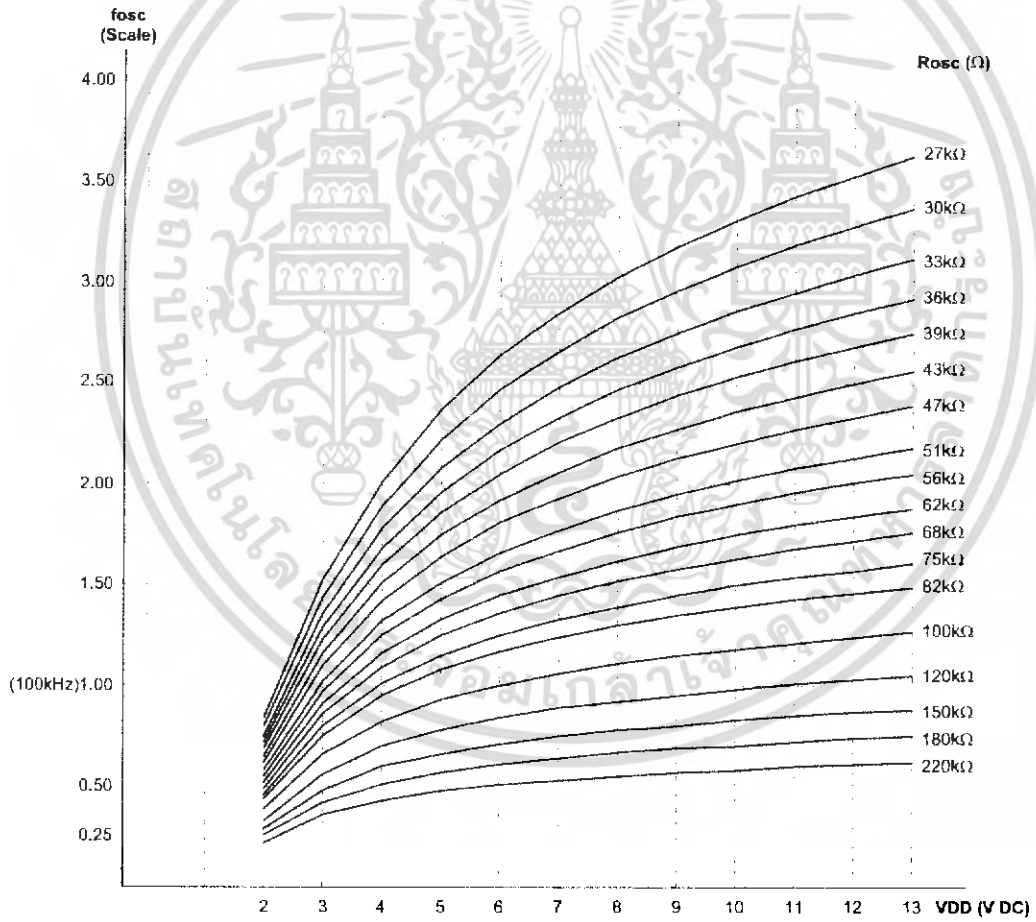
Decoders Part No.	Data Pins	Address Pins	VT	Pair Encoder	Package			
					Encoder		Decoder	
					DIP	SOP	DIP	SOP
HT12D	4	8	√	HT12A HT12E	18	20	18	20
HT12F	0	12	√	HT12A HT12E	18	20	18	20

Address/Data sequence

The following table provides address/data sequence for various models of the 2^{12} series of decoders.

Part No.	Address/Data Bits											
	0	1	2	3	4	5	6	7	8	9	10	11
HT12D	A0	A1	A2	A3	A4	A5	A6	A7	D8	D9	D10	D11
HT12F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11

