ป้ายโฆษณาและประชาสัมพันธ์ โดยชุดแสดงผล LED 1 Column แบบหมุน

ROTATABLE 1 COLUMN LED DISPLAYED

โดย

นายกฤษณะ สันรัตน์

นายชัยวัฒน์ อินทะโส

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ป้ายโฆษณาและประชาสัมพันธ์โดยชุดแสดงผล LED 1 Column แบบหมุน

ROTATABLE 1 COLUMN LED DISPLAYED

โดย

นายกฤษณะ สันรัตน์ เลขประจำตัว 42015164

นายชัยวัฒน์ อินทะโส เลขประจำตัว 42015171

อาจารย์ที่ปรึกษา

อ.อิทธิภูมิ บุญพิคำ

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ป้ายโฆษณาและประชาสัมพันธ์ โดยชุดแสดงผล LED 1 Column แบบหมุน

ผู้จัดทำ

1.นายกฤษณะ สันรัตน์

2.นายชัยวัฒน์ อินทะโส

.....อาจารย์ที่ปรึกษา

(อ.อิทธิภูมิ บุญพิคำ)

ป้ายโฆษณาและประชาสัมพันธ์ โดยชุดแสดงผล LED 1 Column แบบหมุน

กฤษณะ สันรัตน์
ชัยวัฒน์ อินทะโส
ดร.อิทธิภูมิ บุญพิคำ อาจารย์ที่ปรึกษา

บทคัดย่อ

ป้ายโฆษณาและประชาสัมพันธ์ โดยภาคแสดงผล LED 1 Column แบบหมุน โครงงานนี้เสนอป้ายโฆษณา, ประชาสัมพันธ์, และเวลาโดยภาคแสดงผล LED ที่วิ่งในแนวตั้ง และหมุนโดยมอเตอร์เพื่อให้การแสดงข้อมูลเป็นไปอย่างต่อเนื่อง ข้อมูลที่ต้องการแสดงจะถูกส่งผ่านแป้นKeyboard ของ PC ไปยังวงจรภาคส่งแบบ Infrared สู่ภาครับซึ่งอยู่ในหน่วยเดียวกับภาคแสดงผลMicrocontroller MCS-51 จะแปลงข้อมูลที่เข้ามา และส่งออก ไปยังภาค แสดงผลผ่านทาง LED 1 column ที่ควบคุมความเร็วโดย DC Motor เพื่อให้การแสดงผล เป็น ไปตามต้องการ

ROTATABLE 1 COLUMN LED DISPLAYED

Kritsana Sunrut

Chaiwat Intaso

Dr. Ittibhoom Boonpikum Advisor

Abstract

This project pres ents a Vertical moving LED's Displayed sign in English for advertising or information purpose. Displayed information are entered by using a PC keyboard and sent to an IF transmitter in the digital signals form. The receiver unit then passes the signal to the Microcontroller MCS-51 that is the major displayed control unit. The Microcontroller will control both "on/off" status of a column LED's and the motor speed to make displayed information readable.

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 กล่าวนำ

เนื่องจากในปัจจุบันการโฆษณาประชาสัมพันธ์มีบทบาทอย่างมากในองค์กรต่างๆในสังคม เครื่องมือต่างๆถูกนำมาใช้ในการเผยแพร่ข่าวสารข้อมูลออกสู่สาธารณะชน อาทิเช่น โทรทัศน์ วิทยุ ป้ายโฆษณา หนังสือพิมพ์ และที่เห็นอยู่บ่อยครั้งก็คือแผงหลอด LED ซึ่งมักถูกติดตั้งในที่สาธารณะทั่วไปไม่ว่า ห้างสรรพสินค้า สถานที่ราชการ สี่แยกไฟแดง ริมทางเดินเท้า เป็นสื่อที่สามารถเข้าถึงประชาชน และยังเป็นสื่อทันที่ต่อเหตุการณ์ในระดับหนึ่งเมื่อเทียบกับป้ายโฆษณาหรือแผ่นพับ เพราะสามมารถเปลี่ยนแปลงเพิ่มเติมข้อมูลข่าวสารใหม่ลงไปได้ตามต้องการ แต่ยังมือข้อเสียอยู่หลายประการ ที่เห็นได้ชัดคือ ข้อจำกัดในการมองเห็น แผงหลอดLEDสามารถมองเห็นได้จากทิศทางเดียว คือจากทางด้านหน้าเท่านั้น ผู้ที่อยู่ด้านข้างหรือด้านหลังจะไม่สามารถมองเห็นข้อความที่แสดงผ่านแผงหลอด LED ได้ และข้อเสียอีกประการหนึ่งก็คือ มีต้นทุนในการสร้างค่อนข้างสูง แผงหลอด LED ที่ดีควรจะมีความกว้างเพียงพอที่จะแสดงข้อมูลออกมาอย่างต่อเนื่องครบถ้วนสมบูรณ์ เพื่อให้ผู้อ่านสามารถอ่านข้อความได้ง่าย จึงทำให้แผงหลอด LED ต้องมีหลอด LED เป็นจำนวนมากตามไปด้วยทำให้มีต้นทุนในการสร้างค่อนข้างสูง และผลเสียอีกประการหนึ่งที่หลีกเลี่ยงไม่ได้จากจำนวนหลอด LED ที่มาก็คือต้องใช้กระแสไฟมากตามไปด้วย ผู้จัดทำคำนึงถึงข้อเสียเหล่านี้จึงได้จัดทำโครงงานปริญญานิพนธ์นี้ขึ้น

ดังนั้นผู้จัดทำโครงงานปริญญานิพนธ์นี้ จึงได้ศึกษาและสร้างชุดแสดงผลที่เป็นแบบแกนหมุนและใช้ LED ในการแสดงผลเพียง 8 ดวง เพื่อลดต้นทุนในการสร้างและทำให้สิ้นเปลืองพลังงานไฟฟ้าน้อยลง ทั้งนี้ยังลดข้อจำกัดในการมองเห็น โดยจะสามารถมองเห็นได้รอบด้าน

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. เพื่อเสนอการแสดงผลของข้อความแบบหมุนแกนทำให้มุมมอง และพื้นที่กว้างมากขึ้น
2. เพื่อสร้างชุดแสดงผลโดยใช้หลอด LED เพียง 8 หลอดในการแสดงผลเป็นตัวหนังสือภาษาอังกฤษ
3. เพื่อศึกษาซอฟท์แวร์ภาษา Visual Basic เพื่อใช้ควบคุมการทำงานของระบบและแสดงผล
4. เพื่อศึกษาซอฟท์แวร์ของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เพื่อใช้ในการควบคุมหลอด LED ในการแสดงผล
5. เพื่อศึกษาการรับส่งข้อมูลแบบไร้สายโดยใช้แสงอินฟราเรด
6. เพื่อสร้างชุดรับส่งข้อมูลโดยใช้แสงอินฟราเรด

7. ศึกษาการสั่งงานผ่านระบบอินเตอร์เน็ต

1.3 ขอบเขตปริญญานิพนธ์

1. โครงงานปริญญานิพนธ์นี้ใช้ LED 8 หลอด โดยเลือกใช้ชนิด 3 สี

2. ขนาดของแกนหมุน ยาว 30 เซนติเมตร ขนาดบล็อกตัวอักษร 8 × 6 จุด

3. แสดงผลเป็นตัวอักขระภาษาอังกฤษ สามารถแสดงผลเป็นรูปสามเหลี่ยม,สี่เหลี่ยมและบอกเวลาเป็นตัวเลขได้ โดยเคลื่อนที่จากขวาไปซ้ายหรือหยุดนิ่ง

4. ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นตัวควบคุมการแสดงผลและใช้คอมพิวเตอร์เป็นตัวส่งข้อมูล

5. ใช้อินฟราเรดในการรับ-ส่งข้อมูลระยะห่างไม่เกิน 10 เมตร

6. ควบคุมได้ 2 ลักษณะคือเลือกใช้การแสดงผลจากข้อมูลที่ได้ทำการโปรแกรมไว้ไม่เกิน 30 ตัวอักษร หรือเลือกใช้การแสดงผลจากข้อมูลที่ส่งจากคอมพิวเตอร์ละครั้งละไม่เกิน 100 ตัวอักษร

7. การแสดงผลจะสามารถแสดงได้เพียงสีเดียวต่อหนึ่งครั้ง

1.4 ประโยชน์ที่คาดว่าจะได้รับจากปริญญานิพนธ์

1. สามารถนำโปรแกรมภาษา Visual Basic ไปประยุกต์ใช้งานได้

2. สามารถนำ ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ไปประยุกต์ใช้งานได้

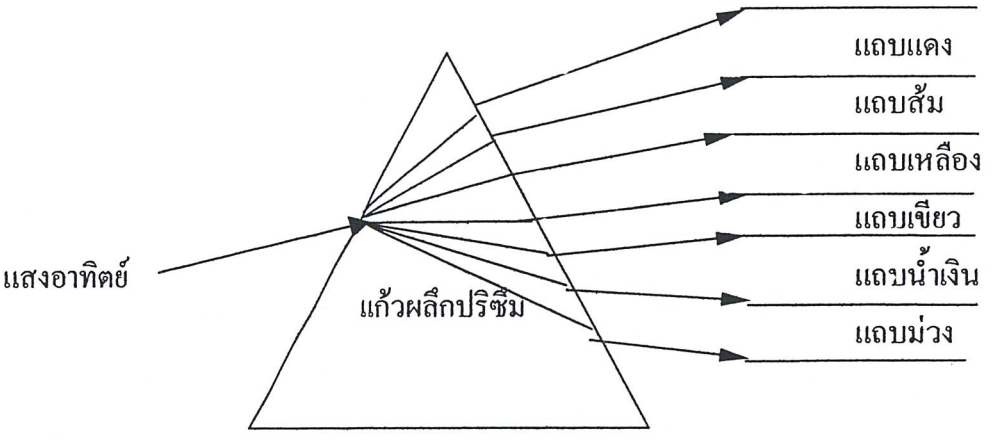
3. สามารถนำหลักการสื่อสารข้อมูลไปใช้ในระบบการรับส่งข้อมูลแบบอินฟราเรดและนำไปประยุกต์ใช้งานได้

4. ได้เครื่องต้นแบบหลอดไฟแสดงผลแบบหมุนควบคุมผ่านระบบอินเตอร์เน็ต และพัฒนาให้ดียิ่งขึ้น

บทที่2
ทฤษฎีที่เกี่ยวข้อง

2.1ทฤษฎีของแสง

ในปี ค.ศ. 1666 เซอร์ไอเซกนิวตัน (Sir Isaac Newton) ได้ค้นพบว่าเมื่อลำแสงอาทิตย์ส่องผ่านแก้วปริซึม (Glass Prism) แสงที่ทะลุผ่านแท่งแก้วไปยังอีกด้านหนึ่งจะกระจายออกเป็นแสงสีต่างๆ ดังรูปที่ 2.1 แสดงว่าแสงอาทิตย์ที่เรามองเห็นเป็นสีขาวหรือสีใสนั้น แท้ที่จริงแล้วประกอบด้วยแสงสีต่าง ๆ หลายสี แสงแต่ละสีที่เห็นนั้นต่างก็มีความถี่หรือความยาวคลื่นใกล้เคียงกัน



