หุ่นยนต์เล่นฟุตบอล

ROBOT SOCCER

โดย

นาย อดิวัชร์ ไพรรัตนากร

นาย อติพล สุเฌอ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

หุ่นยนต์เล่นฟุตบอล

ROBOT SOCCER

โดย

นาย อดิวัชร์ ไพรรัตนากร เลขประจำตัว 41014500

นาย อติเทพ สุเฌอ เลขประจำตัว 41014504

อาจารย์ที่ปรึกษา

ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ ปีการศึกษา 2544

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์เตะฟุตบอล

ผู้จัดทำ

1. นาย อดิวัชร์ ไพรรัตรากร รหัส 41014500
2. นาย อติพล สุเฌอ รหัส 41014504

..... อาจารย์ที่ปรึกษา

ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์

หุ่นยนต์เตะฟุตบอล

นาย อดิวัชร์ ไพรรัตน์นากร รหัส 41014500
นาย อติพล สุเฌอ รหัส 41014504
ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

โครงงานฉบับนี้นำเสนอ หุ่นยนต์เตะฟุตบอล โดยใช้หลักการทำงานอาศัยการประมวลผลสัญญาณภาพ นำมาควบคุมการเคลื่อนไหวของหุ่นยนต์ ซึ่งมีขั้นตอนการทำงานดังนี้ หุ่นยนต์จะอยู่ในสนามที่กำหนด โดยสามารถเคลื่อนไหวได้อิสระในบริเวณสนามนี้ โดยหุ่นยนต์จะเคลื่อนที่เข้าหาลูกบอล และ ยิงลูกบอลเข้าไปในประตู โดยหลักการควบคุมหุ่นยนต์ คือ ส่งภาพของบริเวณภายในสนามผ่านทางกล้องสู่เครื่องคอมพิวเตอร์ และ ทำการประมวลผลภาพที่ได้โดยโปรแกรมที่เขียนจากภาษา delphi และ ทำการควบคุมหุ่นยนต์โดยเครื่องรับส่งไร้สาย

ROBOT SOCCER

Adiwach Prirattanakorn ID 41014500

Atipol Sucher ID 41014504

Dr. Surapan Airphaiboon Advisor

Academic Year 2001

ABSTRACT

This Project aim to present Robot Soccer that consist of Image Processing which can be control movement . Robot can move free at the field by automatic. It will go to the ball and shoot the ball to the goal .The concept of control robot is transfer image from digital camera that locate above the field to computer and process by program to create by **Delphi** language . And send data to control robot by wireless transmitter.

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ คุณพ่อ และคุณแม่ ที่ให้การสนับสนุนด้านการศึกษาคอยดูแลเอาใจใส่และห่วงใยตลอดมาจนการทำงานครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณ ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษาของข้าพเจ้า และ อาจารย์หลายๆท่านที่ให้ความอนุเคราะห์ทั้งทางด้านความรู้ และ อุปกรณ์ในการทำงาน รวมทั้งให้คำปรึกษาและ แนะนำแนวทางการทำโครงงานวิทยานิพนธ์นี้

ขอขอบคุณพี่ๆ และ เพื่อนๆ ทุกคนที่คอยให้คำแนะนำ กำลังใจ และ ความช่วยเหลือ ในการทำงานนี้มาโดยตลอด

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

ในปัจจุบันนี้หุ่นยนต์ที่สามารถปฏิบัติงานได้อย่างอัตโนมัติ ได้เริ่มเข้ามามีบทบาทอย่างมากในชีวิตประจำวัน ทั้งในด้านการอำนวยความสะดวกต่างๆ และมีประโยชน์อย่างมากในด้านอุตสาหกรรม โดยหุ่นยนต์ทั่วๆไปจะทำงานได้ก็ต้องมีผู้เขียนโปรแกรมการทำงานให้ เมื่อเราต้องการศึกษาและทดลองเกี่ยวกับหุ่นยนต์เราก็จำเป็นต้องมีความรู้ในด้านการเขียนโปรแกรมโดยในโครงงานนี้ได้ใช้โปรแกรมภาษาเดลไฟล์ (Delphi)ในการเขียนโปรแกรม

การที่หุ่นยนต์จะทำงานได้นั้นก็จะต้องรับข้อมูลจากภายนอก เช่น เสียง อุณหภูมิ ความชื้น ภาพ มาประมวลผล เพื่อให้ทำงานได้ตรงตามแต่ละเหตุการณ์ที่เราทำการโปรแกรมไว้ ซึ่งในโครงงานนี้มีการรับข้อมูลภาพที่เป็น 2 มิติ เป็นกระบวนการหนึ่งซึ่งเราเรียกว่า ดิจิตอลอิมเมจโปรเซสซิ่ง)

ดิจิตอลอิมเมจโปรเซสซิ่ง โดยทั่วไปหมายถึง การประมวลผลของภาพใน 2 มิติ โดยคอมพิวเตอร์ หรือถ้าจะกล่าวในความหมายที่กว้างขึ้น จะหมายถึง การประมวลผลของข้อมูล 2 มิติ โดยเมื่อใดก็ตามที่มนุษย์หรืออุปกรณ์ใดๆก็ตามได้รับข้อมูลและประมวลผลข้อมูลที่เป็น 2 มิติ เราก็จะเรียกได้ว่าเป็น อิมเมจโปรเซสซิ่ง ดิจิตอลอิมเมจโปรเซสซิ่ง ได้มีการใช้งานอย่างกว้างขวาง เช่น การส่งข้อมูลผ่านดาวเทียมและยานอวกาศ, การส่งข้อมูลและเก็บข้อมูลภาพในธุรกิจ, เรดาร์, โซนาร์, การตรวจสอบชิ้นส่วนอัตโนมัติในโรงงาน การส่งเสริมและบันทึกข้อมูลภาพได้ใช้ในการออกอากาศของโทรทัศน์, การประชุมทางไกล (teleconference) , การรับ - ส่งแฟกซ์ ,การสื่อสารของระบบเครือข่ายคอมพิวเตอร์, โทรทัศน์วงจรปิด, การสื่อสารทางทหาร ในทางการแพทย์ก็เช่น การประมวลผลภาพจาก X-ray และ ultrasonic scanning โดยภาพดังกล่าวสามารถนำมาวินิจฉัยโรคได้

1.1 วัตถุประสงค์ของโครงงาน

เพื่อศึกษาการทำงาน และ นำไปประยุกต์ใช้ ของหุ่นยนต์อัตโนมัติ โดยใช้ความรู้ทางด้านการประมวลผลภาพ การเขียนโปรแกรมประยุกต์ใช้งานด้วย ภาษาเดลไฟล์ โปรแกรมและการประยุกต์ใช้งานของ ไมโครคอนโทรลเลอร์(Micro Controller)

1.2 ขอบเขตของโครงงาน

เนื้อหาของปริญญานิพนธ์นี้ จะเริ่มจากศึกษาการทำงานของหุ่นยนต์โดยใช้คนบังคับ โดยควบคุมผ่านคอมพิวเตอร์ซึ่งสื่อสารผ่านพอร์ตอนุกรมกับไมโครคอนโทรลเลอร์ และการประมวลผลภาพด้วยหลักการเบื้องต้น และได้ทำการพัฒนาจนสามมารถประมวลผลหาตำแหน่งได้อย่างแม่นยำ และหุ่นยนต์สามารถเคลื่อนที่เข้าไปยิงลูกบอลได้อย่างอัตโนมัติ

บทที่ 2
หลักการประมวลภาพ

2.1 หลักการพื้นฐานทางอิมเมจโปรเซสซิ่ง

อิมเมจโปรเซสซิ่ง (image processing) เป็นกระบวนการที่ใช้ในการจัดการข้อมูลที่เป็นรูปภาพต่างๆให้อยู่ในลักษณะของสัญญาณไฟฟ้าเพื่อที่จะได้นำข้อมูลที่เป็นสัญญาณไฟฟ้านั้นไปใช้ประโยชน์ทางอื่น เป็นต้นว่า การตกแต่ง , การส่งรูปภาพไปตามสายนำสัญญาณจากที่แหล่งหนึ่งไปยังอีกแห่งหนึ่ง (ซึ่งก็คือหลักการของโทรสาร), การเก็บข้อมูลรูปภาพไว้ในหน่วยความจำเพื่อทำอัลบัมภาพอิเล็กทรอนิกส์เพื่อใช้ประโยชน์เป็นแฟ้มข้อมูลพนักงาน. แฟ้มอาชญากรรม เป็นต้น นอกเหนือไปจากนี้ก็ยังสามารถนำไปใช้งานด้านการรักษาความปลอดภัย. ตรวจสอบลายนิ้วมือ, การนำข้อมูลไปประมวลผลใช้ในการทำงานของหุ่นยนต์อัตโนมัติได้อีกด้วย

พิกเซล (pixel)

ภาพที่จะถูกแปลงสัญญาณไฟฟ้า จะได้รับการแบ่งรายละเอียดของภาพเป็นตารางเล็กๆและตารางเล็กๆนี้เองคือพิกเซล (pixel) และเมื่อมีการจัดเรียงพิกเซลก็จะเกิดพิกเซลที่เป็นแถว (row) และคอลัมน์ (column) โดยมีจำนวนแถวทางแนวนอนเป็น N แถว และมีจำนวนแถวทางแนวตั้ง M แถว ซึ่งในแต่ละตำแหน่งของพิกเซลจะแทนด้วย P(j,j) โดยที่ I และ j จะเป็นเลขจำนวนเต็มและที่เรียกการจัดเรียงของพิกเซลว่าพิกเซลเมตริกซ์ (pixel matrix)

เมื่อทราบตำแหน่งของพิกเซลแล้วก็จำเป็นต้องทราบว่าที่ตำแหน่งนั้นๆพิกเซลมีค่าเท่าไหร่ ซึ่งค่าที่ว่านี้ก็คือค่าเฉลี่ยความเข้มของแสงที่ตกกระทบบนตำแหน่งของแต่ละพิกเซล เช่น เมื่อทำเป็นภาพที่มีความเข้มสองระดับ ก็จะมีค่าอยู่ระหว่าง 0 กับ 1

ลักษณะข้อมูลภาพ

1. ภาพ 2 ระดับ คือ มีแค่จุดขาวกับดำเท่านั้น โดยแต่ละจุดเป็นข้อมูล 1 บิต

2. ภาพ 16 ระดับ ซึ่งในแต่ละจุดภาพจะเป็นข้อมูล 4 บิต ซึ่งทำให้สามารถแสดงภาพได้ 16 ระดับสี หรือ 16 ระดับสีเทา ขึ้นอยู่กับภาพนั้นเป็นภาพสีหรือภาพขาว -ดำ

3. ภาพ 256 ระดับ ซึ่งในแต่ละจุดภาพจะเป็นข้อมูล 8 บิต ซึ่งทำให้สามารถแสดงภาพได้ 256 ระดับสี หรือระดับสีเทา ขึ้นอยู่กับว่าภาพนั้นเป็นภาพสี หรือขาว - ดำ

4. ภาพ true color ซึ่งในแต่จุดจะเป็นข้อมูลขนาด 24 บิต ทำให้สามารถแสดงผลภาพได้เหมือนภาพจริงที่สุด เพราะสามารถแสดงสีได้ถึง 16,777,216 สี ภาพ true color สามารถแสดงได้เฉพาะภาพสีเท่านั้น ไม่สามารถแสดงภาพขาว -ดำได้

การแสดงภาพนี้ใช้วิธี ตั้งค่าของแม่สีในตารางสี โดยอาจเลือกสีเป็นแบบ 16 สีจาก 64 สี หรือ 16 สี จาก 262,144 สี หรือ 256 สี จาก 262,144 สี ขึ้นอยู่กับโหมด การแสดงผล สำหรับ true color ไม่มีการ

เลือกสี แสดงผลโดยการส่งค่าสี RGB ผ่าน D/A สีละ 8 บิต ออกไปเลย ความแตกต่างของการแสดงภาพสีและภาพขาวดำ คือ ภาพขาวดำจะต้องตั้งให้แม่สีทั้ง 3 สีมีค่าเท่ากัน เนื่องจาก VGA กำหนดให้แม่สีแต่ละสีใช้รีจิสเตอร์ (register) 6 บิต ทำให้แม่สีแต่ละแม่สีแสดงผลได้เพียง 34 ระดับเท่านั้น ยังผลให้เราแสดงภาพ 256 ระดับ ให้เห็นเพียง 64 ระดับเท่านั้น หากต้องการให้เห็นจริงทั้ง 256 ระดับต้องแสดงใน true color mode แล้วให้ RGB มีค่าเท่ากัน ซึ่งโหมดนี้ใช้รีจิสเตอร์ 8 บิต สำหรับแม่สีแต่ละสี

เกรย์เสกล (gray scale)

เกรย์สเกล หมายถึง ความแตกต่างของระดับความเข้มของแสง โดยเกรย์สเกลหนึ่งๆอาจแบ่งเป็น 13 ,20 หรือ 9 ระดับ โดยระดับที่ว่านี้ก็คือ ระดับสีเทา ในภาพหนึ่งๆถ้าต้องการแบ่งระดับความเข้มแสงหรือระดับสีเทาให้มีหลายๆค่านั้น จำเป็นอย่างยิ่งที่จะต้องเพิ่มจำนวนบิต ที่แสดงค่าของพิกเซล ตัวอย่างเช่น ถ้าต้องการภาพที่มีระดับสีเทา 4 ระดับ ต้องแทนด้วยเลขฐานสองจำนวน 2 บิต ถ้าต้องการภาพที่มีระดับสีเทา 16 ระดับ ต้องแทนด้วยเลขฐานสองจำนวน 4 บิต และถ้าต้องการภาพที่มีระดับสีเทา 256 ระดับต้องแทนด้วยเลขฐานสองจำนวน 8 บิต เป็นต้น

จำนวนระดับสีเทาที่ต้องการนี้ก็คือค่ายกกำลังของ 2 นั้นเอง ซึ่งค่าต่ำสุดหรือ 0 จะแทนสีดำหรือไม่มีความสว่างเลย และค่าที่มากที่สุดก็คือค่าที่น้อยกว่าจำนวนระดับสีเทาอยู่ 1 เช่น ค่า 15 ในระดับที่มีระดับสีเทา 16 ระดับก็จะเป็นสีขาวหรือสว่างมากที่สุด เป็นต้น

ในยุคแรกๆระบบการมองเห็น (vision system) จะใช้ระบบเลขฐานสองเพราะสะดวกต่อการนำเซนเซอร์มาใช้ นอกจากนี้การรวบรวมข้อมูล การเก็บรักษาข้อมูลยังสามารถทำได้ง่ายอีกด้วย

ในปัจจุบัน ไมโครโปรเซสเซอร์ที่ใช้กันโดยทั่วไป ขนาดที่เล็กที่สุดก็คือ 8 บิต ดังนั้นเกรย์สเกลขนาด 8 ค่า, 16 ค่า และ 256 ค่า จึงไม่เป็นปัญหาในการประมวลผล

ความสามารถในการแบ่งแยกระดับความแตกต่างของสายตามนุษย์ โดยทั่วๆไปจะอยู่ระหว่าง 10 ถึง 15 ระดับ ดังนั้น เกรย์สเกลขนาด 16 ระดับ จึงถือได้ว่าใกล้เคียงกับสายตามนุษย์หรืออาจจะละเอียดน้อยกว่าสายตามนุษย์บ้างเล็กน้อย (ในบางคน) ในขณะที่เกรย์สเกลขนาด 64 หรือ 256 นั้นละเอียดเกินไปสำหรับมนุษย์

การแสดงภาพนี้ใช้วิธีตั้งค่าของแม่สีในตารางสี โดยอาจเลือกสีเป็นแบบ 16 สี จาก 64 สี หรือ 16 สีจาก 262,144 สีหรือ 256 สีจาก 262,144 สี ขึ้นอยู่กับโหมดการแสดงผลสำหรับทรูคัลเลอร์ จะไม่มีการเลือกสี แสดงผลโดยการส่งค่าสี RGB ผ่าน D/A สีละ 8 บิต ออกไปความแตกต่างของการแสดงภาพสีและขาวดำ คือ ภาพขาวดำจะต้องตั้งให้แม่สีทั้งสามสีมีค่าเท่ากัน เนื่องจาก VGA กำหนดให้แม่สีแต่ละสีใช้ได้เพียง 64 ระดับสีเท่านั้นหากต้องการให้เห็นทั้ง 256 ระดับ ต้องแสดงในโหมดทรูคัลเลอร์ แล้วให้ RGB มีค่าเท่ากัน ซึ่งในโหมดนี้ จะสามารถใช้รีจิสเตอร์ได้ 8 บิต สำหรับแต่ละแม่สี

โดยทั่วไปวิธีการประมวลผลภาพเชิงเลขที่ทำให้คอมพิวเตอร์สามารถรู้จักวัตถุภายในภาพได้นั้นแบ่งได้สองระดับด้วยกันคือ การประมวลผลภาพในระดับต่ำ (Low level Image Processing) และการประมวลผลภาพในระดับสูง (High level Image Processing) การประมวลผลภาพระดับต่ำจะเป็นการ

ประมวลผลเชิงตัวเลขเกือบทั้งหมด เพื่อหาค่าตัวแปรต่างๆมาอธิบายข้อมูลภาพ โดยมีจุประสงค์เพื่อนำตัวแปรเหล่านั้นไปใช้ในการประมวลผลระดับสูงต่อไป

การประมวลผลภาพในระดับสูง เป็นการนำผลลัพธ์ที่ได้จากการประมวลผลระดับต่ำมาตีความหรือเพื่อให้คอมพิวเตอร์สามารถรู้จักและเข้าใจภาพได้สำหรับความแตกต่างของการประมวลผลภาพระดับต่ำและสูง คือ ข้อมูลที่นำมาใช้ในการประมวลผล โดยการประมวลผลภาพระดับต่ำจะใช้ค่าความสว่างของจุดโดยตรงส่วนการประมวลผลภาพระดับสูงนั้นข้อมูลภาพที่นำมาประมวลผลจะถูกแสดงในรูปของสัญลักษณ์ ซึ่งจะแสดงถึงสิ่งต่างๆที่อยู่ในภาพ เช่น ขนาด หรือ รูปร่างของวัตถุในภาพ

ฮีสโตแกรม (histogram)

ฮีสโตแกรมคือกราฟที่บอกให้ทราบถึงจำนวนของระดับสีเทาในภาพหนึ่งๆโดยที่แกน x จะเป็นค่าของระดับสีเทา และแกน y เป็นจำนวนของพิกเซล ในฮีสโตแกรมหนึ่งๆจะประกอบไปด้วย

1. จำนวนพิกเซลทั้งหมดของภาพ
2. จำนวนพิกเซลในแต่ละค่าของระดับสีเทา
3. กราฟที่แสดงจำนวนในแต่ละค่าของระดับสีเทา

กราฟที่ใช้ในฮีสโตแกรมจะเป็นกราฟแท่ง ซึ่งสามารถแสดงจำนวนพิกเซลในแต่ละค่าระดับสีเทาได้เป็นอย่างดี

รูปร่างหรือขนาดของฮีสโตแกรมจะเป็นข้อมูลที่แสดงคุณสมบัติของภาพว่า มีความคมชัด (ccnstrast) มากน้อยเพียงใด ซึ่งข้อมูลนี้ก็คือประโยชน์ของฮีสโตแกรมที่จะใช้ในการกำหนดค่าเทรชโฮล (threshold) ค่าเทรชโฮลนี้จะใช้ในการแปลงรูปภาพให้กลายเป็นภาพที่มีระดับความเข้ม 2 ระดับ คือ ขาวกับดำ หรือ "0" กับ "1" ตามที่ได้กล่าวมาแล้วในเรื่องพิกเซลนั่นเอง

การสร้างฮีสโตแกรม

1. ต้องกำหนดก่อนว่าภาพที่จะนำมาสร้างฮีสโตแกรมนั้น จะแบ่งเป็นกี่พิกเซล
2. สร้างพิกเซลเมตริกซ์จากพิกเซลเล็กๆ
3. นำค่าของพิกเซลในพิกเซลเมตริกซ์ที่ได้จากข้อ 2 มาสร้างตารางความสัมพันธ์ระหว่างค่าระดับสีเท่ากับจำนวนของพิกเซลในแต่ละค่าระดับสีเทาว่ามีกี่พิกเซล
4. นำค่าที่ได้จากตารางในข้อ 3 มาพล็อตเป็นกราฟแท่ง โดยแกนทางแนวนอนเป็นค่าระดับสีเทา และแกนทางแนวตั้งเป็นแกนของจำนวนของพิกเซล และกราฟนี้คือฮีสโตแกรมนั่นเอง

2.2 ความหมายและนิยามของภาพในระบบดิจิตอล

ภาพ (Image) ในเชิงคณิตศาสตร์จะหมายถึง ฟังก์ชัน 2 มิติ f(x,y) โดย x และ y เป็นแกนพิกัดในระนาบ 2 มิติ ค่าฟังก์ชั่น f(x,y) จะเป็นสัดส่วนกับความสว่างหรือความเข้มของภาพที่ตำแหน่ง x,y ซึ่งเราเรียกว่า ระดับสีเทา (Gray Scale) ซึ่งปกติเราจะให้จุดกำเนิดของแกนพิกัด (Coordinate) อยู่ทางมุมซ้ายของภาพ ภาพ 2 มิติที่แทนด้วยฟังก์ชั่น f(x,y) โดย x และ y เป็น แกนในระนาบของภาพ ค่าของฟังก์ชั่น

ที่จุด (x,y) คือความเข้มแสงที่จุดนั้น เนื่องจากแสงเป็นพลังงานรูปหนึ่ง ดังนั้น f(x,y) ต้องไม่เป็นศูนย์และมีค่า (finite) นั่นคือ

$$0 < f(x,y) < a \qquad (2.1)$$

โดยธรรมชาติของแสง ซึ่งจะต้องมีแหล่งกำเนิดแสงและส่วนที่สะท้อนของแสง ดังนั้นเราสามารถแยกฟังก์ชั่น f(x.y) ออกเป็นสองส่วนคือ อิลูมิเนชันคอมโพเนนท์ (illumination component) และรีเฟล็กแทนท์คอมโพเนนท์ (reflectant component) จะได้ว่า

$$f(x.y) = I(x.y)* r(x.y) \qquad (2.2)$$

เมื่อ

$$0 < I(x.y) < a \qquad (2.3)$$

และ

$$0 < r(x.y) < a \qquad (2.4)$$

สมการที่ 2.4 แสดงให้เห็นว่า ฟังก์ชันการสะท้อนถูกจำกัดขอบเขตระหว่าง 0 (ซึ่งหมายถึงการดูดซึมสมบูรณ์) และ 1 (ซึ่งหมายถึง การสะท้อนโดยสมบูรณ์) ธรรมชาติของ I(x.y) ขึ้นอยู่กับแหล่งกำเนิดแสง ในขณะที่ r(x,y) ขึ้นอยู่กับวัตถุที่สะท้อนแสงมาเข้าตา

ดังที่กล่าวมาแล้ว ความเข้มของภาพที่จุด (x.y) เราเรียกว่าระดับสีเทา (gray level) I จากสมการที่ (2.2) ถึง (2.4) จะเห็นว่า I อยู่ในช่วง

$$L_{min} < I < L_{max} \qquad (2.5)$$

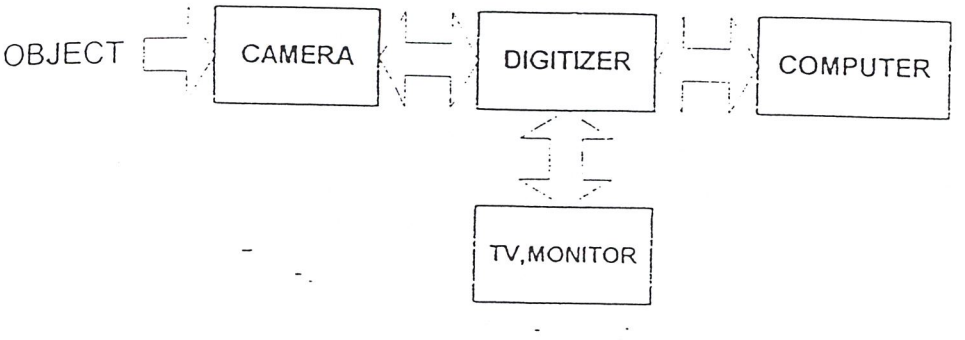
ในทางทฤษฎี L ต้องมีค่าเป็นบวก ในขณะที่ L ต้องมีค่าน้อยกว่า อนันต์ ในทางปฏิบัติ L = Lr และ L = L r ช่วงของ (L.L) เราเรียกว่าช่วงของระดับสีเทา ในทางปฏิบัติโดยใช้หลักคณิตศาสตร์ เรานิยมปรับช่วง (L.L) ให้เป็นช่วง (0.L) โดย L = 0 หมายถึง ดำสนิท และ L = 1 หมายถึงขาว

2.3 การแทนรูปภาพด้วยระบบดิจิตอล

ภาพดิจิตอล (digital image) เป็นภาพที่ถูกแปลงมาจากภาพอนาลอก อยู่ในรูปตัวเลข โดยภาพอนาลอกถูกแบ่งเป็นพื้นที่สี่เหลี่ยมเล็กๆที่เรียกว่าพิกเซล ในแต่ละพิกเซลจะถูกระบุตำแหน่งโดย (x.y) และค่าระดับสีเทาของพิกเซล นั้นคือค่าของ f(x.y)

2.4 ระบบการประมวลผลทางดิจิตอล

ระบบการประมวลผลทางดิจิตอลประกอบด้วย 3 ส่วนใหญ่ๆ คือ ส่วนเปลี่ยนสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล ซึ่งเรียกว่า ดิจิไตเซอร์ (Digitizer) ส่วนประมวลผล (Processing) และส่วนแสดงผล (Display) ดังแสดงในรูป



รูปที่ 2.1 ระบบการประมวลผลทางดิจิตอล

จากรูปที่ 2.1 ส่วนแรกคือ ส่วนที่เปลี่ยนสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล กล้อง (camara) เปรียบเสมือนดวงตาของมนุษย์ ทำหน้าที่เปลี่ยนภาพวัตถุมาเป็นสัญญาณทางไฟฟ้าและส่งให้ดิจิไตเซอร์ (Digitizer) ซึ่งทำหน้าที่เปลี่ยนสัญญาณไฟฟ้าให้เป็นสัญญาณดิจิตอล อุปกรณ์ส่วนนี้ได้แก่ กล้องโทรทัศน์ดิจิไตเซอร์ ซึ่งในภาพประกอบด้วยหลอดที่ทำหน้าที่เป็นสื่อไฟฟ้ากับความสว่างของภาพในตำแหน่งนั้นๆจากนั้น ทำการควอนไตซิ่ง (quantizing) ข้อมูลภาพที่ได้เป็นสัญญาณดิจิตอล

ส่วนประมวลผล คือ คอมพิวเตอร์ ซึ่งเปรียบเสมือนสมอง ทำหน้าที่ประมวลผลและวิเคราะห์ข้อมูลภาพ

ส่วนแสดงผลทำหน้าที่เปลี่ยนข้อมูลตัวเลข (ซึ่งมีระดับเป็นสีเทา) ที่เก็บเป็น อะเรย์ (array) ในคอมพิวเตอร์ ให้อยู่ในรูปที่เหมาะสม และสื่อความหมายกับมนุษย์ได้ คือเป็นภาพที่ปกติทั่วๆไป อุปกรณ์ในส่วนนี้ได้แก่ จอ (monitor) ทีวี เครื่องพิมพ์ดีดที่สามารถผลในรูปกราฟฟิกส์ได้

ภาพ 1 ภาพ ที่ถูกเปลี่ยนจากสัญญาณดิจิตอล สำหรับคอมพิวเตอร์ที่มีขนาดใหญ่ขึ้นอยู่กับความละเอียดของภาพที่ต้องการ และจะมีผลทำให้ใช้เนื้อที่ในหน่วยความจำมากในการเก็บข้อมูล ภาพ 1 ภาพ เช่น การเก็บภาพขนาด 256 *256 จุด ที่มีความแตกต่างของระดับความเข้มของแต่ละจุดเท่ากับ 256 ระดับ จะต้องใช้เนื้อที่ในหน่วยความจำถึง 64,000 ไบท์ ในการเก็บภาพนี้ ดังนั้นในปัจจุบันนี้ได้มีการวิจัยหาวิธีที่จะเก็บภาพด้วยคอมพิวเตอร์ โดยใช้เนื้อที่หน่วยความจำให้น้อยที่สุด และยังคงรักษาความละเอียดของภาพตามการใช้งานได้อีกด้วย

2.5 การสุ่มแบบสม่ำเสมอและควอนไตเซชัน (Uniform sampling and Quantization)

เพื่อที่จะประมวลสัญญาณภาพด้วยระบบคอมพิวเตอร์ ฟังก์ชันของภาพ f(x,y) จะถูกทำให้เป็นสัญญาณไม่ต่อเนื่อง ทั้งระนาบของภาพ ซึ่งเราเรียกว่า การสุ่มภาพ (Image Sampling) ของฟังก์ชันที่ได้เรียกว่า การควอนไตเซชันระดับสีเทา (gray level quantization)

สมมติว่าสัญญาณภาพต่อเนื่อง f(x,y) ถูกดิจิไตซ์ ในระนาบ X Y เป็นช่วงเท่าๆกัน เราสามารถจัด f(x,y) ให้อยู่ในรูปเมตริกซ์ขนาด NxN ได้ดังสมการ

$$f(x,y) = \begin{pmatrix} f(0,0)\ f(0,1)\ \ldots\ldots\ldots\ldots f(0,n-1) \\ f(0,1)\ f(1,1)\ \ldots\ldots\ldots\ldots f(1,n-1) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f(N-1,0)\ f(N-1,1)\ \ldots\ldots f(N-1,n-1) \end{pmatrix} \qquad (2.6)$$

ทางขวามือของสมการ จะเรียกว่า ภาพดิจิตอล และทุกๆสมาชิกของเมตริกซ์จะเรียกว่า พิกเซล จากขบวนการสร้างภาพดิจิตอลข้างต้น จะเห็นว่า เราต้องทราบขนาดของความละเอียดของภาพ NxN พิกเซล และจำนวนระดับของ Gray level ในทางปฏิบัติการทำควอนไตซ์เซชั่นในระบบดิจิตอล จะเป็นค่าของ 2 ยกกำลังจำนวนเต็ม คือ

$$N = 2^n \qquad (2.7)$$

$$G = 2^m \qquad (2.8)$$

เมื่อ G คือ จำนวนระดับ Gray level ดังนั้น จำนวนบิท (bit) ที่ใช้ในการเก็บภาพหนึ่งภาพที่ถูกดิจิไตซ์ คือ

$$B = N \times N \times m \text{ บิท} \qquad (2.9)$$

ดังตัวอย่างขนาด 128 x 128 pixels และระดับ Gray level ความจำขนาด 131,072 บิท

ตารางที่ 2.1 แสดงจำนวนไบท์ที่ใช้ในการเก็บภาพ เมื่อ N และ M เปลี่ยนไป

M / N	1	2	3	4	5	6	7	8
32	128	256	512	512	1024	1024	1024	1024
64	512	1024	2048	2048	4096	4096	4096	4096
128	2048	4096	8192	8192	16384	16384	16384	16384
256	8192	16384	32768	32768	65536	65536	65536	65536
512	32754	65536	131072	131072	262144	262144	262144	262144

2.6 เทคนิคต่างๆในการประมวลภาพ

เทคนิคต่างๆสำหรับการประมวลภาพ แบ่งเป็น 4 พวกใหญ่ๆ คือ

1. อิเมจดิจิไตเซชัน (image digitization)
2. อิเมจเอนฮานเมนต์และรีสตอเรชั่น (image enhancement and restoration)
3. อิเมจเอ็นโค๊ดดิ้ง (image encoding)
4. อิเมจคอนสตรัคชัน (image reconstruction)

2.6.1 อิเมจดิจิไตเซชัน (image digitization)

ดังที่ได้กล่าวมาแล้ว ถึงความหมายของการ digitize ภาพ ซึ่งความละเอียดของภาพที่ได้ก็ขึ้นอยู่กับการจัดระดับภาพ ในปัจจุบันเครื่องมือที่ใช้ทำกระบวนการนี้เรียกว่า ดิจิไตเซอร์ (digitizer) ดิจิไตเซอร์สามารถเปลี่ยนสัญญาณอนาลอกเป็นสัญญาณดิจิตอลได้ ดังนั้นข้อมูลที่ได้จึงเก็บเป็นเลขไบนารี โดยใช้ดิจิไตเซอร์เป็นตัวจัดการ

2.6.2 อิเมจเอนฮานเมนต์และรีสตอเรชั่น (image enhancement and restoration)

อิเมจเอนฮานเมนต์ เป็นการทำภาพให้อยู่ในรูปที่เหมาะสมขึ้น มีความคมชัดมากยิ่งขึ้น สำหรับการนำไปใช้งานเฉพาะอย่าง กล่าวคือ วิธีทำภาพ หรือปรับปรุงสภาพ X-ray อาจจะไม่เป็นวิธีที่ดี เมื่อนำมาปรับปรุงภาพถ่ายดาวเคราะห์ ที่ส่งมาจากการสำรวจทางอวกาศ

วิธีปรับปรุงคุณภาพของภาพ (enhancement) มีหลายวิธี ดังนี้

1. คอนทราสท์เอนฮานเมนต์ (Contrast enhancement) เป็นวิธีที่ทำให้ภาพคมชัดขึ้น โดยอาศัยวิธีฮีสโตแกรม อาจใช้แบบลีเนียร์สเทรท (linear strath) . พีซลีเนียร์สเทรท (piecewise linerar stretch) หรือ อีควอไลเซชัน (Equalization)

2. เอจเอนฮานเมนต์ (Edge enhancement) เป็นการแยกความแตกต่างของจุดภาพที่ใกล้เคียงกัน เพื่อหาขอบเขตของภาพ

3. การประมวลผลภาพสีเทียม (Pseudo - color image processing) เป็นการใช้เทคนิคของการทำ density slicing และการใส่สีเทียมให้กับภาพขาว - ดำ ที่มีระดับ Gray level ต่างๆกัน

อิเมจรีสตอเรชัน (Image restoration)

เป็นขบวนการในการสร้างภาพกลับคืน โดยการหาค่าชดเชย และแก้ความคลาดเคลื่อน เนื่องมาจากข้อมูลในภาพผิดพลาดไป หรือเป็นขบวนการสร้างภาพกลับคืน จากภาพที่ถูกทำให้เสียไป เนื่องจากปรากฏการณ์ต่างๆ โดยใช้หลักการของพีชคณิตเชิงเส้น (linear algrbra)

2.6.3 อิเมจเอ็นโค๊ดดิ้ง (image encoding)

เป็นการใช้เทคนิคต่างๆเพื่อเข้ารหัสข้อมูล เนื่องจากข้อมูลภาพที่ได้จะถูกเก็บในลักษณะเป็นจำนวนไบท์ ดังตารางที่ 2.1 ซึ่งถ้าภาพมีขนาดใหญ่ ก็ต้องใช้พื้นที่ในการเก็บมาก ด้วยข้อจำกัดของไมโครคอมพิวเตอร์ ที่มีขนาดหน่วยความจำจำกัด

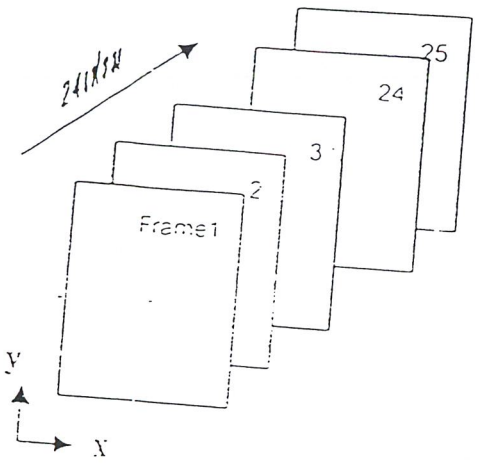
การเข้ารหัสข้อมูลจึงมีประโยชน์ ในด้านการลดพื้นที่ในการเก็บข้อมูลดังกล่าวมาก ผลของการเข้ารหัสข้อมูลนี้ เรียกว่าเป็นการลดข้อมูล (Data reduction หรือ Data compressing) ซึ่งเป็นเนื้อหาที่ทำโครงงานนี้ รายละเอียดของการลดข้อมูลภาพ ได้อธิบายในบทที่ 3 นอกจากนี้ การเข้ารหัสข้อมูลยังมีรหัสข้อมูลยังมีประโยชน์ในการช่วยลดปริมาณข้อมูลภาพ ที่ใช้ในระบบการสื่อสาร เช่น การส่งภาพถ่ายจากอวกาศมายังโลก การส่งข้อมูลภาพผ่านโมเด็ม (modem) เป็นต้น