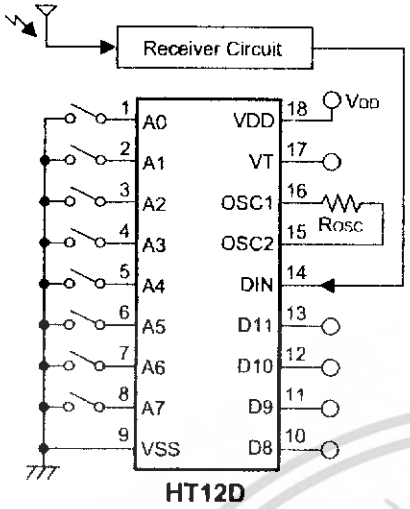
Oscillator frequency vs supply voltage

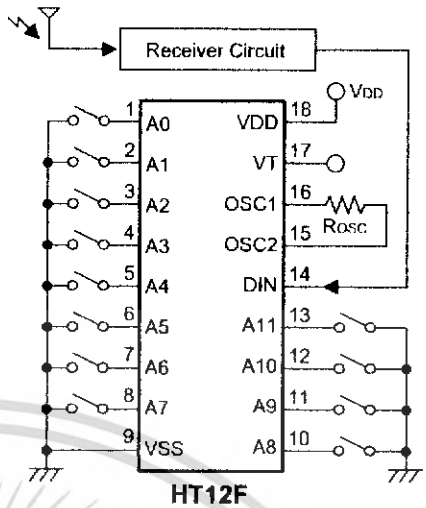


Note: The recommended oscillator frequency is f_{OSCD} (decoder) ≈ 50 f_{OSCE} (HT12E encoder)

≈ $\frac{1}{3}$ f_{OSCE} (HT12A encoder).

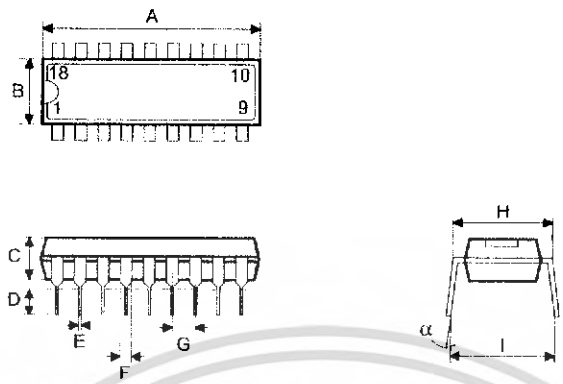
Application Circuits





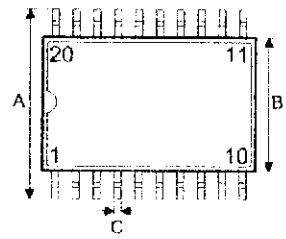
Package Information

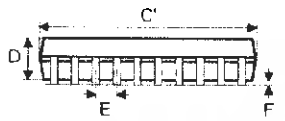
18-pin DIP (300mil) outline dimensions

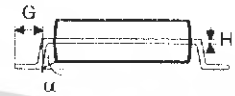


Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	895	—	915
B	240	—	260
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	335	—	375
α	0°	—	15°

20-pin SOP (300mil) outline dimensions



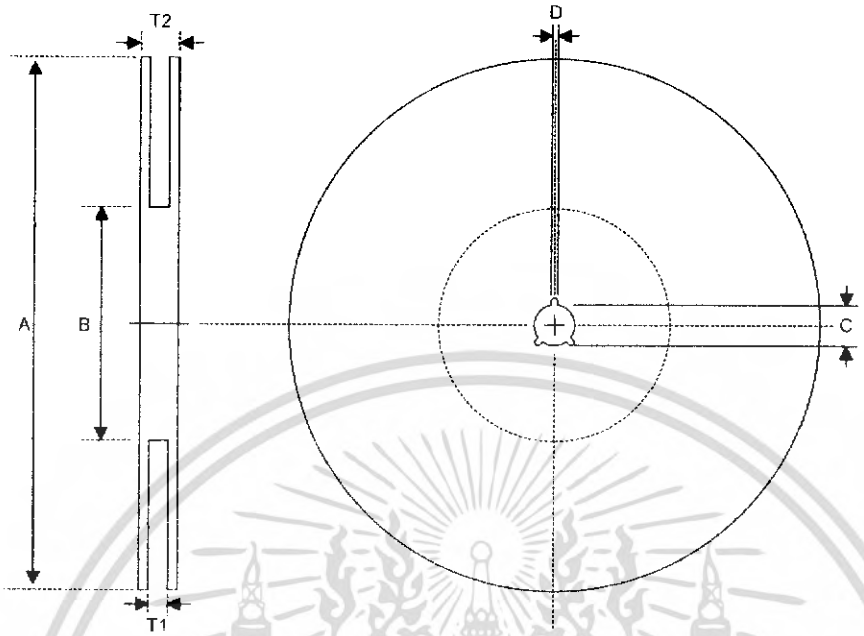




Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C'	490	—	510
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