รูปที่ 2.1 แสดงการแยกแสงอาทิตย์ออกเป็นสเปกตรัมของแสงโดยใช้แก้วผลึกปริซึม

ขอบเขตของความยาวคลื่นของแสงสีต่างๆ มีค่าประมาณดังนี้

สีม่วง	ประมาณ	0.390-0.455	ไมโครเมตร
สีน้ำเงิน	ประมาณ	0.455-0.492	ไมโครเมตร
สีเขียว	ประมาณ	0.492-0.577	ไมโครเมตร
สีเหลือง	ประมาณ	0.577-0.597	ไมโครเมตร
สีส้ม	ประมาณ	0.597-0.622	ไมโครเมตร
สีแดง	ประมาณ	0.622-0.770	ไมโครเมตร

จะเห็นได้ว่าแสงสีม่วง(Violet) มีความยาวคลื่นประมาณ 0.400 ไมโครเมตร สีแดง (Red) มีความยาวคลื่นประมาณ 0.700 ไมโครเมตร แสงในย่านความยาวคลื่น 0.390-0.770 ไมโครเมตร นี้เท่านั้นที่ตามนุษย์มองเห็นได้แสงที่มีความยาวคลื่นอยู่นอกย่านที่ตาเมองเห็นนั้น

ได้แก่แสงอัลตราไวโอเลต (Ultraviolet) หมายถึง แสงที่มีความถี่สูงเลยแสงสีม่วงขึ้นไป และ แสงอินฟราเรด (Infrared) หมายถึงแสงที่มีความถี่ต่ำกว่าแสงสีแดงลงมา

ขอบเขตที่ต่อเชื่อมกันระหว่างสีไม่สามารถชี้เจาะจงลงไปได้ แน่นอนว่าแต่ละสีจะสิ้นสุดลง ณ ที่ใด เพราะต่างก็ค่อยๆ จางลงแล้วเข้ามาเชื่อมติดต่อกันและกัน

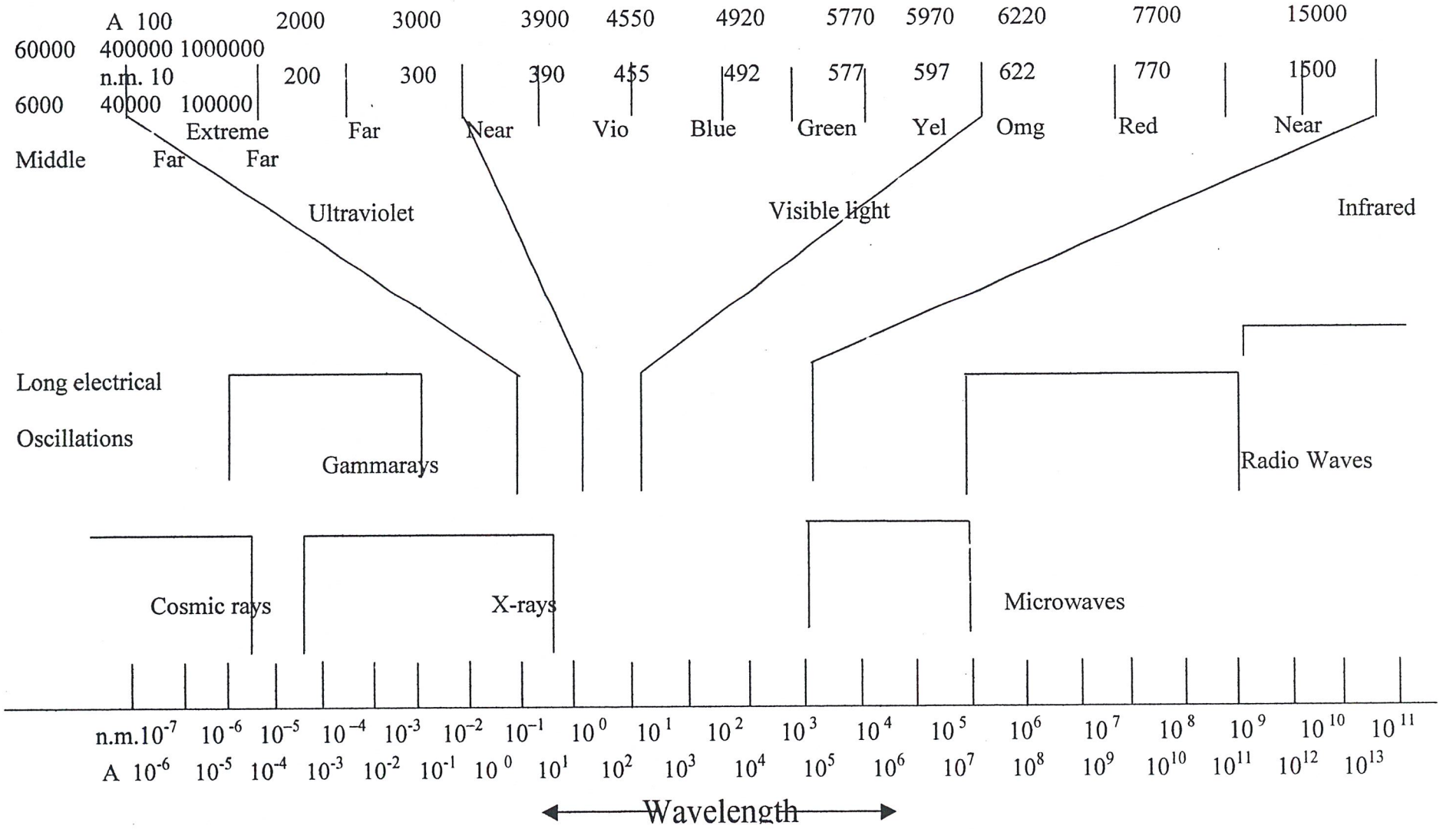
เนื่องจากแสงนั้นมีความถี่ดังนั้นแสงก็เป็นคลื่นแม่เหล็กไฟฟ้าเช่นเดียวกับคลื่นวิทยุ ต่างกันตรงที่ความถี่ และความยาวคลื่น ซึ่งคลื่นแสงอยู่ในย่านความถี่ 4×10^8 เมกะเฮิรตซ์ ซึ่งสูงกว่าคลื่นวิทยุ พิจารณารูปที่ 2.2 จะเห็นว่าย่านคลื่นความถี่คลื่นวิทยุ อยู่ในย่านที่ต่างกันกับย่านความถี่คลื่นแสง แต่มีคุณสมบัติเป็นคลื่นแม่เหล็กไฟฟ้าเหมือนกันแสงมีความเร็ว186,000 ไมล์ต่อวินาทีหรือ 300,000 กิโลเมตรต่อวินาทีและมีความถี่เป็นสัดส่วนผกผันกับความยาวคลื่น สามารถเขียนให้อยู่ในรูปของสมการทางคณิตศาสตร์ได้ดังนี้

$$c = \lambda / f$$

เมื่อ C = ความเร็วคลื่นแสง (เมตร/วินาที)

λ = ความยาวคลื่น (เมตร)

f = ความถี่ (เฮิรตซ์)



รปที่ 2.2 แสดงสเปกตรัมของคลื่นแม่เหล็กไฟฟ้า

2.1.1 แสงอินฟราเรด

แสงอินฟราเรด เป็นแสงที่ไม่สามารถมองเห็น ผู้ค้นพบเป็นนักดาราศาสตร์ชาวอังกฤษชื่อ เซอร์ วิลเลี่ยม เฮอร์เชล (William Herchel Ritter) ในปีค.ศ. 1770 ได้ทำการทดลองเกี่ยวกับเรื่องแถบสีต่างๆ ในสเปกตรัมของแสงแดดว่า แต่ละแถบสีมีคุณสมบัติแตกต่างกันอย่างไรนั้นเฮอร์เชลใช้เทอร์โมมิเตอร์วัดอุณหภูมิของแสงแต่ละแสง โดยเริ่มวัดจากแสงสีม่วงไป ครั้นเลื่อนเทอร์โมมิเตอร์มาถึงแถบสีแดงปรากฏว่าลำปรอทพุ่งขึ้นสูงทันที หลังจากวัดความร้อนในแถบสีแดงจนพอแก่ความต้องการแล้วบังเอิญ เฮอร์เชลได้ทำเทอร์โมมิเตอร์เขยิบออกไปจากแถบ สีแดงอีกเล็กน้อยปรากฏว่าลำปรอทพุ่งขึ้นสูงยิ่งกว่าเดิม แสดงว่าใต้แถบสีแดงต้องมีแสงอีกชนิดหนึ่งที่มองไม่เห็น และแสงชนิดนี้ในปัจจุบัน ทราบว่าเป็นแสงที่ให้ความร้อน

เฮอร์เชล ทำการทดลองแสงที่พบใหม่นี้อยู่หลายครั้ง ก็สรุปผลการทดลองออกว่าแสงที่มองไม่เห็นที่ค้นพบได้นี้ มีคุณสมบัติคล้ายกับแสงอาทิตย์แม้อัตราความเร็วก็เท่ากัน เขาจึงใช้ชื่อเรียกแสงนี้ว่า แสงอินฟราเรด (Infrared) ซึ่งคำว่า Infra เป็นภาษาลาติน แปลว่า "ใต้" คำว่า Red แปลว่า "แดง" ดังนั้น แสงอินฟราเรดจึงเป็นแสงที่อยู่แถบสีแดงในสเปกตรัมของแสงอาทิตย์

ถึงแม้ว่า แสงอินฟราเรดจะเป็นแสงที่ให้ความร้อน แต่เฉพาะตัวแสงอินฟราเรดเองไม่มีความร้อนแต่อย่างใด แสงอินฟราเรดจะไม่ให้ความร้อนใดๆ ถ้าหากแสงอินฟราเรดไม่ส่องไปกระทบเข้ากับวัตถุสิ่งหนึ่งสิ่งใด และจำนวนความร้อนที่ใช้จะมากหรือน้อย ขึ้นอยู่กับกลไกทางเคมีของวัตถุที่แสงอินฟราเรดส่องไปกระทบ คำว่า กลไกทางเคมีก็คือ อุณหภูมิองค์ประกอบของวัตถุนั้นๆ เช่น ที่อุณหภูมิต่างกัน ค่าความร้อนที่แสงอินฟราเรดก็จะให้แตกต่างกันตามไปด้วย

คุณสมบัติของแสงอินฟราเรด

1. เป็นคลื่นแม่เหล็กไฟฟ้าที่มีความยาวคลื่นมากกว่า 7×10^{-7} เมตร
2. เป็นแสงที่ให้ความร้อนที่ร่างกายรับรู้ได้ และสามารถวัดได้โดยใช้เทอร์โมมิเตอร์
3. เป็นแสงที่ตาเปล่ามองเห็น