2.6.4 อิเมจคอนสตรัคชัน (image reconstruction)

เป็นวิธีการสร้างภาพตัดขวางของวัตถุ โดยไม่ต้องผ่า หรือทำลายวัสดุ เพื่อประมวลผลโดยใช้คอมพิวเตอร์ เราเรียกการสร้างภาพตัดขวางด้วยคอมพิวเตอร์ว่า คอมพิวเตอร์โทโมกราฟฟี่ (Computer tomography)

สัญญาณข้อมูลภาพจากดิจิตอลวีดิโอ

การส่งสัญญาณข้อมูลภาพจากวีดิโอ จะมีลักษณะการส่งที่เป็นลำดับภาพเดี่ยวหรือเฟรม (frame) ที่ฉายต่อเนื่องกันดังรูปที่ 2.2 เช่น ระบบวีดิโอ NTSC จะส่งด้วยอัตราเร็ว 30 เฟรมต่อวินาที โดยดิจิตอลวีดิโอแต่ละเฟรมจะเป็นข้อมูลภาพดิจิตอลในลักษณะของเมตริกซ์ ซึ่งแต่ละจุดจะเรียกว่า พิกเซล มีค่าของระดับความเข้มสี โดยทั่วไปจะใช้เกรย์สเกลที่มีค่าตั้งแต่ 0 -255 โดย 0 แทนความมืดมากที่สุด และ 255 แทนความสว่างมากที่สุด



รูปที่ 2.2 ภาพการส่งสัญญาณวีดิโอที่อัตรา 24เฟรมต่อวินาที

2.7 ข้อมูลภาพชนิดบิตแมป

2.7.1 รูปแบบของไฟล์ข้อมูลชนิดบิตแมป

เป็นฟอร์แมทของวินโดว์บิตแมป (Bitmap) ซึ่งเป็นมาตรฐานสำหรับไฟล์กราฟิกบนวินโดว์ ซึ่งจะใช้ในการตัดต่อ หรือสำเนาภาพต่างๆลงบนคลิปบอร์ด (Clipboard) เมื่อเวลาจัดเก็บไฟล์ที่มีสกุล BMP ซึ่งเป็นฟอร์แมตที่ยังสามารถใช้เป็นวอลล์เปเปอร์ได้อีกด้วย

2.7.2 โครงสร้างของไฟล์ข้อมูลภาพชนิดบิตแมป

โครงสร้างของไฟล์ข้อมูลภาพชนิดบิตแมป จะประกอบด้วย 3 ส่วนคือ

1. ข้อมูลเฮดเดอร์ (Header)
2. ข้อมูลจานสี (Palette)
3. ข้อมูลภาพ (Data)

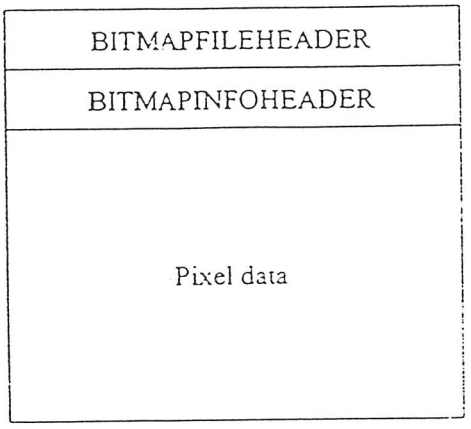
1. ข้อมูลเฮดเดอร์ คือ ข้อมูลที่อยู่บริเวณส่วนหัวของไฟล์ ซึ่งจะประกอบด้วยข้อมูลที่บอกรายละเอียดต่างๆของภาพ เช่น ความกว้าง ความยาวของภาพ จำนวนสี จำนวนบิต ความละเอียด

2. ข้อมูลจานสี คือ ข้อมูลที่บอกถึงชุดของจานสี ที่เกิดจากการผสมแม่สีทั้งสาม คือแดง เขียว และน้ำเงิน มาผสมกันได้เป็นสีต่างๆ ซึ่งถ้ามีจำนวนสีน้อยๆก็จะมีการเก็บค่าจานสีนี้ลงไฟล์ไปด้วย แต่ถ้าเป็นแบบ 24 บิต จะไม่มีค่าจานสี แต่จะใช้วิธีการเก็บค่าแม่สีลงไปเป็นข้อมูลแทน

3. ข้อมูลภาพ คือ ข้อมูลสีของภาพแต่ละจุดที่มาประกอบกันเป็นรูปภาพ ซึ่งค่าที่เก็บนี้จะเป็นค่าที่ใช้ใน การชี้ตาราง Palette หมายเลขอะไร เช่น จุดที่มีค่าเป็น 10 ก็ไม่เปิดดูตารางหมายเลข 10

สมมติว่า R = 0, G = 100, B = 0 ก็จะได้จุดนี้เป็นสีเขียว แต่ถ้าเป็นแบบ 24 บิต จะอ่านข้อมูลขึ้นมา 3 ค่า เป็นค่าแม่สี RGB แล้วนำไปผสมบนหน้าจอแทน

ไฟล์ข้อมูลชนิดบิตแมปมีโครงสร้างดังรูปที่ 2.3 แบ่งออกเป็น 3 ส่วน ได้แก่ Bitmapfileheader เป็นส่วนที่บอกข้อมูลของไฟล์ bitmapinfo เป็นส่วนที่แสดงขนาดและข้อมูลสีของภาพ ส่วนสุดท้ายคือ Pixel data เป็นส่วนเก็บข้อมูลสีแต่ละพิกเซล



รูปที่ 2.3 โครงสร้างของบิตแมปไฟล์

BITMAPINFOHEADER

ตารางที่ 2.2 แสดงข้อมูลใน Bitmapfileheader

Byte	Data	Detail
1 –2	File Type	Must be ASCII text "BM"
3 – 6	Size of file	In double word (32 - bit integer)
7 – 10	Reverse for future	Must be zero
11 – 14	Byte offset to bitmap data	Offset from bitmapfileheader

BITMAPINFO

โครงสร้างของ bitmapinfo เขียนได้เป็นดังนี้

```
typedef struct tagBITMAPINFO {//bmi
        BITMAPINFOHEADER        bmiHEADER ;
        RGBQUAD                 bmiColors [1] ;
}BITMAPINFO ;
```

BITMAPINFO ประกอบด้วย 2 ส่วนคือ BITMAPINFOHEADER เป็นส่วนที่บอกขนาดและข้อมูลสีของภาพบิตแมป และ RGBQUAD ซึ่งจะเก็บค่าตารางสีสำหรับเทียบค่าจากค่าของแต่ละพิกเซล

BITMAPINFOHEADER

โครงสร้างสามารถเขียนได้ดังนี้

```
typedef struct tagBITMAPINFOHEADER {//bmih
        DWORD           biSize :
        LONG            biWidth :
        LONG            biHeight :
        WORD            biPlanes :
        WORD            biBitCount :
        DWORD           biCompression ;
        DWORD           biSizeImage ;
        LONG            biXiPelsPerMeter ;
        LONG            biYiPelsPerMeter ;
        DWORD           biClrUsed :
        DWORD           biClrImportant ;
}BITMAPINFOHEADER ;
```

โดยแต่ละฟิลด์จะมีความหมายดังนี้

biSize	จำนวนไบต์ของ Header file
biWidth , biHeight	บอกขนาดความกว้าง และยาวของภาพในรูปของพิกเซล
biPlanes	เป็น 1 เสมอ
biBitCount	จำนวนบิตต่อ 1 พิกเซล
biCompression	แสดงการบีบอัดข้อมูล
biSizeImage	บอกขนาดของไฟล์
biXiPelsPerMeter	ความยาวแนวนอนในหน่วยพิกเซลต่อเมตร
biYiPelsPerMeter	ความยาวแนวตั้งในหน่วยพิกเซลต่อเมตร

biClrUsed	จำนวนสีในตารางสีที่จะถูกชี้ด้วยค่าพิกเซลในบิตแมป
biClrImportant	เป็นเลขที่แสดงว่าข้อมูลสีมีความสำคัญในการแสดงผลของบิตแมปถ้าเป็นศูนย์แสดงว่าทุกสีมีความสำคัญ

RGBQUAD
มีโครงสร้างดังนี้

```
typedef struct tag RGBQUAD {//rgbq
        BYTE    rgbBlue ;
        BYTE    rgbGreen ;
        BYTE    rgbRed ;
        BYTE    rgbReserved :
} RGBQUAD :
```

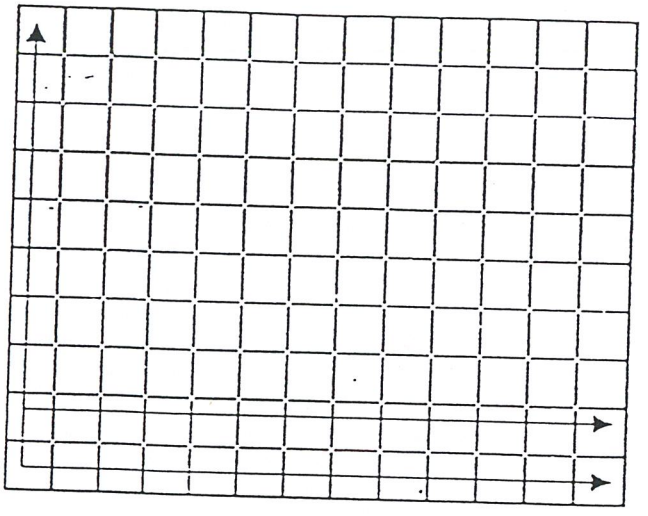
RGBQUAD เป็นโครงสร้างที่แสดงความเข้มของสีแดง เขียว และน้ำเงิน โดยมีความหมายของแต่ละฟิลด์ดังนี้

rgbBlue	แสดงความเข้มของสีน้ำเงิน
rgbGreen	แสดงความเข้มของสีเขียว
rgbRed	แสดงความเข้มของสีแดง
rgbReserved	ต้องมีค่าเป็น 0

ในส่วนของ bmiColors ของโครงสร้าง BITMAPINFO จะประกอบด้วยอาร์เรย์ของ RGBQUAD เพื่อเป็นตารางเปรียบเทียบสีของข้อมูลในแต่ละพิกเซล

Pixel data

เป็นส่วนเก็บข้อมูลสีของแต่ละพิกเซลของภาพ โดยข้อมูลแรกจะเป็นค่าสีของพิกเซลที่อยู่แถวล่างสุดที่ตำแหน่งซ้ายสุด ข้อมูลลำดับต่อไปจะเรียงทางขวาจากแถวล่างจนถึงแถวบนสุด



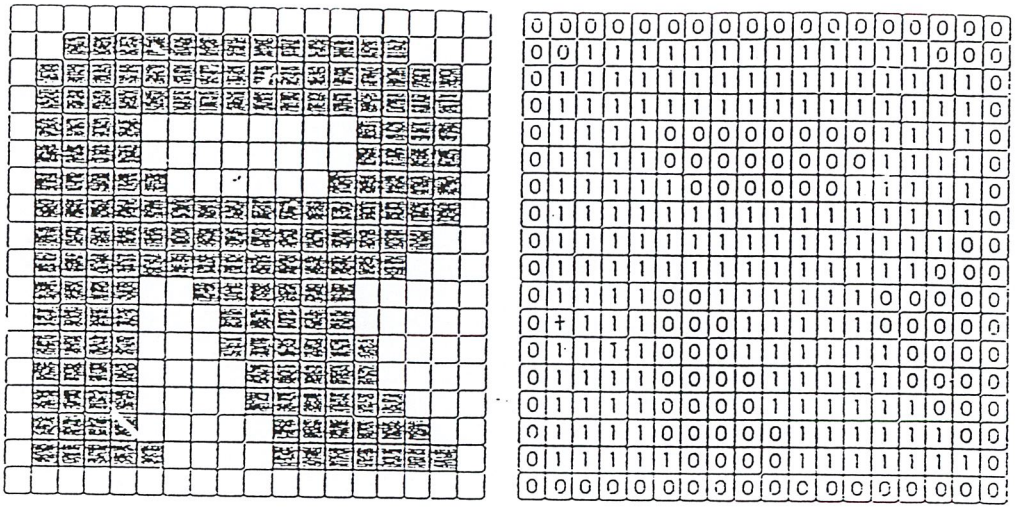
รูปที่ 2.4 แสดงการเก็บข้อมูลของแต่ละพิกเซล

2.8 การสร้างภาพไบนารี

การสร้างภาพไบนารี หมายถึง การแปลงข้อมูลภาพที่มีความเข้มหลายระดับ ให้เป็นภาพที่มีระดับความเข้มเพียง 2 ระดับ คือ 1 จุดภาพ มีค่าได้แค่ 2 ค่าเท่านั้น โดยเป็น 0 กับ 1 ซึ่ง 1 จะหมายถึง จุดภาพที่มีสีดำ และ 0 จะหมายถึงจุดภาพที่มีสีขาว การแปลงข้อมูลภาพหลายระดับไปเป็นภาพไบนารีจึงมีความจำเป็นและมีประโยชน์มากในการแสดงผลภาพที่มีระดับความเข้มของภาพหลายระดับบนอุปกรณ์ที่มีความสามารถในการแสดงผลได้ 2 ระดับ ประโยชน์อีกประการคือ การลดเนื้อที่การเก็บข้อมูลให้เหลือเพียง 8 บิต

จะเห็นได้ว่าการที่จะแก้ปัญหาการแสดงผลภาพที่มีความเข้มหลายระดับบนอุปกรณ์ที่สามารถแสดงผลได้ 2 ระดับนั้น จะทำการแปลงข้อมูลภาพให้เป็นภาพไบนารี่ (Binary Image) ก่อน ซึ่งการสร้างภาพไบนารี่ นั่นก็หมายถึงการแปลงข้อมูลภาพที่ระดับความเข้มหลายระดับ (Multilevel Image) ให้เป็นภาพที่มีระดับความเข้มเพียง 2 ระดับ นั่นคือ 1 จุดภาพมีได้ 2 ค่าเท่านั้น คือ 0 กับ 1 โดยจุดภาพที่แทนด้วย 1 หมายถึง จุดภาพที่มีสีดำ ส่วนจุดที่แทนด้วย 0 จะหมายถึงจุดภาพที่มีสีขาว เมื่อทำการแปลงเป็นภาพไบนารี่แล้วจึงนำภาพนั้นไปแสดงผลบนอุปกรณ์เหล่านั้น จะเห็นได้ว่าการแปลงข้อมูลภาพหลายระดับเป็นภาพไบนารีจึงมีความจำเป็นและมีประโยชน์มากในการแสดงผลภาพที่มีระดับความเข้มของภาพหลายระดับบนอุปกรณ์ที่มีความสามารถในการแสดงผลได้ 2 ระดับ สำหรับประโยชน์อีกประการหนึ่งในการแปลงข้อมูลภาพนั้นเป็นภาพไบนารี่คือการลดเนื้อที่การเก็บข้อมูลจะใช้เนื้อที่ในการเก็บ 8

บิต เมื่อสร้างเป็นภาพไบนารี่แล้วสามารถลดลงได้ถึง 8 เท่า นั่นคือ 1 จุดภาพจะใช้เนื้อที่ในการเก็บ 1 บิต อีกทั้งยังสามารถนำไปประยุกต์ใช้งานได้อย่างแพร่หลาย เช่น นำไปประยุกต์ใช้ในการวิเคราะห์เอกสาร ในขั้นตอนที่เรียกว่าการประมวลผลขั้นต้น (Preprocessing) เป็นต้น



รูปที่ 2.5 ภาพแบบไบนารี่และข้อมูลของแต่ละพิกเซล

ในการสร้างภาพไบนารี สามารถทำได้โดยใช้เทคนิคการทำเทรชโฮล (Thresholding Technics) โดยพิจารณาว่าจุดใดควรจะเป็นจุดขาวหรือจุดดำ จะกระทำโดยการเปรียบเทียบระหว่างจุดภาพเริ่มต้นกับค่าคงที่หนึ่งเรียกว่า ค่าเทรชโฮล เทคนิคนี้ใช้กันมากในกรณีที่ข้อมูลภาพมีลักษณะแตกต่างกันระหว่างวัตถุ และพื้นหลัง โดยค่าของจุดภาพใดๆที่มีค่าน้อยกว่าค่าเทรชโฮลจะถูกกำหนดให้เป็นค่า 1 (จุดสีดำ) และถ้าค่าของจุดภาพมีค่ามากกว่าค่าเทรชโฮลจะถูกเปลี่ยนให้เป็นค่า 0 (จุดสีขาว) ซึ่งการทำงานสามารถแสดงได้ดังสมการที่ 2.10

$$b(x,y) = \begin{cases} 1 : g(x,y) < Thr \\ 0 : g(c,y) \geq Thr \end{cases} \qquad (2.10)$$

b(x,y) ข้อมูลภาพผลลัพธ์เป็นภาพไบนารี่

g(x,y) ข้อมูลภาพอินพุทที่มีระดับความเข้ม 0 ถึง L ระดับ

Thr ค่าเทรชโฮลเป็นค่าคงที่ ระหว่าง 0 ถึง L ระดับ

1 จุดดำ

0 จุดขาว

โดยที่ L คือระดับความเข้มของจุดภาพสูงสุด

ในการสร้างภาพไบนารีโดยใช้เทคนิคเทรชโฮลเพื่อให้ได้ผลลัพธ์ที่ได้เหมาะสมและคมชัด สิ่งสำคัญที่สุด คือค่าเทรชโฮล เนื่องจากถ้าเลือกค่าเทรชโฮลที่ไม่เหมาะสม (ค่าเทรชโฮลที่มีค่าน้อยเกินไปหรือมากเกินไป) ภาพที่ได้อาจจะไม่เหมาะสม ขาดความคมชัดและรายละเอียดบางส่วนขาดหายไป กล่าวคือ ภาพที่ได้อาจจะมืดเกินไป หรือสว่างเกินไป หรือภาพที่ได้มีสิ่งรบกวน (Noise) เกิดขึ้น อันเป็นผลทำให้ภาพผลลัพธ์ที่ได้ไม่สวยงามเท่าที่ควร ดังนั้นปัญหาของการสร้างภาพไบนารีโดยใช้วิธีเทรชโฮลนี้คือ ทำอย่างไรจึงจะสามารถคำนวณหาค่าเทรชโฮลที่เหมาะสมสำหรับแต่ละภาพที่จะนำมาทำการสร้างภาพไบนารี ซึ่งมีวิธีการคำนวณหาค่าเทรชโฮลหลายวิธี โดยแต่ละวิธีเหมาะสมกับลักษณะการทำงานที่แตกต่างกันไป เช่น

2.8.1 การหาค่าเทรชโฮลโดยการกำหนดล่วงหน้า (Preassigned Threshold Value)

การหาค่าเทรชโฮลโดยวิธีการกำหนดล่วงหน้านี้เป็นวิธีที่ง่ายที่สุด เป็นการกำหนดค่าเทรชโฮลเองจากผู้ใช้ ซึ่งจะขึ้นกับประสบการณ์ของผู้ใช้นั้นๆ โดยการเลือกค่าคงที่ค่าหนึ่งจะเป็นค่าที่อยู่ระหว่างค่าต่ำสุดกับค่าสูงสุด ของระดับความเข้มของข้อมูลอินพุท เช่น ภาพข้อมูลอินพุท มีเกรย์สเกล 256 ระดับ จะมีค่าเกรย์สเกลได้ตั้งแต่ 0 - 255 เมื่อเลือกค่าเทรชโฮลได้แล้วสามารถสร้างภาพไบนารีได้

2.8.2 การหาค่าเทรชโฮลจากค่ากลาง (Mid Range Threahold Value)

การหาค่าเทรชโฮลโดยพิจารณาจากค่ากลาง เป็นการหาค่าเทรชโฮลที่แตกต่างจากการหาค่าเทรชโฮลวิธีแรก สำหรับวิธีนี้จะเป็นการคำนวณหาค่าาเทรชโฮลโดยอัตโนมัติโดยไม่ต้องให้ผู้ใช้เป็นผู้กำหนด โดยการหาค่าเทรชโฮลวิธีนี้ได้อาศัยการคำนวณพื้นฐานทางสถิติในเรื่องของการหาค่ากลางหรือค่าเฉลี่ย (mean) มาประยุกต์ใช้ ค่าเทรชโฮลที่คำนวณได้จากค่ากึ่งกลางที่อยู่ระหว่างค่าระดับความเข้มสูงสุด (Maximun Level) และระดับความเข้มต่ำสุด (Minimum Level) ของข้อมูลภาพอินพุท เมื่อทำการคำนวณค่าเทรชโฮลได้แล้ว ก็สามารถสร้างภาพไบนารีได้โดยนำค่าเทรชโฮลที่ได้มาใช้

สำหรับการหาค่ากึ่งกลางจะได้ดังสมการที่ 2.11

$$Thr = \frac{Max(g(x, y) + Min(g(x, y))}{2} \qquad (2.11)$$

โดยที่ Thr ค่าเทรชโฮล

g(x,y) ข้อมูลภาพอินพุท ที่มีระดับความเข้ม 0 ถึง L ระดับ

Max (g(x,y)) ค่าสูงสุดเกรย์สเกลของข้อมูลอินพุท

Max (g(x,y)) ค่าต่ำสุดเกรย์สเกลของข้อมูลอินพุท

การหาค่าเทรชโฮลจากค่าเฉลี่ยเลขคณิต หาได้จากสมการที่ 2.12

$$Thr = \frac{\sum gi(x, y)}{N \times N} \qquad (2.12)$$

2.9 การแยกวัตถุจากภาพ (Segmenttation)

กระบวนการสำคัญอีกขั้นตอนหนึ่งในการประมวลผลภาพเบื้องต้นก่อนที่จะนำไปสู่การจดจำรูปแบบ ก็คือ กระบวนการแยกวัตถุออกจากพื้นหลัง ซึ่งในที่นี้จะเป็นการแยกข้อมูลภาพที่เป็นตัวอักษรออกจากข้อมูลภาพทั้งหมด โดยแยกออกมาทีละตัวอักษรเพื่อนำมาสู่กระบวนการจดจำรูปแบบซึ่งสามารถประมวลผลได้ทีละหนึ่งตัวอักษรเท่านั้น

2.9.1 การแยกภาพด้วยการพิจารณาการต่อเนื่องของข้อมูล

เมื่อรับข้อมูลภาพที่ได้จากการเปลี่ยนข้อมูลเป็นรูปแบบไบนารี่ ที่มีค่า 0 กับ 1 เรียบร้อยแล้วซึ่งข้อมูล 0 จะแทน ส่วนที่เป็นพื้นหลัง และ 1 แทนส่วนที่เป็นตัวอักษร หลักการเบื้องต้นคือการหาค่าพิกเซลที่เป็น 0 ที่ต่อเนื่องกันตลอดแนวตั้ง และแนวนอนทำให้ได้ขนาดของกรอบ (Block) ข้อมูลภาพวัตถุที่มีขนาดต่างๆกัน จากนั้นจะพิจารณาเลือกขนาดของกรอบที่ต้องการจากความแตกต่างของจำนวนพิกเซล ความสูงความกว้าง และตำแหน่ง เป็นต้น ซึ่งจะได้กรอบของตัวอักษรที่ต้องการ

2.9.2 การแยกภาพด้วยวิธี Region labeling

ในการจำแนกภาพโดยวิธีนี้ได้ถือว่าบริเวณที่อยู่ข้างเคียงเป็นบริเวณที่สำคัญมาก จุดภาพที่อยู่ข้างเคียงกันมักจะมีคุณสมบัติทางสถิติที่คล้ายกันหรือใกล้เคียงกันสำหรับจุดรอบข้างที่มาเชื่อมต่อกัน ในวิธีนี้จะทำการพิจารณาถึงความเป็นเนื้อเดียวกันร่วมกัน การรวมตัวกันจะสิ้นสุดลงเมื่อพื้นที่ข้างเคียงไม่สามารถพิจารณาถึงความเป็นเนื้อเดียวกันได้ แต่ถ้าจุดภาพที่อยู่ใกล้เคียงกันนั้นตรวจสอบแล้วไม่อยู่ในเกณฑ์การรวม จุดภาพนั้นจะไม่ถูกรวมเข้าไปในส่วนนั้นของภาพแต่จะถูกเลือกให้เป็นจุดเริ่มต้นส่วนอื่นๆต่อไป และหลังจากที่จุดภาพทุกจุดได้รวมตัวกันเป็นกลุ่มเรียบร้อยแล้ว

ในกรณีนี้จะกล่าวถึงภาพที่มีวัตถุในภาพมาก วิธีการที่จะแยกแต่ละวัตถุออกจากกันจะทำได้โดยพิจารณาจากการติดกันของพิกเซลที่เป็น 1 โดยสามารถพิจารณาได้ดังนี้

- การติดกันแบบ 4 จุด จะพิจารณา 4 พิกเซล รอบข้างทางด้านแนวนอนและแนวตั้ง
- การติดกันแบบ 8 จุด จะพิจารณา 8 พิกเซล รอบข้างด้านแนวนอนและแนวตั้ง

วิธีการแยกวัตถุแบบ Region labeling นี้จะพิจารณาแบบไบนารีเฉพาะพิกเซลที่มีค่า 1 ทีละแถวจากซ้ายไปขวาและจากบนลงล่าง ซึ่งเมื่อพิจารณาที่พิกเซลใดพิกเซลหนึ่ง ถ้าพิกเซลแถวบนและทางซ้ายซึ่งผ่านการกำหนด Label แล้ว มีค่าไม่เป็น 1 ก็จะกำหนด Label ใหม่ให้กับพิกเซลนี้ ถ้าพิกเซลใดพิกเซลหนึ่งมีค่าเป็น 1 ก็จะกำหนด Label เหมือนกับพิกเซลข้างเคียงที่เป็น 1 แต่ถ้าในกรณีที่มีพิกเซลข้างเคียงมีค่า 1 มากกว่า 1 พิกเซล และแต่ละพิกเซลมี Label ต่างกันก็จะกำหนด Label ให้กับพิกเซลที่พิจารณาอยู่

เหมือนกับ Label ที่สมมูลกันให้เหมือนกัน ซึ่งจะสามารถทราบถึงความแตกต่างของแต่ละวัตถุในภาพโดยดูจาก Label ที่ต่างกัน

2.10 เทคนิคการติดตามรอยขอบภาพ (Countour Following)

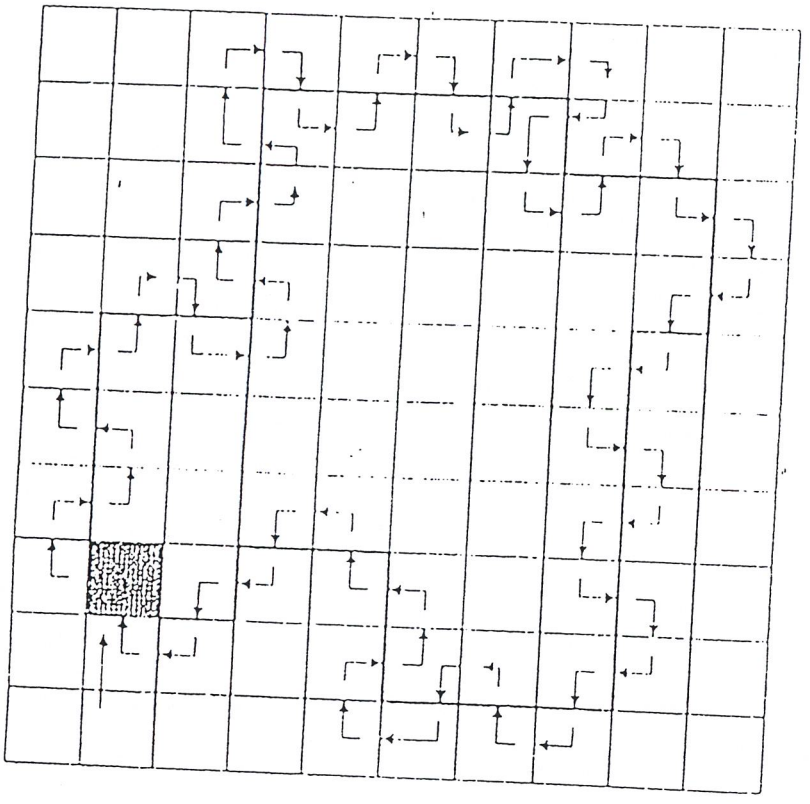
เทคนิคการติดตามรอยขอบภาพ ถูกนำมาใช้ในการแยกและคัดลอกส่วนของรูปภาพใดๆที่อยู่บนรูปใหญ่ ข้อมูลภาพที่จะนำมาประมวลผลด้วยเทคนิคนี้ต้องอยู่ในรูปของข้อมูลไบนารีนั่นคือ จุดภาพจะแสดงด้วยตัวเลข 0 กับ 1 เท่านั้น

การทำงานของเทคนิคการติดตามรอยขอบภาพ เป็นการเดินไต่ไปตามขอบระหว่างส่วนที่เป็นรูปภาพ กับส่วนที่เป็นพื้นเบื้องหลัง โดยจะตรวจกวาดไปทุกๆพิกเซล โดยจะเริ่มจากจุดมุมซ้ายของภาพตรวจกวาดไปในทิศทางจากซ้ายไปขวา และเลื่อนจากบนลงล่าง เมื่อตรวจกวาดมาพบจุดใดๆที่มีค่าของจุดภาพเป็น 1 ก็จะเปลี่ยนลักษณะการเคลื่อนที่ไปยังจุดถัดไปใหม่โดยมีเงื่อนไขการเคลื่อนที่ดังนี้

1. ถ้าจุดที่อยู่ปัจจุบันเป็นจุดของภาพหรือมีค่าของจุดเป็น 1 ให้เลี้ยวซ้ายแล้วก้าวเดินตรงไปข้างหน้ายังจุดถัดไป

2. ถ้าจุดที่อยู่ในปัจจุบันเป็นพื้นเบื้องหลัง หรือมีค่าของจุดเป็น 0 ให้เลี้ยวขวา แล้วก้าวเดินตรงไปข้างหน้าไปยังจุดถัดไป

3. การเคลื่อนที่จะสิ้นสุดลง เมื่อจุดที่อยู่ปัจจุบันเป็นจุดเดียวกันกับจุดเริ่มต้น



รูปที่ 2.6 แสดงลักษณะการเดินตามรอยขอบภาพ

ในรูปที่ 2.6 ได้แสดงลักษณะการทำงานของเทคนิคการติดตามรอยขอบภาพซึ่งจะแสดงการเคลื่อนที่ไปตามจุดต่างๆที่เป็นขอบภาพ เริ่มจากจุดที่ถูกแรเงาไว้ซึ่งเป็นจุดของภาพแรกที่ตรวจกวาดตามเงื่อนไขที่กำหนด เมื่อการเคลื่อนที่วนกลับมาถึงจุดเริ่มต้น ก็จะทราบถึงจุดที่เป็นขอบของภาพทั้งหมด ขณะที่เดินไต่ขอบนั้นก็จดจำค่าพิกัดไปด้วย เมื่อเดินถึงจุดเดิม ก็นำค่าพิกัดที่น้อยสุดและมาสุดมากำหนดขอบเขตของขนาดของวัตถุเพื่อใช้เปรียบเทียบกันต่อไป

บทที่ 3.
โปรแกรมภาษาเดลไฟล์ (Delphi)

Delphi เป็นเครื่องมือสำหรับสร้างแอพพลิเคชั่นสำหรับรันบน Windows 95/98/2000 ที่ผลิตโดยบริษัท Imprise (ชื่อเดิมก็คือ Borland) ซึ่งเป็นบริษัทที่แวดวงของนักพัฒนาแอพพลิเคชั่นรู้จัก และยอมรับในตัวผลิตภัณฑ์เป็นอย่างดี

Delphi นั้นเป็นเครื่องมือพัฒนาแอพพลิเคชั่นแบบ Visual Programming (เหมือนกับVisual .Visual C++, PowerBuilder ฯลฯ)ซึ่งทำให้เราสามารถเห็นผลลัพธ์การทำงานไปพร้อมๆ กับการลงมือสร้างแอพพลิเคชั่น

จุดเด่นที่สำคัญมากของความเป็น Visual Programming คือช่วยลดการสร้างแอพพลิเคชั่น นั้นเพราะแทนที่เราจะทุ่มเทเวลาไปปรับแต่งส่วนติดต่อผู้ใช้ หรืองานที่ไม่จำเป็น หรืองานซ้ำๆซากๆ เราก็มอบภาระเหล่านี้ให้ Delphi เสียสำหรับเราก็มุ่งเข้าไปแก้ไขปัญหาที่เป็นหัวใจการทำงานของแอพพลิเคชั่นดีกว่า

3.1ความสามารถของ Delphi 5.0

Delphi ได้พัฒนามาจนถึงเวอร์ชั้น 5.0 ซึ่งประกอบไปด้วยเครื่องมือช่วยในการออกแบบ ,สร้าง และทดสอบแอพพลิเคชั่นที่หลากหลาย ช่วยให้ผลงานออกมาได้อย่างรวดเร็ว ,สะดวกสบายจนน่าประหลาดใจ สำหรับความสามารถของ Delphi 5.0 ที่เราน่าจะรู้จักได้แก่

3.1.1 สร้างแอพพลิเคชั่นจาก VCL

การสร้างแอพพลิเคชั่นแบบ Visual Programming นั้นเกิดจากการนำเอาออบเจ็กซ์ต่างๆ มาประกอบกันเป็นแอพพลิเคชั่น ซึ่งออบเจ็กต์เหล่านั้นส่วนใหญ่จะอยู่ในรูปของส่วนติดต่อผู้ใช้ (User Interface) ซึ่งทำให้เราสามารถสร้างแอพพลิเคชั่นได้อย่างรวดเร็วใช้งานได้ง่ายและสวยงาม

3.1.2 เครื่องมือสร้างแอพพลิเคชั่นมีปริมาณมากเพียงพอ

Delphi ได้เข้ามาช่วยในการพัฒนาแอพลิเคชั่นเป็นเรื่องที่ไม่ยุ่งยากอีกต่อไป เพราะมีการรวบรวมเอาเครื่องมือที่จำเป็นต่างๆ ไว้ใช้งานกันอย่างครบถ้วน และเครื่องมือแต่ละตัวก็ยังมีความน่าใช้งานอีกด้วย

3.1.3 สร้างแอพพลิเคชั่นใช้งานฐานข้อมูล

Delphi นั้นได้ชื่อว่าเป็นเครื่องมือสำหรับสร้างแอพพลิเคชั่นเพื่อใช้งานร่วมกับฐานข้อมูลที่ได้รับการยอมรับเป็นอย่างมาก ซึ่งประกอบไปด้วยเครื่องมืออำนวยความสะดวกในการออกแบบ การสร้างและการทดสอบแอพพลิเคชั่นฐานข้อมูล

3.1.4 การสร้างแอพพลิเคชั่นใช้งานร่วมกับอินเตอร์เน็ต

อินเตอร์เน็ตนั้น ไม่ได้มีเพียงเว็บเพจที่สร้างมาจากภาษา HTML เท่านั้น การที่จะทำให้แอพพลิเคชั่นที่ใช้งานอินเตอร์เน็ตได้รับการยอมรับ นอกจากความสวยงามแล้ว สิ่งที่เพิ่มเข้ามาคือ ความสามารถในการทำงานของแอพพลิเคชั่นต้องสามารถประยุกต์ใช้กับงานต่างๆ ได้เป็นอย่างดี ซึ่ง Delphi5.0 ได้เตรียมความสะดวกสบายสำหรับการสร้างแอพพลิเคชั่นเพื่อใช้งานร่วมกับอินเตอร์เน็ตไว้อย่างมากมาย