Product Tape and Reel Specifications

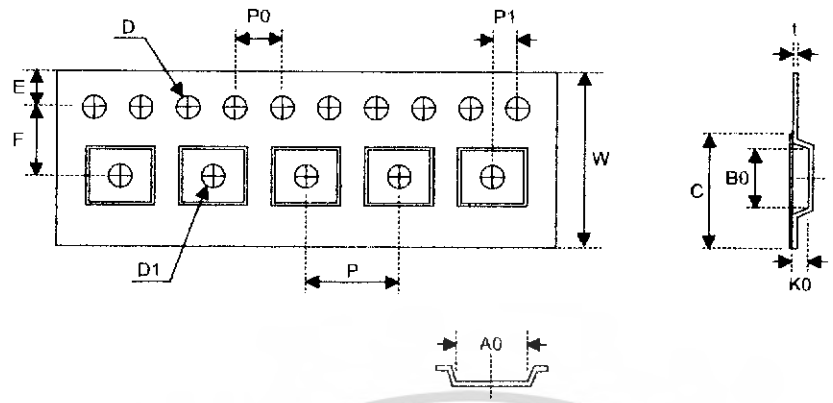
Reel dimensions



SOP 20W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1.0
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13.0+0.5 −0.2
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8+0.3 −0.2
T2	Reel Thickness	30.2±0.2

Carrier tape dimensions



SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0+0.3 −0.1
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	21.3

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor (Shanghai) Inc.
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China
Tel: 021-6485-5560
Fax: 021-6485 0313
http://www.holtek.com.cn

Holtek Semiconductor (Hong Kong) Ltd.
Block A, 3/F, Tai On Industrial Building, 77-79 Cheung Sha Wan Rd., Kowloon, Hong Kong
Tel: 852-2-745-8288
Fax: 852-2-742-8657

Holmate Semiconductor, Inc.
46712 Fremont Blvd., Fremont, CA 94538
Tel: 510-252-9880
Fax: 510-252-9885
http://www.holmate.com

ภาคผนวก

ข. โปรแกรมที่ใช้งาน

```
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
// Program          : Servo Motor Control " RECEIVER PROGRAM "
// Description      : 2 servo motor control
//                  : Data rate 9600 bps,X tal 11.0592MHz
//                  :
// For              :
// Filename         : servo.c
// Version          : 1.1Rev. Edit for use parallel data use P1.0-P1.3
//                  :
// Compiler         : Keil 7.03
//-------------------------------------------------------------------------
// include file
//-------------------------------------------------------------------------
#include<reg52.h>
//#include<intrins.h>
//-------------------------------------------------------------------------
// define Port&Pin
//-------------------------------------------------------------------------
sbit    m1 = P2^0;      // control servo 1 for up and down
sbit    m2 = P2^1;      // control servo 2 for left and right

sbit    kk = P2^3;
//-------------------------------------------------------------------------
// Deafine constance value
//-------------------------------------------------------------------------
//#define res   3       // resolution for move

//-------------------------------------------------------------------------
// delay milisec use timer 0 (mode 1 16 bit timer)
//-------------------------------------------------------------------------
void delay_msec(unsigned int time)
    {
       unsigned int delay;
       for (delay = 0; delay<time; delay++)
{
   TMOD = 0x21;
     TH0  = 0xfc;              /*11.0592 MHz = 0xfc 12MHz = 0xfc*/
     TL0  = 0x66;              /*11.0592 MHz = 0x66 12MHz = 0x17*/
     TF0  = 0;
     TR0  = 1;
     while (TF0 == 0);
     TR0 = 0;
   }
    }
//-------------------------------------------------------------------------
// delay for center position of servo m1 time = 1.5ms
// value = 64200 = 0xfac8
//-------------------------------------------------------------------------
void delay15_m1(void)
    {
    TMOD = 0x21;
    TH0  = 0xfa;
    TL0  = 0xc8;
    TF0  = 0;
    TR0  = 1;
    while (TF0 == 0);
    TR0 = 0;
    }
//-------------------------------------------------------------------------
// delay for center position of servo m2 time = 1.5ms
// value = 64250 = 0xfafa
//-------------------------------------------------------------------------
void delay15_m2(void)
    {
   TMOD = 0x21;
   TH0  = 0xfa;
    TL0  = 0xfa;
   TF0  = 0;
   TR0  = 1;
  while (TF0 == 0);
  TR0 = 0;
    }
//-------------------------------------------------------------------------
// delay for servo time   2ms for m1 maximum time max
```