2.1.2 ไดโอดเปล่งแสง

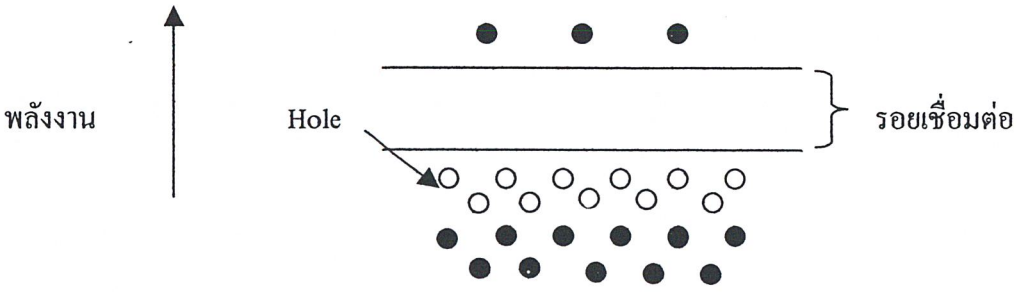
ไดโอดเปล่งแสง (Light Emitting Diode) เรียกย่อๆ ว่า LED คือไดโอดซึ่งสามารถเปล่ง แสงออกมาได้ แสงที่เปล่งออกมาประกอบด้วยคลื่นความถี่เดียว และเฟสต่อเนื่องกันซึ่งต่างกับแสงธรรมดาที่ตาคนมองเห็นอันประกอบด้วยคลื่นซึ่งมีเฟส และความถี่ต่างๆ กันมารวมกันไดโอดซึ่งสามารถให้แสงออกมาได้ทั้งชนิดสารกึ่งตัวนำของเหลว ก๊าซ ในที่นี้จะกล่าวถึงชนิดที่เป็น สารกึ่งตัวนำเท่านั้น โดยไดโอดเปล่งแสงนั้นเหมือนกับไดโอดทั่วๆ ไป ที่ประกอบด้วยสารกึ่งตัวนำ ชนิด พี และชนิด เอ็น ซึ่งโดยทั่วไปมี 2 ชนิดใหญ่ๆ คือ ไดโอดเปล่งแสงชนิดที่เปล่งแสงในย่าน ที่ตาของ

มนุษย์สามารถมองเห็นได้ และไดโอดเปล่งแสงชนิดที่ตาของมนุษย์ไม่สามารถมองเห็นเช่น แสงอินฟราเรด

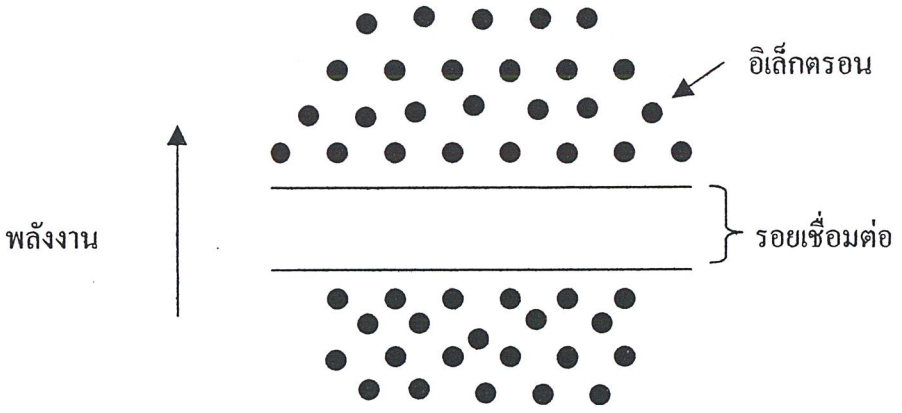
เกี่ยวกับโครงสร้างของอุปกรณ์ปล่อยแสงของไดโอด ซึ่งเกิดจากการนำสารกึ่งตัวนำ ชนิด พี และ ชนิด เอ็น มาเชื่อมต่อกันนั้นเป็นสิ่งสำคัญยิ่ง ดังนั้นเพื่อเป็นการทำให้เข้าใจขั้นตอนการปล่อยแสงอันเนื่องมาจากโครงสร้างของอุปกรณ์แสงที่เป็นสารกึ่งตัวนำ ควรจะศึกษาถึง คุณสมบัติของสารกึ่งตัวนำชนิด พี และ สารกึ่งตัวนำชนิด เอ็น ด้วย

สำหรับสารกึ่งตัวนำนั้นปกติจะรวมตัวกันอยู่ในชั้นพลังงานที่เรียกว่า วาเลียน แบนด์ ซึ่งสารกึ่งตัวนำที่มีสภาพอย่างนี้เรียกว่า จีเนียส เซมิคอนดักเตอร์ แต่สารที่เรียกว่าสารกึ่งตัวนำชนิดพี มีสภาพที่จำนวนโฮล (อนุภาคที่เป็นประจุบวก)มาก และมีอิเล็กตรอน (อนุภาคที่มีประจุลบ)น้อยดังแสดงในรูปที่ 2.3 วนในสารที่เรียกว่า สารกึ่งตัวนำชนิด เอ็น จะมีสภาพตรงกันข้ามกับ สารกึ่งตัวนำชนิด พี คือมีจำนวนอิเล็กตรอนมากกว่าโฮล และจำนวนอิเล็กตรอนที่มากเกินนี้จะรวมตัวกันอยู่ในชั้นพลังงานที่เรียกว่า คอนดักชั่น แบนด์ ดังรูปที่ 2.4

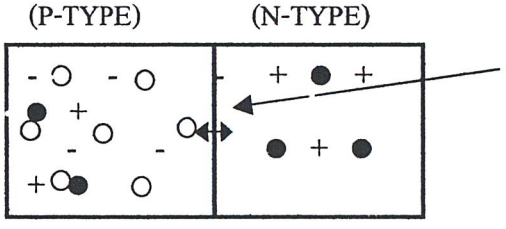
จากนั้นนำสารกึ่งตัวนำทั้งสองชนิดมาเชื่อมต่อกัน ที่บริเวณที่เชื่อมต่อกันนั้นอิเล็กตรอนที่มีจำนวนมากในสารกึ่งตัวนำชนิด เอ็น จะรวมตัวกับโฮล ที่มีจำนวนมากในสารกึ่งตัวนำชนิด พี และจากปริมาณการรวมตัวที่เพิ่มขึ้นเรื่อยๆ นั้น ทำให้เกิดประจุไฟฟ้าลบในสารกึ่งตัวนำชนิด พี เพราะว่าโฮล ลดลงเหลือแต่อิเล็กตรอน และเกิดประจุไฟฟ้าบวกขึ้นสารกึ่งตัวนำชนิด เอ็น เพราะว่าอิเล็กตรอนรวมตัวกับโฮล จากผลอันนี้ทำให้เกิดสภาพ ดีเพิลชั่น โซน ขึ้นตรงบริเวณใกล้ๆ รอยเชื่อมต่อ และได้โครงสร้างของระดับพลังงาน เกี่ยวกับโครงสร้างของระดับพลังงานนี้มี ความหมายสำคัญยิ่งยวดในการนำไปพิจารณาโครงสร้างของไดโอดเปล่งแสง โดยไดโอดเปล่งแสงจะเกิดการปล่อยแสงขึ้นในชั้น แอกทีป เลเยอร์ และไดโอดเปล่งแสงทั่วไปจึงให้แสงปล่อยออกมา ข้างนอกจากด้านหน้าที่ต่อกับขั้วไฟฟ้า ดังรูปที่ 2.5 เมื่อมีการจ่ายกระแสทำให้เกิดการกระตุ้นให้เปลี่ยนระดับชั้นพลังงานทำให้เกิดการปล่อยพลังงานออกมาอยู่ในรูปของแสงที่เรียกว่า โฟตอน



รูปที่ 2.3 แสดงระดับพลังงานของสารกึ่งตัวนำชนิด พี



รูปที่ 2.4 แสดงระดับพลังงานของสารกึ่งตัวนำชนิด เอ็น



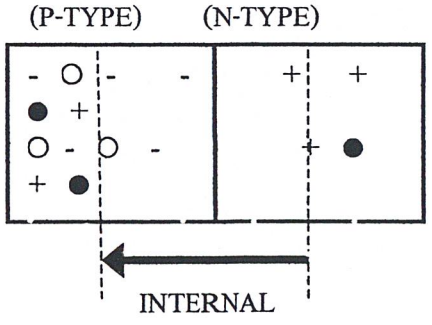
อิเล็กตรอนกับ Hole รวมกันโดยการกระจายทำให้หมดไป

○ : Hole

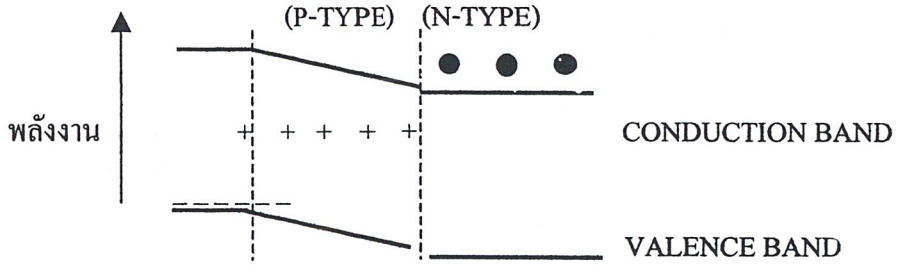
● : Electron

\+ : อะตอมที่มีประจุบวก

\- : อะตอมที่มีประจุลบ



แรงที่จะทำให้เกิดการกระจายเท่ากันกับแรงของสนามไฟฟ้าภายในทำให้อิเล็กตรอนกับ Hole แยกห่างจากกัน



รูปที่ 2.5 สนามไฟฟ้าภายในและระดับพลังงานของรอยต่อ พีเอ็น

2.2ไมโครคอนโทรลเลอร์ MCS-51

2.2.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งเป็นตระกูลที่นิยมกันมากซึ่งมีคุณสมบัติดังนี้

- สามารถนำเอาข้อมูลมา AND,OR หรือทำ Complement ทั้งแบบทีละ 8 บิต และ 1 บิต

- สามารถใช้กับหน่วยความจำสำหรับโปรแกรม (Program Memory)ซึ่งเป็นหน่วยความจำที่ใช้สำหรับเก็บคำสั่งที่จะใช้ MCS-51 ทำงานได้สูงสุด 64 กิโลไบต์ (Kilobyte) (64×1024 ไบต์) ทำให้เขียนโปรแกรมควบคุมการทำงานได้มาก

- สามารถติดต่อกับหน่วยความจำสำหรับข้อมูล (Data Memory) ซึ่งเป็นหน่วยความจำสำหรับเก็บข้อมูลในระหว่างการทำงานของโปรแกรมได้สูงสุด 64 กิโลไบต์

- ใน 8051 และ 8751 มีหน่วยความจำสำหรับโปรแกรมจำนวน 4 กิโลไบต์ ใน 8052 และ 8752 มีหน่วยความจำสำหรับโปรแกรมจำนวน 8 กิโลไบต์ (ใน8031 และ 8032 ไม่มีหน่วยความจำชุดนี้) อยู่ภายในวงจรรวมทำให้ไม่ต้องต่อหน่วยความจำสำหรับโปรแกรมอยู่ ภายนอก ระบบรวมทั้งหมดจึงมีขนาดเล็ก และไม่มีสัญญาณรบกวนจากภายนอกซึ่งจะทำให้ MCS-51 ทำงานผิดพลาดได้ยาก

- มีพอร์ตแบบขนาน (Parallel Port) สำหรับข้อมูลเข้าและออกจำนวน 32 บิต ที่ข้อมูล แต่ละบิตเป็นอิสระต่อกัน