3.2 จุดเด่นของ Delphi 5.0

ถ้าหากเราเคยใช้งาน Delphi ในเวอร์ชั่นที่ผ่านๆมา เราจะพบว่า Delphi 5.0 มีความสามารถเพิ่มมากกว่าที่เรารู้จักโดยเฉพาะความสามารถด้านการสร้างแอพพลิเคชั่นสำหรับอินเตอร์เนต อีกทั้งจุดเด่นของการปรับปรุงประสิทธิภาพของเครื่องไม้เครื่องมือ ระบบเอกสารช่วยเหลือให้เพิ่มขึ้นจนเป็นเครื่องมือที่มีประโยชน์ในการใช้งานมาก

3.2.1 ความเร็วของแอพพลิเคชั่นที่สร้างจาก Delphi5.0

Delphi นั้นได้ชื่อว่าเป็นคอมไพเลอร์โดยสมบูรณ์ในตัว ช่วยให้แอพลิเคชั่นที่สร้างขึ้นเป็นไฟล์ .EXE ที่ใช้งานได้ทันที (ไม่ต้องเสียเวลาแปล เหมือนแอพพลิเคชั่นในลักษณะสคริปต์ เช่น Visual Basic ที่ต้องแปลทุกครั้ง) และใช้งานได้ในทุกเครื่อง โดยไม่จำเป็นต้องมีไฟล์พิเศษเพิ่มเติมเลย ทำให้แอพพลิเคชั่นที่สร้างจาก Delphi มีขนาดเล็ก กินทรัพยากรของระบบน้อย จึงทำงานได้รวดเร็วมาก

3.2.2 ความสามารถด้านการเขียนโปรแกรม

การเขียนโปรแกรมนั้นเป็นเรื่องของความมีระบบระเบียบโดยธรรมชาติอยู่แล้ว แต่เมื่อความซับซ้อนของโปรแกรมมีมากขึ้นทำให้การเขียนโปนแกรมเป็นเรื่องที่ยุ่งยากอย่างหลีกเลี่ยงไม่ได้ ทำให้มีหลาย ๆ วิธีที่คิดขึ้นมาเพื่อลดความซับซ้อนลงซึ่ง Delphi ก็มีหลายวิธีที่ช่วยให้เราเขียน และทดสอบโปรแกรมที่เขียนได้อย่างสะดวกสบาย

3.2.3 ความสามารถด้านการตรวจสอบโปรแกรม

เมื่อความซับซ้อนของโปรแกรมมีมากขึ้น ความผิดพลาดย่อมเพิ่มมากขึ้นตามตัว ทำให้เราต้องมีวิธีการที่ช่วยให้การเขียนโปรแกรมกลั่นกรองเอาข้อผิดพลาดออกไปให้มากที่สุด เพื่อการทำงานที่ถูกต้องของแอพพลิเคชั่น ซึ่ง Delphi 5.0 มีความสามาถนี้อย่างเต็มเปี่ยม

และเมื่อแอพพลิเคชั่น กับผู้เขียนโปรแกรมนั้นอยู่คนละที่ เช่นในการทำงานแบบเครือข่าย หรือในอินเตอร์เนต นั่นการตรวจสอบและแก้ไขข้อผิดพลาดต่างๆ จึงเป็นเรื่องที่ยุ่งยากไม่ใช่น้อย แต่นับว่ายังโชคดีที่ Delphi มีความสามารถในการตรวจสอบ และแก้ไขข้อผิดพลาดจากระยะไกลให้เราใช้งานด้วย

3.2.4 ความสามารถด้านการสร้างแอพพลิเคชั่นอินเตอร์เน็ต

อินเตอร์เนตในวันนี้ไม่ใช้เรื่องที่เราจะมองข้าม ถือได้ว่าเป็นหนทางรอดของเราเลยทีเดียว Delphi 5.0 นั้นถึงกับเน้นความเป็นเครื่องมือพัฒนาแอพพลิเคชั่นสำหรับอินเตอร์เน็ตที่น่าใช้งาน จากเดิมที่เน้นที่การสร้างแอพพลิเคชั่นเกี่ยวกับฐานข้อมูลและแอพพลิเคชั่นใช้งานทั่วไป

3.2.5 เข้ากับเทคโนโลยีของไมโครซอฟท์มากขึ้น

อย่างที่ทราบกันว่าไมโครซอฟท์เข้าไปซื้อหุ้นในบริษัท Inprise และนั่นก็อาจส่งผลให้ Delphi ใกล้ชิดกับเทคโนโลยีของไมโครซอฟท์มากขึ้น แต่นั่นอาจไม่ใช่เหตุผลที่ถูกต้องทั้งหมด เราเองก็คงยอมรับกันว่าซอฟท์แวร์ที่ใช้งานส่วนใหญ่ รวมถึงมาตรฐานการใช้งานนั้นส่วนใหญ่จะเป็นไมโครซอฟท์ จึงเป็นเรื่องน่ายินดีด้วยซ้ำที่เราจะสร้างแอพพลิเคชั่นได้เข้ากับคนส่วนใหญ่ได้ดีกว่า Delphi เวอร์ชั่นที่ผ่านๆ มา

3.3 การใช้งานโปรแกรมภาษาเดลไฟล์

3.3.1 สร้างแอพพลิเคชั่นแบบ Event- Driven

แอพพลิเคชั่นที่สร้างจากDelphi นั้นมีวิธีการนั้นมีวิธีการสร้างที่แตกต่างจากการเขียนโปรแกรมแบบเดิมๆ ที่เราเคยเรียนรู้มาก่อน หากเราอาจเคยเขียนโปรแกรมาจากภาษา BASIC, Cหรือ Pascal ก็จะเห็นว่ามีแนวคิดที่ค่อนข้างต่างกัน ซึ่งเราจะเรียกวิธีการที่สร้างแอพพลิเคชั่น ด้วย Delphi ว่า Even-Driven (แปลว่า เหตุการณ์พาไป)

Even-Driven ที่จริงก็คือการเขียนโปรแกรมในลักษณะที่ว่า “ถ้ามีเหตุการณ์เกิดขึ้น เราจะจัดการกับมันอย่างไร” เช่นถ้าผู้ใช้คลิกที่ปุ่ม Exit เราจะทำอย่างไร? เราอาจจะ ถามผู้ใช้ว่าแน่ใจแล้วนะว่าจะจบการทำงานของโปรแกรม ถ้าผู้ใช้ยืนยันก็จบไป แต่ถ้าไม่ก็ให้แอพพลิเคชั่นทำงานต่อไป เป็นต้น

3.3.2 ระบบMessage Loop ของ Windows

Even-Driven ถือได้ว่าเป็นเหมือนหลักการทำงานของ Windows ก็ว่าได้ที่จริงแล้วเบื้องหลังการทำงานของWin-down ก็คือจะมีการรอรับการทำงานจากผู้ใช้, จากฮาร์ดแวร์ต่างๆ ส่งเหตุการณ์ต่างๆ เข้ามาสู่ Message Loop ซึ่งจะพิจารณาว่ามีเหตุการณ์อะไรเกิดขึ้น ซึ่งแต่ละเหตุการณ์ Windows ได้เตรียมวิธีการจัดการแต่ละเหตุการณ์เอาไว้แล้ว

จากแอพพลิเคชั่นที่ผ่านมาเราจะเห็นว่าแอพพลิเคชั่นประกอบไปด้วย วินโดว์,ปุ่มกด หรือตัวเลือก ซึ่งเราจะเรียกสิ่งต่างๆ ที่นำสร้างเป็นแอพพลิเคชั่นนี้ว่า ออบเจ็กต์(Object)

ตารางที่ 3.1 ส่วนประกอบของออบเจ็กต์ต่างๆ ของ แอพพลิเคชั่นตัวอย่างนั้นด้วยดังต่อไปนี้

ออบเจ็กต์	ลักษณะของออบเจ็กต์
ฟอร์ม(Form)	เป็นออบเจ็กต์ที่เรามองเห็นเป็นวินโดว์ ซึ่งมีออบเจ็กต์อื่นๆ วางอยู่ภายในมีข้อความอยู่ด้านบน
ปุ่มกด(Buttom)	เป็นออบเจ็กต์ที่ใช้การคลิกเมาส์เพื่อรับคำสั่งหรือกด Enter
ปุ่มตัวเลือก(Radio Button)	เป็นออบเจ็กต์ที่ให้เราคลิกเมาส์เพื่อเลือกตัวใดตัวหนึ่ง
รูปภาพ(Image)	เป็นออบเจ็กต์ที่ทำหน้าที่แสดงรูปภาพ

3.2.3 พรอพเพอร์ตี้(Property)

จะเห็นได้ว่าแต่ละออบเจ็กต์มีลักษณะเฉพาะตัวที่แสดงความเป็นออกเจ็กต์ เช่น ปุ่มกด ก็จะมีข้อความที่อยู่บนตัวมัน (Caption)เช่น จบการทำงาน หรือ บอกเวลา ซึ่งเราจะเรียกลักษณะเฉพาะตัวนี้ว่า พรอพเพอร์ตี้(Property)

ในออบเจ็กต์ชนิดเดียวกันก็จะมีพรอพเพอร์ตี้อย่างเดียวกัน แต่อาจจะมีค่าของพรอพเพอร์ตี้ที่แตกต่างกัน โดยเราสามารถกำหนดค่าพรอพเพอร์ตี้ได้ใน Object Inspector

ลักษณะการกำหนดพรอพเพอร์ตี้ใน Delphi นั้นมี 3รูปแบบ ได้แก่

1. การกำหนดค่าอย่างอิสระ- เช่นพรอพเพอร์ตี้ Name, พรอพเพอร์ตี้ Captionเป็นต้น
2. การกำหนดค่าจากตัวเลือกที่มีให้ใน Object Inspector- เช่น พรอพเพอร์ตี้ Cursor, พรอพเพอร์ตี้ BackColor เป็นต้น
3. การกำหนดค่าจากไดอะล็อบ็อกซ์- เช่นพรอพเพอร์ตี้Fontเป็นต้น

นอกจากการระบุค่าพรอพเพอร์ตี้จาก Object Inspector ซึ่งถือเป็นการกำหนดค่าก่อนการทำงานของแอพพลิเคชั้น(Design-Time)แล้ว เรายังสามารถกำหนดค่า หรือเปลี่ยนค่าพรอพเพอร์ตี้ได้โดยการเขียนลงไปในโปรแกรมของเรา ซึ่งก็จะทำให้มีการกำหนดค่า หรือเปลี่ยนแปลงค่าในขณะที่แอพพลิเคชั่นทำงาน(Run –Time) ได้เช่นกัน

3.2.4 เมธอด (Method)

นอกจากใช้พรอพเพอร์ตี้ในการบอกความแตกต่างของออบเจ็กต์แต่ละตัวแล้ว ยังมีสิ่งที่หนึ่งที่ออบเจ็กต์แต่ละตัวมักจะมีนั่นก็คือเมธอด (Method)

เมธอดก็คือ ความสามารถในการทำงานอย่างใดอย่างหนึ่งของออบเจ็กต์ เช่น ออบเจ็กต์Form มีความสามารถในการวาดวงกลมได้เป็นต้น ในการเรียกใช้เมธอดเราจะเรียกโดยการเขียนโปรแกรม

3.2.5 อีแวนต์ (Even)

อีแวนต์ ก็คือเหตุการณ์ที่เกิดขึ้นกับแต่ละออบเจ็กต์ เช่น ปุ่ม ก็จะมีเหตุการณ์ที่ผู้ใช้คลิกเมาส์(อีแวนต์ OnClick) หรือกรอบข้อความ จะมีเหตุการณ์เมื่อผู้ใช้กรอกข้อความ(อีแวนต์ OnKeyPress)

เราสามารถเขียนคำสั่งต่างๆ เพื่อจัดการกับเหตุการณ์ที่เกิดขึ้น โดยการเลือกเหตุการณ์ที่ต้องการจากแท็บ Events ใน Object Inspector

3.2.6 ออบเจ็กต์กับคอมโพเนนต์ของDelphi

ข้อดีของการเขียนโปรแกรมเพื่อสร้างแอพพลิเคชั่นใน Delphi ก็คือ เมื่อเราสมมุติสิ่งต่างๆเป็นออบเจ็กต์ที่มีพรอพเพอร์ตี้, เมธอดแล้ว เราก็สามารถนำมาใช้งานได้เรื่อยๆ (Resuable Object) ทำให้ไม่ต้องเสียเวลามาสร้างกันใหม่ทุกครั้งที่สร้างแอพพลิเคชั่น

Delphi จะเรียกออบเจ็กต์ต่างๆ ที่สามารถนำกลับมาใช้ในการสร้างแอพพลิเคชั้นว่าคอมโพเนนต์ (Component) ซึ่งจะถูกเก็บไว้ใน Component Palette พร้อมให้เรานำมาใช้งาน

3.2.7 ส่วนประกอบการทำงานของ Delphi

การทำงานเพื่อสร้างแอพพลิเคชั่นกับ Delphi 5.0 นั้นเราจะอยู่ภายในสภาวะแวดล้อมการพัฒนาแบบครบครัน นั่นคือประกอบด้วยเครื่องมือชนิดต่างๆที่นำมาสร้างและทดสอบแอพพลิเคชั่น

1. Menu Bar Menu Bar เป็นส่วนที่เก็บคำสั่ง เพื่อใช้เรียกงาน Delphi 5.0 ในรูปเมนู ซึ่งอยู่บนสุดของการแสดงผล

2. Form Designer Form Designer เป็นส่วนที่เราออกแบบหน้าตาของฟอร์ม ซึ่งถือเป็นพื้นที่แสดงผลหลัก ที่เราสามารถนำเอาออบเจ็กต์ต่างๆ มาวางไว้ภายในให้เป็นแอพพลิเคชั่นตามที่เราต้องการได้

3. Object Inspectorเราสามารถกำหนดค่าพรอพเพอร์ตี้ของออบเจ็กต์ต่างๆผ่านทาง Object Inspector นอกจากนี้เรายังเริ่มต้นจัการกับอีแวนต์ต่างๆ ของออบเจ็กต์ผ่านทางObject Inspector เช่นเดียวกัน

4. Component PaletteComponent Palette คือส่วนที่เก็บคอมโพแนนต์ต่างๆ แยกเป็นหมวดหมู่เอาไว้ พร้อมให้เรานำไปประกอบในแอพพลิเคชั่นที่จะสร้างขึ้น และไม่ใช่จะมีเพียงแต่คอมโพแนนต์ที่ Delphi 5.0 เตรียมไว้ให้เท่านั้น เรายังสามารถสร้างคอมโพแนนต์มาใช้งานเอง แล้วบรรจุไว้ในComponent Palette ได้ด้วย

5. Code EditorCode Editor เป็นส่วนที่เราใช้ในการเขียนคำสั่งต่างๆ เข้าไปเพื่อกำกับการทำงานของแอพพลิเคชั่นให้สอดคล้องกันซึ่งคำสั่งต่างๆ ในภาษาปาสคาล ที่อยู่ในCode Editor จะประกอบไปด้วยส่วนที่ Delphi เตรียมไว้ให้และส่วนที่เราต้องเขียนเพิ่มเติมเข้าไป โดยใน Code Editor เรามักจะเห็นCode Explorer ควบคู่ไปด้วย ซึ่ง Code Explorer จะช่วยให้เราเห็นตัวแปร,โปรแกรมย่อย,ออบเจ็กต์,พรอพเพอร์ตี้ เมธอดของออบเจ็กต์ต่างๆที่ประกอบกันเป็นแอพพลิเคชั่น

บทที่ 4
ไมโครคอนโทรลเลอร์ MCS-51

4.1 คุณสมบัติทั่วไปของไมโครคอนโทรลเลอร์ MCS- 51

คุณสมบัติทั่วไปที่สำคัญของไมโครคอนโทรเลอร์ตระกูล MCS -51 ดังนี้

- เป็นไมโครคอนโทรลเลอร์ ขนาด 8 บิต
- มีวงจรออสซิลเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
- มีขาสัญญาณอินพุตจำนวน 32 บิต
- สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (external data memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 K
- มีหน่วยความจำโปรแกรมภายในตัว (on-chip pro-gram memory) ขนาด 4K โดยเฉพาะเบอร์ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 8K สำหรับเบอร์ 831 และM 8032 จะไม่มีหน่วยความจำในส่วนนี้
- มีหน่วยความมูลภายในตัว (on-chip data memory)ขนาด 128 ไบต์ โดยเฉพาะเบอร์ 8032 และ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 ไบต์

ตารางที่ 4.1 แสดงคุณสมบัติของไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูล MCS-51

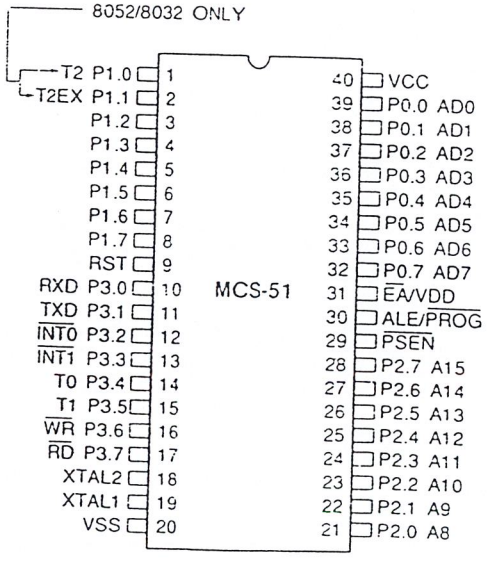
ชื่อเบอร์	หน่วยความจำภายใน		จำนวนไทเมอร์ / เคาน์เตอร์	จำนวนอินเตอร์รัปต์
	เก็บโปรแกรม	เก็บข้อมูล		
8052AH	8K x 8 ROM	256 x 8 ROM	3 x 16 – Bit	6
8051AH	4K x 8 ROM	128 x 8 ROM	2 x 16 – Bit	5
8051	4K x 8 ROM	128 x 8 ROM	2 x 16 – Bit	5
8032AH	ไม่มี	256 x 8 ROM	3 x 16 – Bit	6
8031AH	ไม่มี	128 x 8 ROM	2 x 16 – Bit	5
8031	ไม่มี	128 x 8 ROM	2 x 16 – Bit	5
8751H	4K x 8 EPROM	128 x 8 ROM	2 x 16 – Bit	5
8751H-12	4K x 8 EPROM	128 x 8 ROM	2 x 16 - Bit	5

- หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะบิตทำได้ง่าย ส่งผลการเขียนโปรแกรมทำได้ง่ายมากขึ้น

- มีไทเมอร์ / เคาน์เตอร์ (time-counter)ขนาด 16 บิตจำนวน 2ตัว โดยเฉพาะเบอร์ 8032หรือ 8052 จะมีไทเมอร์ / เคาร์เตอร์จำนวน 3 ตัว
- การอินเตอร์รัปต์สามารถทำได้จากรแหล่งกำเนิดโดยเฉพาะเบอร์ 8032 และ 8052 จะทำการอินเตอร์รัปต์ได้ จาก 6 แหล่งกำเนิดโดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้เป็น 2 ระดับ
- มีพอร์ตสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานเป็นแบบฟลูดูเพล็กซ์ (full duplex)
- มีคำสั่งในการคำนวณทางคณิตศาสตร์และทางตรรกศาสตร์
- คำสั่งโดยส่วนใหญ่ใช้เวลาทำการเพียง 1 ไมโครวินาที เมื่อใช้ คริสตอลความถี่ 12 เมกะเฮริตซ์
- ต้องการแหล่งจ่ายไฟ 5 โวลต์เพียงชุดเดียว

4.2 โครงสร้างภายนอก ของ MCS-51

ไมโครคอนโทรเลอร์ตระกูล MCS- 51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูปที่ 4.1 สำหรับหน้าที่การใช้งานของแต่ละขามีดังนี้



รูปที่ 4.1 แสดงการจัดตำแหน่งขาต่างๆ ของไมโครคอนโทรลเลอร์ตระกูล MCS-51

- ขา V_{CC} เป็นขาป้อนแรงดันไฟเลี้ยง + 5 โวลต์
- ขา V_{SS} เป็นขากราวด์
- ขาพอร์ต O (Port O) มี 8 ขาได้แก่ $P_{0.0}$ - $P_{0.7}$ เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตเหล่า ค่า 1 ไปยังพอร์ตแต่ละบิตของพอร์ตเพื่อ

กำหนดให้ขาพอร์ตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่สามารถนำมาใช้เป็นพอร์ตอินพุตอิมพีแดนซ์สูงได้ นอกจากพอร์ตนี้ใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้านที่ในการกำหนดตำแหน่งแอดเดรสไบต์ต่ำ (A_0-A_7) ซึ่งจะใช้งานเป็นแบบมัลติเพล็กซ์กับการรับส่งข้อมูลขนาด 8 บิต (D_0- D_7)

- ขาพอร์ต 1 (Port 1) มี 8 ขา ได้แก่ $P_{1.0}$-$P_{2.7}$ เป็นขาพอร์ตอินพุตเอาต์พุตพอร์ตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียน ค่า 1 ค่า ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากนี้สำหรับเบอร์ 8032 และ 8052 ขาพอร์ต $P_{1.0}$ และ $P_{1.1}$ จะถูกนำมาใช้เป็นขาT2 และ T2EX ตามลำดับด้วย

- ขาพอร์ต 2 (Port2) มี 8 ขาได้แก่ ขา $P_{2.0}$-$P_{2.7}$ เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้ถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบต์สูง (A8 -A_{15})

- ขาพอร์ต 3 (Port 3)มี8 ขา ได้แก่ P 3.0- P 3.7 เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้ามใช้งานเป็นอินพุตพอร์ต ต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะใช้เป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในหน้าที่พิเศษต่างๆ ดังแสดงในตารางที่ 4.2

- ขารีเซต(RST)ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสถานะเป็น1 อย่างน้อย 2 แมชีนไซเกิล ในขณะที่ออสซิลเตอร์ยังทำงานอยู่

- ขา ALE/ $\overline{\text{PROG}}$ เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตซ์(latch) ค่าตำแหน่งแอดเดรสไบส์ต่ำ(ad-dress Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่แนอินพุตรับพัลส์ในการโปรแกรม (Program pulse input)ในส่วนของหน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ ในตระกูล MCS-51 ที่มีหน่วยความจำโปรแกรมภายในเป็น EPROM

ตารางที่ 4.2 แสดงหน้าที่พิเศษของแต่ละขาของพอร์ต P_3

ขาพอร์ต	หน้าที่พิเศษ
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interupt 0)
P3.3	$\overline{\text{INT1}}$ (external interupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

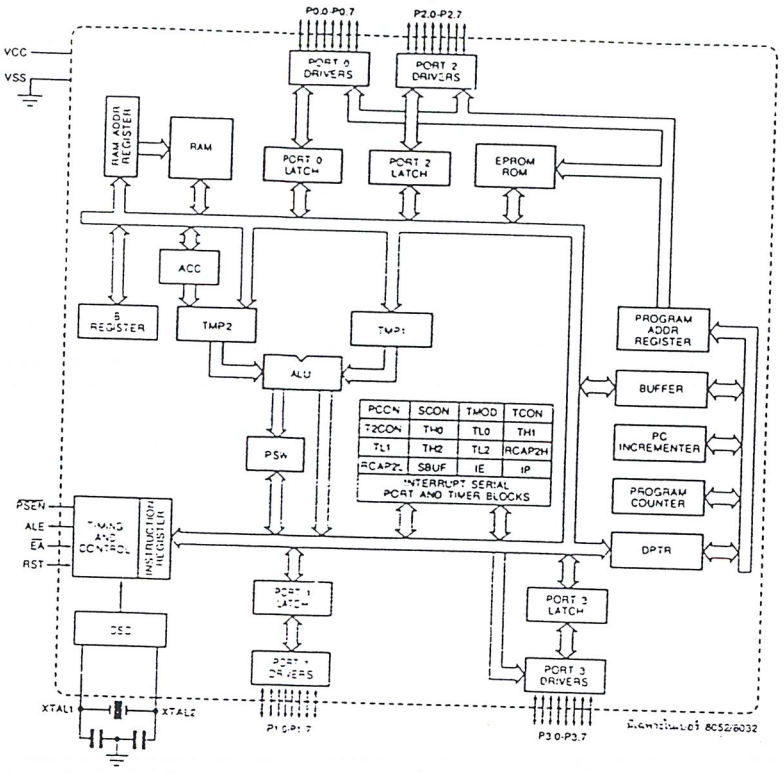
- ขา $\overline{\text{PSEN}}$(Program Store Enable) ทำหน้าที่เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรบจำนวน 2 ครั้งในแต่ละแมชชีนไซเกิล แต่ในขณะที่ติดต่อกับหน่วยความจำภายนอกจะไม่มีการส่งสัญญาณสโตรบแต่อย่างใด

- ขา $\overline{\text{EA}}$ / VPP (External Access enable / VPP) เป็นขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมจากภายในหรือจากภายนอก โดยถ้ามีสถานะเป็น 0 จะหมายถึงให้ไมโครคอนโทรลเลอร์รับคำสั่งจากหน่วยความจำภายนอกที่ตำแหน่งแอดเดรส 0-0FFFH (0-1 FFFH ถ้าเป็นเบอร์ 8052) อย่างไรก็ถ้าบิตป้องกัน(security bit) ในหน่วยความจำEPROM ถูกโปรแกรมไว้ ไมโครคอนโทรลเลอร์จะไม่รับคำสั่งจากหน่วยความจำภายนอกเลย นอกจากนี้ขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (V PP) ขนาด 21 โวลต์ เพื่อใช้ในระหว่างโปรแกรม EPROM

- ขา XTAL 1 และ ขา XTAL 2 เป็นขาอินพุตและเอาต์พุตของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (inverting oscllator amplifier) สำหรับใช้ต่อร่วมกับคริสตอล ภายนอก

4.3 โครงสร้างภายในของ MCS-51

โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แสดงดังในรูปที่ 2.2 โดยส่วนที่มีเครื่องหมายดอกจัน(*) จะมีเฉพาะในเบอร์ 8032 และ 8052 เท่านั้น



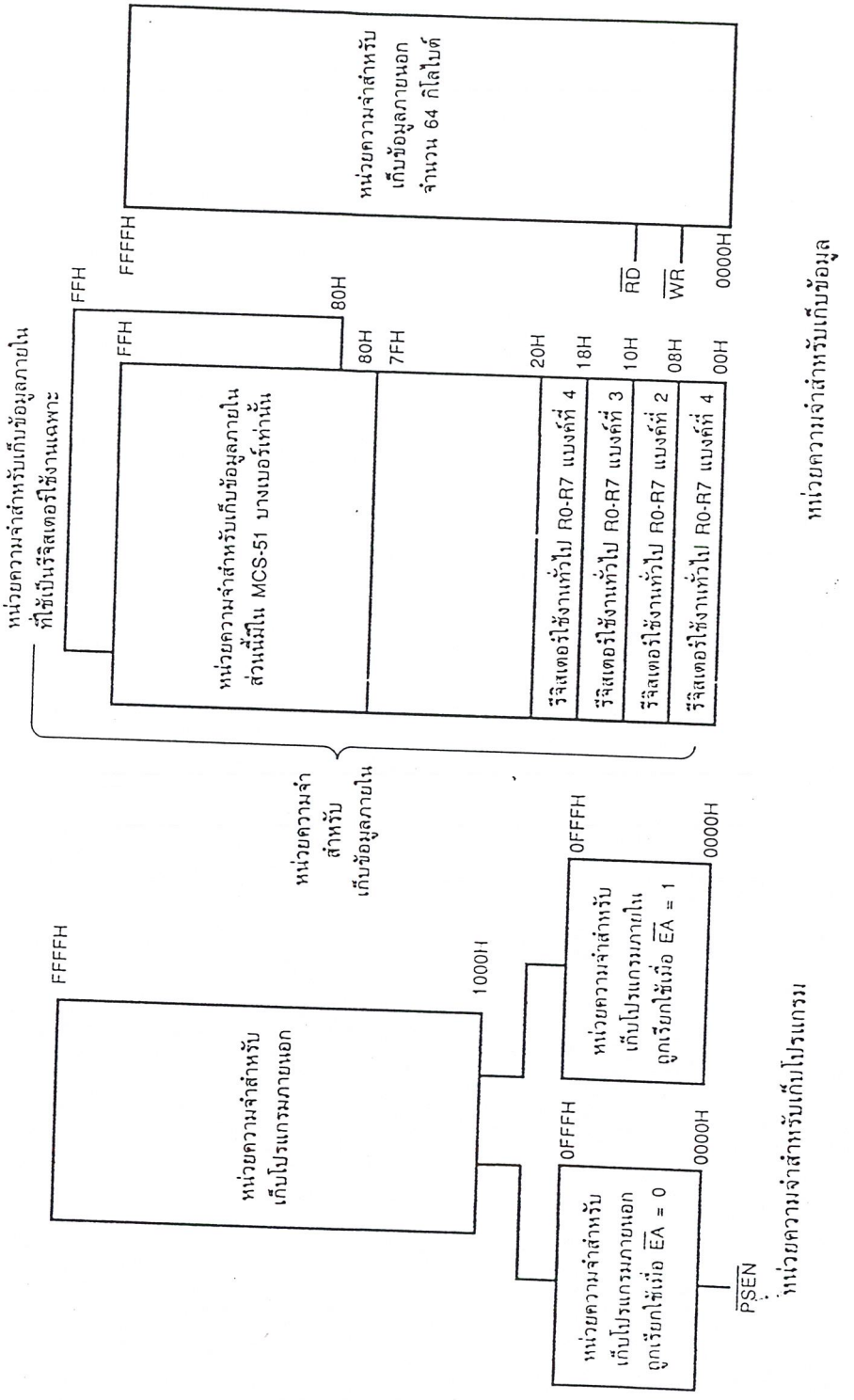
รูปที่ 4.2 แสดงโครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51

4.3.1การจัดหน่วยความจำ

ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบ่งชนิดหรือหน้าที่ของหน่วยความจำออกเป็น 2 ส่วนคือหน่วยความจำโปรแกรม (program memory) และหน่วยความจำข้อมูล (data memory)

หน่วยความจำโปรแกรมจะใช้สำหรับเก็บโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ ซึ่งบางเบอร์จะมีหน่วยความจำในส่วนนี้อยู่ภายในตัว โดยอาจจะมีขนาดไม่เท่ากันหรือเป็นหน่วยความจำในส่วนนี้เลย โปรแกรมการทำงานจะถูกเก็บไว้ยังหน่วยความทรงจำโปรแกรมภายนอกทั้งหมด

สำหรับหน่วยความจำข้อมูลจะใช้เก็บข้อมูลหรือค่าตัวแปรต่างๆ จากการทำงานของโปรแกรม ซึ่งใน MCS-51 ทุกเบอร์จะมีหน่วยความจำในส่วนนี้อยู่จำนวนหนึ่ง แต่อาจมีขนาดมากน้อยต่างกันไปในแต่ละเบอร์สำหรับการจัดโครงสร้างของหน่วยความจำ ทั้งในส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลแสดงไว้ในรูปที่ 4.3



รูปที่ 4.3 แสดงการจัดโครงสร้างของหน่วยความจำทั้งในส่วนของหน่วยความจำโปรแกรม
และ หน่วยความจำข้อมูล

4.3.1.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมสามารถแบ่งออกได้เป็น2ส่วนคือ หน่วยความจำโปรแกรมและหน่วยความจำโปรแกรม หน่วยความจำโปรแกรมภายในจะถูกใช้งานถ้าขาสัญญาณถ้าขาสัญญาณEA มีค่าเป็น 1 โดยจะถูกใช้งานถ้าขาสัญญาณ EA มีค่าเป็น 1 โดยจะถูกใช้งานในช่วงแอดเดรส 0-0FFFH (หรือช่วงแอดเดรส0-1FFFH ในเบอร์ 8052) นอกเหนือจากช่วงแอดเดรสนี้จะใช่หน่วยความจำโปรแกรมภายนอกทั้งหมด ในกรณีตรงกันข้ามถ้าขาสัญญาณ EA มีค่าเป็น 0 ในช่วง แอดเดรส 0-0FFFH(หรือช่วงสัญญาณ 0-1 FFFHในเบอร์ 8052) จะถูกใช้จากหน่วยความจำภายนอก หรือกล่าวได้ว่าถ้าขาสัญญาณ EA มีค่าเป็น 0 จะเป็นการเลือกใช้หน่วยความจำโปรแกรมภายนอกทั้งหมดตลอดช่วงแอดเดรส

4.3.1.2 หน่วยความจำข้อมูล

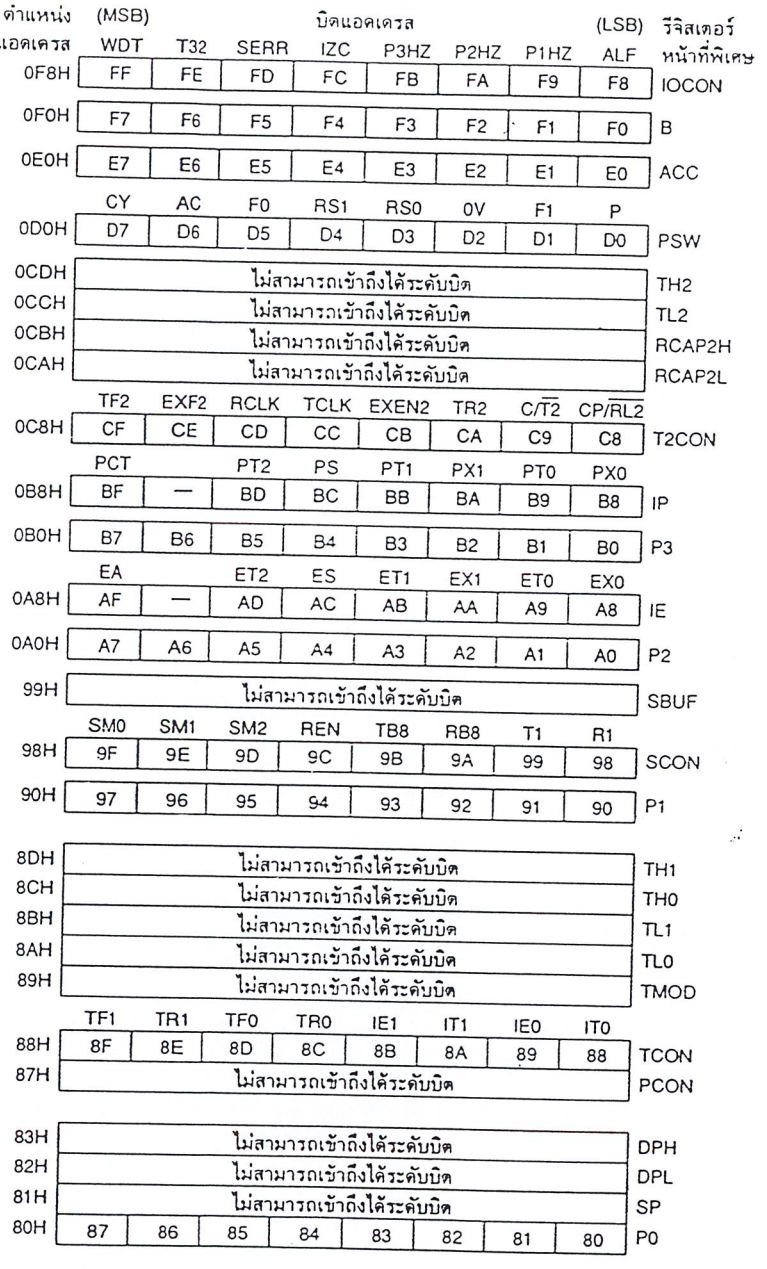
หน่วยความจำข้อมูลสามารถแบ่งออกได้เป็น 2 ส่วน คือหน่วยความจำข้อมูลภายในและหน่วยความจำข้อมูลภายนอก สำหรับหน่วยความจำข้อมูลภายในข้อมูลยังแบ่งออกได้เป็น 2ส่วนย่อยคือส่วนที่ใช้เก็บข้อมูลทั่วไปและหน่วยที่ใช้เป็นรีจิสเตอร์หน้าที่พิเศษหรือ SFR (Special Function Register) โดยส่วนที่ใช้เก็บข้อมูลทั่วไปจะถูกใช้สำหรับเก็บข้อมูลหรือคาตัวแปรต่างๆ จากการทำงานของโปรแกรม ส่วนรีจิสเตอร์หน้าที่พิเศษจะถูกใช้งานเป็นรีจิสเตอร์ควบคุมการทำงานและบอกสถานะของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในขนาด 158 ไบต์เป็นอย่างน้อย และบางเบอร์อาจมีถึงขนาด 256 ไบต์