: 1

```
// sub time 750us
//---------------------------------------------------------------------
void delay15_m1_max(void)
    {
   TMOD = 0x21;
   TH0  = 0xf7;
   TL0  = 0xc7;
   TF0  = 0;
   TR0  = 1;
   while (TF0 == 0);
   TR0 = 0;
    }
//
// delay for servo time = 2ms for m2 maximum time max
// sub time 750us
//---------------------------------------------------------------------
void delay15_m2_max(void)
    {
  TMOD = 0x21;
  TH0  = 0xf8;
  TL0  = 0x01;
  TF0  = 0;
  TR0  = 1;
  while (TF0 == 0);
  TR0 = 0;
    }
//---------------------------------------------------------------------
// delay for  servo time = 1ms for m1 minimum timr min
// add time 750us
//---------------------------------------------------------------------
void delay15_m1_min(void)
    {
   TMOD = 0x21;
   TH0  = 0xfd;
   TL0  = 0xa3;
   TF0  = 0;
   TR0  = 1;
    while (TF0 == 0);
    TR0 = 0;
    }
//---------------------------------------------------------------------
// delay for servo time = 1ms for m2 minimum time min
// add time 750us
//---------------------------------------------------------------------
void delay15_m2_min(void)
    {
    TMOD = 0x21;
    TH0  = 0xfd;                    // 1000 us
    TL0  = 0xdd;
    TF0  = 0;
    TR0  = 1;
    while (TF0 == 0);
    TR0 = 0;
    }
//---------------------------------------------------------------------
// delay for up and down of servo 1
//---------------------------------------------------------------------
void m11(unsigned int time)
    {
    unsigned int temp;
    temp = time;
    temp = temp & 0x00ff;
    TL0 = temp;
    temp = time;
    temp = temp >> 8;
    temp = temp & 0x00ff;
    TH0 = temp;

    TMOD = 0x21;
//            TH0  = 0xfa;
//            TL0  = 0xb5;
       TF0    0;
       TR0  = 1;
       while (TF0 == 0);
       TR0 = 0;
```