- มีวงจร Timer/Counter ขนาด 16 บิต 2 ชุด (8052 มี 3 ชุด) ที่ทำงานในโหมดต่างๆ ได้ถึง 4 โหมด

- มี Universal Asynchonous Receiver Transmitter (UART) สำหรับรับและส่งข้อมูลอนุกรม (Serial) แบบ Full Duplex ที่สามารถเลือกรูปแบบการรับ-ส่งข้อมูลได้ 4 แบบ

- มีแหล่งกำเนิดสัญญาณขอขัดจังหวะการทำงานของโปรแกรม (Interrupt Service Signal) 6 แหล่ง ซึ่งสามารถทำการกระโดดไปทำงานตอบสนองการขัดจังหวะ (Interrupt Service Routine) ได้ต่างๆ กัน 5 ตำแหน่ง

- สามารถเลือกการทำงานให้อยู่ในโหมดของ Idle และ Power Down ซึ่งจะประหยัดการใช้กำลังไฟในการทำงาน

ซึ่งจากคุณสมบัติดังกล่าว จึงทำให้ MCS-51 เป็นที่นิยมนำมาใช้ในการควบคุมระบบอัตโนมัติมากคุณสมบัติดังกล่าวบรรจุในวงจรรวมเดียว (Single Chip) 40 ขา ดังนั้นจึงสามารถออกแบบให้ระบบทั้งหมดมีขนาดเล็ก และการที่ทั้งหมดบรรจุภายในวงจรรวมเดียวจึงทำให้การตรวจสอบหาข้อผิดพลาดในระบบง่ายไม่สลับซับซ้อน รวมทั้งลดปัญหาเรื่องการที่มีสัญญาณ รบกวนในระบบจนทำให้การทำงานผิดพลาดไป

2.2.2 โครงสร้างของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีสมาชิกในตระกูลหลายตัวด้วยกัน แต่ละเบอรจะมีคุณสมบัติพิเศษบางอย่างแตกต่างกัน เช่น มีหน่วยความจำสำหรับเก็บโปรแกรม และข้อมูลภายในชิพเพิ่มขึ้น มีวงจรเปลี่ยนค่าสัญญาณแอนะลอกเป็นดิจิตอลในตัว สามารถรับสัญญาณ อินเตอร์รัปต์ได้หลายชนิด ทำกระบวนการ DMA (Direct Memory Access) ได้ในตัว มีรีจิสเตอร์สำหรับเป็นไทม์เมอร์ หรือเคาน์เตอร์เพิ่มขึ้น คุณสมบัติพิเศษที่แตกต่างกันของ ไมโครคอนโทรลเลอร์ แต่ละเบอร์ในตระกูลนี้ดังแสดงในตารางที่ 2.1

Device	ROMless Version	EPROM Version	ROM Byte	Ram Byte	8-Bit I/O Port	16-Bit Timer / Counters	Programmable Counter Array (PCA)	UART	Serial Expansion Port (SEP)	Gobal Serial Channel (GSC)	DMA Channels	A/D Chaanel	Interrupt Sources / Vectors	Power Down And Idle Modes
8051	8031	-	4K	128	4	2		3					6/5	
8051AH	8031AH	8751H 8751BH	4K	128	4	2		3					6/5	
8052AH	8032AH	8752BH	8K	256	4	3		3					8/6	
80C51BH	80C31BH	87C51	4K	128	4	2		3					6/5	3
83C51FA	80C51FA	87C51FA	8K	256	4	3	3	3					14/7	3
83C51FB	80C51FA	87C51FB	16K	256	4	3	3	3					14/7	3
83C51GA	80C51GA	87C51GA	4K	128	4	2		3	3			8	8/7	3
83C152JA	80C152JA	-	8K	256	5	2		3		3	2		19/11	3
-	80C152JB	-	-	256	7	2		3		3	2		/19/11	3
83C152JC	80C152JC	-	8K	256	5	2		3		3	2		19/11	3
-	80C152JD	-	-	256	7	2		3		3	2		19/11	3
83C451	80C451	-	4K	128	7	2		3					6/5	3

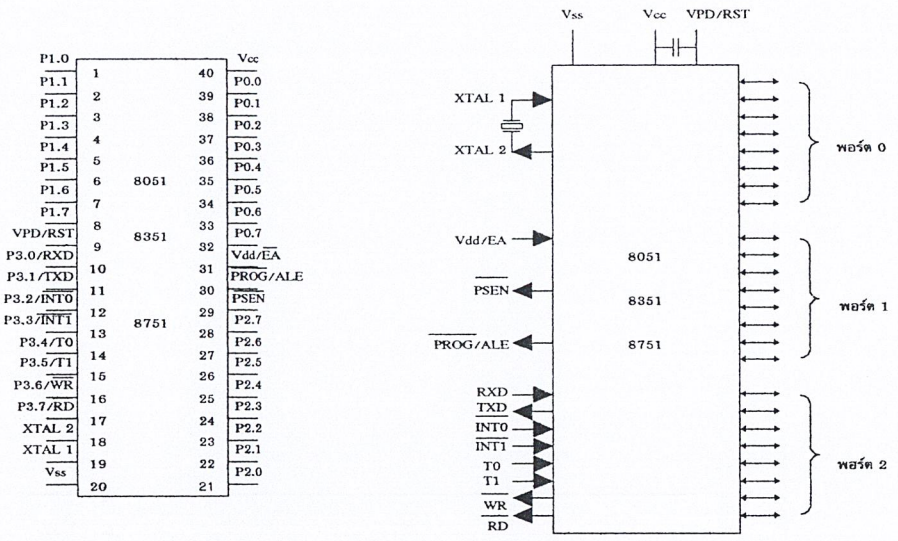
ตารางที่ 2.1 ตารางแสดงความแตกต่างของสมาชิกไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ที่นับได้ว่าเป็นเบอร์พื้นฐานสำหรับตระกูล MCS-51 นี้ได้แก่เบอร์ 8051,8031,8751 โดยเบอร์ 8051 จัดเป็นสมาชิกตัวแรกในตระกูลซึ่งมีหน่วยความจำสำหรับเก็บโปรแกรมในชิพเป็น ROM ขนาด 4 กิโลไบต์ และหน่วยความจำสำหรับเก็บข้อมูลทั่วไป (RAM) จำนวน 128 ไบต์ มีพอร์ตขนาด 8 บิต 4 พอร์ต มีรีจิสเตอร์สำหรับใช้ไทม์เมอร์ หรือ เคาร์เตอร์ขนาด 16 บิตรวม 2 ตัว รับสัญญาณอินเตอร์รัปต์จากภายนอกได้ 2 ชนิด สามารถรับและส่งข้อมูลแบบอนุกรมผ่านทางพอร์ตสื่อสารแบบอนุกรม มีวงจรออสซิลเลเตอร์เพื่อสร้างสัญญาณนาฬิกาควบคุมการทำงานในตัวเอง ส่วน 8751 จะมีคุณสมบัติเหมือนเบอร์ 8051 ทุกอย่างต่างกันเพียงชนิดของหน่วยความจำสำหรับเก็บโปรแกรมในชิพเบอร์ 8751 จะเป็น EPROM แทนที่จะเป็น ROM ส่วน

เบอร์ 8031 จะเหมือนกับเบอร์ 8051 ต่างกันเพียงในเบอร์ 8031 ไม่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิพ

2.2.3 ตำแหน่งขาของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูป 2.6



รูปที่ 2.6 แสดงตำแหน่งขาของชิพไมโครคอนโทรลเลอร์ตระกูล MCS-51

- ขา Vss (ขา 20) สำหรับต่อลงกราวด์

- ขา Vcc (ขา 40) สำหรับต่อแหล่งจ่ายแรงดันกระแสตรงขนาด 5 โวลต์

- ขาพอร์ต 0 (ขา 31-39) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 0 ขนาด 8 บิต (P0.0-P0.7) แบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นอินพุต เอาต์พุตพอร์ตทั่วไปได้แล้วพอร์ต 0 ยังใช้ในการติดต่อหน่วยความจำสำหรับเก็บโปรแกรม และข้อมูลภายนอกชิพด้วยโดย ส่งค่าแอดเดรสบัสไบต์ (A0-A7) และมัลติเพลกซ์กับการรับส่งข้อมูล (D0-D7) จากหน่วยความจำภายนอกในระหว่างการเขียน หรืออ่านข้อมูลโดยมีวงจรพูลอัพภายใน

- ขาพอร์ต 1 (ขา 1-8) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 1 (P1.0-P1.7) สามารถใช้เป็นอินพุตหรือเอาท์พุตทั่วไปได้ ขา P1.0,P1.1 ในเบอร์ 8052 จะใช้งานในหน้าที่อย่างอื่นนอกเหนือจากใช้เป็นอินพุต หรือเอาร์พุตพอร์ตทั่วไปได้

- ขาพอร์ต 2 (ขา21-28) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 2 (P1.0-P1.7) ขนาด 8 บิตแบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นอินพุต เอาต์พุตพอร์ตทั่วไปได้แล้วพอร์ต 2 ยังใช้ติดต่อหน่วยความจำสำหรับเก็บโปรแกรม และข้อมูลภายนอกชิพด้วย โดยส่งค่าแอดเดรสบัสไบต์สูง(A8-A15) และมีวงจรพูลอัพภายใน

- ขาพอร์ต 3 (ขา 10-17) มี 8 ขาใช้เป็นขาสำหรับพอร์ต 3 (P3.0-P3.7) สามารถใช้เป็นอินพุต หรือเอาต์พุตทั่วไปได้ นอกจากนี้ยังใช้งานในหน้าที่พิเศษต่างๆ อีกหลายอย่างดังนี้

ขา P3.0	RXD	ใช้รับข้อมูลจากภายนอกแบบอนุกรม
ขา P3.1	TXD	ใช้ส่งข้อมูลออกไปภายนอกแบบอนุกรม
ขา P3.2	INT0	ใช้เป็นอินพุตหรือรับสัญญาณอินเตอร์รัปต์ชนิด 0
ขา P3.3	ITN1	ใช้เป็นอินพุตหรือรับสัญญาณอินเตอร์รัปต์ชนิด 1
ขา P3.4	T0	สัญญาณอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 0
ขา P3.5	T1	สัญญาณอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 1
ขา P3.6	WR	ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิพ
ขา P3.7	RD	ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลไปยังหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิพ

- ขา RST (ขา 9) ใช้สำหรับรีเซตทุกอย่างภายในชิพ เพื่อเริ่มต้นการทำงานใหม่ การรีเซตใช้เมื่อเริ่มจ่ายพลังงาน หรือโปรแกรมเกิดการผิดพลาด เมื่อต้องการรีเซตชิพ MCS-51 ขานี้ต้องมีสภาวะเป็น 1 เป็นอย่างน้อย 2 แมชชีนไซเกิลระหว่างที่ออสซิลเลเตอร์ยังทำงานอยู่

- ขา ALE/ROG (ขา 30) เป็นขาสำหรับใช้ส่งสัญญาณไปภายนอก เพื่อควบคุมการแลตช์ค่าแอดเดรสไบต์ต่ำ (address latch enable) จากพอร์ต 0 ในระหว่างการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอก

- ขา PSEN (ขา 29) ใช้ส่งสัญญาณสโตรบเพื่ออ่านคำสั่งจากโปรแกรมที่เก็บไว้ในหน่วยความจำภายนอกชิพ (program strobe enable) เมื่อชิพทำงานด้วยโปรแกรมจากภายนอกขานี้ จะส่งสัญญาณสโตรบสองครั้งในแต่ละแมชชีนไซเกิล แต่ในช่วงการเขียน หรือการอ่านข้อมูลกับหน่วยกวามจำที่เก็บโปรแกรมภายนอก หรือเมื่อใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมภายในชิพจำไม่มีสัญญาณออกมาจากขานี้

- ขา EA/Vpp (ขา 31) เป็นขาสำหรับใช้เลือกให้ MCS-51 ทำงานจากโปรแกรมที่อยู่ ภายในหรือภายนอกชิพ โดยหากขานี้มีสภาวะเป็น 0 หมายถึงให้ใช้โปรแกรมจากหน่วยความจำ ที่เก็บโปรแกรมภายนอก หากขานี้สภาวะเป็น 1 หมายถึงบังคับให้ MCS-51 ใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมภายในชิพ

-ขา XTAL 1 (ขา 19) ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุตเข้าสู่วงจรออสซิลเลเตอร์

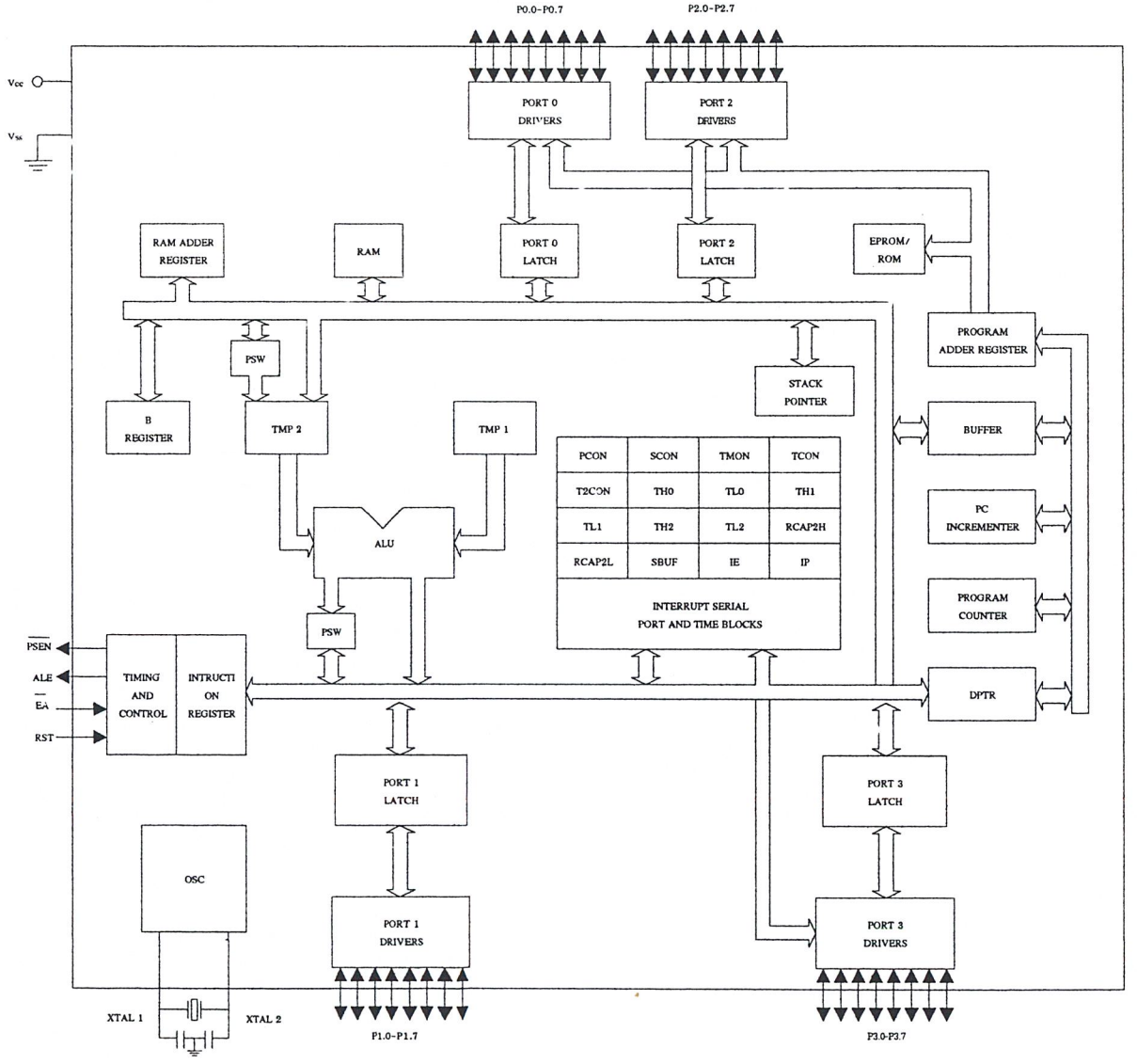
-ขา XTAL2 (ขา 18) ใช้ต่อคริสตอลภายนอก โดยเป็นเอาต์พุตออกจากวงจร ออสซิลเลเตอร์

2.2.4 โครงสร้างภายในของ MCS-51

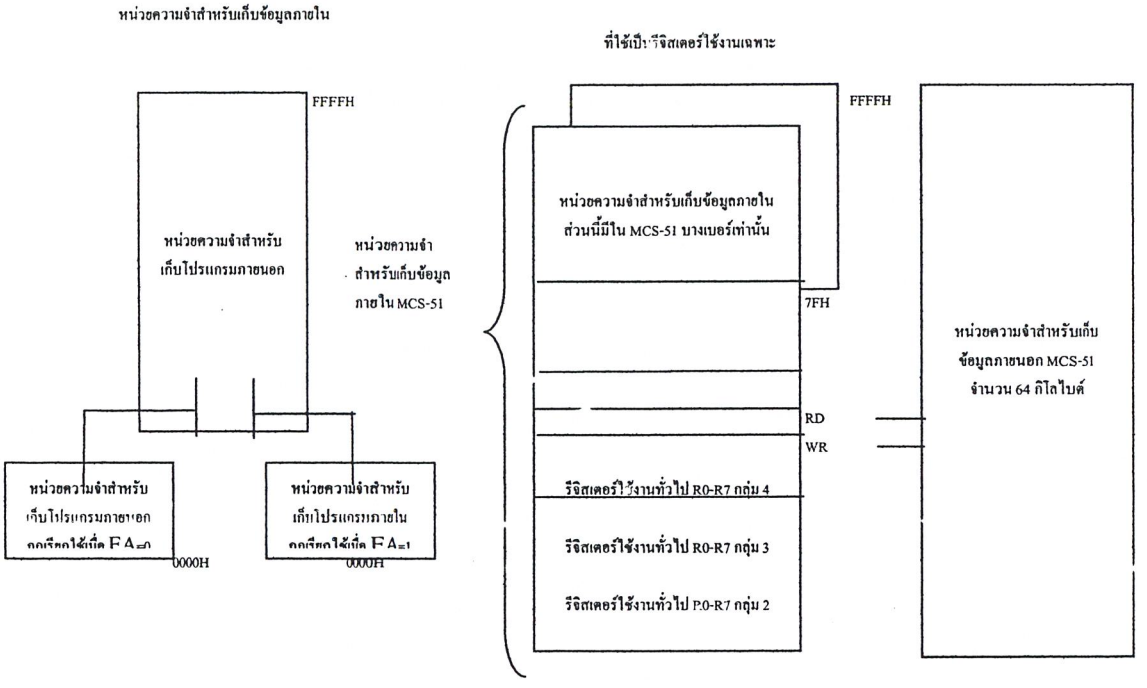
โครงสร้างภายในชิพไมโครคอนโทรลเลอร์ตระกูล MCS-51 แสดงดังรูปที่ 2.7

โครงสร้างหน่วยความจำภายใน MCS-51 แสดงดังรูปที่ 2.8 ซึ่งไมโครคอนโทรลเลอร์ ในตระกูล MCS-51 ทุกเบอร์จะแบ่งหน่วยความจำออกเป็น 2 ส่วน คือ

- หน่วยความจำสำหรับเก็บโปรแกรม (program memory)
- หน่วยความจำสำหรับเก็บข้อมูล (data memory)



รูปที่ 2.7 แสดงโครงสร้างภายในของชิพไมโครคอนโทรลเลอร์ MCS-51



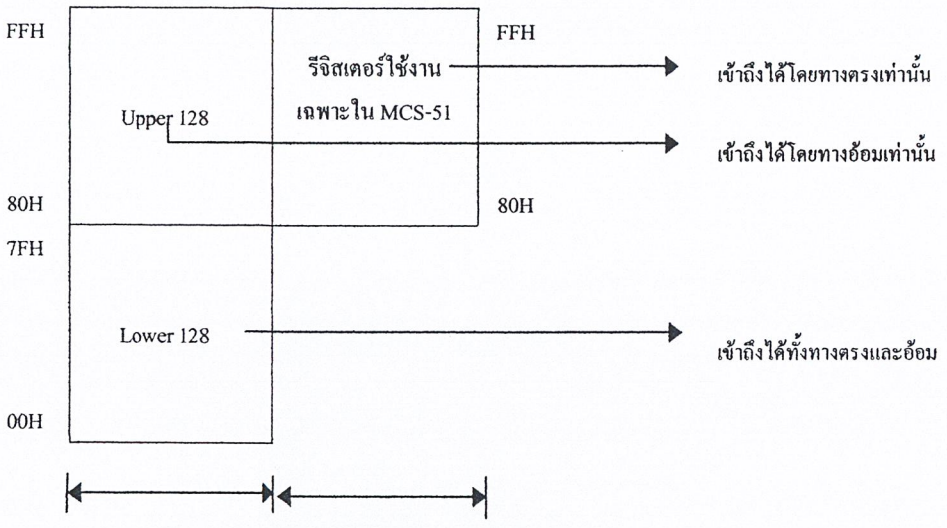
รูปที่ 2.8 แสดงโครงสร้างหน่วยความจำทั้งหมดของ MCS-51

2.2.4.1หน่วยความจำสำหรับเก็บโปรแกรม หน่วยความจำสำหรับเก็บโปรแกรมภายในตัว MCS-51 จะแบ่งออกเป็น 2 ส่วนคือ หน่วยความจำสำหรับเก็บโปรแกรมภายในชิพ (internal program memory) และหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิพ (external program memory)

2.2.4.2หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บข้อมูลของ MCS-51 จะแบ่งออกเป็น 2 ส่วนคือ หน่วยความจำสำหรับเก็บข้อมูลภายในชิพ และหน่วยความจำสำหรับเก็บ ข้อมูลภายนอกชิพ หน่วยความจำสำหรับเก็บข้อมูลภายในชิพของ MCS-51 ยังแบ่งออกเป็น 2 ส่วนย่อย คือ