4.3.2 รีจีสเตอร์

4.3.2.1 รีจิสเตอร์หน้าที่พิเศษ (SFR)

รีจิสเตอร์หน้าที่พิเศษ มีบทบาทอย่างมากในการควบคุมการทำงานของไมโครคอนโทรลเลอร์ และทำให้การเขียนโปรแกรมสามารถทำได้สะดวกมากขึ้น รีจิสเตอร์หน้าที่พิเศษที่สำคัญคือ การควบคุมการทำงานในส่วนต่างๆ ภายในไมโครคอนโทรลเลอร์และทำหน้าที่แสดงสถานะการทำงาน ซึ่งในรีจิสเตอร์หน้าที่พิเศษบางตัวยังสามารถเข้าถึงในระดับบิต (bit addressable)ด้วย ดังแสดงรูปการจัดหน่วยความจำและตำแหน่งของริจิสเตอร์หน้าที่พิเศษต่างๆ ในรูปที่ 4.4



รูปที่ 4.4 แสดงการจัดหน่วยความจำ และ ตำแหน่งของรีจีสเตอร์หน้าที่พิเศษต่างๆ

4.3.2.2 รีจิสเตอร์ใช้งานทั่วไป

รีจิสเตอร์ใช้งานทั่วไปมีไว้สำหรับให้ผู้เขียนโปรแกรมสามารถนำข้อมูลไปพักไว้ชั่วคราว หรือใช้งานทั่วไปได้ตามต้องการ ซึ่งรีจิสเตอร์ใช้งานทั่วไปนี้มีอยู่ด้วยกัน 8 ตัวคือรีจิสเตอร์ R_0- R7 โดยรีจิสเตอร์ทั้ง 8ตัวถูกจัดให้อยู่รวมกันและมีให้เลือกใช้ถึง 4 แบงค์ (bank)นั่นคือมีรีจิสเตอร์ใช้งานทั่วไปถึง 32 ตัวให้ใช้งาน เพียงแต่การเลือกใช้รีจิสเตอร์ R_0- R7 ในแบงค์ใดแบงค์หนึ่งจะถูกกำหนดจากบิต RSO . RS1 ในรีจิสเตอร์หน้าที่พิเศษ PSW ดังนั้นการเลือกใช้จึงเลือกได้เพียงแบงค์เดียวในขณะใดขณะหนึ่ง อย่างไรก็ตามค่าข้อมูลที่เก็บไว้ในรีจิสเตอร์ที่มีชื่อเดียวแต่อยู่คนละแบงค์จะไม่มีผลซึ่งกันและกันเลย ทำให้ผู้เขียนโปรแกรมใช้งานรีจิสเตอร์ทั่วไปได้ทั้ง 32 ตัว อย่างเต็มที่และไม่ยุ่งยากในการเขียนโปรแกรม

4.3.2.3 การใช้งานรีจิสเตอร์

โดยปกติไมโครคอนโทรเลอร์ตระกูล MCS-51 จะทำการประมวลผลข้อมูล ครั้งละ 1 ไบต์ ซึ่งกระทำเก็บรีจิสเตอร์ภายในที่รีจิสเตอร์แต่ละตัวเก็บข้อมูลได้ขนาด 1 ไบต์เช่นกัน เช่นรีจิสเตอร์ A ซึ่งเป็นอแกคิวมูเลเตอร์ (ac-cumulator) ทำหน้าที่เป็นรีจิสเตอร์กลางสำหรับการคำนวณทางคณิตศาสตร์หรือทางลอจิกของตัวกระทำ 2 ตัว ตัวอย่างเช่น ถ้าต้องการบวกค่า 10 กับข้อมูลตัวหนึ่ง ให้ทำการโหลดข้อมูลไปเก็บไว้ในรีจิสเตอร์ A ก่อน จากนั้นให้ใช้คำสั่งนำค่า 10 ไปบวกกับA ผลที่ได้จากการบวกข้อมูลและค่า 10 จะถูกเก็บไว้ในรีจิสเตอร์ A นอกจากรีจิสเตอร์ A ทำการบวกด้วยการกำหนดค่าโดยตรงแล้ว มันยังทำการคำนวณร่วมกับรีจิสเตอร์ขนาด 8 บิตตัวอื่น ๆ ได้อีกด้วย

ทั้งไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์จะมีรีจิสเตอร์สำหรับใช้งานในคำสั่งพิเศษ โดยผู้เขียนโปรแกรมอาจกำหนดขึ้นได้เอง โดยที่กำหนดให้อยู่ในตอแหน่งแอดเดรสพิเศษ ในที่นี้มีค่ามากกว่า 07FHขึ้นไป ตัวอย่างเช่นแอกคิวเตอร์ถูกกำหนดให้ใช้หน่วยความจำภายในที่0E0Hรีจิสเตอร์เหน่านี้เรียกว่า รีจิสเตอร์หน้าที่พิเศษ (spe- cial function registers หรือ SFR_S) จำนวนของรีจิสเตอร์พิเศษอาจจะมีไม่เท่ากันในไมโครคอนโทรลเลอร์แต่ละเบอร์ ในตระกูล MCS-51 ขึ้นอยู่กับคำสั่งที่ทำการตั้งค่าไว้ เพราะริจิสเตอร์พิเศษเหล่านี้ถูกรวมอยู่หรือใช้พื้นที่ในส่วนของหน่วยความจำภายในของไมโครคอนโทรลเลอร์ ซึ่งหน่วยความจำภายในหรือแรมภายในจะมีขนาดไม่เท่ากันในแต่ละเบอร์

นอกจากรีจิสเตอร์พิเศษหรือ SFR แล้วยังมีรีจิสเตอร์ สำหรับใช้ในงานทั่งไปอีก 8 ตัว คือรีจิสเตอร์ R 0ถึงR 7 รีจิสเตอร์ ทั้ง 8 ตัว ถูกบรรจุอยู่ภายในแรมภายในไมโครคอนโทรลเลอร์ในรูปของแบงค์ (bank) และใช้สำหรับเก็บข้อมูลชั่วคราวระหว่างการประมวลผล ในที่นี้จะใช้รีจิสเตอร์เฉพาะแบงค์ศูนย์เท่านั้น และหลังจากรีเซตระบบทุกครั้งรีจิสเตอร์ที่แบงค์ศูนย์จะถูกเลือกโดยอัตโนมัติ

4.4 การจัดสรรหน่วยความจำบน MCS-51 บอร์ด

ไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูล MCS-51 มีขนาดของหน่วยความจำไม่เท่ากัน ทำให้การจัดสรรพื้นที่ในหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลที่แตกต่างกัน

4.4.1หน่วยความจำสำหรับเก็บโปรแกรม

หน่วยความจำสำหรับเก็บโปรแกรมสามารถขยายได้สูงถึง 64 กิโลไบต์มีหน้าที่เก็บคำสั่งต่างๆ สำหรับไมโครคอนโทรลเลอร์ มันสามารถใช้เก็บตารางข้อมูลและค่าคงที่ได้ในการใช้งาน ในที่นี้จะใช้งานไมโครคอนโทรลเลอร์โดยใช้หน่วยความจำโปรแกรมภายนอกเท่านั้น ที่ขา 31 หรือค่า EA (external access enable) จึงถูกต่อลงกราวด์ไว้เพื่อกำหนดให้ไม่ใช้งานหน่วยความจำโปรแกรมภายในที่มีอยู่แล้ว และเมื่อไมโครคอนโทรลเลอร์ต้องการติดต่อกับหน่วยความจำโปรแกรมภายนอกมันจะส่งสัญญาณลอจิก Low ที่ ขา29 หรือขา $\overline{\text{PSEN}}$ ออกมา

หน่วยความจำโปรแกรมไม่จำเป็นเสมอไปว่าต้องเป็นรอมหรืออีพรอม เช่นเดียวกับตำแหน่งแอดเดรสที่ว่างแต่ละแอดเดรส อาจอยู่ในรูปของหน่วยความจำหรือเป็นตำแหน่งของพอร์ตอินพุตเอาต์พุตก็ได้ หน่วยความจำโปรแกรมในที่นี้ถูกแบ่งออกเป็น 2 ช่วงดังนี้ คือช่วงแอดเดรสต่ำ 00000H ถึง 04000H เป็นอีพรอมIC $_5$ และช่วงแอดเดรสจาก 04000H ถึง 08000H เป็นหน่วยความจำแรม IC $_6$ ของระบบ คำสั่งต่างๆ จะถูกป้อนให้ไปเก็บไว้และทำการประมวลผลจากที่แรมนี้

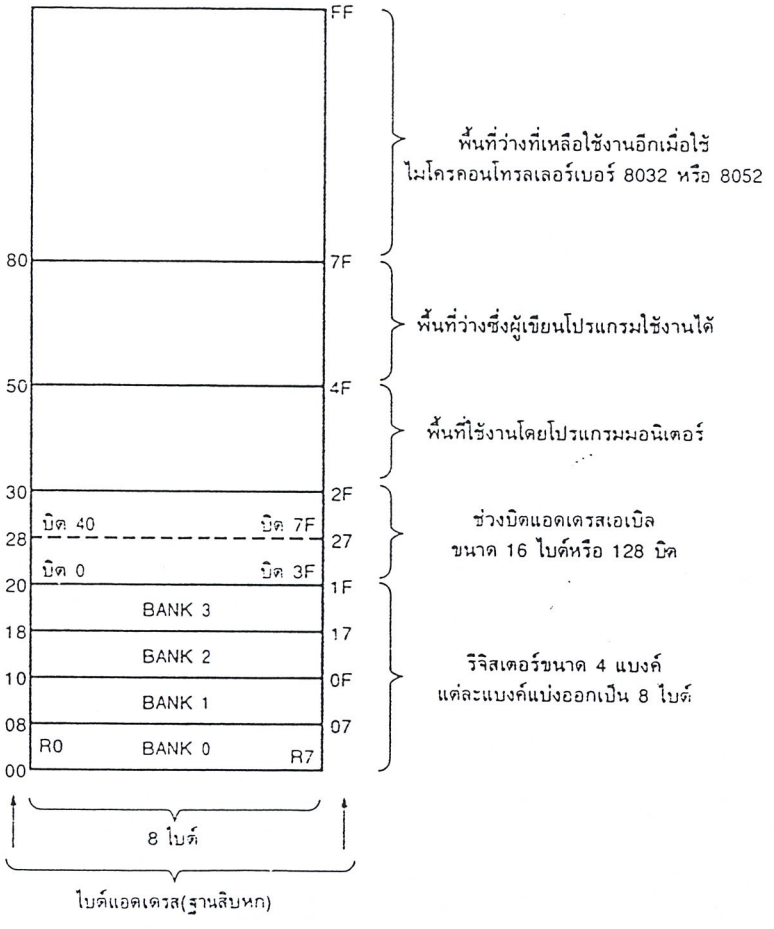
4.4.2 หน่วยความจำสำหรับเก็บข้อมูล

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 สามารถทำการอ่านและเขียนข้อมูลจากหน่วยความจำข้อมูลที่มีขนาดสูงสุดได้ 64 กิโลไบต์ หน่วยความจำในส่วนนี้ทำหน้าที่เก็บข้อมูลใช้งานจำนวนมากเป็นส่วนใหญ่ ซึ่งในบางครั้งอาจเรียกได้ว่าแรมบน MCS-51 บอร์ด หรือ IC 6 เป็นผู้ทำหน้าที่นี้ หน่วยความจำข้อมูลบนบอร์ด MCS-51 กำหนดให้มีตำแหน่งใช้งาน ตั้งแต่ 04000H ถึง 08000H ซึ่งตามที่กล่าวมาแล้วหน่วยความจำสำหรับเก็บโปรแกรมโดยใช้แรมถูกกำหนดให้เริ่มต้นที่ตำแหน่ง 04000H เป็นต้นไป นั่นคือ โปรแกรมทดลองหรือโปรแกรมที่ดาวน์โหลดจากคอมพิวเตอร์จะต้องเริ่มประมวลผลที่ตำแหน่ง 04000H ขึ้นไปเสมอ

การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ส่วนใหญ่แล้วสามารถทำงานได้โดยไม่ต้องใช้หน่วยความจำข้อมูลมากเท่าใดนัก ทำให้เมื่อต้องการใช่หน่วยความจำก็อาศัยแรมภายในที่มีอยู่แล้วมากกว่าที่จะใช้แรมที่อยู่ภายนอกขนาดของแรมภายในมีขนาด 128 ไบต์ สำหรับไมโครคอนเทรลเลอร์เบอร์ 8031 และ 8051 และมีขนาด 256 ไบต์สำหรับเบอร์ 8032 และ 8052 แต่ในที่นี้จะใช้แรมภายในนี้สูงสุดไม่เกิน 128 ไบต์ ดังนั้นจึงไม่มีปัญหาไม่ว่าผู้อ่านจะใช้ไมโครคอนโทรลเลอร์เบอร์ใดมาทำการศึกษา

ในส่วนของแรมภายในประกอบด้วยรีจิสเตอร์ของไมโครคอนโทรลเลอร์ หน่วยความจำสแต็คสำหรับใช้งานและจัดการระบบภายในซับรูทีนก็อยู่ในส่วนของแรมภายในด้วย ดังนั้นขนาดของหน่วยความจำภายในที่ผู้เขียนโปรแกรมใช้งานได้จริงจึงน้อยกว่า 128 ไบต์ ในช่วงแอดเดรสระหว่าง 20H ถึง 22FH เรียกว่าบิตอแดเดรสเอเบิล(bit addressable range) ในส่วนนี้ใช้งานในการจัดแจงหรือโยกย้ายถ่ายเทบิตข้อมูลของคำสั่งไปยังแอดเดรส, เปลี่ยนหรือเรียกใช้บิตใดบิตหนึ่ง ส่วนประกอบสุดท้ายที่ใช้แรม

ภายในก็คือ โปรแกรมมอนิเตอร์ซึ่งบรรจุอยู่ในอีพรอม ซึ่งการประมวลผลในส่วนนี้ต้องใช้งานแรมภายในบางส่วนด้วยเช่นกัน รูปที่ 4.5 แสดงการแบ่งช่วงแอดเดรสของแรมภายใน



รูปที่ 4.5 แผนที่แสดงการจัดแบ่งช่วงแอดเดรสใช้งานของแรมภายใน ซึ่งมีขนาด 128 ไบต์ หรือ 256 ไบต์ ขึ้นอยู่กับเบอร์ของไมโครคอนโทรลเลอร์

บทที่ 5
การสื่อสารผ่านพอร์ตอนุกรม กับ ไมโครคอนโทรลเลอร์ MCS-51

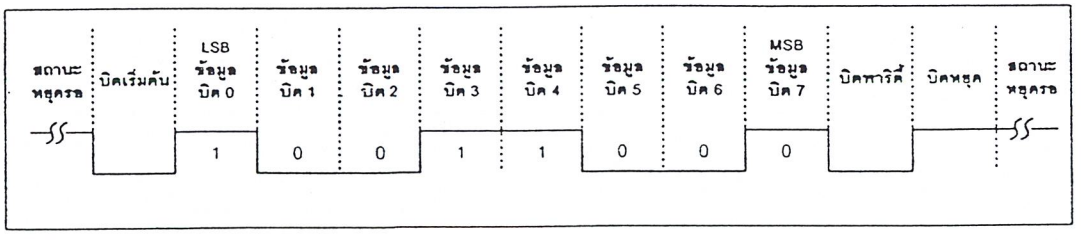
ไมโครคอนโทรลเลอร์ตระกูล MXS-51 มีวงจรสื่อสารอนุกรมแบบฟูลดูพเล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟลูดูเพล็กซ์ หมายถึง วงจรสื่อสารที่สามารถทำการรับและส่งข้อมูลในลักษณะ 2 ทิศทางได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขาP3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P 3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบเฟรชเป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS- 232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐาน RS-422 หรือ RS-485 ได้แล้ว โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

5.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารแบบอะซิงโครนัส คือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอดหรือ บอดเรต(baud rate) มีหน่วยเป็นบิตต่อวินาที(bit per second: bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น(start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมมีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity bit)มีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด(stop bit)มีขนาด 1 บิท



รูปที่ 5.1 รูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส

รูปที่ 5.1 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูลDATA จะมีสถานะลอจิก"1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stag) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้

ขาDATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1บิต เรียกบิตนี้ว่า บิตเริ่มต้น (start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิตLSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิต จากนั้นตามด้วยบิตพาริตี้ (parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งออกไปคือ บิตปิดท้ายหรือบิตหยุด (stop bit) โดยจะเป็นการทำให้ขาDATA มีสถานะลอจิก"1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต. 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่า ได้แก่ 110. 150. 300. 600. 1,200. 2,400. 4,800, 9,600 และ 19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่าข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1บิตและบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1ไบต์จะมีความยาวเท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9.600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่(odd) แบบคู่(even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมบิตพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ99H หรือ 10011001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัว ซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของบิตพาริตี้จะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asymchronous Receiver Transmitter: เป็นอุปกรณ์ที่ใช้ในการรับส่งข้อมูลอนุกรม) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก 1 ทั้งหมดรวมทั้งบิตจะทำการตรวจสอบค่าพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ ทำการส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

5.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน MCS-51

ในการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์MCS-51 มีรีจิสเตอร์ที่ต้องเกี่ยวข้องอยู่ 2 ตัว ดังมีรายละเอียดต่อไปนี้

5.2.1 ริจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือSBUF(Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของริจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต มีการแบ่งเป็น 2 ส่วนคือ ริจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล(transmit buffer register)และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากริจิสเตอร์SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือP3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลช

5.2.2 รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial port Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์SFRสามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0- SM1 (Serial port mode bit0-1): ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์MCS-51 ดังมีรายละเอียดต่อไปนี้

SM2 : ใช้ในการอีนาเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ (multiprocessor) ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทลเลอร์ MCS-51 ในโหมด2และ3 ถ้าบิตนี้เป็น"1" บิต RIจะไม่แอกตีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น "0" (ข้อมูลบิตที่9 เก็บไว้ที่บิต RB8) ในการทำงานโหมด 1 ถ้าบิตนี้เซต บิต RI จะไม่แอกตีฟถ้ายังไม่ได้รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN (Enable serial recption) : ใช้ในการอีนาเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้มีการรับข้อมูลต้องเซตบิตนี้ให้เป็น"1"

TB8: ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RB 8: ใช้สำหรับรับข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แต่ถ้าพอร์ตอนุกรมทำงานอยู่ในโหมด 1และบิต SM2 เป็น"0" ข้อมูลที่บิต RS 8: คือข้อมูลของบิตหยุด(Stop bit) สำหรับในการทำงานโหมด 0 บิตนี้จะไม่ใช้งานบิต RB 8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์เท่านั้น

TI(transmit Interrupt flage): ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์แวร์ เมื่อทำการส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานของโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟแวร์เท่านั้น

RI(Receive Interrupt flage) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์แวร์ เมื่อทำการรับข้อมูลบิตที่ 8 เรียบร้อย แล้วในการทำงานโหมด0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีสามารถรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิตSM2 มีการเซต บิตนี้จะสามารถเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟแวร์เท่านั้น

5.3 โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชสามารถเลือกโหมดการทำงานได้ถึง 4 โหมดคือ

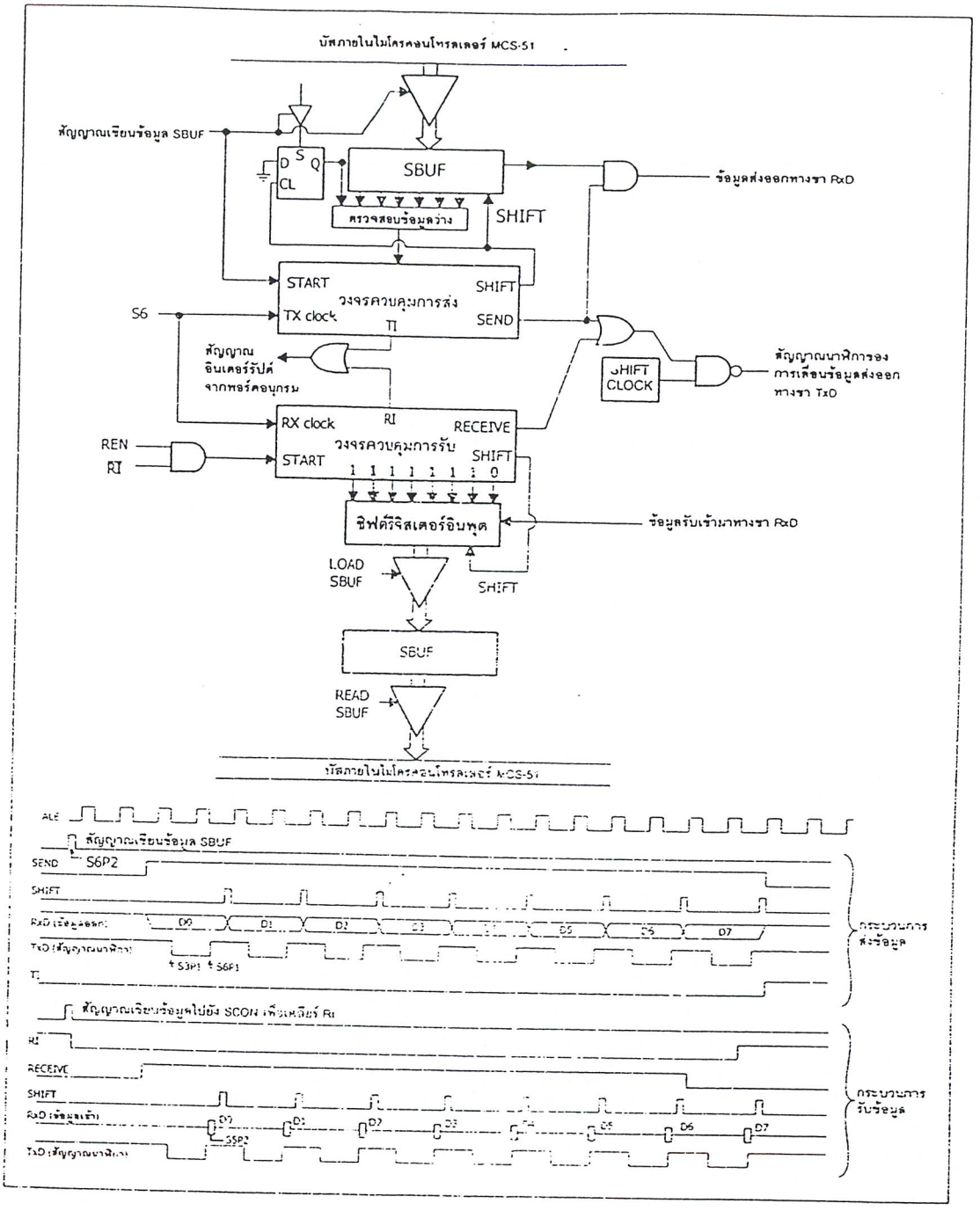
1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์
2. โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้
3. โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่
4. โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดการทำงานของวงจรพอร์ตในไมโครคอนโทรลเลอณ์ MCS-51 กระทำได้โดยการกำหนดข้อมูลให้แก่บิต SM0 และ SM1 ในรีจิสเตอร์ SCON

5.3.1 การทำงานในโหมด 0 ของวงจรพอร์ตอนุกรม

มีไดอะแกรมการทำงานและไดอะแกรมเวลาแสดงในรูปที่ 5.2 ข้อมูลอนุกรมเวลาจะผ่านเข้าและออกทางขา RxD ส่วนขา TxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล (shift clock)ในโหมดนี้มีจำนวนข้อมูล 8 บิต โดยทำการรับและส่งข้อมูลในบิต LSB ก่อน อัตราในการรับส่งข้อมูลหรืออัตราบอดถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่สัญญาณนาฬิกา

เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์SBUF สัญญาณเขียนข้อมูล SUBF แอกตีฟเป็น"1" ที่สเตต 6 เฟส 2 (S6P2) ของแมชีน ไซเกิล ส่งมายังวงจรควบคุมการส่ง(TX control) ทำให้วงจรควบคุมการส่งเริ่มต้นการส่งข้อมูลสัญญาณ SEND จะแอกตีฟเป็น "1" ตลอดกระบวนการส่งข้อมูล



รูปที่ 5.2 ไดอะแกรมการทำงานในโหมด 0 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

ข้อมูลจากรีจิสเตอร์ SUBF จะถูกเลื่อนออกจากขาที่P3.0 หรือขา RxD ครั้งละบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทางขา P3.1 หรือ TxD โดยสัญญาณนาฬิกาของการเลื่อนข้อมูลจะมีขอบขาลงที่สัญญาณที่สเตต 3 เฟส และมีขอบขาขึ้นของสัญญาณที่สเตต 6 เฟส 1 ของแต่ละแมชชีนไซเกิลในกระบวนการส่งข้อมูล จนกระทั่งเมื่อส่งข้อมูลครบ 8 บิต แล้ว แล้วบิต TIในรีจิสเตอร์SCON จะเกิดการเซตเป็นการแจ้งให้ทราบว่าส่งข้อมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับ

การอีนาเบิลไว้ ก็จะเกิดการอินเตอณ์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ SEND จะกลายเป็น"0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

ในกระบวนการรับข้อมูล เริ่มต้นด้วยการเซต RENให้เป็น "1" และเคลียร์บิต RI ในรีจิสเตอร์ SCONก่อน ที่สเตต 6 เฟส 2 ของแมชชีนไซเกิลถัดไป วงจรควบคุมการรับ(RX control) จะทำการเขียนข้อมูล 11111110 ไปยังชิฟต์รีจิสเตอร์ สำหรับรับข้อมูลและทำการแอกตีฟสัญญาณ RECEIVEให้เป็น "1" ในสัญญาณนาฬิการลูกถัดไป

เมื่อสัญญาณRECEIVE แอกตีฟ ก็จะเกิดการส่งสัญญาณนาฬิกาของการเลื่อนข้อมูล(Shift clock) ขึ้นผ่านทางขาP3.1 หรือTxD เพื่อทำการกำหนดจังหวะการรับข้อมูลครั้งละบิต โดยสัญญาณนาฬิกาจะเกิดขึ้นในช่วงสเตต 3 เฟส 1 ถึง สเตต6 เฟส 1 ของแต่ละแมชีนไซเกิล การรับข้อมูลเข้ามาทางขา P3.0 หรือ RxD จะเกิดขึ้นที่สเตต 5 เฟส 2 ในแมชีนไซเกิลเดียวกับสัญญาณนาฬิกาของการเลื่อนข้อมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต บิต RI จะได้รับการเซตเพื่อแจ้งการเสร็จสิ้นกระบวน การรับข้อมูล หากการอินเอตร์รัปต์จากพอร์ตอนุกรมได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นการรับข้อมูล สัญญาณ RECEIVEจะกลายเป็น "0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

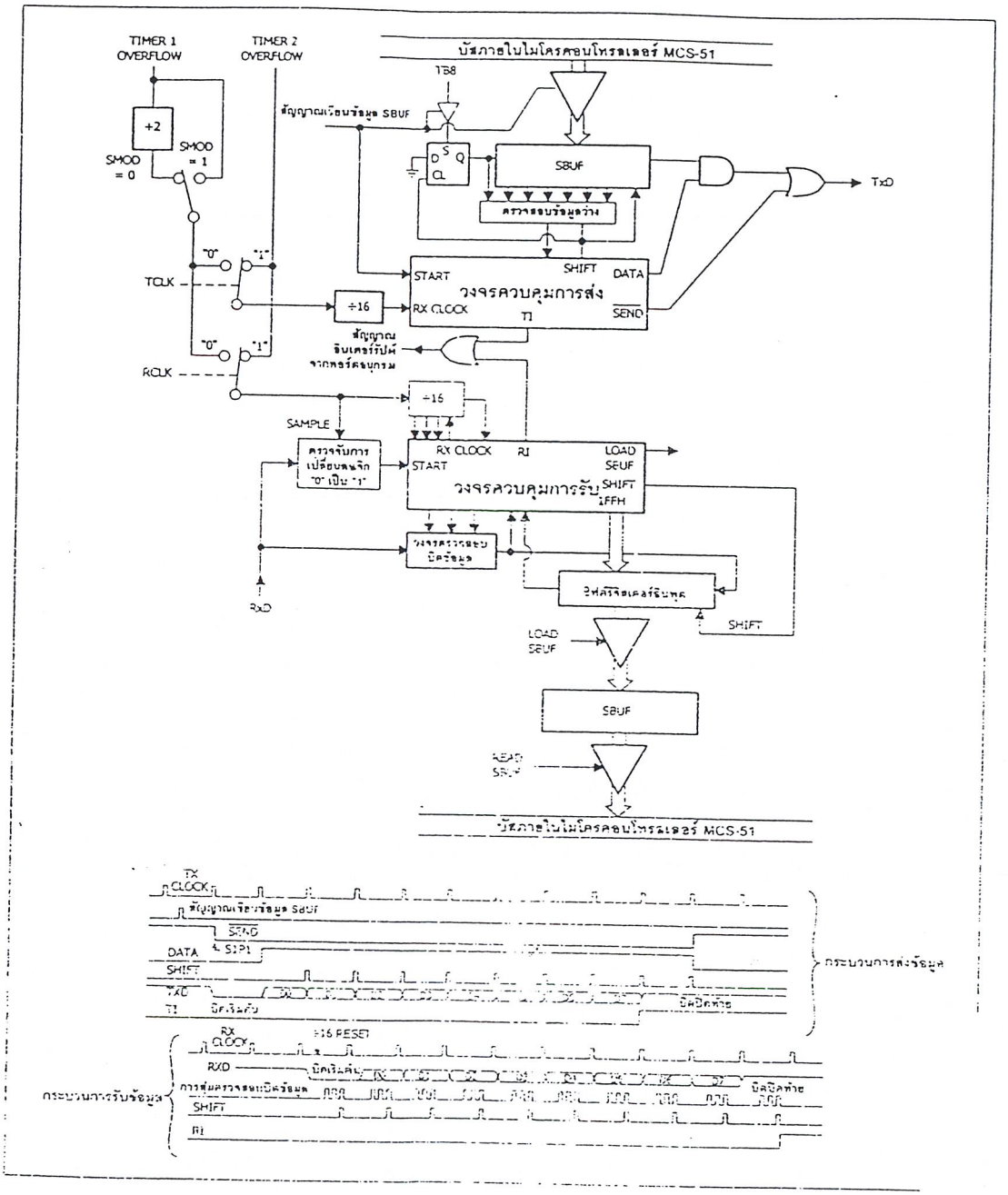
การทำงานในโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 จะใช้ประโยชน์ในการเชื่อมต่อกับไอซีริจิสเตอร์ภายนอกเพื่อทำการขยายจำนวนพอร์ตอินพุตหรือเอาต์พุต แต่ไม่เป็นที่นิยมใช้งานมากนัก เนื่องจากไมโครคอนโทรลเลอร์ MCS-51 เองมีพอร์ตอยู่ค่อนข้างมาก และสามารถติดต่อกับพอร์ตเหล่านั้นได้ง่ายและเร็วกว่ามาก

5.3.2การทำงานในโหมด 1 ของวงจรพอร์ตอนุกรม

มีไดอะแกรมการทำงานและไดอะแกรมเวลาแสดงในรูปที่ 5.3 ในโหมดนี้จะใช้ในการรับส่งข้อมูลรวม 10 บิตโดยส่งข้อมูลออกทางขา P 3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10บิตประกอบด้วย บิตเริ่มต้น (มีค่าเป็น"0") 1 บิต บิตข้อมูล 8 บิต โดยรับหรือส่งข้อมูลในบิตLSB ก่อน และบิตหยุดหรือปิดท้าย(มีค่าเป็น "1") ในการรับข้อมูล บิตหยุดจะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON อัตราบอดในโหมดนี้ได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 ใน AT89C51 ส่วนในไมโครคอนโทรลเลอร์เบอร์ AT89C52และในอนุกรม AT89Sxx สามารถเลือกใช้อัตราการเกิดโอเวอร์โฟลวของไทเมอร์1 หรือ ไทเมอร์2 ในการกำหนดเป็นอัตราบอดได้

กระบวนการส่งข้อมูลเริ่มต้นด้วยการแอกตีฟสัญญาณเขียนข้อมูลมายังรีจิสเตอร์ SBUF ส่งมายังวงจรควบคุมการส่ง(TX control) จากนั้นวงจรควบคุมการส่งจะทำการแอกตีฟสัญญาณSEND ที่สเตต 1 เฟส 1 ของแมชชีนไซเกิลโดยสัญญาณ SEND จะเป็น "0" ตลอดกระบวนการส่งข้อมูลเมื่อสัญญาณ SEND แอกตีฟ จะทำการส่งบิตเริ่มต้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 แมชชีนไซเกิล จากนั้นตามด้วยการส่งบิตข้อมูล 8 บิตเรียงลำดับจากบิต LSB โดยข้อมูลที่ทำการส่งนี้จะถูกเรียกออกมาจากรีจิสเตอร์บัฟเฟอร์สำหรับการส่งข้อมูล ในทุกๆ บิตข้อมูลที่ทำการส่งออกไปจะเกิดสัญญาณ

พัลส์ SHIFT ขึ้นเพื่อให้เรียกข้อมูลในแต่ละบิตจากรีจิสเตอร์ บัฟเฟอร์ การกำหนดจังหวะการส่งข้อมูลจะใช้สัญญาณนาฬิกาการส่ง (TX clock) เป็นตัวกำหนด โดยสัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณTCLK จากไทเมอร์ 1 ด้วย 16 หลังจากการส่งบิตข้อมูลก็จะทำการส่งบิตหยุดหรือบิตปิดท้ายขนาด 1 บิต ดังนั้นการส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการส่งข้อมูลได้รับการอินเทอร์รัปต์ไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ



รูปที่ 5.3 ไดอะแกรมการทำงานในโหมด 1 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

หลังจากที่ทำการบริการอินเตอร์รัปต์หรือส่งข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต TIก่อน เป็นอันดับแรกเพื่อให้สามารถเริ่มต้นกระบวนการรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ด้านการรับข้อมูล จะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก"1" เป็น"0" ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของอัตราบอด เมื่อตรวจจับพบ ไทเมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดอัตราบอดรีเซต และทำการเขียนข้อมูลIFFH ไปยังชิฟต์รีจิสเตอร์อินพุต

ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามาเป็น"0" หรือ"1" จะใช้ผลการสุ่มค่อนข้างมาก โดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 สเตต การสุ่มข้อมูลจะทำการสุ่มสเตตที่ 7. 8 และ 9 หาก 2 ใน 3 ของการสุ่มพบว่าข้อมูลเป็นลอจิกใด จะตีความข้อมูลในบิตนั้นเป็นตามเสียงข้างมาก ยกตัวอย่างสุ่มพบลอจิก "1" 2 ใน 3 ครั้ง จะตีความว่าบิตของข้อมูลที่ได้รับนั้นเป็น "1"

ลำดับของการรับข้อมูลมีลักษณะเดียวกับการส่งข้อมูลคือ เริ่มด้วยบิตเริ่มต้นก่อน ตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุกๆ การราบข้อมูลได้ 1 บิต นะมีพับลส์ SHIFT เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู่รีจิสเตอร์บัฟเฟอร์การรับข้อทมูล การกำหนดจังหวะการรับข้อมูลใช้สัญญาณนาฬิการการรับข้อมูล (RX clock) หลังจากสัญญาณนาฬิกาลูกสุดท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์SBUF และบิต RB 8ในรีจิสเตอร์ SCON โดยข้อมูลในบิต RB 8 ก็คือข้อมูลของบิตหยุดนั่นเอง พร้อมกันนั้นยังทำการเซตบิตRI ในรีจิสเตอร์ SCONด้วย หากการอินเตอร์รัปต์จากพอร์ตอนุกรมาได้รับการ อีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ

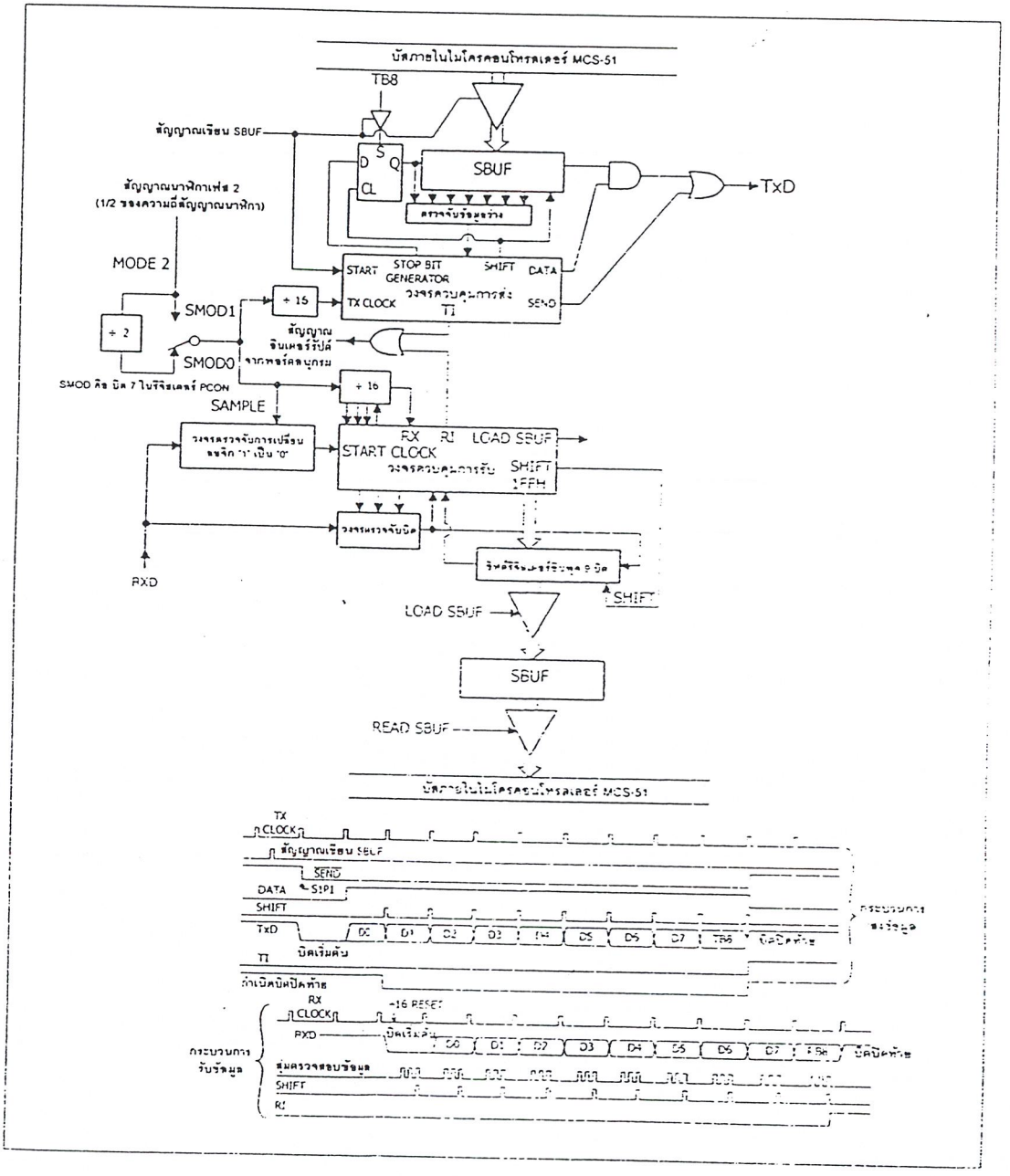
หลังจากที่ทำการบริการอินเตอร์รัปต์หรือรับข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RIก่อน เป็นอันดับแรก เพื่อให้สามารถเริ่มต้นกระบวนการรับส่งข้อมูลทางพอร์ตอนุกรมาดำเนินต่อไปได้

การทำงานในโหมดนี้ได้รับความนิยมสูงสุด เนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

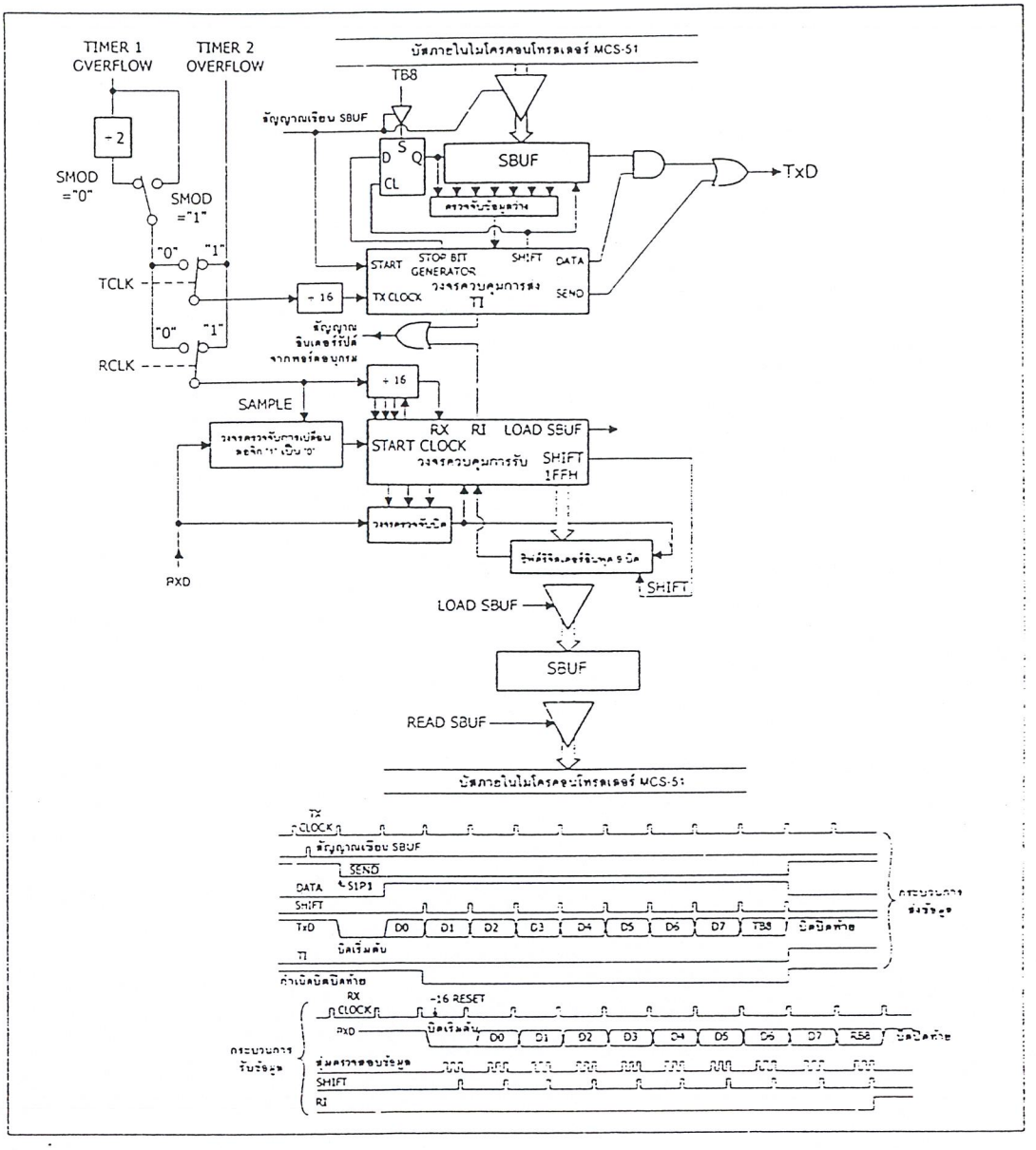
5.3.3การทำงานในโหมด 2 และ 3 ของวงจรพอร์ตอนุกรม

ในทั้งสองโหมดนี้จะใช้รูปแบบการข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้น มีค่าเป็น"0" จำนวน 1 บิต .บิตข้อมูล 8 บิต โดยทำการรับและส่งบิต LSB ก่อน .บิตข้อมูลบิตที่ 9 และบิตปิดท้ายมีค่าเป็น"1" จำนวน 1บิต ในการส่งข้อมูล ข้อมูลบิตที่ 9 จะได้รับการเก็บไว้ในบิต TB 8 ในรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะนำไปเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับอัตราบอดในโหมด 2 จะคงที่โดยเลือกได้ 2 ค่าคือ 1/32 หรือ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1

ในรูปที่ 5.4 และ 5.5 เป็นไดอะแกรมการทำงานและไดอะแกรมการทำงานในโหมด 2 และ 3 ของพอร์ตอนุกรม การทำงานโดยรวมจะคล้ายกับการทำงานในโหมด 1 ส่วนที่แตกต่างกันคือจำนวนบิตของข้อมูลที่ในโหมด 2 และ 3 จะมีเพิ่มมาอีก 1 บิต โดยส่วนใหญ่จะใช้เป็นบิตตรวจสอบพาริตี้



รูปที่ 5.4 ไดอะแกรมการทำงานในโหมด 2 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51



รูปที่ 5.5 ไดอะแกรมการทำงานในโหมด 3 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

5.4 อัตราการบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์MCS-51

5.4.1 โหมด 0

อัตราบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

อัตราบอดในโหมด0 = ความถี่ของสัญญาณนาฬิกา / 12หน่วยเป็นบิตต่อวินาที

5.4.2 โหมด 1 และ 3

เนื่องจากทั้งสองโหมดนี้สามารถเลือกแหล่งกำเนิดอัตราบอดได้ 2 แหล่ง คือ จากอัตราโอเวอร์โฟลวของไทเมอร์ 1และ 2 สำหรับอัตราบอดเมื่อใช้การโอเวอร์โฟลวของไทเมอร์ 1 จะต้องใช้ค่าของบิต SMOD ในรีจิสเตอร์ PCON (จะกล่าวถึงรายละเอียดของรีจิสเตอร์ตัวนี้ในบทที่ว่าด้วยการทำงานในโหมดประหยัดพลังงาน)มาพิจารณาประกอบด้วย โดยสามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = $(2^{\text{ค่าในรีจิสเตอร์SMOD}}/32)$X อัตราโอเวอร์โฟลวของไทเมอร์ 1

ตารางที่ 5.1 การเลือกอัตราบอดของวงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

อัตราบอด (บิตต่อวินาที : bps)	ความถี่สัญญาณนาฬิกา	SMOD	ไทเมอร์ 1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0 : สูงสุด 1 MHz	12 MHz	X	X	X	X
โหมด 2 : สูงสุด 375K	12 MHz	1	X	X	X
โหมด 1,3 : 62.5K	12 MHz	1	0	2	FFH
19.2K (19,200)	11.0592 MHz	1	0	2	FDH
9.6K (9,600)	11.0592 MHz	0	0	2	FDH
4.8K (4,800)	11.0592 MHz	0	0	2	FAH
2.4K (2,400)	11.0592 MHz	0	0	2	F4H
1.2K (1,200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

ถ้าหากในไทเมอร์1ไม่ได้อีนาเบิลการอินเตอร์รัปต์ไว้ สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = $(2^{\text{ค่าในรีจิสเตอร์SMOD}}/32)$X(ความถี่สัญญาณนาฬิกา /{12 x [256-(TH1)]}

ในตารางที่ 9.1 แสดงการกำหนดอัตราบอดโดยใช้ไทเมอร์ 1

ในกรณีที่ใช้ไทเมอร์ 2 ในการกำหนดอัตราบอด โดยกำหนดให้ไทเมอร์ 2 ทำงานในโหมดกำเนิดอัตราบอด(baud rate generator) สามารถคำนวณหาอัตราบอด ได้จาก

อัตราบอด = อัตราโอเวอร์โฟลวของไทเมอร์ 2/16 หน่วยเป็น บิตต่อวินาที

ถ้าหากกำหนดให้ไทเมอร์ 2 ทำงานในโหมดไทเมอร์หรือเคาน์เตอร์ตามปกติ สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = ความถี่ของสัญญาณนาฬิกา/ (32x (65536-(RCAP2H, RCAP2L)))

โดยที่(RCAP2H, RCAP2L) เป็นค่าของรีจิสเตอร์RCAP2H และ RCAP2L มีขนาด 16 บิตไม่คิดเครื่องหมาย

5.4.3 โหมด 2

ในโหมดนี้การกำหนดอัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD เป็น"0" อัตราบอดจะเท่ากับ 1/64 ของความถี่สัญญาณนาฬิกา ในกรณีที่ SMOD เป็น "1" อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสูตรคำนวณทางคณิตศาสตร์ได้ดังนี้

อัตราบอด =($2^{\text{ค่าของบิตSMOD}}$ /64)Xความถี่สัญญาณนาฬิกา

5.5 การกำหนดค่าของไทเมอร์เพื่อเลือกอัตราบอด

ในการใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สิ่งที่ต้องให้ความสนใจมากที่สุดประการหนึ่งคืออัตราการถ่ายทอดหรืออัตราบอด ซึ่งการกำหนดอัตราบอดนั้นจะขึ้นอยู่กับค่าความถี่ของสัญญาณนาฬิกาเป็นหลัก สำหรับโหมดการทำงานของพอร์ตอนุกรมที่สามารถเลือกอัตราบอดได้อย่างอิสระคือในโหมด 1 และ 3 โดยกำหนดได้จากอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 ถ้าหากไทเมอร์ 1 มีการเกิดโอเวอร์โฟลวในอัตราที่สูงมาก สามารถถ่ายทอดข้อมูลได้อย่างรวดเร็ว ดังนั้นการกำหนดค่าและโหมดการทำงานของไทเมอร์ 1 จึ่งเป็นสิ่งที่มีผลต่อการเปลี่ยนแปลงของอัตราบอดด้วย

ในการใช้ไทเมอร์ 1 เพื่อกำหนดอัตราบอดในโหมด 1และ 3 ของพอร์ตอนุกรมจะต้องกำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2หรือโหมด8บิตแบบตั้งค่าการนับอัตโนมัติ และการกำหนดค่ารีโหลดให้แก่รีจิสเตอร์TH 1จึงเป็นตัวแปรหลักที่ใช้ในการกำหนดอัตราบอดให้แก่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์MCS-51

เริ่มต้นด้วยการเคลียร์บิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON ให้เป็น "0" ค่าของการรีโหลดให้แก่ TH1สามารถคำนวณได้จาก

TH1 = 256 – ((ค่าความถี่ของคริสตอล /384)/ อัตราบอด)

แต่ถ้าบิตSMOD เกิดการรีเซต จะเป็นการเกิดอีนาเบิลทวีคูณของอัตราบอด ดังนั้นการกำหนดค่าให้แก่ TH1 จึงต้องคำนวณจาก

TH1 =256 – ((ค่าความถี่ของคริสตอล /192)/ อัตราบอด)

ยกตัวอย่างถ้าหากในไมโครคอนโทรลเลอร์ AT89C51 ใช้คริสตอล 11.0592MHz ต้องการกำหนดอัตราบอดของพอร์ตอนุกรมของไมโครคอนโทรเลอร์ไว้ที่ 19,200 บิตต่อวินาที ในกรณีที่ไม่อีนาเบิลการทวีคูณของอัตราบอดค่ารีโหลดของไมโครคอนโทรเลอร์จะเท่ากับ

TH1 =256 – ((ค่าความถี่ของคริสตอล /384)/ อัตราบอด)

= 256 –((11059200/384)/19200)

= 256-(28800/19200)

= 256-1.5 = 254.5

เนื่องจากผลลัพธ์ที่ได้เป็นค่าที่ไม่ใช่จำนวนเต็ม ถ้าหากกำหนดค่าของ TH1 เป็น 254 เมื่อทำการแทนค่าเพื่อคำนวณหาค่าอัตราบอดจะได้อัตราบอดเท่ากับ 14,400บิตต่อวินาที และถ้าหากกำหนดค่าของ TH1 เป็น 255อัตราบอดจะมีค่าเท่ากับ28,800บิตต่อวินาที ดังนั้นจะเห็นได้ค่าของTH1 ที่ไม่เป็นจำนวนเต็มจะไม่สามารถทำให้เกิดอัตราบอดตามที่ต้องการได้

ทางแก้ไขคือ ทำให้การอีนาเบิลทวีคูณอัตราบอด โดยการเซตบิตSMOD ในรีจิสเตอร์PCON ให้เป็น "1" จากนั้นทำการแทนค่าลงในสมการหาค่า TH1 เมื่อมีการเซตบิต SMODได้ผลดังนี้

TH1 =256 – ((ค่าความถี่ของคริสตอล /192)/ อัตราบอด)

= 256 –((11059200/192)/19200)

= 256-(57600/19200)

= 256-3 = 253

นำค่าของTH1ที่ได้ทำการแทนค่าหาอัตราบอดจะได้เท่ากับ 19,200 บิตต่อวินาที

สามารถสรุปขั้นตอนในการเลือกอัตราบอดโดยการกำหนดค่าของไทเมอร์ 1 ได้ดังนี้

1. กำหนดให้พอร์ตอนุกรมของไมโครคอนโทรเลอร์ MCS-51 ทำงานในโหมด 1 หรือ 3
2. กำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2หรือโหมด 8 บิตตั้งค่าอัตโนมัติ
3. กำหนดข้อมูลให้แก่ TH1 เท่ากับ 253 เพื่อให้สามารถกำเนิดอัตราบอดได้ 19,200 บิตต่อวินาทีตามที่ต้องการ
4. ทำการเซตบิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON เพื่ออีนาเบิลการทวีคูณอัตราบอด

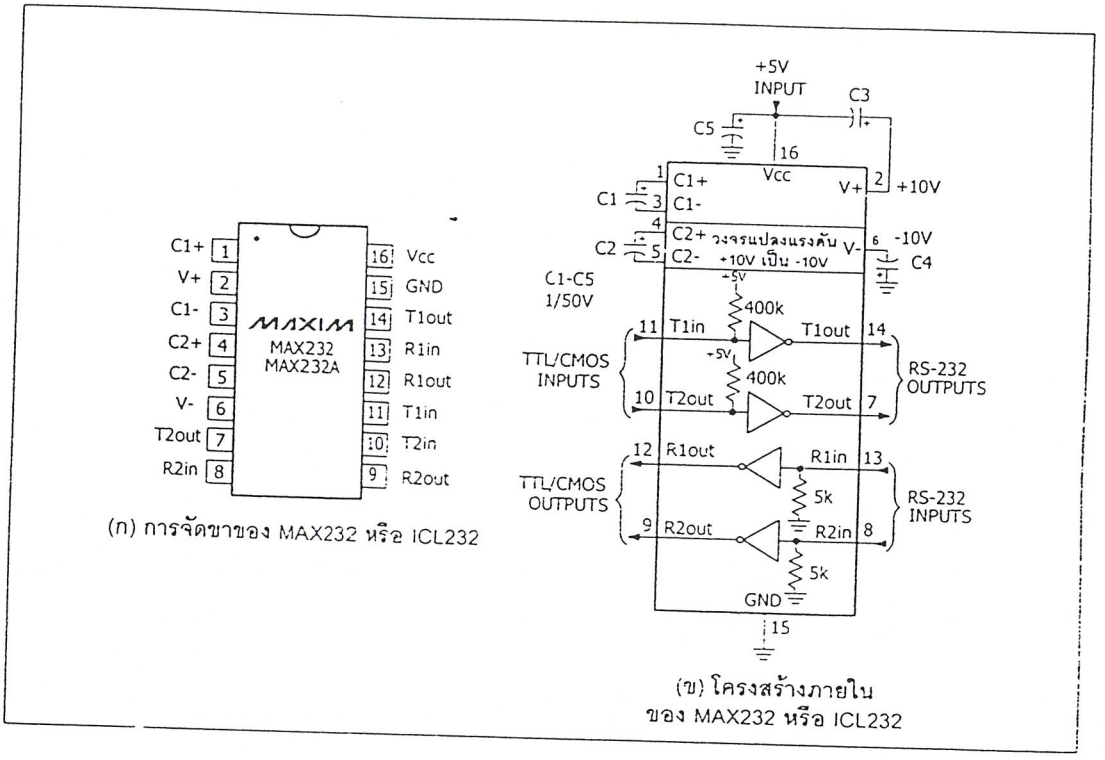
5.6 การเขียนหรือส่งข้อมูลออกจากพอร์ตอนุกรม

ข้อมูลที่ต้องการส่งออกทุกค่าต้องนำไปเก็บรีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรม ซึ่งก็คือรีจิสเตอร์ SBUF ดังตัวอย่าง

MOV SBUF, # 'A'

จากคำสั่งข้างต้นเป็นการส่งข้อมูลของตัวอักษรA ออกไปยังพอร์ตอนุกรมของไมโครคอนโทรเลอร์อย่างไรก็ตามก่อนทำการส่งข้อมูลทุกครั้ง ต้องแน่ใจว่าบิต TI เคลียร์หรือมีค่าเป็น "0" และเมื่อทำการส่งข้อมูลเรียบร้อยแล้ว ก็จะเกิดการเซตบิต TI เพื่อแจ้งให้ทราบ ดังตัวอย่างโปรแกรมต่อไปนี้

```
CLR  TI          ; เคลียร์บิตTI เพื่อเตรียมการส่งข้อมูลออก
MOV  SBUF, #'A'  ; ส่งข้อมูลของตัวอักษร A ไปยังพอร์ตอนุกรม
JNB  TI, S       ; รอการเซตของบิต TI เพื่อการแจ้งการส่งข้อมูลที่เสร็จสมบูรณ์
```

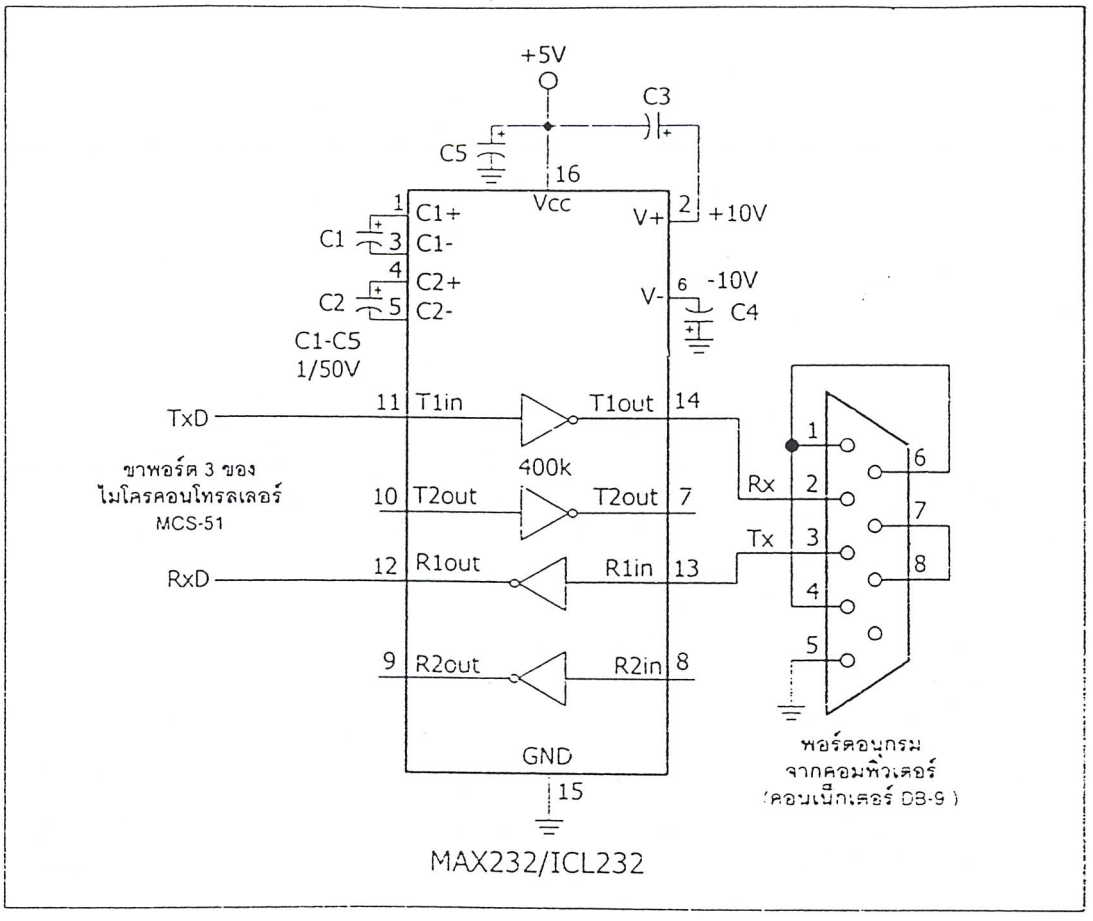


รูปที่ 5.6 รายละเอียดเบื้องต้นของไอซีแปลงสัญญาณเพื่อเชื่อมต่อพอร์ตอนุกรมของคอมพิวเตอร์

5.7 การอ่านหรือรับข้อมูลจากพอร์ตอนุกรม

การรับข้อมูลจากพอร์ตอนุกรมสามารถกระทำได้ง่ายมาก เพียงทำการตรวจสอบบิต RI เกิดการเซตขึ้นหรือไม่ ถ้าพบว่ามีการเซตเกิดขึ้นแล้ว ให้ทำการอ่านค่าจากรีจิสเตอร์ SBUF โดยต้องทำการโอนย้ายข้อมูลผ่านทางแอกคิวมูเลเตอร์หรือรีจิสเตอร์ A ดังตัวอย่าง

```
CLR RI          : เคลียร์บิตTI เพื่อเตรียมการส่งข้อมูลออก
JNB RI, S       : รอคอยการเซตของบิต RIอันเป็นการแจ้งให้ทราบว่า การรับข้อมูล
                : เสร็จสมบูรณ์และมีข้อมูลเกิดขึ้นที่รีจิสเตอร์ SBUF
MOV A, SBUF     : อ่านค่าจากรีจิสเตอร์ โดยการโอนย้ายข้อมูลผ่านทางรีจิสเตอร์ A
CLR RI          : หลังจากทำการอ่านข้อมูลเรียบร้อยแล้วต้องทำการเคลียร์บิต RI เสมอ
```



รูปที่ 5.7 วงจรเชื่อมต่อ MAX 232 หรือ ICL 232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ และ ไมโครคอนโทรลเลอร์ MCS-51

5.8 การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมพื้นฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ $\pm$ 3 ถึง $\pm$12 V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

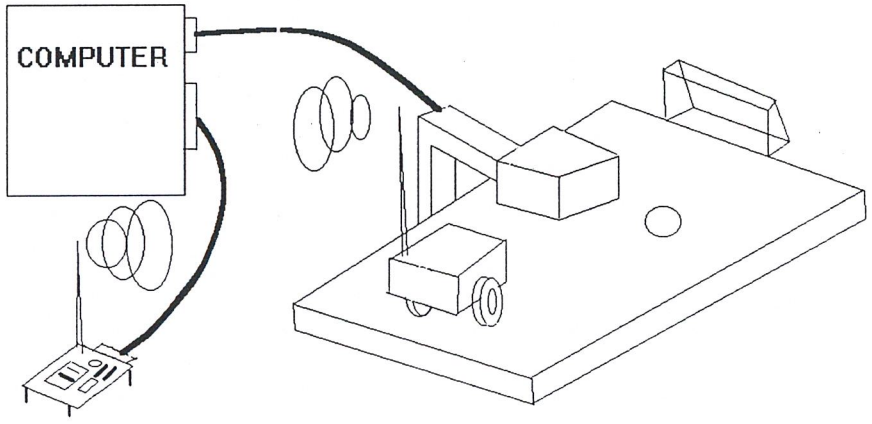
ไอซีที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้ต้องทำการแปลงข้อมูลส่งไปไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีอยู่ด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จาก MAXIM หรือ ICL232 จาก HARRIS เป็นต้น ในรูปที่ 5.6 แสดงการจัดขาของไอซี

ICL232 ซึ่งใช้ในการแปลงสัญญาณ RS-232 ส่วนวงจรของการต่อกับไมโครคอนโทรลเลอร์ MCS-51 แสดงในรูปที่ 5.7

บทที่ 6
หลักการทำงานของหุ่นยนต์เตะฟุตบอล

6.1 โครงสร้างการทำงานของหุ่นยนต์เล่นบอล

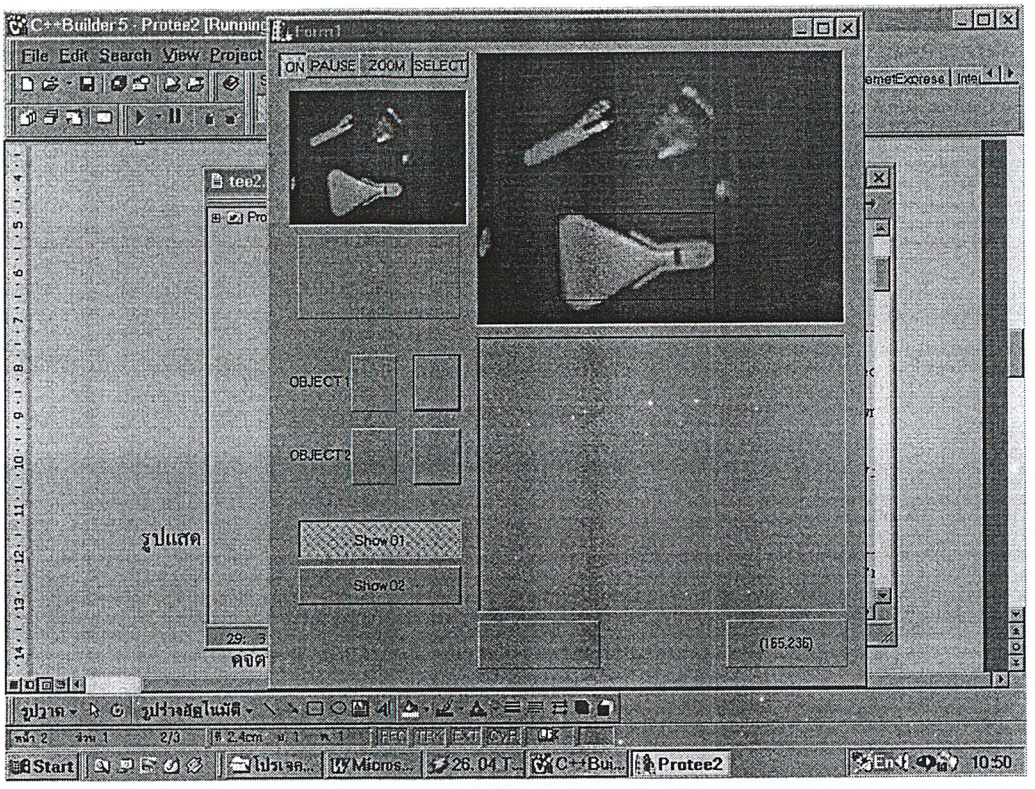
หลักการทำงานของหุ่นยนต์เล่นบอลนี้ คือ ในการทำงานของหุ่นยนต์จะมีกล่องติดอยู่เหนือสนาม จะส่งข้อมูลภาพมายัง computer เพื่อทำการประมวลผล และสั่งการไปยังหุ่นยนต์ทาง port อนุกรม ผ่านอุปกรณ์รับซึ่งไร้สาย ใช้หุ่นยนต์เคลื่อนที่ไปหาลูกบอลและส่งลูกแบลเข้าเขตประตูที่กำหนดไว้



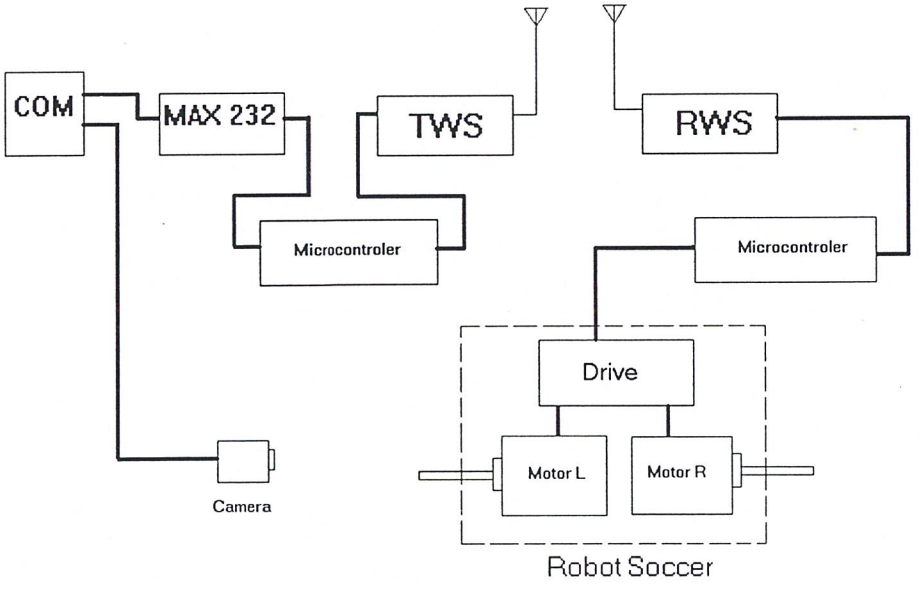
รูปที่ 6.1 แสดงรูปร่างของโครงงาน

โดยหลักการทำงานในขั้นแรกนั้นได้เริ่มจากการสั่งงานหุ่นยนต์ผ่านทาง port ให้เคลื่อนที่ไปยังตำแหน่งที่ต้องการ โดยการสั่งการจากผู้บังคับ (Manual) โดยใช้ Keyboard ควบคุม และส่วนของการหาตำแหน่งนั้นทำได้โดยการทดลองแยกวัตถุที่ถ่ายได้จากกล้องด้วยสี ทำการตีกรอบวัตถุที่แยกสีนั้นออกมา ซึ่งจะใช้เป็นหลักการเบื้องต้นง่ายๆในการนำข้อมูลภาพไปประมวลผลหาตำแหน่งหุ่นยนต์ ลูกบอล เพื่อนำไปใช้ควบคุมหุ่นยนต์ให้ทำงานอัตโนมัติต่อไป

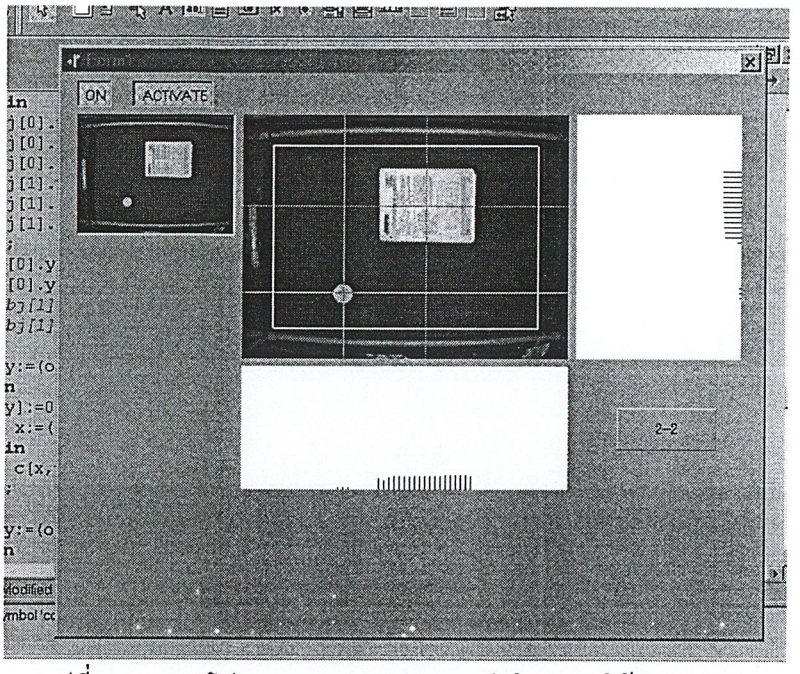
หลักจากการใช้หลักการทำงานแบบเบื้องต้น และการหาตำแหน่งอย่างง่ายแล้วได้ทำการพัฒนาการตีกรอบเพื่อหาตำแหน่งวัตถุมาเป็นการวัดโดยการวัดจากการหาค่าความสว่างเพื่อบอกตำแหน่งของวัตถุ โดยการหาตำแหน่งสามารถทำได้อย่างแม่นยำยิ่งขึ้น และมีการสั่งการหุ่นย์ให้สามารถยิงลูกบอลได้อย่างอัตโนมัติเต็มรูปแบบอีกด้วย



รูปที่ 6.2 แสดงโปรแกรมตีกรอบวัตถุด้วยสี



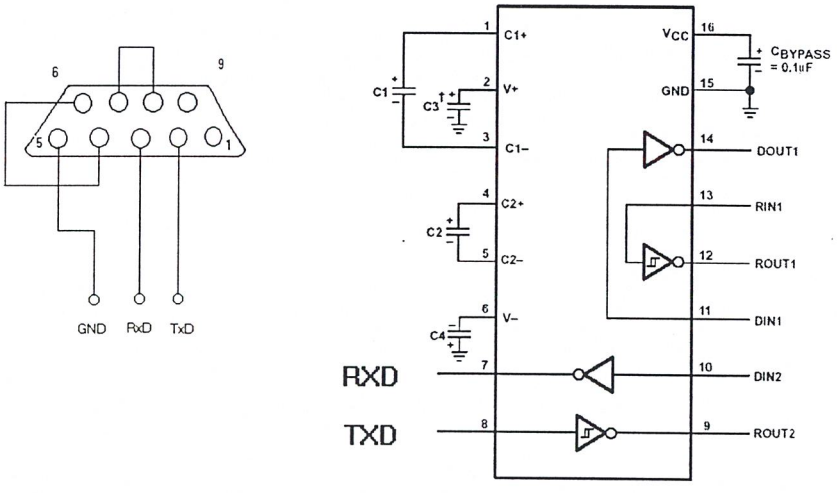
รูปที่ 6.3 แสดงการจับข้อมูลและการสั่งควบคุมการทำงานของ Computer



รูปที่ 6.4 แสดงโปรแกรมการบอกตำแหน่งโดยการใช้ค่าความสว่าง

6.2 การทำงานของหุ่นยนต์

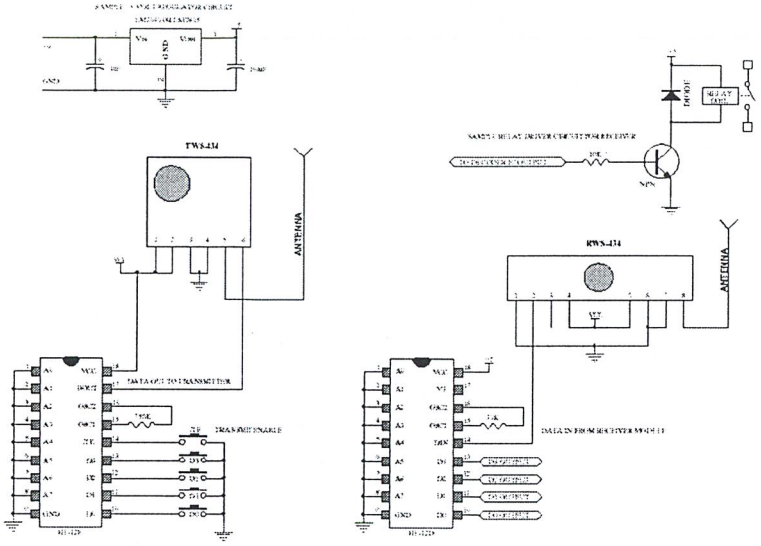
1. กล้องทำหน้าที่ส่งข้อมูลภาพแล้วสั่งมายัง Computer
2. Max 2 3 2 ทำหน้าที่แปลงสัญญาที่ออกมาจาก Computer ซึ่งเป็นสัญญาณ ดิจิตอล ระดับ 0 – 12 V เป็นสัญญาณ Digital 0.5V.