: 2

```
    }
//-----------------------------------------------------------------------
// delay for left and right of servo 2
//-----------------------------------------------------------------------
void m22(unsigned int time)
    {
    unsigned int temp;
    temp = time;
    temp = temp & 0x00ff;
    TL0 = temp;

    temp = time;
    temp = temp >> 8;
    temp = temp & 0x00ff;
    TH0 = temp;

    TMOD = 0x21;
//            TH0  = 0xfa;
//            TL0  = 0xef;
    TF0  = 0;
    TR0  = 1;
    while (TF0 == 0);
    TR0 = 0;
    }

//-----------------------------------------------------------------------
// dmsec
//-----------------------------------------------------------------------
void dmsec (unsigned int count) {                  // mSec Delay
    unsigned char i;                          // for Keil CA51 (Speed x 1)
    while (count)
    {
        i = 115; while (i>0) i--;
        count--;
    }
    }
/*
//-----------------------------------------------------------------------
// send text to rs232
// input ascii
//-----------------------------------------------------------------------
void send_txt(unsigned char dat)
{
    TI = 0;
    SBUF = dat;
    while(~TI);          // Wait for send data finish (TI = 1)
    TI = 0;
}
*/
//-----------------------------------------------------------------------
// initial
//-----------------------------------------------------------------------
void init(void)
{
    unsigned char i;
    delay_msec(150);
    m1 =0;
    m2 =0;
    P1 = 0x0f;

    TMOD = 0x21;          // Timer1 Mode2(8 bit auto reload) for serial port
//  SCON = 0x50;          // Serial port TX and RX data Mode


//  TH1 = 0xFA;           // Set 4800 TH1 and TL1 = FA
//  TL1 = 0xFA;

//  TH1 = 0xFD;           // Set 9600 TH1 and TL1 = FD
//  TL1 = 0xFD;

//  RI = 0;               //  RI flag ; 1 = not rx,0 = readv to rx
//  TI = 0;               //  TI flag ; 1 = not tx,0 = ready to tx
//  EA = 1;               // Enable interrupt all
//  ES = 1;               // Enable interrupt serial port
//  TF0 = 0;
```

```
//  TR1 = 1;              // Start Timer1


    //intitial servo 1 up or down to center
    for(i=0;i<100;i++)
    {
    m2 = 1;
    delay15_m1();
    m1 = 0;
    dmsec(24);
    }

    //initial servo 2 left or right to center
    for(i=0;i<100;i++)
    {
    m2 = 1;
    delay15_m2();
    m2 = 0;
    dmsec(24);
    }
//-------------------------------------------------------------------
// test max servo
//-------------------------------------------------------------------
    for(i=0;i<100;i++)
    {
    m1 = 1;
    delay15_m1_max();
    m1 = 0;
    dmsec(24);
    }

    for(i=0;i<100;i++)
    {
    m2 = 1;
    delay15_m2_max();
    m2 = 0;
    dmsec(24);
    }

//-------------------------------------------------------------------
//test min servo
//-------------------------------------------------------------------
    for(i=0;i<100;i++)
    {
    m1 = 1;
    delay15_m1_min();
    m1 = 0;
    dmsec(24);
    }

    for(i=0;i<100;i++)
    {
    m2 = 1;
    delay15_m2_min();
    m2 = 0;
    dmsec(24);
    }

}
//-------------------------------------------------------------------
// main function
//-------------------------------------------------------------------
void main(void)
{
    unsigned char dat=0,i;
    unsigned int del_11=0xfab4,del_22=0xfafa;       // initial value for servo at center
    init();
//  delay_msec(250);

//-----start at center
    for(i=0;i<75;i++)
    {
    m1 = 1;
    m11(del_11);
```

```
        m1 = 0;
        dmsec(24);
        }

        for(i=0;i<75;i++)
        {
        m2 = 1;
        m22(del_22);
        m2 = 0;
        dmsec(24);
        }

        while(1)                        // Infinite loop
        {
        P1 = 0x0f;
        dat = P1;

    /*  switch(dat)
        {

            case 0x02:  del_11 = del_11 - 1;

                if(del_11 <= 63500)
                {
                del_11 = 63500;
                }

                for(i=0;i<1;i++)
                {
                m1 = 1;
                m11(del_11);
                m1 = 0;
                dmsec(10);
                }
                break;

            case 0x08:

            del_11 = del_11 + 1;

            if(del_11 >= 64950)
            {
            del_11 = 64950;
            }

            for(i=0;i<1;i++)
            {
            m1 = 1;
            m11(del_11);