- ส่วนที่ใช้เก็บข้อมูลทั่วไป (internal ram)
- ส่วนที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (special function register)

หน่วยความจำส่วนที่ใช้เก็บข้อมูลทั่วไปภายในชิพเป็นหน่วยความจำสำหรับเก็บข้อมูลที่มีภายใน MCS-51 หน่วยความจำส่วนนี้มีไว้สำหรับเก็บข้อมูลในขณะทำงาน หน่วยความจำสำหรับเก็บข้อมูลภายในชิพที่เป็นรีจิสเตอร์ใช้งานเฉพาะเป็นเพื่อควบคุมการทำงาน และบอกสถานะของซีพียู แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลสองบริเวณมีดังในรูปที่ 2.9



รูปที่ 2.9 แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิพ MCS-51

MCS-51 ทุกเบอร์จะมีหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิพอย่างน้อยที่สุด 128 ไบต์ ไปจนถึง 256 ไบต์ทั้งนี้ขึ้นอยู่กับเบอร์ของชิพ หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิพบริเวณ 128 ไบต์แรกมีชื่อเรียกว่า lower และในบริเวณ 128 ไบต์หลังที่มีเพิ่มมาในบางเบอร์มีชื่อเรียกว่า upper 128 ดังที่แสดงในรูปที่ 2.9

หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิพบริเวณ 128 ไบต์หลัง (ตำแหน่ง 80H ขึ้นไป) จะมีตำแหน่งตรงกับหน่วยความจำสำหรับเก็บข้อมูลภายในชิพที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (ตำแหน่ง 80H ขึ้นไป) โดยมีการเข้าถึงข้อมูลในหน่วยความจำทั้งสองส่วนไม่เหมือนกัน

หน่วยความจำสำหรับเก็บข้อมูลภายในส่วนนี้มีใน MCS-51 บางเบอร์	FFH 80H 7FH 2FH
บริเวณหน่วยความจำที่ใช้ได้ระดับบิต จำนวน 16 ไบต์ × 8 = 128	20H
รีจิสเตอร์ใช้งานทั่วไป R0-R7 กลุ่ม 4	18H
รีจิสเตอร์ใช้งานทั่วไป R0-R7 กลุ่ม 3	10H
รีจิสเตอร์ใช้งานทั่วไป R0-R7 กลุ่ม 2	08H
รีจิสเตอร์ใช้งานทั่วไป R0-R7 กลุ่ม 1	00H

รูปที่ 2.10 แสดงหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิพทั้งสองส่วน

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	(T_2CON)		($RCAP_2L$)	($RCAP_2H$)	(TL_2)	(TH_2)			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPH	DPL				PCON	87

รูปที่ 2.11 แสดงโครงสร้างและตำแหน่งของรีจิสเตอร์ใช้งานเฉพาะใน MCS-51

2.2.4.3 รีจิสเตอร์ใช้งานเฉพาะ เนื่องจาก MCS-51 ถูกออกแบบไว้สำหรับใช้ควบคุมระบบโดยเฉพาะ จึงทำให้มีความสามารถเฉพาะตัวหลายอย่าง การควบคุมการทำงานของวงจรภายใน ไมโครคอนโทรลเลอร์จะกระทำผ่านรีจิสเตอร์ที่ถูกกำหนดหน้าที่ไว้แล้ว ดังนั้นจำเป็นต้องทราบหน้าที่การทำงานของรีจิสเตอร์ใช้งานเฉพาะแต่ละตัวให้ละเอียด รีจิสเตอร์ใช้งานเฉพาะทั้งหมดใน MCS-51 มีดังนี้แสดงในรูปที่ 2.11

หน้าที่พิเศษต่างๆของรีจิสเตอร์ใช้งานเฉพาะที่ใช้งานดังจะแสดงต่อไปนี้

- ACC (Accumulator) มีแอดเดรสอยู่ที่ 0E0H เป็นรีจิสเตอร์ที่ใช้ในการคำนวณต่างๆและอาจใช้เก็บข้อมูลชั่วคราวได้ สามารถอ้างตำแหน่งแบบบิตได้

- B (B Register) มีแอดเดรสอยู่ที่ 0F0H เป็นรีจิสเตอร์ที่ใช้ร่วมกับ ACC ในคำสั่ง การ คูณ การหาร สามารถอ้างตำแหน่งแบบบิตได้

- SP (Stack Pointer) มีแอดเดรสอยู่ที่ 81H ใช้ชี้ตำแหน่งหน่วยความจำที่ใช้สำหรับ เก็บแอดเดรสเดิมของโปรแกรมก่อนคำสั่ง CALL หรือตำแหน่งที่จะใช้เก็บข้อมูลด้วยตำแหน่ง PUSH และตำแหน่งที่จะอ่านข้อมูลออกมาด้วยคำสั่ง POP ไม่สามารถอ้างแบบตำแหน่งบิตได้

- DPH (Data Pointer High Byte) มีแอดเดรสอยู่ที่ 83H

- DPL (Data Pointer Low Byte) มีแอดเดรสอยู่ที่ 82H ทั้ง DPH และ DPL จะประกอบกันเป็นรีจิสเตอร์ขนาด 16 บิตชื่อ DPTR ซึ่งใช้ในการอ้างแอดเดรสของหน่วยความจำภายนอก เมื่อต้องการอ่าน หรือเขียนข้อมูลลงไป ไม่สามารถอ้างตำแหน่งแบบบิตได้

- TH0 (Timer/Counter 0 High Byte) มีแอดเดรสอยู่ที่ 8CH

- TL0(Timer/Counter 0 Low Byte) มีแอดเดรสอยู่ที่ 8AH ทั้ง TH0 และ TL0 ใช้ในการเก็บค่าเริ่มต้นในการนับของ Timer/Counter 0 ไม่สามารถอ้างตำแหน่งแบบบิตได้

- TH1 (Timer/Counter 1 High Byte) มีแอดเดรสอยู่ที่ 8DH

- TL1(Timer/Counter 1 Low Byte) มีแอดเดรสอยู่ที่ 8BH ทั้ง TH1 และ TL1 ใช้ในการเก็บค่าเริ่มต้นในการนับของ Timer/Counter 0 ไม่สามารถอ้างตำแหน่งแบบบิตได้

- P0 (Port 0) มีแอดเดรสอยู่ที่ 0A0H พอร์ตทั้ง 3 เป็นพอร์ตแบบขนาน สามารถ อ้างตำแหน่งแบบบิตได้ โดยที่มี P0 และ P2 เป็นแอดเดรส และดาต้าบัส ส่วน P1 ใช้งานได้เอนกประสงค์

- PSW (Program Status Word) มีแอดเดรสอยู่ที่ 0D0H เป็นรีจิสเตอร์ที่บอกสภาวะของผลลัพธ์ที่ได้จากการทำขบวนการทางคณิตศาสตร์ (Arithmetic Operation) เช่น การบวก ลบ คูณ หาร หรือคำสั่งที่เกี่ยวกับลอจิกต่างๆ (Logic Operation) รวมถึง การบวก Bank ของ R0-R7 ที่ใช้อยู่ในขณะนั้นด้วย สามารถอ้างตำแหน่งแบบบิตได้

CY	AC	F0	RS1	RS0	0V	-	P

บิต ชื่อบิต

PSW.7 CY Carry flag จะเซตเมื่อหลังจากบวก ลบ แล้วเกิดการทดจากบิต 7 หรือ บิต 7 มีการขอยืมตัวทด

PSW.6 AC Auxiliary Carry flag จะเซตเมื่อหลังจากการบวก ลบแล้วเกิดการทด จากบิต 3 หรือบิต 3 มีการขอยืมตัวทด

PSW.5 F0 Flag 0 เป็นแฟล็กที่ผู้ใช้สามารถใช้งานได้เอนกประสงค์

PSW.4 RS1 ใช้เป็นตัวเลือก Bank ของ R0-R7 ที่ใช้งานอยู่ในขณะนั้น

PSW.3 RS0 ใช้เป็นตัวเลือก Bank ของ R0-R7 ที่ใช้งานอยู่ในขณะนั้น

RS1	RS0	Bank	แอดเดรส
1	1	1	20H
1	0	2	18H
0	1	1	10H
0	0	0	00H

PSW.2 0V Overflow flag จะเซตเมื่อหลังการบวก ลบแล้วเกิดการทด หรือยืมจาก

บิต 7 แต่ไม่มีกรทด หรือยืมจากบิต 6 หรือ เกิดการทด หรือยืมจากบิต 6 แต่ไม่มีการทด หรือยืมจากกบิต 7

PSW.1 - ไม่ได้ใช้งาน

PSW.0 P Parity flag จะเซต หรือรีเซต เมื่อรีจิสเตอร์ ACC มีจำนวนบิตที่เป็น "1" เลขคู่หรือคี่

- IE (Interrupt Enable Register) มีแอดเดรสอยู่ที่ 0A8H เป็นรีจิสเตอร์ที่ใช้ในการ Enable Disable การเกิด Interrupt ทุกชนิด ซึ่งได้แก่ Timer/Counter Interrupt, External Interrupt และ Serial Port Interrupt สามารถอ้างตำแหน่งแบบบิต โดยถ้าเป็น "0" จะหมายถึง Disable และถ้าเป็น "1" จะหมายถึง Enable

EA	-	ET	ES	ET1	EX1	ET0	EX0

บิต	ชื่อบิต	
IE.7	EA	เป็นบิตที่ใช้ในการ Enable หรือ Disable การ interrupt จากแหล่งหมาย ความว่าถ้าบิตนี้เป็น "0" จะไม่เกิดการ Interrupt จากตัวใดเลย
IE.6	-	ไม่ได้ใช้งาน
IE.5	ET2	Enable Timer 2 ใช้ในการ Enable หรือ Disable Time/Cornter 2 (มีเฉพาะใน 8052)
IE.4	ES	Enable Serial Port ใช้ในการ Enable หรือ Disable การ Interrupt จาก Serial Port
IE.3	ET1	Enable Timer 1 ใช้ในการ Enable หรือ Disable การ interrupt จาก Timer/Counter1
IE.2	EX1	Enable External Interrupt 1 ใช้ในการ Enable หรือ Disable การ Interrupt จาก External Interrupt 1
IE.1	ET0	Enable Timer 0 ใช้ในการ Enable หรือ Disable การ Interrupt จาก Timer/Counter 0
IE.0	EX0	Enable External Interrupt 0 ใช้ในการ Enable หรือ Disable การ Interrupt จาก External Interrupt 0

เมื่อเกิดการ Interrupt ขึ้น ไมโครคอนโทรลเลอร์จะกระโดดการทำงานไปยัง Vector Address ของแหล่งที่ทำให้เกิด Interrupt ซึ่งเป็นตำแหน่งที่แน่นอนดังแสดงในตาราง จะเห็นได้ว่าแต่ละ Vector จะห่างกันเพียง 8 ไบต์ ซึ่งส่วนใหญ่จะไม่เพียงพอสำหรับการ Interrupt Service Routine ดัง

นั้นที่ Vector Address จะใช้คำสั่ง Jump กระโดดการทำงานไปยังตำแหน่งที่มีเนื้อที่ ที่กลับไปทำงาน ณ ตำแหน่งเดิมก่อนการเกิด Interrupt