รูปที่ 6.5 การส่งข้อมูลต่างๆทาง Port อนุกรม ผ่านอุปกรณ์รับส่ง Max 232

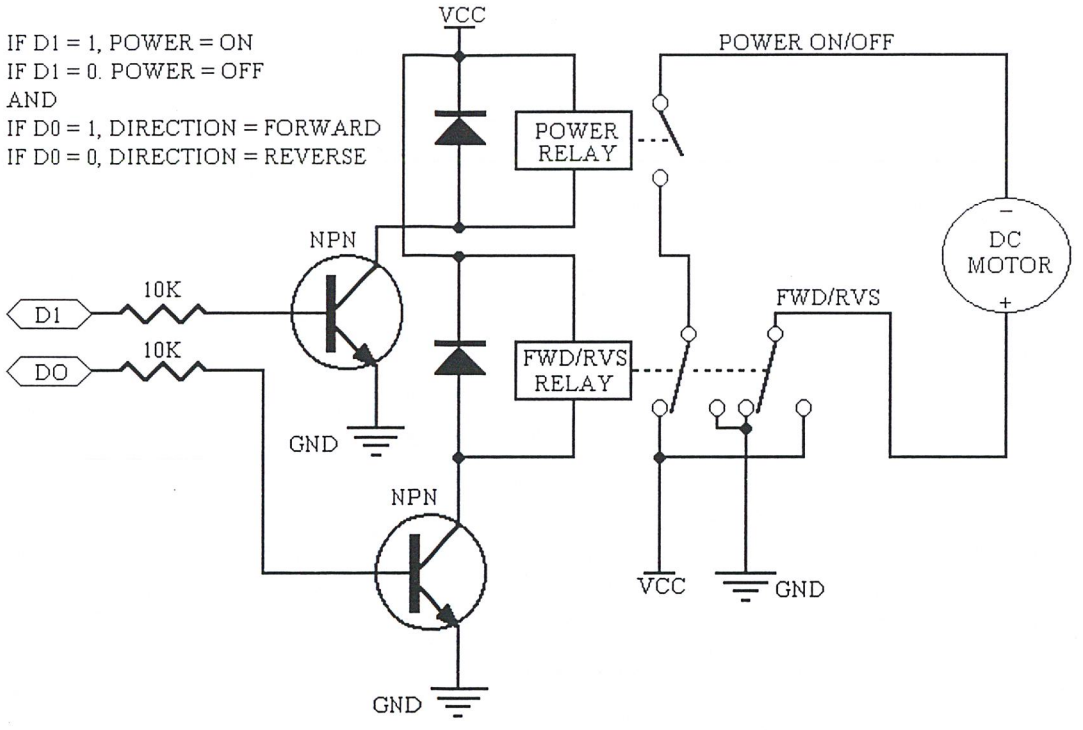
3. เครื่องส่งสัญญาณไร้สาย ประกอบด้วย 2 ส่วน คือ
   - ส่วนทำหน้าที่เข้ารหัส แปลงข้อมูล 4 บิต ที่ขนาดกันมาเป็นข้อมูลอนุกรม

- ส่วน Modulation ข้อมูลอนุกรมที่ได้กับเคลื่อนพาหะขนาด 300 MHz ได้วิธี Amplitude Modulation



รูปที่ 6.6 วงจรรับ-ส่ง

4. เครื่องรับสัญญาณ ไร้สาย ประกอบด้วย 2 ส่วน คือ
   - ส่วนทำหน้าที่ถอดรหัส แปลงข้อมูลจากที่เป็นข้อมูลอนุกรม เป็นข้อมูล 4 บิต ขนานกัน
   - ส่วน Modutation แยกข้อมูลอนุกรมออกมาจากคลื่นพาห์
5. Micracontroller ทำหน้าที่ รับข้อมูลที่เป็นอนุกรมจาก คอมพิวเตอร์ แล้วแปลงมาเป็นข้อมูล 4 บิต ขนานกัน
6. ส่วนหุ่นยนต์จะประกอบด้วย ตัว ขับเคลื่อน คือ มอเตอร์ 2 ตัว เมื่อมีคำสั่งเดินหน้า – ถอยหลัง ก็จะหมุน พร้อมๆ กัน ไปตามทิศทางที่สั่ง เมื่อมีคำสั่งเลี้ยวซ้ายหรือถอยหลังก็จะหยุดมอเตอร์ด้านที่ต้องการเลี้ยว และเดินมอเตอร์อีกตัว ในการที่จะสั่งให้มอเตอร์เดินหน้า ถอยหลังได้ ก็จะต้องมีวงจรขับ ดังรูป คือ

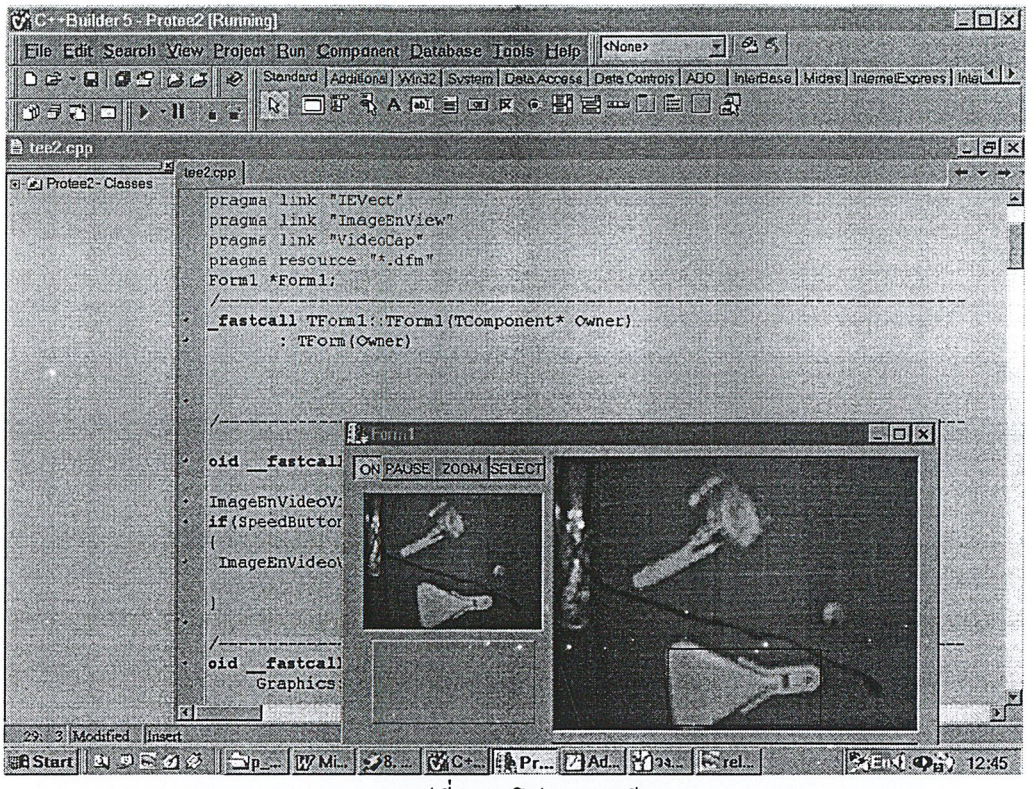


รูปที่6.7 วงจรขับมอเตอร์ เดินหน้า – ถอยหลัง

8.3 ส่วนของโปรแกรม

8.3.1 โปรแกรมตีกรอบของวัตถุ โดยการแยกสี

ใช้หลักการจำสีของ พิกเซล เป็นค่า R(red) G(green) B(blue) โดยจะต้องทำการเลือกพื้นที่สีที่ต้องการจะแยกก่อน ทำการบันทึกค่า RGB ของแต่ล่ะ พิกเซลไว้ แล้วนำมาหาค่าเฉลี่ย เก็บเป็นค่ามาตรฐานไว้ เมื่อเราสั่งให้โปรแกรมทำงาน โปรแกรมจะทำการสแกนหาค่าจุดที่มีค่าสีใกล้เคียง กับที่บันทึกไว้ แล้วโปรแกรมจะตีกรอบ บริเวณที่มีค่าจุดสีใกล้เคียงกับค่ามาตรฐานที่เก็บไว้นั้น โดยโปรแกรมนี้ต้องทำการเก็บค่ามาตรฐานใหม่ทุกครั้งที่เรียกใช้โปรแกรมเพื่อป้องกันความผิดพลาดจากการปรับสภาพแสงไม่เท่ากัน



รูปที่ 6.8 โปรแกรมตีกรอบ

8.3.2 โปรแกรมสื่อสารผ่านพอร์ตอนุกรม

การติดต่อกับพอร์ตสื่อสารอนุกรม RS-232 ก่อนเริ่มการใช้งานจำเป็นต้องกำหนดคุณสมบัติพื้นฐานบางอย่างให้กับพอร์ต

การสร้างออบเจ็กต์

1. การเปิดพอร์ตสื่อสาร มีขั้นตอนดังนี้
   - เปิดพอร์ตสื่อสาร
   - อ่านค่าคุณสมบัติ (พารามิเตอร์) มาตรฐานของพอร์ตในปัจจุบัน
   - เปลี่ยนแปลงพารามิเตอร์บางส่วนให้ตรงตามความต้องการ และกำหนดให้พอร์ตมีค่าตามพารามิเตอร์
2. การปิดพอร์ตสื่อสาร การปิด หรือยกเลิกการใช้พอร์ตสื่อสารในระบบ WIN32 สามารถทำได้ด้วยการใช้คำสั่ง CloseHanndle ด้วยพารามิเตอร์ hCommFile ที่ใช้ควบคุมออบเจ็กต์สื่อสาร
3. การรับข้อมูลจากพอร์ตสื่อสาร การอ่านข้อมูลจากพอร์ตสื่อสารในระบบ WIN32 ทำได้ด้วยฟังก์ชัน ReadFileFile โดยระบุบัฟเฟอร์ที่ใช้รุบข้อมูลด้วย dataP และ ขนาดของบัฟ

เฟอร์ไว้ใน dataLen หลังจากการใช้ฟังก์ชั่น ReadFile ถ้าการประมวลผลสำเร็จ จะให้ค่าเป็นจริง (True)

4. การส่งข้อมูลออกพอร์ตสื่อสาร การส่งข้อมูลออกพอร์ตสื่อสาร สามารถใช้ฟังก์ชัน WriteFile ผ่านพารามิเตอร์ควบคุมด้วยแฮนเดิล hCommFile ระบุตำแหน่งข้อมูลที่ต้องการส่งด้วย dataP ขนาดข้อมูลที่ส่งด้วย dataLen และ ตัวแปรสำหรับรับจำนวนข้อมูลที่ส่งจริง byteSend ในการตรวจสอบข้อมูลที่ส่งออด้วยฟังก์ชัน WritFile ว่าเป็นไปตามที่ต้องการหรือไม่สามารถตรวจสอบได้จากตัวแปร byteSend หรือ เช็คค่าจากตัวแปร overlapWrite ด้วยฟังก์ชัน GetOverlappedResult

5. การทำกิจกรรมเสริมอื่น ๆ เป็นกิจกรรมที่สร้างขึ้นเพื่อสนับสนุนการใช้งานออบเจ็กต์ เช่น การกำหนดค่าเบื้องต้อให้กับระบบของออบเจ็กต์ หรือ กิจกรรมที่ใช้แสดงข้อมูลของโครงสร้างข้อมูลจต่างๆ ที่กำหนดให้กับพอร์ตสื่อสาร เป็นต้น

6.3.3 โปรแกรมการหาตำแหน่งวัตถุโดยการวัดจากค่าความสว่าง

โปรแกรมจะเริ่มการทำงานจากการหาค่าของความสว่างจุดของภาพจากกล้องอย่างหยาบก่อน โดยจะหาค่าความสว่างที่จุดบนภาพทุกๆ 5 จุดทั้งแกน x และ y ของทั้งภาพ โดยสมการ

$$\text{Brightness} = (R+G+B) / 3 \qquad (7.1)$$

โดย

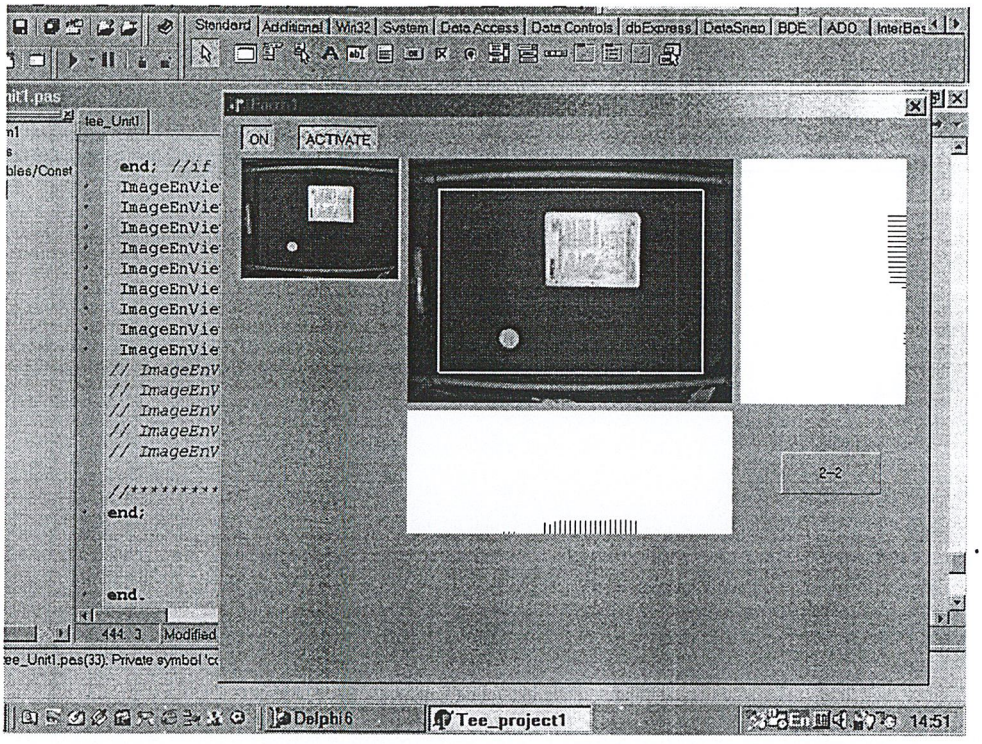
Brightness คือ ค่าความสว่างของจุด Pixel นั้น

R คือ ค่า RED COMPONENT เป็นองค์ประกอบสีแดงในแต่ล่ะ Pixel

G คือ ค่า GREEN COMPONENT เป็นองค์ประกอบสีเขียวในแต่ล่ะ Pixel

B คือ ค่า BLUE COMPONENT เป็นองค์ประกอบสีน้ำเงินในแต่ล่ะ Pixel

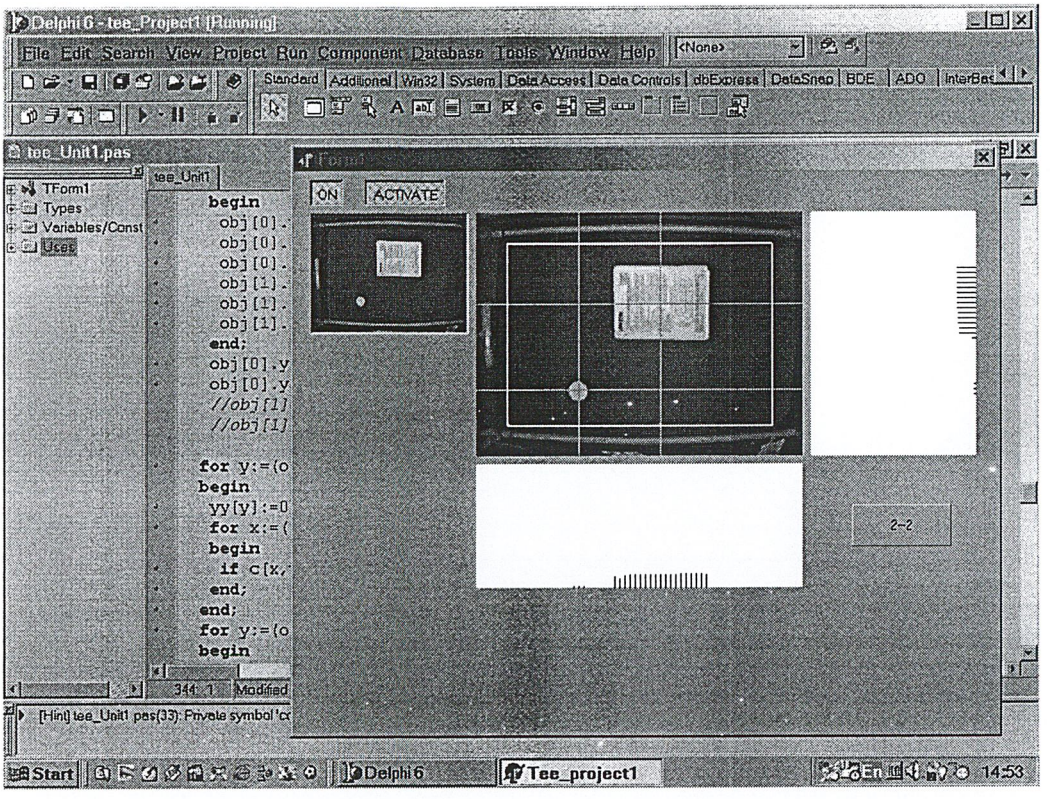
แล้วหลังจากนั้นจะทำการพล็อตค่าจำนวนจุดที่มีความสว่างมากกว่าค่าที่กำหนดทุกๆ ระยะห่าง 5 จุด ของทั้งแกน x และ y ซึ่งจะได้ผลตามรูป



รูปที่ 6.9 กราฟแสดงจำนวนจุดที่มีค่าความสว่างมากกว่าค่าที่กำหนดทุกๆ ระยะบนแกน x และ y

เมื่อเราได้กราฟดังในรูปที่ 6.9 เรา0ก็จะสามารถหาตำแหน่งของวัตถุอย่างคร่าวๆ ได้ หลังจากนั้นเราก็จะทำการหาตำแหน่งอย่างละเอียดจากการอบคร่าวของวัตถุโดยการหาค่าความสว่างอย่างละเอียดขยายออกมาจากกรอบคร่าวๆ นั้น เนื่องจากกรอบคร่าวๆที่เราหาจากทุกๆ 5 จุดบนภาพ จะมีขนาดเล็กกว่าขนาดจริง

หลังจากเราได้ตำแหน่งขอบจริงๆ ของวัตถุแล้ว เราจะนำมาหาค่ากึ่งกลางซึ่งก็จะได้ค่าซึ่งเป็นตำแหน่งของวัตถุนั้นนั่นเอง ตามรูปที่ 6.10



รูปที่ 6.10 แสดงการหาตำแหน่งของวัตถุ

6.3.4 โปรแกรมการเคลื่อนที่เข้าไปยิงลูกบอลอย่างอัตโนมัติของหุ่นยนต์

เริ่มจากการหามุมของหุ่นยนต์ก่อน เพื่อสามารถรู้ได้ว่าหุ่นยนต์หันหน้าไปในทิศทางใด โดยก่อนที่จะเริ่มให้หุ่นยนต์หมุนหมุน หรือ เคลื่อนที่ไปในทิศทางใด จะสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหน้าตรงๆ ก่อน แล้วถอยหลังกลับมายังตำแหน่งเดิมเพื่อเก็บค่าพิกัดตำแหน่ง x และ y เพื่อใช้คำนวณมุมจาก

$$\theta = \tan^{-1}[(y_2-y_1)/(x_2-x_1)] \qquad (7.2)$$

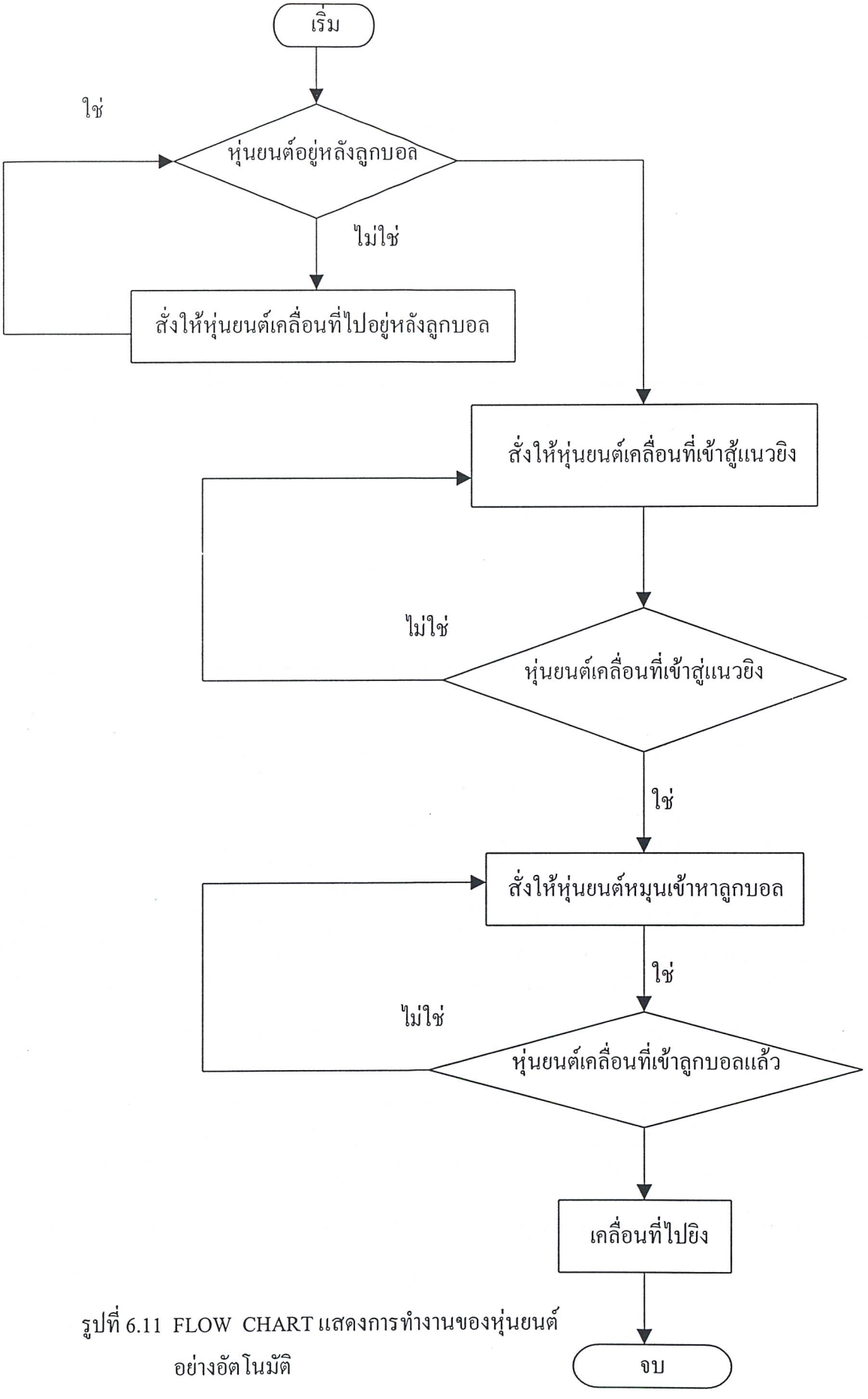
โดย

θ คือ มุมของทิศทางรถที่ทำกับแกน x ของสนาม

y_2, x_2 คือ ค่าพิกัดที่ตำแหน่งแรก

y_2, x_1 คือ ค่าพิกัดของตำแหน่งที่รถเคลื่อนที่ไปก่อนถอยกลับมาที่ตำแหน่งแรก

ส่วนของการเคลื่อนทีไปยิงลูกบอลอย่างอัตโนมัตินั้น จะทำการตรวจสอบก่อนว่าหุ่นยนต์อยู่หลังลูกบอลแล้วหรือยัง เมื่อสั่งให้หุ่นยนต์เข้าไปอยู่หลังลูกบอลแล้วก็จะเคลื่อนที่เข้าไปสู่แนวยิง และทำการหมุนให้หันหน้าเข้าหาลูกบอล แล้วเคลื่อนที่ไปข้างหน้าพร้อมกับทำการยิงบอลให้เข้าประตู



รูปที่ 6.11 FLOW CHART แสดงการทำงานของหุ่นยนต์อย่างอัตโนมัติ

บทที่ 7

การทดลอง

7.1 การทดลองตีกรอบของวัตถุด้วยสี

1. ติดตั้งกล้อง เข้ากับที่ตั้ง
2. ติดตั้งไฟ 1 ดวงเหนือสนาม
3. นำตัวอย่างวัตถุที่ทดลองสีต่างๆมาวางให้กล้องจับภาพที่กึ่งกลางสนาม
4. ทำการ Run โปรแกรม ตีกรอบของวัตถุ
5. สังเกตกรอบที่ตีได้ให้จอแสดงผล เปรียบเทียบกับขนาดจริงของวัตถุตัวอย่าง
6. เลื่อนวัตถุไปยังขอบของสนามทำการทดลองเปรียบเทียบ
7. เปลี่ยนวัตถุเป็นชนิดผิวสะท้อนต่างๆ สีต่างๆ สังเกตผล
8. ทดลองเพิ่มแสงไฟเป็น 2 ดวงโดยติดตั้งไว้ให้อยู่ที่ตำแหน่ง $\frac{1}{4}$ และ $\frac{3}{4}$ ของความยาวสนาม แล้วทำการทดลองซ้ำข้อ 3-7

ตารางที่ 7.1 แสดงผลการทดลองใช้โปรแกรมตีกรอบวัตถุ

ชนิดของวัตถุ ที่นำมาทดลอง	% ของพื้นผิววัสดุที่ตีกรอบได้			
	หลอดไฟดวงเดียว		หลอดไฟ 2 ดวง	
	วางวัตถุไว้กึ่งกลางสนาม	เลื่อนวัตถุไว้ที่ขอบสนาม	วางวัตถุไว้กึ่งกลางสนาม	เลื่อนวัตถุไว้ที่ขอบสนาม
1. พลาสติกผิวด้านสีเขียว	80 %	50 %	85 %	85 %
2. หนังสือห่อปกสีเหลือง	15 %	5 %	10 %	10 %
3. กระดาษสีแดง	97.5 %	50 %	97.5 %	95 %
4. โลหะสีเงิน	5 %	0 %	4 %	4 %
5. พลาสติกผิวด้านสีฟ้า	85 %	55 %	85 %	80 %
6. กระดาษสีขาว	99 %	60 %	99 %	99 %
7. กระดาษสีดำ	99 %	99 %	99 %	99 %

7.2การทดลองการเคลื่อนที่ของหุ่นยนต์

1. ประกอบส่วนการทำงานต่างๆของหุ่นยนต์
2. ทำการ Run โปรแกรมควบคุมการเคลื่อนที่ของหุ่นยนต์
3. ทดลองการวิ่งของหุ่นยนต์ในแนวตรง โดยกำหนดแนวเส้นตรง ยาว 3 เมตร นำหุ่นยนต์มาวางบนเส้นโดยให้เส้นอยู่ระหว่าง 2 ล้อ ใช้ทิศทางของหุ่นยนต์ขนานกับเส้นตรง
4. กดปุ่มสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหน้า วัดระยะทางที่เข้าออกจาแนวเส้นตรงที่ระยะ0.5 m ,1 m ,1.5 m ,2 m ,2.5 m ,3 m
5. ทำแบบเดียวกับข้อ 3 – 4 โดยใช้หุ่นวิ่งถอยหลัง
6. ทดลองสั่งเลี้ยวซ้าย – ขวา จับเวลาในการเลี้ยวจนครบ 1 รอบ
7. ทดลองถ่วงน้ำหนัก แล้ววิ่งเลี้ยวซ้าย – ขวา จับเวลาในการเลี้ยวจนครบ 1 รอบ

ตารางที่ 7.2 แสดงระยะทางที่รถวิ่งไปแล้วเลื่อนออกจากแนวกลาง

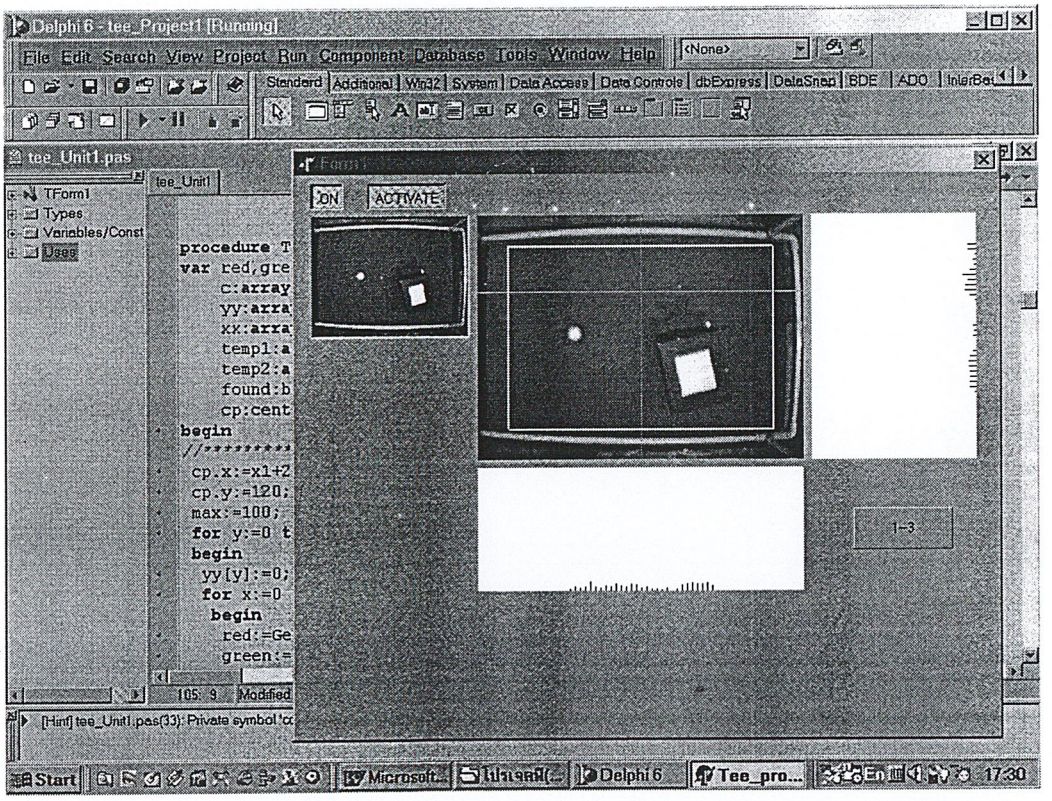
ระยะจากจุดเริ่ม	0.5		1.0		1.5		2.0		2.5		3.0	
(m)	FWD	REV	FWD	REV	FWD	REV	FWD	REV	FWD	REV	FWD	REV
ระยะที่เลื่อนออกจากแนวกึ่งกลาง (cm)	2.0	2.0	5.5	5.0.	9.5	9.0	16.0	14.0	20.0	18.0	25.0	24.0

ตารางที่ 7.3 แสดงผลการทดลองจับเวลาหุ่นยนต์ในการเคลื่อนที่หมุนครบ 1 รอบ

ทิศทางการหมุน	เวลาในการหมุนครบ 1 รอบ (s)		
	ไม่ได้ถ่วงน้ำหนัก	ถ่วงน้ำหนัก	
		ล้อซ้าย	ล้อขวา
หมุนซ้าย	15	20	16
หมุนขวา	12	14	18