            m1 = 0;
            dmsec(10);
            }
            P1 = 0x0f;
            dat = P1;       // read P1 againt.
            break;

    case 0x04:
        del_22 = del_22 - 1;

            for(i=0;i<1;i++)
                {
                m2 = 1;
                m22(del_22);
                m2 = 0;
                dmsec(10);
                }

            if(del_22 <= 63500)
            {
            del_22 = 63500;
            }
            P1 = 0x0f;
            dat = P1;       // read P1 againt.
            break;
```

```
case 0x06:
    del_22 = del_22 + 1;

        if(del_22 >= 65000)
        {
        del_22 = 65000;
        }

        for(i=0;i<1;i++)
        {
        m2 = 1;
        m22(del_22);
        m2 = 0;
        dmsec(10);
        }

        P1 = 0x0f;
        dat = P1;        // read P1 againt.
        break;

case 0x05:

    for(i=0;i<3;i++)             // send fix servo 1
        {
        m1 = 1;
        m11(del_11);
        m1 = 0;
        dmsec(10);
        }

        for(i=0;i<3;i++)             // send fix servo 2
        {
        m2 = 1;
        m22(del_22);
        m2 = 0;
        dmsec(10);
        }
        P1 = 0x0f;
        dat = P1;        // read P1 againt.
        break;
    }
*/

        if(dat == 0x02)              // if recieve num#2
        {
            del_11 = del_11 - 3;

            if(del_11 <= 63500)
                {
                  del_11 = 63500;
                }
            for(i=0;i<1;i++)
                {
                m1 = 1;
                m11(del_11);
                m1 = 0;
                dmsec(25);
                }
        dat = P1;        // read P1 againt.
        }   // end if dat == 2


        if(dat == 0x08)              //if recieve num#8
        {
        del_11 = del_11 + 3;

            if(del_11 >= 64950)
            {
                del_11 = 64950;
            }

            for(i=0;i<1;i++)
            {
                m1 = 1;
```

```
            m11(del_11);
            m1 = 0;
            dmsec(25);
        }
    dat = P1;       // read P1 againt.
    }


    if(dat == 0x04)             //if recive num#4
    {
    del_22 = del_22 - 3;
        if(del_22 <= 63500)
        {
        del_22 = 63500;
        }
        for(i=0;i<1;i++)
        {
            m2 = 1;
            m22(del_22);
            m2 = 0;
            dmsec(25);
        }
    dat = P1;       // read P1 againt.
    }


    if(dat == 0x06)             // if recieve num#6
    {
    del_22 = del_22 + 3;

        if(del_22 >= 65000)
        {
            del_22 = 65000;
        }
        for(i=0;i<1;i++)
        {
            m2 = 1;
            m22(del_22);
            m2 = 0;
            dmsec(25);
        }
    dat = P1;       // read P1 againt.
    }

    if(dat == 0x05)             // if number #5 servo 1 & servo 2 to fix
    {
    for(i=0;i<2;i++)            // send fix servo 1
        {
        m1 = 1;
        m11(del_11);
        m1 = 0;
        dmsec(25);
        }

    for(i=0;i<2;i++)            // send fix servo 2
        {
        m2 = 1;
        m22(del_22);
        m2 = 0;
        dmsec(25);
        }
    dat = P1;       // read P1 againt.
    }

    if(dat == 0x00)             // if number #5 servo 1 & servo 2 to fix
    {
    m1 = 0;
    m2 = 0;
    P1 = 0x0f;
    dat = P1;       // read P1 againt.
    }
    }   // end loop while
}   // end loop main
//---------------------------------------------------------------------------
```

```
// PROGRAM      :   SERVO MOTOR CONTROL " TRANSMITTER PROGRAM "
//#include<reg52.h>
#include<AT89x051.H>
#include<intrins.h>

//-----------------------------------------------------------------
// delay milisec use timer 0 (mode 1 16 bit timer)
//-----------------------------------------------------------------
void delay_msec(unsigned int time)
    {
          unsigned int delay;
          for (delay = 0; delay<time; delay++)
         {
            TMOD = 0x21;
            TH0  = 0xfc;                 /*11.0592 MHz = 0xfc 12MHz = 0xfc*/
            TL0  = 0x66;                 /*11.0592 MHz = 0x66 12MHz = 0x17*/
            TF0  = 0;
            TR0  = 1;
            while (TF0 == 0);
            TR0 = 0;
          }
    }
//-----------------------------------------------------------------
// initial
//-----------------------------------------------------------------
void init(void)
{
//  unsigned char i;