Interrupt Source	Vector Address
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial Port	0023H
Timer 2	002BH

ตารางที่ 2.2 แสดง Vector Address ของ Interrupt Source ต่างๆ

- IP (Interrupt Priority Register) มีแอดเดรสอยู่ที่ 0B8Hนอกจากเราจะสามารถ Enable หรือ Disable การตอบสนอง interrupt ที่มาจากแหล่งต่างๆ ได้แล้ว สำหรับในกรณีที่มีการInterrupt มาพร้อมกับสองแหล่ง ไมโครคอนโทรลเลอร์จะไม่สามารถตอบสนองต่อสัญญาณ Interrupt ได้พร้อมกัน ทำให้ต้องมีตัวตัดสินว่าจะให้ไมโครคอนโทรลเลอร์ตอบสนองสัญญาณใด ก่อน ซึ่งจะใช้รีจิสเตอร์ IP นี้เป็นตัวจัดลำดับความสำคัญก่อนหลังในการตอบสนองการ Interrupt สามารถอ้างตำแหน่งแบบบิตได้

-	-	PT2	PS	PT1	PX1	PT0	PX0

บิต	ชื่อบิต	
TCON.7	TF1	Timer 1 Overflow Flag จะเซตตัวเองโดยอัตโนมัติ เมื่อ Timer/Counter 1 ทำการส่งสัญญาณ Interrupt ออกไป และจะรีเซตตัวเองการตอบสนองสัญญาณ Interrupt คือ เมื่อไมโครคอนโทรลเลอร์กระโดดการทำงานไปที่ตำแหน่ง 001BH
TCON.6	TR1	Timer 1Run Control Bit เมื่อการเซต,รีเซตจะเป็นการสั่งให้ Timer/Counter 1 เริ่มนับหรือหยุดนับ ซึ่งเป็นการสั่งทาง Software
TCON.5	TF0	Timer 0 Overflow Flag เช่นเดียวกับ TF1 แต่การตอบสนอง Interrupt จะกระโดดไปที่ตำแหน่ง 000BH

TCON.4	TR0	Timer 0 Run Control Bit เช่นเดียวกับ TR1 แต่จะใช้ควบคุมการทำงานของ Timer/Counter 0
TCON.3	IE1	External Interrupt 1 Edge flag จะเซตตัวเองโดยอัตโนมัติ เมื่อมีการตรวจพบสัญญาณ Interrupt ที่ขา INT1(ขา 13) และจะรีเซตตัวเองเมื่อมีการตอบสนองต่อสัญญาณ Interrupt คือ เมื่อไมโครคอนโทรลเลอร์กระโดดการทำงานไปที่ตำแหน่ง 0013H
TCON.2	IT1	Interrupt 1 Type Control Bit จะเซต, รีเซตด้วย Software ใช้ในการกำหนดลักษณะการตรวจจับสัญญาณ Interrupt จากภายนอกของ External Interrupt 1 โดยถ้าเซตจะตรวจสอบสัญญาณแบบ Transition Activated คือ สัญญาณเปลี่ยนจาก “1” เป็น “0” (ขอบขาลง) และถ้ารีเซต จะตรวจสอบสัญญาณแบบ Level Activated คือ สัญญาณใน ระดับ “1”
TCON.1	IE0	External Interrupt 0 Edge Flag เช่นเดียวกับ IE1 แต่จะตรวจจับสัญญาณ Interrupt ที่ขา Int0 (ขา 12) และเมื่อมีการตอบสนองต่อสัญญาณ Interrupt จะกระโดดการทำงานไปที่ตำแหน่ง 0003H
TCON.0	IT0	Interrupt 0 Type Control Bit เช่นเดียวกับ IT1 แต่ใช้ในการกำหนดลักษณะการตรวจจับสัญญาณ Interrupt จากภายนอกของ External Interrupt 0

- TMOD (Timer/Counter Mode Control Register) มีแอดเดรสอยู่ที่ 89H ไม่สามารถอ้างตำแหน่งแบบบิตได้ ใช้ในการเลือก Mode การทำงาน Timer/Counter จะเห็นได้ว่า TMOD ถูกแบ่งออกเป็นสองส่วนที่เหมือนกัน ซีกซ้าย (TMOD.4-TMD.7) ใช้ควบคุม Timer/Counter 1 และซีกขวา (TMOD.0-TMOD.3) ใช้ควบคุม Timer/Counter 0

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 2			

บิต	ชื่อบิต	
TMOD.7	GATE	เป็นบิตที่ใช้เลือกการควบคุมการทำงานของ Timer/Counter 1 ว่าจะควบคุมด้วย Hardware หรือ Software โดยเมื่อ TR1 ของ TCON เป็น “1” และ Gate = “1” Timer/Counter 1 จะเริ่มทำงานเมื่อ INT1 (ขา 13) เป็น “1” เป็นลักษณะการควบคุมทาง Hardware ถ้า GATE เป็น “0” และ

		TR1 เป็น “1” Timer/Counter 1 จะทำงานทันทีโดยไม่สนใจสัญญาณที่ขา INT1 เป็นลักษณะ การควบคุมทาง Software
TMOD.6	C/T	เป็น ”0” จะหมายถึง การทำงานเป็น Timer โดยอินพุตจะเป็นสัญญาณนาฬิกาจากภายในของไมโครคอนโทรลเลอร์และถ้าเป็น “1”จะหมายถึง การทำงานเป็น Counter โดยจะรับสัญญาณทางขา T1 (ขา 15) แทน
TMOD.5	M1	ใช้เลือก Mode การทำงานของ Timer/Counter 1
TMOD.4	M0	ใช้เลือก Mode การทำงานของ Timer/Counter 1
TMOD.3	GATE	
TMOD.2	C/T	มีการทำงานเหมือนกัน แต่จะเกี่ยวข้องกับ TR0, INT0, Timer/Counter 0
TMOD.1	M1	
TMOD.0	M0	

M1	M0	Mode
0	0	0
0	1	1
1	0	2
1	1	3

ตารางที่ 2.3 แสดงการกำหนดการเลือก Mode

2.2.4.4 Mode การทำงานของ Timer/Counter ไมโครคอนโทรลเลอร์มีรีจิสเตอร์ขนาด 16 บิต อยู่ด้วยกัน 2 ชุด คือ TH0, TL0 และ TH1,TL1 โดยทำหน้าที่เป็น Timer มักจะทำการเพิ่มค่าของรีจิสเตอร์ขึ้น 1 ทุกๆ 1/12 ของความถี่ออสซิลเลเตอร์ แต่ถ้าทำหน้าที่เป็น Counter รีจิสเตอร์จะเพิ่มค่าขึ้น 1 ทุกครั้งที่ขา T1, T0 มีการเปลี่ยนแปลงจาก “1” เป็น “0” (ขอบขาลง) และทุกครั้งที่ทุกบิตของรีจิสเตอร์เป็น “1” ทั้งหมดเปลี่ยนแปลงเป็น “0” ทั้งหมด (การเกิด Overflow) จะเกิดการส่งสัญญาณ Interrupt 1 ครั้ง เราสามารถเลือกลักษณะการทำงานได้ทั้งหมด 4 Mode ดังต่อไปนี้

1) Mode 0 (13 Bit Timer/Counter) การทำงานใน Mode นี้จะใช้รีจิสเตอร์ขนาด 13 บิต เป็นตัวนับ โดยจะใช้ 8 บิตของ THx และ 5 บิตล่างของ TLx (บิต 0-4 ของ TLx) ประกอบกันเป็นตัวนับขนาด 13 บิต โดย 3 บิตที่เหลือ TLx จะจะไม่มีผลต่อการนับ โดยที่การทำงานของมันจะสัมพันธ์กับการตั้งค่าใน TMOD ด้วย เมื่อรีจิสเตอร์ขนาด 13 บิตนี้เปลี่ยนจาก 1111 1111 1111 1 เป็น 0000 0000 0000 0 หรือที่เรียกว่าเกิด Overflow มันจะทำการส่งสัญญาณ Interrupt และจะแสดงที่บิต TFx ของ TCON ด้วย

2) Mode 1 (16 Bit Timer/Counter) การทำงานใน Mode นี้ทั้งนี้ Timer 1 และ Timer 0 จะคล้ายกับการทำงานใน Mode 0 เพียงแต่ว่าตัวนับจะเป็นรีจิสเตอร์ขนาด 16 บิต

3) Mode 2 (8 Bit Auto Reload) ใน Mode 0 และ Mode 1 ที่ผ่านมาจะต้องทำการตั้งค่าเริ่มต้นของตัวนับใหม่ทุกครั้งที่ Timer/Counter ส่งสัญญาณ Interrupt ส่วนใน Mode 2 นี้ จะใช้ตัวนับขนาดเพียง 8 บิตเท่านั้น โดยมี TLx เป็นตัวนับ และหลังจากที่ Timer/Counter ส่งสัญญาณ Interrupt ก็จะทำการ Load ค่าเริ่มต้นใหม่โดยอัตโนมัติจาก THx ให้กับ TLx

4) Mode 3 (Two 8 Bit Timer) ใน Mode นี้ Timer/Counter 0 จะถูกแยกเป็นตัวนับ 2 ชุดที่เป็นอิสระต่อกัน โดยชุดแรกจะใช้ตัวนับ TL0 ซึ่งสามารถทำงานเป็นตัวนับขนาด 8 บิตและการทำงานก็จะเกี่ยวข้องกับ GATE, TR0, INT0, T0 และ TF0 คือสามารถทำงานในลักษณะของ Timer โดย มีอินพุตเป็นความถี่ของระบบหรือทำงานในลักษณะของ Counter ที่มีอินพุทจาก ภายนอกโดยผ่านขา T0 และชุดที่สองตัวนับ THx ซึ่งจะทำงานในลักษณะของ Timer ได้เพียงอย่างเดียวเท่านั้น และจะควบคุมการปิดเปิดการนับโดยใช้ TR1 เมื่อเกิด Over Flow ก็จะ ส่งสัญญาณ Interrupt ออกทาง TF1 สำหรับ Timer/Counter 1 จะหยุดนับใน Mode นี้

2.2.5 โครงสร้างและการทำงานของพอร์ต

พอร์ตใน MCS-51 ซึ่งมีขนาด 8 บิตรวมทั้งสิ้น 4 พอร์ตแต่ละพอร์ตมีหน้าที่ที่ติดต่อกับวงจรภายนอก โดยสามารถรับ และส่งข้อมูลกับภายนอกได้ คือ ทั้ง 4 พอร์ต จะเป็นพอร์ตแบบสองทิศทาง (bidirection port) โดยโครงสร้างแต่ละพอร์ตจะประกอบด้วย

1.วงจรแลตช์ ซึ่งสถานะของเอาต์พุตของวงจรแลตช์จะปรากฏอยู่ที่รีจิสเตอร์ใช้งานเฉพาะ P0 ถึง P3

2.วงจรเอาร์พุตไดรเวอร์

3.วงจรอินพุทบัฟเฟอร์

เอาต์พุตไดรเวอร์ของพอร์ต 0 และพอร์ต 2 กับอินพุตไดรเวอร์ของพอร์ต 0 จะถูกใช้การติดต่อกับหน่วยความจำสำหรับเก็บโปรแกรมภายในชิพ โดยมีการทำงานดังนี้