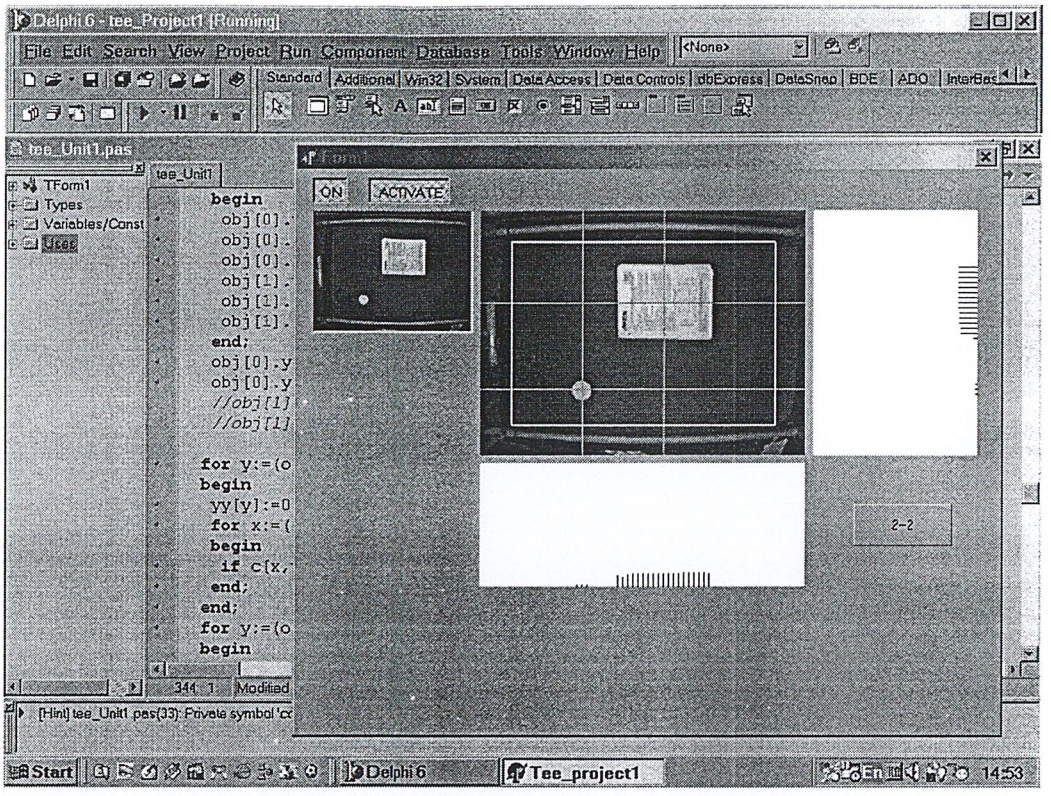
7.3 การทดลองการหาตำแหน่งของวัตถุโดยใช้ค่าความสว่าง

1. ติดตั้งกล้อง เข้ากับที่ตั้ง
2. ติดตั้งหลอดไฟยาว 1 ดวง เหนือสนาม
3. นำหุ่นยนต์ซึ่งทำการกำหนดสัญลักษณ์ไว้แล้วพร้อมกับลูกบอล มาวางไว้ในสนาม
4. ทำการ RUN โปรแกรมหาตำแหน่งวัตถุโดยค่าความสว่าง
5. ทดลองหาตำแหน่งโดยกำหนดค่า ความสว่าง ค่าต่างๆ
6. สังเกตตำแหน่งของวัตถุที่โปรแกรมแสดงออกมา

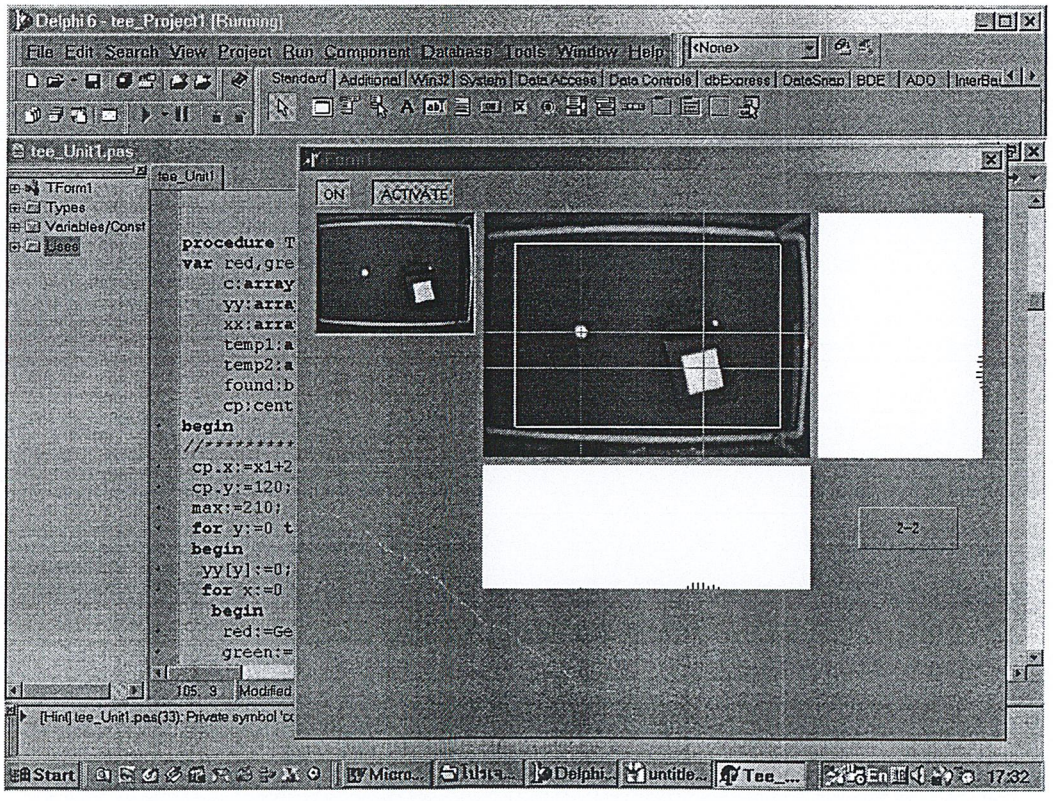


รูปที่ 7.1 ผลการทดลองกำหนดค่าความสว่างน้อยกว่า 150

จากรูปที่ 7.1 เราสามารถสังเกตเห็นได้ว่าเมื่อกำหนดค่าความสว่างขั้นต่ำไว้น้อยเกินไปจะทำให้บริเวณซึ่งไม่ใช่บริเวณของสัญลักษณ์ เช่น บริเวณที่มีแสงจ้าเนื่องจากอยู่ใกล้แหล่งกำเนิดแสงหรือถูกรบกวนจากแสงภายนอก ถูกนับไปด้วยทำให้การแสดงบริเวณของขอบวัตถุมีความคลาดเคลื่อนมาก



รูปที่ 7.2 ผลการทดลองกำหนดค่าความสว่างเท่ากับ 150



รูปที่ 7.3 ผลการทดลองกำหนดค่าความสว่างมากกว่า 150

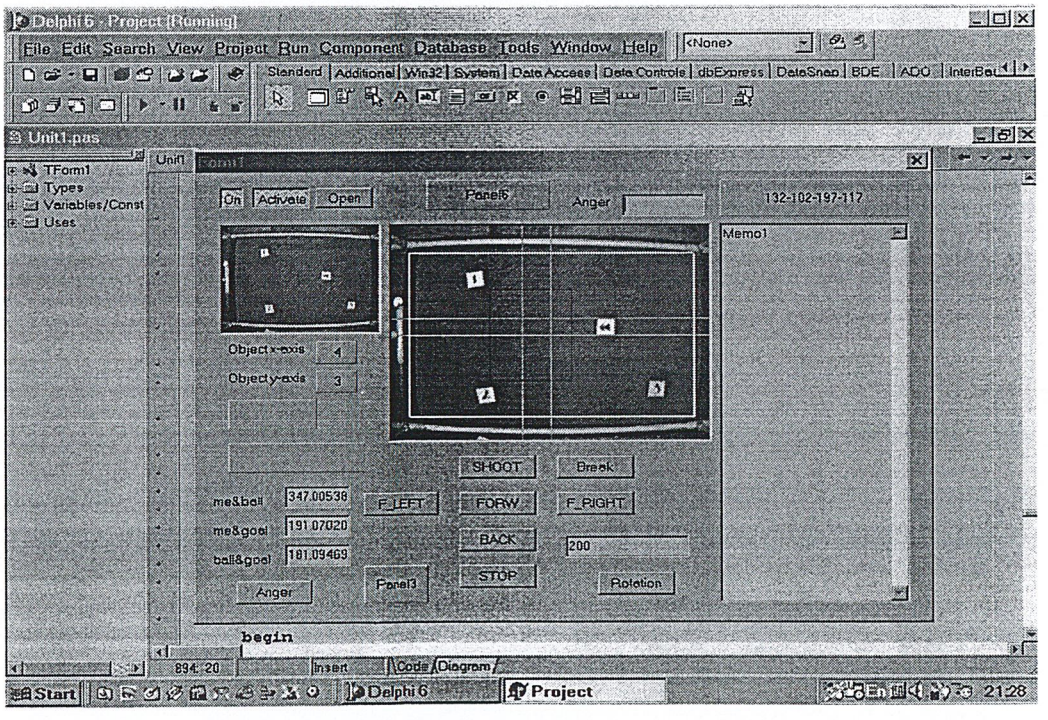
จากรูปที่ 7.3 จะสังเกตเห็นว่าการกำหนดค่าความสว่างขั้นต่ำมีค่าสูงจะทำให้บางบริเวณของสัญลักษณ์ไม่ถูกนับ ทำให้มีความคลาดเคลื่อนเล็กน้อยของขอบวัตถุเกิดขึ้น

ตารางที่ 7.4 แสดงผลการทดลองการหาตำแหน่งของวัตถุโดยใช้ค่าความสว่าง

ค่าความสว่างที่กำหนด	ตำแหน่งที่โปรแกรมแสดงออกมาเทียบกับตำแหน่งจริง
50	คลาดเคลื่อนมาก
100	คลาดเคลื่อนมาก
150	คลาดเคลื่อนน้อยมาก
200	คลาดเคลื่อนเล็กน้อย
250	คลาดเคลื่อนค่อนข้างมาก

7.4 การทดลองการสั่งให้รถเคลื่อนที่ไปยังตำแหน่งที่สามารถยิงลูกโดยอัตโนมัติ

1. ติดตั้งกล้อง เข้ากับที่ตั้ง
2. ติดตั้งหลอดไฟยาว 1 ดวง เหนือสนาม
3. กำหนดตำแหน่งเริ่มต้นของหุ่นยนต์บนสนาม 3 ตำแหน่งตามรูป



รูปที่ 7.4 แสดงตำแหน่งที่กำหนดเป็นจุดเริ่มต้นของหุ่นยนต์ และจุดสิ้นสุด ซึ่งต้องหันไปตามทิศของลูกศร

4. นำหุ่นยนต์ไปปล่อยที่ตำแหน่ง ที่กำหนด แล้วทดลอง RUN โปรแกรมสั่งให้หุ่นยนต์เคลื่อนที่มายังตำแหน่ง และ หันหน้าไปตามทิศทางที่กำหนด ทำการทดลองตำแหน่งล่ะ 5 ครั้ง

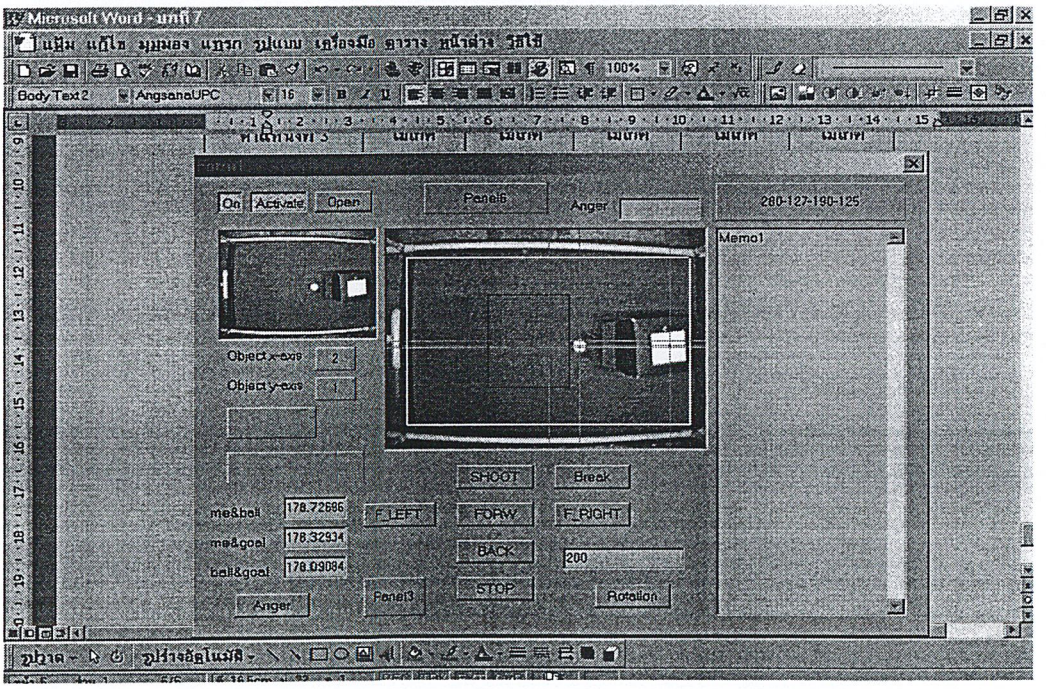
5. สังเกตผลการทดลอง และ หาสาเหตุของความผิดพลาด พร้อมทั้งแก้ไขสาเหตุ

ตารางที่ 7.5 แสดงผลการทดลอง**การสั่งให้รถเคลื่อนที่ไปยังตำแหน่งที่สามารถยิงลูกโดยอัตโนมัติ**

ตำแหน่งเริ่มต้นของการทดลอง	การเกิดความผิดพลาดของการทดลอง				
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5
ตำแหน่งที่ 1	เกิด	เกิด	ไม่เกิด	ไม่เกิด	ไม่เกิด
ตำแหน่งที่ 2	เกิด	เกิด	ไม่เกิด	ไม่เกิด	ไม่เกิด
ตำแหน่งที่ 3	ไม่เกิด	ไม่เกิด	ไม่เกิด	ไม่เกิด	ไม่เกิด

การทดลองที่ 7.5 การทดลองหาประสิทธิภาพของการยิงลูกบอลเข้าประตู

1. ติดตั้งกล้อง เข้ากับที่ตั้ง
2. ติดตั้งหลอดไฟยาว 1 ดวง เหนือสนาม
3. นำหุ่นยนต์ และ ลูกบอลไปวางไว้ในตำแหน่งที่กำหนด



รูปที่ 7.5 แสดงตำแหน่งที่วางลูกบอล และ หุ่นยนต์

4. ทำการ RUN โปรแกรมให้หุ่นยนต์ เคลื่อนที่ไปข้างหน้าพร้อมกับยิงลูกบอล
5. ทำการทดลองซ้ำจำนวน 50 ครั้งเพื่อบันทึกจำนวนครั้งที่ยิงลูกบอลเข้าประตู

ตารางที่ 7.6 แสดงประสิทธิภาพของการยิงลูกบอลเข้าประตู

จำนวนครั้งที่ยิง	จำนวนลูกบอลที่ยิงเข้าประตู	เปอร์เซ็นต์ความแม่นยำ
50	15	30 %

บทที่ 8

สรุปและวิจารณ์ผลการทดลอง

8.1 สรุปและวิจารณ์ผลการทดลอง

1. การทดลองตีกรอบรอบวัตถุ

จากการทดลอง จะเห็นว่าปัจจัยที่มีผลกระทบต่อการทำงานของระบบ ซึ่งทำให้เกิดข้อผิดพลาดในการทำงาน และได้ข้อจำกัดในการใช้งานระบบโดยระบบจะทำงานได้ไม่สมบูรณ์ถ้าข้อมูลภาพมีลักษณะดังนี้

- แสงจากภายนอกระบบที่มีความเข้มสูง เข้ามารบกวนระบบ
- การติดตั้งกล้องจะต้องได้มุม และระดับไม่อยู่ในตำแหน่งเดียวกัน
- การให้แสงภายในระบบมีค่าไม่เหมาะสม
- ลักษณะพื้นผิวมีการสะท้อนแสงมาก

จากผลการทดลองจะพบว่าแสงที่ฉายให้แก่ระบบ เป็นปัจจัยหลักที่จะทำให้เกิดความผิดพลาดมากที่สุด ซึ่งสังเกตได้จาก ขนาดพื้นที่ของภาพที่หาได้เมื่อ ระดับความเข้มแสงแตกต่างกัน ดังนั้นในการทำงานจริงจะต้องมีการควบคุมระบบ และป้องกันไม่ให้แสงจากภายนอก มีผลกระทบมากกว่าแสงของระบบ

ส่วนประสิทธิภาพของการตีกรอบ ถ้าพิจารณาเปอร์เซ็นต์ที่ตีกรอบได้ เทียบกับพื้นที่จริงของวัตถุ เมื่อมีการจัดสภาพแวดล้อมให้ดีพบว่า มีความใกล้เคียงกับพื้นที่จริงของวัตถุแต่เมื่อจัดแสดงไม่ดี เช่นไม่สม่ำเสมอ หรือมีแสงมามากจนทำให้วัตถุบางชนิดสะท้อนแสดงจะทำให้บริเวณที่สะท้อนแสงซึ่งกล้องจับได้เป็นสีเขาว เกิดการแยกแยะผิดพลาดขึ้น

แต่ผลที่ได้จากการตีกรอบก็สามารถบอกตำแหน่งของวัตถุตามวัตถุประสงค์ที่เราต้องการได้ แต่ก็จะสามารถทำให้มีความแม่นยำมากขึ้น โดยจัดแสงให้เหมาะสม และเลือกกำหนดสีของวัสดุให้เป็นสีด้านที่ไม่สะท้อนแสง

2. การทดลองเคลื่อนที่ของหุ่นยนต์

จากการทดลองพบว่า หุ่นยนต์เดินไม่ตรง ซึ่งมีสาเหตุมาจาก

- การถ่วงน้ำหนักของหุ่นยนต์ 2 ข้าง ไม่สมดุล
- มอเตอร์ขับเคลื่อนทั้ง 2 ตัว หมุนเร็วไม่เท่ากัน

ซึ่งสาเหตุที่สามารถแก้ได้คือ นำน้ำหนักมาถ่วงจนทำให้หุ่นยนต์สามารถเคลื่อนที่ตรงได้ หรือเพิ่มส่วนของ Software คอยตรวจสอบการเคลื่อนที่ของหุ่นยนต์โดยปรับให้เข้าแนวตรง ผ่านทางกล้องที่ติดตั้งไว้ด้านบน

3. การทดลองการหาตำแหน่งของวัตถุโดยใช้ค่าความสว่าง

จากการทดลองจะพบว่า การใช้ค่าความสว่างในการหาตำแหน่งของวัตถุพบว่าปัจจัยต่าง ๆ

ซึ่งจะทำให้การประมวลผลคลาดเคลื่อน ดังเช่นการใช้สีตีกรอบวัตถุ จะถูกทำให้มีผลรบกวนต่อการประมวลผลน้อยลง เพราะ เราจะใช้วัตถุผิวด้านสีดำห่อหุ้มตัวหุ่นไว้ ส่วนสัญลักษณ์ที่ใช้ในการแสดงตำแหน่งก็เป็นสีขาว ทำให้เกิดความคลาดเคลื่อนเนื่องจากการจัดแสง แสงจากภายนอก ลักษณะพื้นผิวที่สะท้อนแสง ไม่เกิดขึ้น ส่วนค่าความสว่างที่เหมาะสมก็คือค่า 150 ถ้าเรากำหนดค่าความสว่างน้อยกว่านี้จะทำให้มีจุดซึ่งไม่ใช่สัญลักษณ์ที่เรากำหนดถูกนับ ดังจะเห็นได้จากกราฟของจำนวนจุดที่มีค่ามากกว่าค่าความสว่างที่กำหนดมีลักษณะไม่เป็นไปตามลักษณะของสัญลักษณ์ และถ้าหากเราใช้ค่าความสว่างมีค่ามากกว่า 150 แล้ว ก็จะทำให้บางจุดบนสัญลักษณ์ที่เรากำหนดทำให้การหาตำแหน่งเกิดการผิดเพี้ยนได้

4. การทดลองการสั่งให้รถเคลื่อนที่ไปยังตำแหน่งที่สามารถยิงลูกโดยอัตโนมัติ

จากผลการทดลองพบว่า ในการเคลื่อนที่ของหุ่นยนต์อย่างอัตโนมัติ เข้าไปยังเป้าหมายที่เรากำหนดสามารถทำได้อย่างมีประสิทธิภาพ โดยทำการทดลองหลายครั้งหลายกรณีทำให้เราสามารถหาสาเหตุความผิดพลาดและแก้ไขได้ โดยความผิดพลาดที่เกิดขึ้นนั้นมีสาเหตุมาจาก ความผิดพลาดจากการเขียนโปรแกรม การใช้แบตเตอรี่ที่มีกำลังอ่อนทำให้การกำหนดระยะเวลาการทำงานของมอเตอร์ผิดพลาด รวมทั้งความผิดพลาดที่เกิดมาจากความบิดเอียงของภาพที่รับมาจากกล้อง เราจึงจำเป็นที่จะต้องหามุมและตำแหน่งของหุ่นยนต์ทุกๆ เฟรมภาพก่อนที่จะทำการสั่งให้มีการเคลื่อนที่ เพื่อให้เกิดความผิดพลาดน้อยที่สุด

5. การทดลองหาประสิทธิภาพของการยิงลูกบอลเข้าประตู

จากการทดลองพบว่าเมื่อเรากำหนดให้ตำแหน่ง และ ทิศทางการของหุ่นยนต์สามารถยิงลูกบอลให้เข้าประตูแน่ๆแล้วแต่ผลการทดลองพบว่ามีบางครั้งที่ไม่สามารถยิงบอลเข้าประตูได้ แสดงว่ายังมีข้อจำกัดอื่นๆที่ทำให้ประสิทธิภาพการยิงลูกบอลไม่เป็นไปตามที่เราคลาดไว้ ดังเช่น การทำงานของมอเตอร์ขับเคลื่อนทั้งสองข้างไม่เท่ากัน หรือ ปัจจัยจากภายนอกซึ่งทำให้ลูกบอลไม่สามารถเคลื่อนที่ไปได้ตรงๆ เช่น พื้นสนามไม่ได้ระดับสม่ำเสมอ จุดที่เครื่องยิงสัมผัสลูกบอลตอนยิงไม่ตรงกับจุดศูนย์ถ่วงของลูกบอลพอดี

8.2 สิ่งที่ได้ดำเนินการในภาคเรียนที่ 1

1. ส่วนของ Software
   - เขียนโปรแกรมการรับภาพจากกล้อง
   - เขียนโปรแกรมการตีกรอบวัตถุด้วยการแยกสี
   - เขียนโปรแกรมการเคลื่อนที่ของหุ่นยนต์โดยการบังคับแบบ manual

2. ส่วนของ Hardware
   - ตัวหุ่นยนต์
   - วงจร รับ – ส่ง ไร้สาย
   - วงจรเชื่อมโยงปรับระดับสัญญาณ Max 232
   - ชุดวงจร Microcontroler
   - สนามแข่งขัน

8.3 สิ่งที่ได้ดำเนินการในภาคเรียนที่ 2

- ทำการควบคุมความสว่างของสนามทดลองใช้สม่ำเสมอ ให้สีของหุ่นยนต์และลูกบอล เป็นสีด้านไม่สะท้อนแสง โดยกำหนดให้มีเพียง 2 สี คือ สีขาว และ สีดำ
- การปรบปรุง Software ให้หุ่นยนต์เดินตรง
- พัฒนาโปรแกรมในการหาตำแหน่งเป็นโปรแกรมการหาตำแหน่งแบบใช้ค่าความสว่าง
- เปลี่ยน Motor เพิ่มความเร็วขึ้น และ พัฒนาโครงสร้างของหุ่นยนต์ให้แข็งแรงขึ้น
- ติดชุดยิงบอลในหุ่นยนต์
- เก็บข้อมูลในการบังคับ แบบ Manual เพื่อไปใช้ในการควบคุมแบบอัตโนมัติ

หนังสืออ้างอิง

1. Maher . A . SidAhhmed , " Image Processing theory algorithms & architecture " , McGraw-Hill International , 1992
2. Philip E. Mattisonn , " Practical digital video with programming examples in C " , John Wiley & Sons , Inc. , 1994
3. สัจจะ จรัสรุ่งรวีวร, " Delphi 5 ฉบับสมบูรณ์ " , Info Press , 2543
4. ชัยวัฒน์ ลิ้มพรจิตรวิไล , " เรียนรู้และปฏิบัติการ ไมโครคอนโทรลเลอร์ " , Innovative Experament , Inc , 1988
5. ไกรวุฒิ โรจน์ประเสริฐสุด , " วิเคราะห์เจาะลึกทั้งทฤษฎี และ ปฏิบัติ MCS-51 ", ซีเอ็ดยูเคชั่น , 2539

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ คุณพ่อ และคุณแม่ ที่ให้การสนับสนุนด้านการศึกษาคอยดูแลเอาใจใส่และห่วงใยตลอดมาจนการทำงานครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณ ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษาของข้าพเจ้า และ อาจารย์หลายๆท่านที่ให้ความอนุเคราะห์ทั้งทางด้านความรู้ และ อุปกรณ์ในการทำงาน รวมทั้งให้คำปรึกษาและ แนะนำแนวทางการทำโครงงานวิทยานิพนธ์นี้

ขอขอบคุณพี่ๆ และ เพื่อนๆ ทุกคนที่คอยให้คำแนะนำ กำลังใจ และ ความช่วยเหลือ ในการทำงานนี้มาโดยตลอด

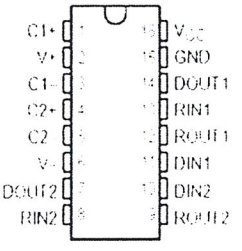
ภาคผนวก

3-V TO 5.5-V MULTICHANNEL RS-232 LINE DRIVER/RECEIVER

SLLS341N – JANUARY 2000 – REVISED [illegible]

- Meets or Exceeds the Requirements of TIA/EIA-232-F and ITU v.28 Standards
- Operates With 3-V to 5.5-V V_{CC} Supply
- Operates up to 250 kbit/s
- Low Supply Current . . . 300 µA Typical
- External Capacitors . . . 4 × 0.1 µF
- Accepts 5-V Logic Input With 3.3-V Supply
- Designed to Be Interchangeable With Maxim MAX3232
- RS-232 Bus-Pin ESD Protection Exceeds ±15 kV Using Human-Body Model (HBM)
- Applications
  - Battery-Powered Systems, PDAs, Notebooks, Laptops, Palmtop PCs, and Hand-Held Equipment
- Package Options Include Plastic Small-Outline (D, DW), Shrink Small-Outline (DB), and Thin Shrink Small-Outline (PW) Packages

D, DB, DW, OR PW PACKAGE
(TOP VIEW)



description

The MAX3232 device consists of two line drivers, two line receivers, and a dual charge pump circuit with ±15-kV ESD protection pin to pin (serial-port connection pins, including GND). The device meets the requirements of TIA/EIA-232-F and provides the electrical interface between an asynchronous communication controller and the serial-port connector. The charge pump and four small external capacitors allow operation from a single 3-V to 5.5-V supply. The devices operate at data signaling rates up to 250 kbit/s and a maximum of 30-V/µs driver output slew rate.

The MAX3232C is characterized for operation from 0°C to 70°C. The MAX3232I is characterized for operation from –40°C to 85°C.

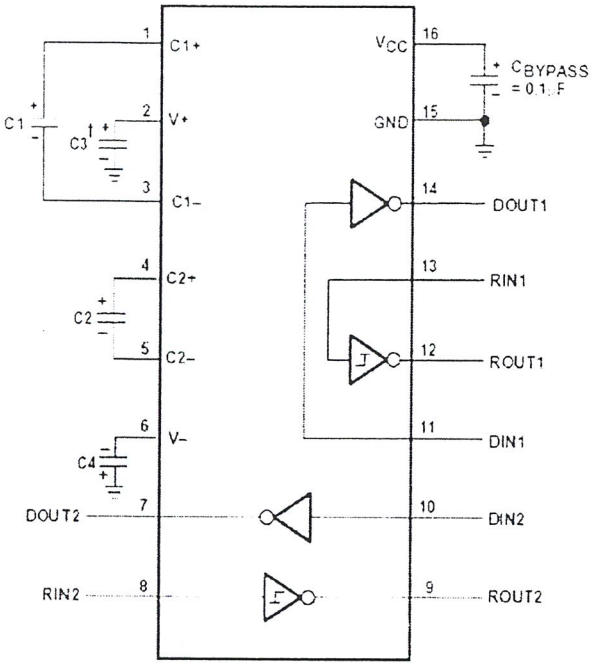
AVAILABLE OPTIONS

T_A	PACKAGED DEVICES			
	SMALL OUTLINE (D)	SHRINK SMALL OUTLINE (DB)	SMALL OUTLINE (DW)	THIN SHRINK SMALL OUTLINE (PW)
0°C to 70°C	MAX3232CD	MAX3232CDB	MAX3232CDW	MAX3232CPW
–40°C to 85°C	MAX3232ID	MAX3232IDB	MAX3232IDW	MAX3232IPW

The D, DB, DW, and PW packages are available taped and reeled. Add the suffix R to device type (e.g., MAX3232CDR).

SLES104D – JANUARY 2000 – REVISED AUGUST 2004

APPLICATION INFORMATION



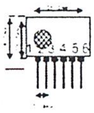
V_{CC} vs CAPACITOR VALUES

V_{CC}	C1	C2, C3, C4
3.3 V ± 0.3 V	0.1 µF	0.1 µF
5 V ± 0.5 V	0.047 µF	0.33 µF
3 V to 5.5 V	0.1 µF	0.47 µF

Figure 4. Typical Operating Circuit and Capacitor Values

TWS-434 TRANSMITTER

NEW!



pin 1 : Vcc
pin 2 : Vcc
pin 3 : Gnd
pin 4 : Gnd
pin 5 : RF Output
pin 6 : Code Input

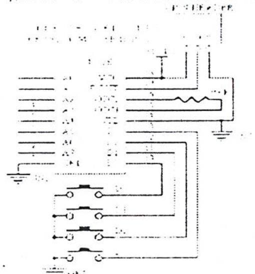
Frequency 300 - 433MHz

Moulation : AM (Code)
Supply Voltage : 1.5v - 12 v dc
RF Output Power : 8mW

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating supply voltage		1.5	-	12	V
Icc	Peak Current		-	5	[illegible]	mA
Vih	Input High Voltage	I(Data) = 100uA (High)	Vcc-0.5	-	Vcc	V
Vil	Input Low Voltage	I(Data) = [illegible] (Low)	-	-	0.3	V
[illegible]	Absolute Frequency		314.8	315	433.[illegible]	MHz
Po	RF Out Power Into 50Ω		-3	0	+2	dBm
	Modulation Bandwidth	External Encoding	-	5		KHz
Tr	Modulation Rise Time		-	-	100	uS
Tf	Modulation Fall Time		-	-	100	uS

Notes : (Case Temperature = 25°C ± 2°C Test Load Impedance = 50 Ω)

Application Circuit: 4-bit RF transmitter using the Holtek HT-12E encoder



Connections:
1. TWS Gnd to header-pin #3
2. TWS Vcc to header-pin #1
3. TWS Code-Input to header-pin #2.

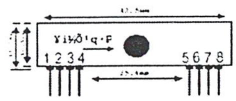
NOTE: TE can be switched to ground to provide transmit on button-press

More detailed information is available from http://www.rentron.com

For 8-bit RF remote control substitute the HT-12E with HT-640 encoder

Maximum range obtained with 5-watt antenna, 35cm in length on TWS & RWS.

RWS-434 RECEIVER



pin 1 : Gnd
pin 2 : Digital Output
pin 3 : Linear Output
pin 4 : Vcc
pin 5 : Vcc
pin 6 : Gnd
pin 7 : Gnd
pin 8 : Ant (About 30 - 35 cm)

Frequency 300 - 433MHz

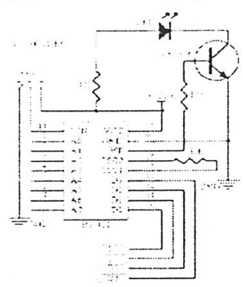
Moulation : AM
Supply Voltage : 4.5v - 5.5 v dc
Sensitivity : 3uVrms
Output : Digital & Linear
Application : Radio Remote Control

Vcc	Operating supply voltage		-	[illegible]	-	
V Data	Data out	I Data = +200 uA (High)	Vcc-0.5		Vcc	V
		I Data = -10 uA (Low)	-	-	0.3	V

Electrical Characteristics

Characteristics	SYM	Min	Typ	Max	Unit
Operation Radio Frequency	FC	300 - 434			MHz
Sensitivity	Pref			-106	dBm
Channel Width		±500			KHz
Noise equivalent BW	NEB		5	4	KHz
Baseboard data rate				3	KB/s

Application Circuit: 4-bit RF Receiver using the Holtek HT-12D decoder



Connections
1. RWS Gnd to header-pin #1
2. RWS Vcc to header-pin #3
3. RWS Digital-Output to header-pin #2

HT-12E encoder data transfers to HT-12D data-pins via RF transmission

VT-pin transitions from low to high on valid reception & address

More detailed information is available from http://www.rentron.com

For 8-bit RF remote control substitute the HT-12D with HT-648L decoder

Reynolds Electronics
3101 Eastridge Lane, Canon City, Co. 81212 http://www.rentron.com
Tel: (719) 269-3469 Fax: (719) 276-2853 e-mail webmaster@rentron.com

HOLTEK

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
- Pair with Holtek's 2^{12} series of encoders
- 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

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.

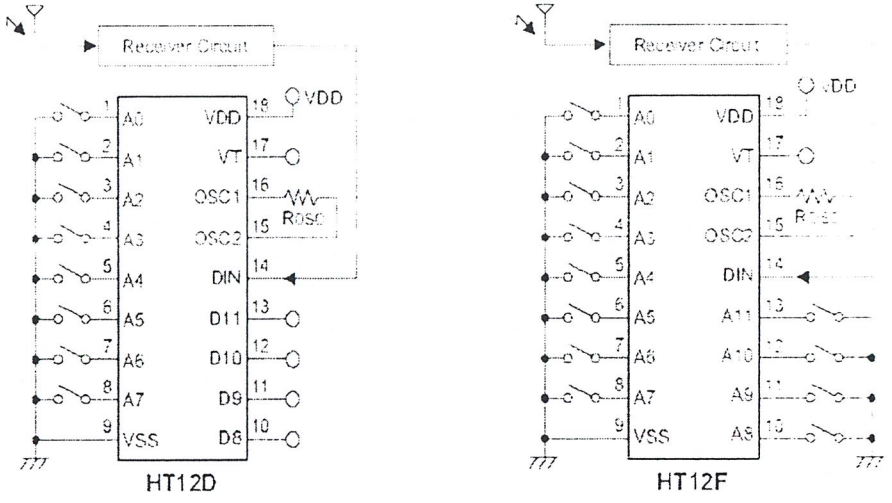
Selection Table

Part No. \ Function	Address No.	Data No.	Data Type	VT	Oscillator	Trigger	Package
HT12D	8	4	L	1	RC oscillator	DIN active "Hi"	18 DIP/20 SOP
HT12F	12	0	—	1	RC oscillator	DIN active "Hi"	18 DIP/20 SOP

Notes: Data type: L stands for latch type data output.

VT can be used as a momentary data output.

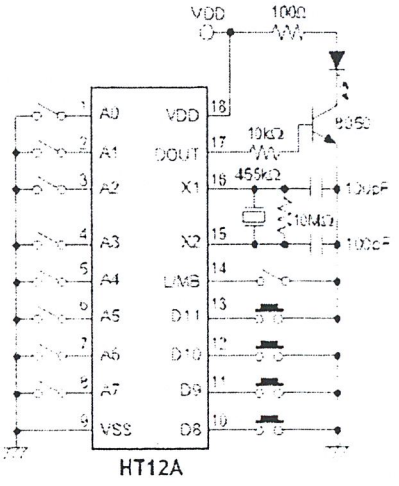
Application Circuits

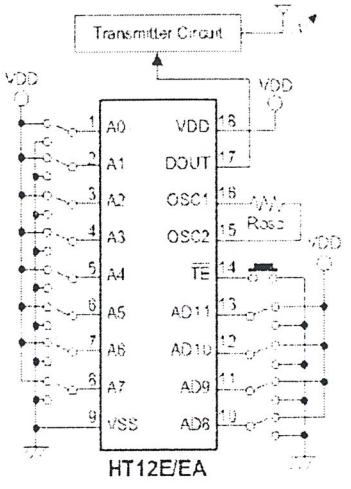


Notes: Typical infrared receiver: PIC-12043TYPIC-12043S (KODESHI CORP.)
or LTM9052 (LITEON CORP.)