    delay_msec(150);

    TMOD = 0x21;          // Timer1 Mode2(8 bit auto reload) for serial port
    SCON = 0x50;          // Serial port TX and RX data Mode


//  TH1 = 0xFA;           // Set 4800 TH1 and TL1 = FA
//  TL1 = 0xFA;

    TH1 = 0xFD;           // Set 9600 TH1 and TL1 = FD
    TL1 = 0xFD;

    RI = 0;               //  RI flag ; 1 = not rx,0 = ready to rx
    TI = 0;               //  TI flag ; 1 = not tx,0 = ready to tx
    EA = 1;               // Enable interrupt all
    ES = 1;               // Enable interrupt serial port
//  TF0 = 0;
    TR1 = 1;              // Start Timer1
}
//-----------------------------------------------------------------
void sevice_serial() interrupt 4    // Vector interrupt for serial port
//-----------------------------------------------------------------
{
        unsigned char dat;
        RI = 0;                 // Clear RI flag
        dat = SBUF;           // Load data to dat

            if(dat == 0x35)               // if recieve num#5
            {
            P1=0x05;
            }

            if(dat == 0x32)               // if recieve num#2
            {
            P1=0x02;
            }

            if(dat == 0x38)               // if recieve num#8
            {
            P1=0x08;
            }

            if(dat == 0x34)               // if recieve num#4
            {
            P1=0x04;
            }
```

```
            if(dat == 0x36)                 // if recieve num#6
            {
            P1=0x06;
            }

            if(dat == 0x30)                 // if recieve num#0
            {
            P1 = 0x00;
            }

}
//------------------------------------------------------------
void main(void)
//------------------------------------------------------------
{
    init();
    P1 = 0x00;
    while(1);                       // Infinite loop
}
//------------------------------------------------------------
```

```
// PROGRAM   :   " ประมวลผลภาพ "
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, CPort, StdCtrls, CPortCtl, ieview, imageenview, ievect, videocap,
ExtCtrls, Buttons;
type
TForm1 = class(TForm)
ImageEnVideoView1: TImageEnVideoView;
Button1: TButton;
Button2: TButton;
Button3: TButton;
Button4: TButton;
Button5: TButton;
Image1: TImage;
Timer1: TTimer;
head: TRadioButton;
tail: TRadioButton;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Edit4: TEdit;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Button6: TButton;
ComPort: TComPort;
Button7: TButton;
Button8: TButton;
Button9: TButton;
Button10: TButton;
Button11: TButton;
Button12: TButton;
Button13: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure Image1Click(Sender: TObject);
procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
procedure Button6Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Button8Click(Sender: TObject);
procedure Button9Click(Sender: TObject);
procedure Button12Click(Sender: TObject);
procedure Button10Click(Sender: TObject);
procedure Button11Click(Sender: TObject);
procedure Button13Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
color_mark:integer;
r_mark :integer;
g_mark :integer;
b_mark :integer;
headcolorr,headcolorg,headcolorb :integer;
tailcolorr,tailcolorg,tailcolorb :integer;
headcolorr2,headcolorg2,headcolorb2 :integer;
tailcolorr2,tailcolorg2,tailcolorb2 :integer;
dotx,doty,dotx1,doty1,dotx2,doty2,xt,yt,xh,yh,xt2,yt2,xh2,yh2:integer;
Str: String;
```

```
implementation

{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
ImageEnVideoView1.DoConfigureSource;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
ImageEnVideoView1.DisplayMode := dmOverlay;
ImageEnVideoView1.ShowVideo := True;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
ImageEnVideoView1.DoConfigureFormat;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
if ImageEnVideoView1.Frozen = True then
begin
ImageEnVideoView1.Frozen := False;
Button4.Caption := 'Freez';
end
 else
begin
ImageEnVideoView1.Frozen := True;
Button4.Caption := 'Unfreez';
 end;
end;

procedure TForm1.Button5Click(Sender: TObject);
var
x:integer;
y:integer;
begin
for x:=0 to image1.Width -1 do
for y:=0 to image1.Height-1 do
image1.Canvas.Pixels[x,y]:=ImageEnVideoView1.IEBitmap.Canvas.Pixels[x,y];
end;

procedure TForm1.Timer1Timer(Sender: TObject);   //  Start Program
var
hxmax,hymax,hxmin,hymin,txmax,tymax,txmin,tymin,x,y,r,g,b,pix_div,color,adh,adt,adh2,adt2:integer;
hxmax2,hymax2,hxmin2,hymin2,txmax2,tymax2,txmin2,tymin2,bot1,bot2:integer;
adbotx1,adboty1,adbotx2,adboty2:Extended;
m,ys:Single;
Str:String;
begin
ImageEnVideoView1.Frozen := True;
 pix_div :=8;       // scan pixs
hxmax:=0;
hymax:=0;
hymin:=300;
hxmin:=480;
txmax:=0;
tymax:=0;
tymin:=300;
txmin:=480;
for x:=0 to (image1.Width div pix_div)-1 do
for y:=0 to (image1.Height div pix_div)-1 do
begin
color:=  ImageEnVideoView1.IEBitmap.Canvas.Pixels[(x*pix_div),(y*pix_div)];
b:=(color and $FF0000)div $10000;
g:=(color and $FF00)div $100;
```

```
r:=color and $FF;
if (r>headcolorr-35)and(r<headcolorr+35)and(g>headcolorg-35)and(g<headcolorg+35)
and(b>headcolorb-35)and(b<headcolorb+35)
 then
begin
if x*pix_div > hxmax then
hxmax:=x*pix_div;
if y*pix_div > hymax then
hymax:=y*pix_div;
if x*pix_div < hxmin then
hxmin:=x*pix_div;
if y*pix_div < hymin then
hymin:=y*pix_div;
end;
if (r>tailcolorr-35)and(r<tailcolorr+35)and(g>tailcolorg-35)and(g<tailcolorg+40)
and(b>tailcolorb-35)and(b<tailcolorb+35) then
begin
if x*pix_div > txmax then
txmax:=x*pix_div;
 if y*pix_div > tymax then
tymax:=y*pix_div;
if x*pix_div < txmin then
 txmin:=x*pix_div;
 if y*pix_div < tymin then
tymin:=y*pix_div;
end;
end;
if hxmax>0 then
if hymax>0 then
if hymin<hymax then
if hxmin<hxmax then
begin
xh:=(hxmax+hxmin)div 2;
yh:=(hymax+hymin)div 2;
end;
 if txmax>0 then
if tymax>0 then
if tymin<tymax then
 if txmin<txmax then
begin
xt:=(txmax+txmin)div 2;
yt:=(tymax+tymin)div 2;
end;
Image1.Canvas.Ellipse(xh-3,yh-3,xh+3,yh+3);    //:= clBlack;
Image1.Canvas.Pixels[xt,yt]:= clCream;
if(xh<10)or(xh>470)or(yh<10)or(yh>290)then  //edge pixel
 begin
Str:='0';
if ComPort.Connected then
ComPort.WriteStr(Str);
end
else
begin            //process
 if xh<190 then    //width
            begin
Str:='4';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;
if xh>290 then    //width
begin
Str:='6';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;
if yh<100 then  //height
begin
```

```
Str:='8';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;
 if yh>200 then   //height
begin
Str:='2';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;
if ((xh>=190)and(xh<=290)and(yh>=100)and(yh<=200)) then  // stop bit
begin
Str:='0';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;
            end;
edit1.Text:= IntToStr(xh);
edit2.Text:= IntToStr(xt);
edit3.Text:= IntToStr(yh);
edit4.Text:= IntToStr(yt);
end;

procedure TForm1.Image1Click(Sender: TObject);
begin
form1.Caption:='read pixel';
end;

procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,Y: Integer);
begin
if form1.Caption='read pixel' then
begin
color_mark:=image1.Canvas.Pixels[x,y] ;
b_mark:=(color_mark and $FF0000)div $10000;
g_mark:=(color_mark and $FF00)div $100;
r_mark:=color_mark and $FF;
form1.Caption:='form1';
end;
if head.Checked=True Then
begin
headcolorr:=r_mark;
headcolorg:=g_mark;
headcolorb:=b_mark;
end;
if tail.Checked=True Then
begin
tailcolorr:=r_mark;
tailcolorg:=g_mark;
tailcolorb:=b_mark;
end;
            end;
procedure TForm1.Button6Click(Sender: TObject);
begin
if ComPort.Connected then
begin
ComPort.Close;
Button6.Caption := 'Start';
end
else
begin
ComPort.Open;
Button6.Caption := 'Stop';
end;
            end;

procedure TForm1.Button7Click(Sender: TObject);
begin
```

```
ComPort.ShowSetupDialog;
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
Str:='8';
if ComPort.Connected then
ComPort.WriteStr(Str);
        end;

procedure TForm1.Button9Click(Sender: TObject);
begin
Str:='4';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;

procedure TForm1.Button12Click(Sender: TObject);
begin
Str:='0';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;

procedure TForm1.Button10Click(Sender: TObject);
begin
Str:='6';
if ComPort.Connected then
ComPort.WriteStr(Str);
end;

procedure TForm1.Button11Click(Sender: TObject);
begin
Str:='2';
 if ComPort.Connected then
ComPort.WriteStr(Str);
end;

procedure TForm1.Button13Click(Sender: TObject);
begin
if Button13.Caption='Manual' then
begin
Button13.Caption:='Auto';
Button8.Enabled := true;
Button9.Enabled := true;
Button10.Enabled := true;
Button11.Enabled := true;
Button12.Enabled := true;
end
else
begin
 Button13.Caption:='Manual';
 Button8.Enabled := false;
 Button9.Enabled := false;
 Button10.Enabled := false;
 Button11.Enabled := false;
 Button12.Enabled := false;
end;
end;
end.
```