Typical RF receiver: JR-209 (JUWA CORP.)
RE.99 (MING MICROSYSTEM, U.S.A.)

Application Circuits





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

Features

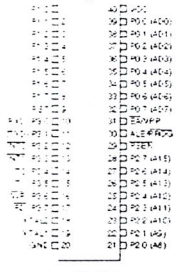
- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

Description

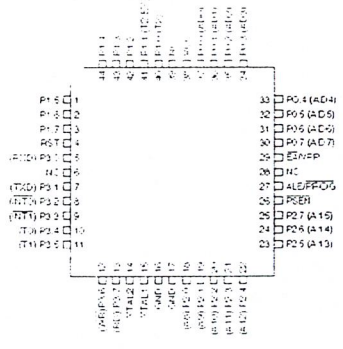
The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

Pin Configurations

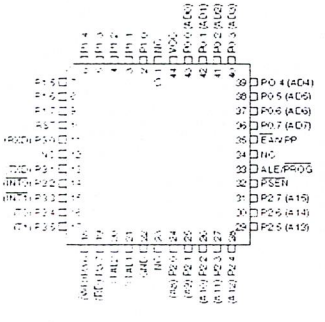
PDIP



PQFP/TQFP



PLCC



8-bit Microcontroller with 4K Bytes Flash

AT89C51

Rev. 0265B–11/97

Flash Programming Modes

Mode		RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{pp}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L		H/12V	H	H	H	H
	Bit - 2	H	L		H/12V	H	H	L	L
	Bit - 3	H	L		H/12V	H	L	H	L
Chip Erase		H	L		H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Note: 1. Chip Erase requires a 10 ms $\overline{PROG}$ pulse

Figure 3. Programming the Flash

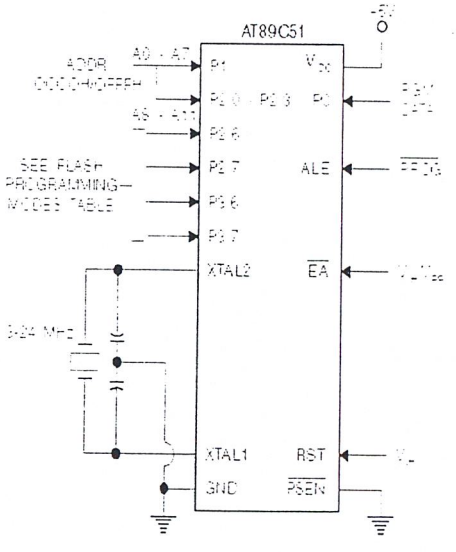
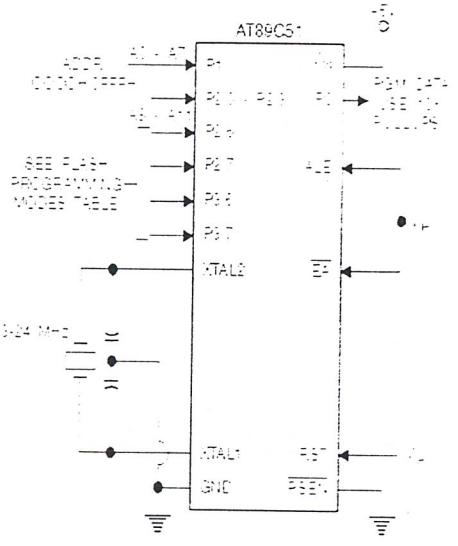
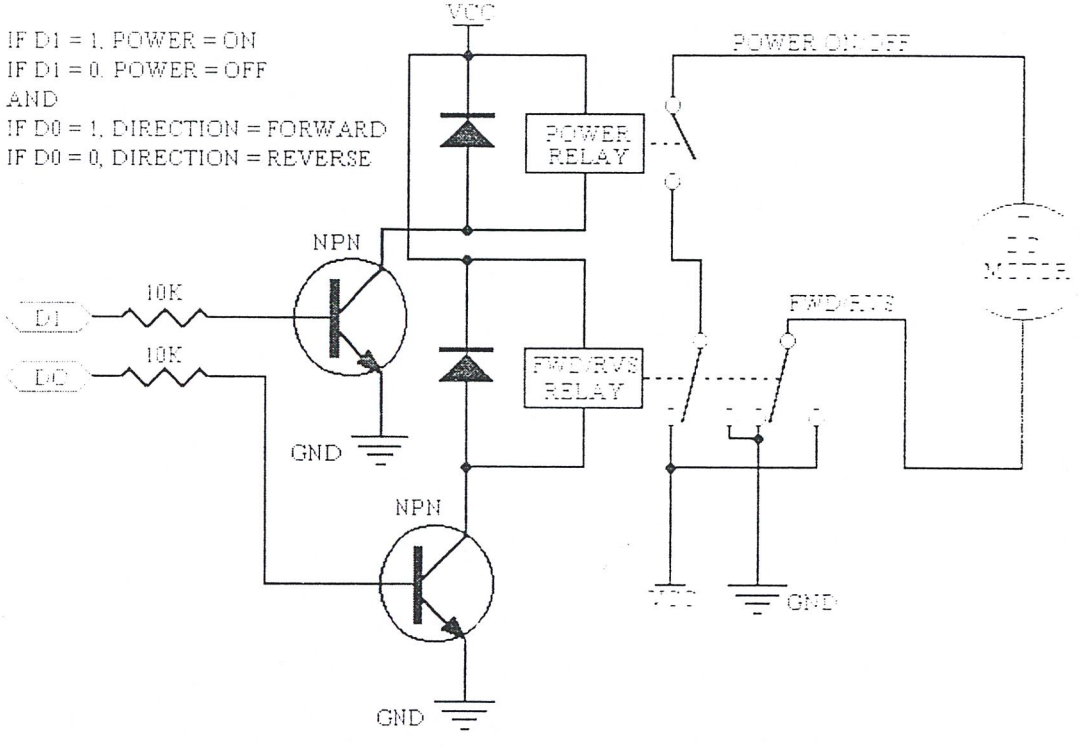


Figure 4. Verifying the Flash




IF D1 = 1, POWER = ON
IF D1 = 0, POWER = OFF
AND
IF D0 = 1, DIRECTION = FORWARD
IF D0 = 0, DIRECTION = REVERSE
VCC
POWER ON/OFF
POWER RELAY
NPN
10K
D1
10K
D0
FWD/RVS RELAY
FWD/RVS
DC MOTOR
GND
NPN
VCC
GND
GND

```
unit uComm32;
//ออปเจ็กต์สำหรับส่งข้อมูลผ่านพอร์ตสื่อสาร
interface

uses Classes, Dialogs, Messages, SysUtils, Windows;

const
  // define max comm. port possible
  MAX_COMM_PORT = 4;

type

  TCommInfo = record
    hCommFile : THandle;
    commPort  : byte;
    rxBuffer  : integer;
    txBuffer  : integer;
    baudRate  : integer;
    byteSize  : integer;
    parity    : integer;
    stopBits  : integer;
    errorFlg  : boolean;
  end;

  Tcomm32 = class( tobject )
  public
     myCommInfo : TCommInfo;
    Constructor Create( commInfo:TCommInfo );
    Destructor  Free;

    function  SendData(dataP:pointer; dataLen:dword) : integer;
    function  ReadData(dataP:pointer; dataLen:dword) : integer;
    procedure CloseComm32;
  private
     commDcb     : Tdcb
     commProp    : TCommProp
     evtMask     : dword
     commTimeOuts : TCommTimeouts;
     dispStr     : Tstrings
     overlapWrite : TOverlapped;
     overlapRead  : TOverlapped;
    procedure Echo( content : shortString );

    procedure DispComDCB( var dspDcb : Tdcb );
    procedure DispComProperties( var dspProp : TCommProp );
    procedure DispComTimeouts( var dspComTmeOut : TCommTimeouts );

    procedure VerifyCommInfo( var commInfo : TCommInfo );
```

```
    Echo( '  XonChar   = "'+XonChar+'"' )
    Echo( '  XffChar   = "'+XoffChar+'"' )
    Echo( '  ErrorChar = "'+errorChar+'"' );
    Echo( '  EofChar   = "'+EofChar+'"' )
    Echo( '  EvtChar   = "'+EvtChar+'"' )
    Echo( '  wReserved1= '+IntToStr( wReserved1 ) ); Echo( ' ' )
  end;
end;

procedure Tcomm32.DispComProperties( var dspProp : TCommProp );
begin
  with dspProp do
  begin
    Echo('Communications Properties : ' );
    Echo('  wPacketLength      = '+IntToStr(wPacketLength))
    Echo('  wPacketVersion     = '+IntToStr(wPacketVersion))
    Echo('  dwServiceMask      = '+IntToStr(dwServiceMask))
    Echo('  dwReserved1        = '+IntToStr(dwReserved1))
    Echo('  dwMaxTxQueue       = '+IntToStr(dwMaxTxQueue))
    Echo('  dwMaxRxQueue       = '+IntToStr(dwMaxRxQueue))
    Echo('  dwMaxBaud          = '+IntToStr(dwMaxBaud))
    Echo('  dwProvSubType      = '+IntToStr(dwProvSubType))
    Echo('  dwProvCapabilities = '+IntToStr(dwProvCapabilities));
    Echo('  dwSettablePrrams   = '+IntToStr(dwSettableParams));
    Echo('  dwSettableBaud     = '+IntToStr(dwSetTableBaud))
    Echo('  wSettableData      = '+IntToStr(wSetTableData))
    Echo('  wSetTableStopParity= '+IntToStr(wSetTableStopParity));
    Echo('  dwCurrentTxQueue   = '+IntToStr(dwCurrentTxQueue ))
    Echo('  dwCurrentRxQueue   = '+IntToStr(dwCurrentRxQueue ))
    Echo('  dwProvSpec1        = '+IntToStr(dwProvSpec1))
    Echo('  dwProvSpec2        = '+IntToStr(dwProvSpec2))
    Echo(' ' );
  end;
end;

procedure Tcomm32.DispComTimeouts(var dspComTmeOut:TCommTimeouts);
begin
  with dspComTmeOut do
  begin
    Echo('Communications Timeouts information : ' );
    Echo('  ReadIntervalTimeout         = '+
        IntToStr( ReadIntervalTimeout ) )
    Echo('  ReadTotalTimeoutMultiplier  = '+
        IntToStr( ReadTotalTimeoutMultiplier ) )
    Echo('  ReadTotalTimeoutConstant    = '+
        IntToStr( ReadTotalTimeoutConstant ) )
    Echo('  WriteTotalTimeoutMultiplier = '+
        IntToStr( WriteTotalTimeoutMultiplier ) )
```

```
    procedure OpenComm32( var commInfo : TCommInfo );
 //   procedure CloseComm32;
  end;

implementation

//-------------------------------------------------------------

constructor Tcomm32.Create(commInfo:TCommInfo);
begin
  inherited Create;
//  dispStr:=dspStr;
  fillchar( commDcb, sizeof( Tdcb ),0 );
  fillchar( commProp, sizeof( TCommProp ),0 );
  fillchar( commTimeOuts, sizeof( TCommTimeouts ),0 );
  fillchar( overlapWrite, sizeof( TOverlapped ),0 );
  fillchar( overlapRead, sizeof( TOverlapped ),0 );
  OpenComm32( commInfo );
end;

destructor Tcomm32.Free;
begin
  CloseComm32;
end;

//-------------------------------------------------------------

procedure Tcomm32.Echo( content : shortString );
begin
  if ( dispStr=nil ) then exit;
  if ( dispStr.count>300 ) then dispStr.delete(0);
  dispStr.add( content );
end;

procedure Tcomm32.DispComDCB( var dspDcb : Tdcb );
begin
  with dspDcb do
  begin
    Echo( 'Communications Device Control Block information : ' )
    Echo( '  DCBLength = '+IntToStr( DCBLength ) )
    Echo( '  BaudRate  = '+IntToStr( BaudRate ) )
    Echo( '  Flags     = '+IntToStr( Flags ) )
    Echo( '  wReserved = '+IntToStr( wReserved ) )
    Echo( '  XonLim    = '+IntToStr( XonLim ) )
    Echo( '  XoffLim   = '+IntToStr( XoffLim ) )
    Echo( '  ByteSize  = '+IntToStr( byteSize ) )
    Echo( '  Parity    = '+IntToStr( Parity ) )
    Echo( '  StopBits  = '+IntToStr( StopBits ) )
```

```
    Echo('  WriteTotalTimeoutConstant     = '+
        IntToStr( WriteTotalTimeoutConstant ) )
    Echo(' ');
  end;
end;

//-------------------------------------------------------------

procedure Tcomm32.VerifyCommInfo( var commInfo : TCommInfo );
begin
  with commInfo do
  begin
    if ( not commPort in [1..MAX_COMM_PORT] ) then commPort:=1
    if ( rxBuffer<4096 ) then rxBuffer:=4096;
    if ( txBuffer<1024 ) then txBuffer:=1024;
    if ( baudRate<1 ) then baudRate:=CBR_9600;
    if ( not byteSize in [ 4..8 ] ) then byteSize:=8
    if ( not parity in [ EVENPARITY,MARKPARITY,NOPARITY,
      ODDPARITY ] ) then parity:=NOPARITY;
    if not stopBits in [ONESTOPBIT,ONE5STOPBITS,TWOSTOPBITS] then
      stopBits:=ONESTOPBIT;
  end;
end;

procedure Tcomm32.OpenComm32( var commInfo : TCommInfo );
var commPortP : pchar; commPortStr,s : ShortString;
begin
  myCommInfo:=commInfo; commPortP:=@commPortStr[1];
  VerifyCommInfo( myCommInfo );
  fillchar( commPortStr, sizeof( ShortString ), 0 );
  with myCommInfo do
  begin
    commPortStr:='COM'+IntToStr( commPort );
    hCommFile:=CreateFile(commPortP, GENERIC_READ+GENERIC_WRITE,
              0, nil, OPEN_EXISTING, FILE_FLAG_OVERLAPPED, 0)
    // comm. handle is valid?
    if ( hCommFile=INVALID_HANDLE_VALUE ) then
    begin
      s:='Error on open '+commPortStr;
      Echo( s+', error code = '+IntToStr( GetLastError ) )
      ShowMessage( s ); exit;
    end;
    if ( GetFileType( hCommFile )<>FILE_TYPE_CHAR ) ther
    begin
      s:='Handle is not COM port.'; Echo( s )
      ShowMessage( s ); exit;
    end;
    // get all comm. properties
```

```
    GetCommState( hCommFile, commDcb );
    GetCommProperties( hCommFile, commProp );
    GetCommMask( hCommFile, evtMask );
    GetCommTimeOuts( hCommFile, commTimeOuts );
    // config necessary comm. properties
    SetupComm( hCommFile, rxBuffer, txBuffer );
    commTimeOuts.ReadIntervalTimeout:= MAXDWORD; // set timeout
    commTimeOuts.ReadTotalTimeoutMultiplier:=0;
    commTimeOuts.ReadTotalTimeoutConstant:=0;
    SetCommTimeouts( hCommFile, commTimeOuts );
    Echo( 'evtmask = '+IntToStr( evtMask ) );
    evtMask:=EV_RXCHAR;
    SetCommMask( hCommFile, evtMask );
    commDcb.baudRate:=baudRate;
    commDcb.parity:=Parity;
    commDcb.stopBits:=StopBits;
    commDcb.byteSize:=byteSize;
    SetCommState( hCommFile, commDcb );
    errorFlg:=false;
    // Display comm. properties
    Echo( commPortStr+' open successfully.' );
    DispComDCB( commDcb );
    DispComProperties( commProp );
    DispComTimeouts( commTimeOuts );
  end;
end;

procedure Tcomm32.CloseComm32;
begin
  if ( myCommInfo.hCommFile>0 ) then
    CloseHandle( myCommInfo.hCommFile );
end;

function Tcomm32.SendData( dataP : pointer;
                    dataLen : dword ) : intege
var byteSend : dword;
begin
  result:=-1; byteSend:=0;
  if ( not WriteFile( myCommInfo.hCommFile, dataP^, dataLen,
    byteSend, @overlapWrite ) ) then exit;
  result:=byteSend;
end;

function Tcomm32.ReadData( dataP : pointer;
                     dataLen : dword ) : intege
var byteRead : dword; overlapReadP : POverlapped;
    nNumberOfBytesRead : dword; ret : boolean;
begin
```

```
  byteRead:=0; overlapReadP:=@overlapRead; nNumberOfBytesRead:=0;
  if ( not ReadFile( myCommInfo.hCommFile, dataP^, dataLen,
    byteRead, overlapReadP ) ) then exit;
  ret:=GetOverlappedResult( myCommInfo.hCommFile, overlapReadP^,
        nNumberOfBytesRead, true );
  result:=byteRead;
end;

end.
```

```
unit tee_utscomm32;
//โปรแกรมส่งข้อมูลทางพอร์ตอนุกรม
{$H-}
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, uComm32, StdCtrls, Buttons, ExtCtrls, Gauges;

type
  TFtsCom32 = class(TForm)
    SpeedButton1: TSpeedButton;
    RadioGroup1: TRadioGroup;
    Edit1: TEdit;
    SpeedButton8: TSpeedButton;
    Edit2: TEdit;
    Label1: TLabel;
    SpeedButton2: TSpeedButton;
    Edit3: TEdit;
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton8Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
  private
    { Private declarations }
    commInfo:TcommInfo;
  public
    { Public declarations }
      commProc : Tcomm32;
      sp:shortstring;
    function binarytochar(st:shortString):integer;
    procedure send(data:shortstring);

  end;

var
  FtsCom32: TFtsCom32;

implementation

uses uComProc;

{$R *.DFM}

//---------------------------------------------------------

procedure TFtsCom32.Send(data:shortstring);
var t,m:shortString;  p:pchar;    i,a,b:integer;
begin
```

```
 i:=binarytochar(data);
 t:=chr(i+48);
 p:=@t[1];
 a:=commProc.SendData(p,length(t));
 Edit2.Text:=p[0]+'--'+p[1];
 Edit3.Text:=inttostr(a);
 sleep(100);
end;

procedure TFtsCom32.SpeedButton1Click(Sender: TObject);
begin
  with commInfo do
  begin
   commPort:=RadioGroup1.ItemIndex+1;
   rxBuffer:=1024;
   txBuffer:=1024;
   baudRate:=CBR_9600;
   byteSize:=8;
   parity:=NOPARITY;
   stopBits:=ONESTOPBIT;
  end;
  commProc:=Tcomm32.Create(commInfo);
end;

function TFtsCom32.binarytochar(st:shortString):integer;
var i:integer;
begin
 i:=(strtoint(st[1])*8)+(strtoint(st[2])*4)+(strtoint(st[3])*2)+(strtoint(
st[4])*1);
 result:=i;
end;


procedure TFtsCom32.SpeedButton8Click(Sender: TObject);
begin
 //commProc:=Tcomm32.Create(commInfo);
 commProc.CloseComm32;
end;

procedure TFtsCom32.SpeedButton2Click(Sender: TObject);
begin
 send(Edit1.Text);
end;

end.
```

```
;โปรแกรมรับข้อมูลจากพอร์ตอนุกรม (ในคอนโทรลเลอร์)
 org 000h
        mov p2,#00h
        call int_ser
  start: call rx_b
       mov p2,a
       ajmp start
int_ser: mov a,#0fdh
        mov th1,a
        mov tl1,a
        mov tmod,#00100000b
        clr es
        clr et1
        setb tr1
        mov scon,#01010000b
        ret

rx_b:   push ie
       clr es
       jnb ri,$
       clr r
       mov a,sbuf
       jnb ti,$

       pop ie
       ret

      end
```

```
//โปรแกรมตีกรอบออปเจ็กต์
#include<vcl.h>
#include<iostream.h>
class object
{
 private:
  int R,G,B;
  int X1,Y1,X2,Y2;
  int range;
 public:
  object(void) {R=0;G=0;B=0;}
  init(int rr,int gg,int bb);
  int getR(void);
  int getG(void);
  int getB(void);
  bool position(Graphics::TBitmap *bitmap);
  int getX1(void);
  int getY1(void);
  int getX2(void);
  int getY2(void);
  setrange(int r);
  int getrange(void);
};

object::init(int rr,int gg,int bb)
{
  R=rr;
  G=gg;
  B=bb;
  range=20;
}
object::setrange(int r)
{
 range=r;
}
int object::getrange(void)
{
 return(range);
}
int object::getR(void)
{
 //read red color
 return(R);
}
int object::getG(void)
{
 //read green color
 return(G);
```

```
}
int object::getB(void)
{
 //read blue color
 return(B);
}
bool object::position(Graphics::TBitmap *bitmap)
{
 //find X1,Y1,X2,Y2
 int x,y,r,g,b,add,num=0;
 bool ok=false;
 while((num<=3)&&(!ok))
 {
  if(num==0) {y=0;add=10;}
  else if(num==1) {y=5;add=10;}
  else if(num==2) {y=0;add=2;}
  else if(num==3) {y=1;add=2;}
  while((y<=240)&&(!ok))
  {
   if(num==0) {x=0;}
   else if(num==1) {x=5;}
   else if(num==2) {x=0;}
   else if(num==3) {x=1;}
   while((x<=320)&&(!ok))
   {

    r=GetRValue(bitmap->Canvas->Pixels[x][y]);
    if((r>(R-range))&&(r<(R+range)))
    {
     g=GetGValue(bitmap->Canvas->Pixels[x][y]);
     if((g>(G-range))&&(g<(G+range)))
     {
      b=GetBValue(bitmap->Canvas->Pixels[x][y]);
      if((b>(B-range))&&(b<(B+range))) {ok=true;}
     }
    }
   x=x+add;
   }
  y=y+add;
  }
num++;
}
//////////////////////////////////////////////////
int xx,yy;
x=x-add;
y=y-add;
bool ok1;
//find x1//////////////////////////////////////
```

```
 if(ok)
{
 ok1=true;
 xx=x;
 while((xx>=0)&&(ok1))
 {
  yy=y;
  ok1=false;
  while((yy<=240)&&(!ok1))
  {
   r=GetRValue(bitmap->Canvas->Pixels[xx][yy]);
   if((r>(R-range))&&(r<(R+range)))
   {
    g=GetGValue(bitmap->Canvas->Pixels[xx][yy]);
    if((g>(G-range))&&(g<(G+range)))
    {
     b=GetBValue(bitmap->Canvas->Pixels[xx][yy]);
     if((b>(B-range))&&(b<(B+range))) {ok1=true;}
    }
   }
   yy++;
  }
  xx--;
 }
X1=xx++;
}
//find y1!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
if(ok)
{
 ok1=true;
 yy=y;
 while((yy<=240)&&(ok1))
 {
  xx=x;
  ok1=false;
  while((xx<=320)&&(!ok1))
  {
   r=GetRValue(bitmap->Canvas->Pixels[xx][yy]);
   if((r>(R-range))&&(r<(R+range)))
   {
    g=GetGValue(bitmap->Canvas->Pixels[xx][yy]);
    if((g>(G-range))&&(g<(G+range)))
    {
     b=GetBValue(bitmap->Canvas->Pixels[xx][yy]);
     if((b>(B-range))&&(b<(B+range))) {ok1=true;}

    }
```

```
    }
    xx++;
   }
   yy--;
  }
 Y1=yy++;
 }
//find x2////////////////////////////////////////////
 if(ok)
 {
  ok1=true;
  xx=x;
  while((xx<=320)&&(ok1))
  {
   yy=y;
   ok1=false;
   while((yy<=240)&&(!ok1))
   {
    r=GetRValue(bitmap->Canvas->Pixels[xx][yy]);
    if((r>(R-range))&&(r<(R+range)))
    {
     g=GetGValue(bitmap->Canvas->Pixels[xx][yy]);
     if((g>(G-range))&&(g<(G+range)))
     {
      b=GetBValue(bitmap->Canvas->Pixels[xx][yy]);
      if((b>(B-range))&&(b<(B+range))) {ok1=true;}
     }
    }
   yy++;
   }
  xx++;
  }
X2=xx--;
}
//find y2////////////////////////////////////////////
if(ok)
{
 ok1=true;
 yy=y;
 while((yy<=240)&&(ok1))
 {
  xx=x;
  ok1=false;
  while((xx<=X2)&&(!ok1))
  {
    r=GetRValue(bitmap->Canvas->Pixels[xx][yy]);
 if((r>(R-range))&&(r<(R+range)))
 {
```

```
    g=GetGValue(bitmap->Canvas->Pixels[xx][yy]);
    if((g>(G-range))&&(g<(G+range)))
    {
     b=GetBValue(bitmap->Canvas->Pixels[xx][yy]);
     if((b>(B-range))&&(b<(B+range))) {ok1=true;}
    }
   }
   xx++;
  }
  yy++;
  }
 Y2=yy--;
 }
 return(ok);
}
int object::getX1(void)
{
 //read X1
 return(X1);
}
int object::getY1(void)
{
 //read Y1
 return(Y1);
}
int object::getX2(void)
{
 //read X2
 return(X2);
}
int object::getY2(void)
{
 //read Y2
 return(Y2);
}
```

```
//โปรแกรมหลักของโปรแกรมจำสีและตีกรอบออปเจ็กต์
//---------------------------------------------------------------------------
---

#include <vcl.h>
#include "File1.cpp"
#pragma hdrstop

#include "tee2.h"
//---------------------------------------------------------------------------
---
#pragma package(smart_init)
#pragma link "IEVect"
#pragma link "ImageEnView"
#pragma link "VideoCap"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
---
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------
---

void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
 ImageEnVideoView1->ShowVideo=SpeedButton1->Down;
 if(SpeedButton1->Down)
 {
  ImageEnVideoView1->OnVideoFrame=ImageEnVideoView1VideoFrame;

 }
}
//---------------------------------------------------------------------------
---
void __fastcall TForm1::ImageEnVideoView1VideoFrame(TObject *Sender,
      Graphics::TBitmap *Bitmap)
{
 ImageEnView1->Assign(Bitmap);
 ImageEnView2->Clear();
 int x,y,r,g,b,range1=o1.getrange(),range2=o2.getrange();
 char str[20];
 bool t1=o1.position(ImageEnView1->Bitmap);
 bool t2=o2.position(ImageEnView1->Bitmap);
 if(t1)
```

```
{
 ImageEnView1->Bitmap->Canvas->MoveTo(o1.getX1(),o1.getY1());
 ImageEnView1->Bitmap->Canvas->LineTo(o1.getX1(),o1.getY2());
 ImageEnView1->Bitmap->Canvas->MoveTo(o1.getX1(),o1.getY1());
 ImageEnView1->Bitmap->Canvas->LineTo(o1.getX2(),o1.getY1());
 ImageEnView1->Bitmap->Canvas->MoveTo(o1.getX2(),o1.getY2());
 ImageEnView1->Bitmap->Canvas->LineTo(o1.getX1(),o1.getY2());
 ImageEnView1->Bitmap->Canvas->MoveTo(o1.getX2(),o1.getY2());
 ImageEnView1->Bitmap->Canvas->LineTo(o1.getX2(),o1.getY1());
}
if(t2)
{
 ImageEnView1->Bitmap->Canvas->MoveTo(o2.getX1(),o2.getY1());
 ImageEnView1->Bitmap->Canvas->LineTo(o2.getX1(),o2.getY2());
 ImageEnView1->Bitmap->Canvas->MoveTo(o2.getX1(),o2.getY1());
 ImageEnView1->Bitmap->Canvas->LineTo(o2.getX2(),o2.getY1());
 ImageEnView1->Bitmap->Canvas->MoveTo(o2.getX2(),o2.getY2());
 ImageEnView1->Bitmap->Canvas->LineTo(o2.getX1(),o2.getY2());
 ImageEnView1->Bitmap->Canvas->MoveTo(o2.getX2(),o2.getY2());
 ImageEnView1->Bitmap->Canvas->LineTo(o2.getX2(),o2.getY1());
}
/* if((t2)&&(SpeedButton6->Down))
{
 sprintf(str,"(%d,%d),(%d,%d)",o2.getX1(),o2.getY1(),o2.getX2(),o2.getY2(
));
 Panel5->Caption=str;
 for(y=o2.getY1();y<=o2.getY2();y++)
 {
  for(x=o2.getX1();x<=o2.getX2();x++)
  {
   r=GetRValue(ImageEnView1->Bitmap->Canvas->Pixels[x][y]);
   if((r>(R-range2))&&(r<(R+range2)))
   {
    g=GetGValue(ImageEnView1->Bitmap->Canvas->Pixels[x][y]);
    if((g>(G-range2))&&(g<(G+range2)))
    {
     b=GetBValue(ImageEnView1->Bitmap->Canvas->Pixels[x][y]);
     if((b>(B-range2))&&(b<(B+range2)))
     {
      ImageEnView2->Bitmap->Canvas->Pixels[x][y]=RGB(r,g,b);
     }
    }
   }
  }
 }
}*/

ImageEnView2->Update();
```

```
}
//---------------------------------------------------------------------
---
TForm1::findRGB(Graphics::TBitmap *bitmap,int x1,int y1,int x2,int y2)
{

 int x,y;
 int r,g,b,rmax=0,rmin=255,gmax=0,gmin=255,bmax=0,bmin=255;
 int rum=0,rom=0,gum=0,gom=0,bum=0,bom=0;
 if(y1>y2)
 {
  y=y1;
  y1=y2;
  y2=y;
 }
 if(x1>x2)
 {
  x=x1;
  x1=x2;
  x2=x;
 }
 for(y=y1;y<=y2;y++)
  {
   for(x=x1;x<=x2;x++)
   {
    r=GetRValue(bitmap->Canvas->Pixels[x][y]);
    g=GetGValue(bitmap->Canvas->Pixels[x][y]);
    b=GetBValue(bitmap->Canvas->Pixels[x][y]);
    if(r>rmax) rmax=r;
    if(r<rmin) rmin=r;
    if(g>gmax) gmax=g;
    if(g<gmin) gmin=g;
    if(b>bmax) bmax=b;
    if(b<bmin) bmin=b;
   }
  }
////////////////////////////////////////////////////////////////////////
 for(y=y1;y<=y2;y++)
 {
  for(x=x1;x<=x2;x++)
  {
   r=GetRValue(bitmap->Canvas->Pixels[x][y]);
   g=GetGValue(bitmap->Canvas->Pixels[x][y]);
   b=GetBValue(bitmap->Canvas->Pixels[x][y]);
   if(r>((rmax+rmin)/2)) rom++;
   if(r<=((rmax+rmin)/2)) rum++;
   if(g>((gmax+gmin)/2)) gom++;
   if(g<=((gmax+gmin)/2)) gum++;
```

```
    if(b>((bmax+bmin)/2)) bom++;
    if(b<=((bmax+bmin)/2)) bum++;
   }
  }
  int qr1,qr3,qg1,qg3,qb1,qb3;
  qr1=((rmax-rmin)/4)+rmin;
  qr3=(3*((rmax-rmin)/4))+rmin;
  qg1=((gmax-gmin)/4)+gmin;
  qg3=(3*((gmax-gmin)/4))+gmin;
  qb1=((bmax-bmin)/4)+bmin;
  qb3=(3*((bmax-bmin)/4))+bmin;
  R=(int)(((qr3*rom)+(qr1*rum))/(rum+rom));
  G=(int)(((qg3*gom)+(qg1*gum))/(gum+gom));
  B=(int)(((qb3*bom)+(qb1*bum))/(bum+bom));
}
//---------------------------------------------------------------------
---
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
 if(SpeedButton2->Down)
  ImageEnVideoView1->OnVideoFrame=0;
 else if(SpeedButton1->Down)
 {
  ImageEnVideoView1->OnVideoFrame=ImageEnVideoView1VideoFrame;
  ImageEnView1->DeSelect();
 }
}
//---------------------------------------------------------------------
---

void __fastcall TForm1::ImageEnView1MouseUp(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
 if(SpeedButton2->Down)
 {
  end=Point(X,Y);
  ImageEnView1->Select(start.x,start.y,end.x,end.y,iesbBitmap);
 }
}
//---------------------------------------------------------------------
---

void __fastcall TForm1::ImageEnView1MouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
 if(SpeedButton2->Down)
 {
```

```
   start=Point(X,Y);
   ImageEnView1->DeSelect();
  }
 }
 //---------------------------------------------------------------------
 ---


 void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
 {
  if(ImageEnView1->Selected)
  {
   findRGB(ImageEnView1->Bitmap,ImageEnView1->SelX1,ImageEnView1->SelY1,Ima
 geEnView1->SelX2,ImageEnView1->SelY2);
   Panel1->Color=RGB(R,G,B);
  }
 }
 //---------------------------------------------------------------------
 ---

 void __fastcall TForm1::Button1Click(TObject *Sender)
 {
  o1.init(R,G,B);
  Panel2->Color=RGB(R,G,B);
 }
 //---------------------------------------------------------------------
 ---

 void __fastcall TForm1::Button2Click(TObject *Sender)
 {
  o2.init(R,G,B);
  Panel3->Color=RGB(R,G,B);
 }
 //---------------------------------------------------------------------
 ---



 void __fastcall TForm1::SpeedButton4Click(TObject *Sender)
 {
  if(ImageEnView1->Selected&&SpeedButton4->Down)
  {
   ImageEnView1->ZoomSelection();
  }
  else if(!SpeedButton4->Down)
  {
   ImageEnView1->Fit();
  }
```

```
}
//---------------------------------------------------------------------------
---



void __fastcall TForm1::ImageEnView1MouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
 char str[10];
 sprintf(str,"(%d,%d)",X,Y);
 Panel5->Caption=str;
}
//---------------------------------------------------------------------------
---
```

```
// เฮดเดอร์ไฟล์ของโปรแกรมจำสีและหาตำแหน่งภาพ
//---------------------------------------------------------------------------
---

#ifndef tee2H
#define tee2H
//---------------------------------------------------------------------------
---
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "IEVect.hpp"
#include "ImageEnView.hpp"
#include "VideoCap.hpp"
#include <Buttons.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
---
class TForm1 : public TForm
{
__published:    // IDE-managed Components
        TSpeedButton *SpeedButton1;
        TImageEnVideoView *ImageEnVideoView1;
        TImageEnView *ImageEnView1;
        TPanel *Panel1;
        TSpeedButton *SpeedButton2;
        TSpeedButton *SpeedButton3;
        TPanel *Panel2;
        TPanel *Panel3;
        TLabel *Label1;
        TLabel *Label2;
        TButton *Button1;
        TButton *Button2;
        TSpeedButton *SpeedButton4;
        TImageEnView *ImageEnView2;
        TSpeedButton *SpeedButton5;
        TSpeedButton *SpeedButton6;
        TPanel *Panel4;
        TPanel *Panel5;
        void __fastcall SpeedButton1Click(TObject *Sender);
        void __fastcall ImageEnVideoView1VideoFrame(TObject *Sender
          Graphics::TBitmap *Bitmap);
        void __fastcall SpeedButton2Click(TObject *Sender);
        void __fastcall ImageEnView1MouseUp(TObject *Sender
          TMouseButton Button, TShiftState Shift, int X, int Y);
        void __fastcall ImageEnView1MouseDown(TObject *Sender
          TMouseButton Button, TShiftState Shift, int X, int Y);
```

```
        void __fastcall SpeedButton3Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall SpeedButton4Click(TObject *Sender);
        void __fastcall ImageEnView1MouseMove(TObject *Sender,
          TShiftState Shift, int X, int Y)
private:        // User declarations
public:         // User declarations
        __fastcall TForm1(TComponent* Owner);
        findRGB(Graphics::TBitmap *bitmap,int x1,int y1,int x2,int y2)
        int R,G,B;
        int X1,Y1,X2,Y2;
        TPoint start,end;
        object o1,o2;
};
//---------------------------------------------------------------------------
---
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
---
#endif
```