พอร์ต	หน้าที่
P1.0 (T2)	ใช้เป็นอินพุตให้เคาน์เตอร์ของไทเมอร์ 2
P1.1 (T2EX)	ใช้เป็นสัญญาณควบคุมการทำงานของไทเมอร์ 2
P3.0 (RXD)	ใช้รับข้อมูลจากภายนอกแบบอนุกรม
P3.1 (TXD)	ใช้ส่งข้อมูลออกไปภายนอกแบบอนุกรม
P3.2 (INT0)	ใช้เป็นอินพุตเพื่อรับสัญญาณอินเตอร์รัปต์ชนิด 0
P3.3 (INT1)	ใช้เป็นอินพุตเพื่อรับสัญญาณอินเตอร์รัปต์ชนิด 1
P3.4 (T0)	สัญญาณอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 0
P3.5 (T1)	สัญญาณอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 1
P3.6 (WR)	ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอกชิพ
P3.7 (RD)	ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิพ

ตารางที่ 2.4 แสดงหน้าที่พิเศษอื่นๆ ของพอร์ต 1 และพอร์ต 3

ก) พอร์ต 0 ใช้ส่งค่าแอดเดรสไบต์ต่ำ (low byte (A0-A7)) ของตำแหน่งหน่วยความจำภายนอก (หน่วยความจำสำหรับเก็บโปรแกรม หรือข้อมูลกับหน่วยความจำภายนอกชิพ)และยังถูกใช้เป็นดาต้าบัส (data bus) ซึ่งจะใช้ทั้งในการรับ และส่งข้อมูลกับหน่วยความจำภายนอก โดยจะผลัดกันใช้คนละเวลาแบบ time multiplex

ข) พอร์ต 2 ใช้ส่งค่าแอดเดรสไบต์สูง (high byte (A8-A7)) ของตำแหน่งหน่วยความจำภายนอก (หน่วยความจำสำหรับเก็บโปรแกรม หรือข้อมูลกับหน่วยความจำภายนอกชิพ) เมื่อมีการใช้แอดเดรสขนาด 16 บิต และถ้าพอร์ต 2 ไม่ถูกใช้งาน (ติดต่อหน่วยความจำภายนอกน้อยกว่า 256 ไบต์) คือถูกใช้ไม่ครบ 16 บิต (ใช้หน่วยความจำภายนอกไม่ครบ 64 กิโลไบต์) พอร์ต 2 บิตที่ไม่ได้ใช้ก็จะส่งภายในรีจิสเตอร์ใช้งานเฉพาะ P2 ออกมาที่แต่ละขา (ซึ่งก็คือค่าที่แลตช์อยู่ภายใน)

ค) พอร์ต 3 ทั้งหมดรวมถึงพอร์ต 1 ของ 8052 อีก 2 ขา จะไม่เพียงแต่ใช้เป็นพอร์ต อินพุต เอาต์พุตเท่านั้น เพราะแต่ละบิตของพอร์ต 3 และ พอร์ต 1 บางบิตยังมีหน้าที่พิเศษอื่นอีกดังแสดงในตารางที่ 2.4

หน้าที่พิเศษเหล่านี้จะใช้ได้ก็ต่อเมื่อ ค่าในบิตแลตช์ของพอร์ตในบิตนั้นๆ มีค่าเป็น “1” มิฉนั้นที่ขาของพอร์ตภายนอกชิพจะมีค่าเป็น 0

ฉ. พอร์ตสื่อสารข้อมูลแบบอนุกรม

พอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51 ประกอบด้วยรีจิสเตอร์ขนาด 8 บิตจำนวนสองตัว แต่ละตัวมีชื่อเรียกดังนี้คือ

-รีจิสเตอร์สำหรับข้อมูล ใช้รับข้อมูลที่ส่งมาจากภายนอก

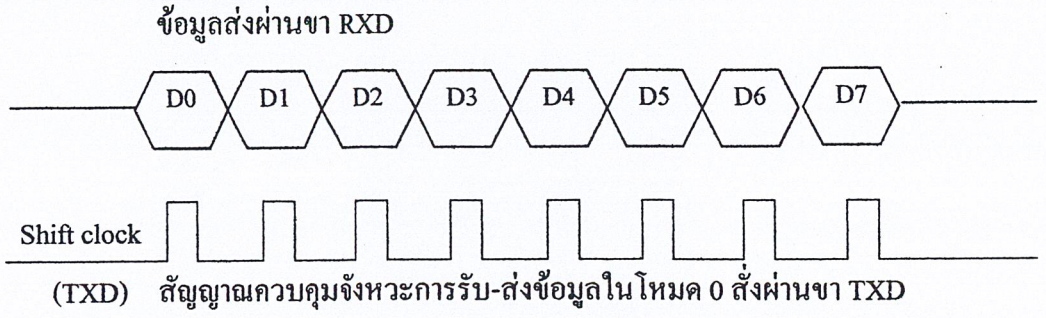
-รีจิสเตอร์สำหรับส่งข้อมูล (transmit register) ใช้ส่งข้อมูลจาก MCS-51 ออกไปภายนอก

รีจิสเตอร์ทั้งสองมีตำแหน่งแอดเดรสเดียวกันในรีจิสเตอร์ใช้งานเฉพาะ คือ จะตรงกับตำแหน่งของ รีจิสเตอร์ใช้งานเฉพาะ SBUF (ตำแหน่ง 99H) ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิพที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ การเข้าถึงข้อมูลในรีจิสเตอร์แต่ละตัว MCS-51 จะทราบเองว่าต้องการติดต่อกับรีจิสเตอร์ตัวใดโดยตรวจสอบจากรหัสคำสั่ง ทั้งนี้เพราะในการเขียนข้อมูลไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ SBUF หมายถึงการโหลดข้อมูลไปที่รีจิสเตอร์สำหรับ ส่งข้อมูลเพื่อส่งข้อมูลออกไปภายนอก ส่วนการอ่านข้อมูลจากรีจิสเตอร์ใช้งานเฉพาะ SBUF จะหมายถึงค่าที่รับเข้ามาได้จากภายนอกที่เก็บไว้ในรีจิสเตอร์สำหรับข้อมูลมาใช้งาน

การใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51 มีความสะดวก และคล่องตัวสูงทั้งนี้เพราะความสามารถกำหนดการทำงานที่แตกต่างกันได้ 4 ประเภท โดยสามารถกำหนดได้จาก ค่าของบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON การใช้งานที่แตกต่างกัน 4 ประเภทนี้มีจุดประสงค์เพื่อความคล่องตัวในการรับ หรือส่งข้อมูลแบบอนุกรมแต่ละประเภทดังนี้

ก) โหมด 0

การทำงานของพอร์ตสื่อสารข้อแบบอนุกรมในโหมด 0 นี้ขา RXD จะใช้สำหรับรับและ ส่งข้อมูล ส่วนขา TXD มีไว้เพื่อสร้างสัญญาณ shift clock เพื่อกำหนดจังหวะทำงานในการรับและส่งข้อมูล ในโหมด 0 การรับและส่งข้อมูลจะเป็นแบบ 8 บิต โดยจะเริ่มส่งจากบิตต่ำสุดก่อน(LSB) อัตราการรับส่งข้อมูล ในการทำงานโหมด 0 จะถูกกำหนดไว้ที่ 1/12 ของความถี่ ออสซิลเลเตอร์ที่ใช้ การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 จะไม่มีบิตเริ่มต้นของข้อมูล (start bit) และบิตสิ้นสุดของข้อมูล (stop bit) เพราะจังหวะการรับ และส่งข้อมูลถูกกำหนดจากสัญญาณ shift clock แล้ว แสดงดังรูปที่ 2.12

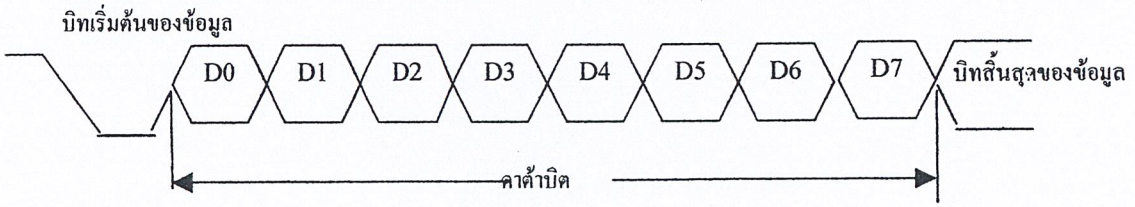


รูปที่ 2.12 แสดงข้อมูลที่รับ และส่งในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0

ข) โหมด 1

การทำงานในโหมดนี้มีการรับ และส่งข้อมูลครั้งละ 10 บิต ข้อมูลจะส่งออกไปภายนอก ผ่านทางขา TXD และรับข้อมูลเข้ามาทางขา RXD ข้อมูลทั้ง 10 บิต ประกอบด้วยบิตเริ่มต้นของข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) และบิตสิ้นสุดของข้อมูลอีก 1 บิต (มีค่าเป็น 1 เสมอ) ในขณะทำการรับข้อมูล ค่าบิตสิ้นสุดของข้อมูลที่รับได้จะไปอยู่ในบิต RB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON แสดงดัง

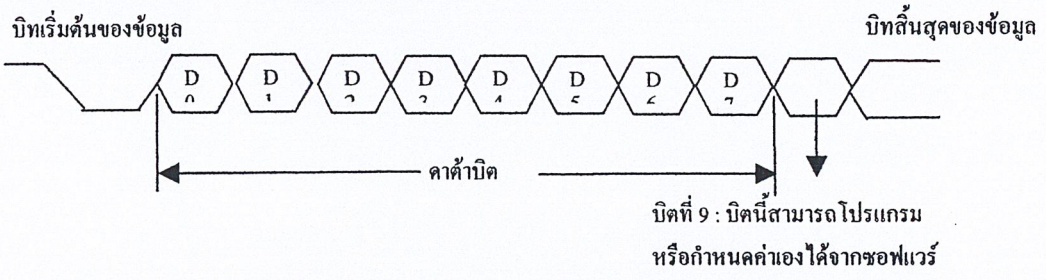
รูปที่ 2.13



รูปที่ 2.13 แสดงข้อมูลที่รับและส่งในการทำงานของพอร์ตสื่อสารข้อแบบอนุกรมโหมด 1

ค) โหมด 2

การทำงานในโหมดนี้จะมีการรับ และส่งข้อมูลที่ละ 11 บิต ข้อมูลจะส่งออกไป ภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาทางขา RXD ข้อมูลที่รับและส่งทั้ง 11 บิต ประกอบไปด้วยบิตเริ่มต้นของข้อมูล (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) ตามด้วย บิตที่ 9 (ต่อจากบิตข้อมูลสุดท้าย) ซึ่งเป็นบิตที่สามารถกำหนดให้มีค่าเป็นศูนย์หรือหนึ่งได้ (programmable 9th data bit) และบิตสิ้นสุดของข้อมูล (มีค่าเป็น 1 เสมอ) ดังนั้นจำนวนบิตที่ รับส่งทั้งหมด 11 บิตจะประกอบไปด้วยบิตต่างๆ ดังรูปที่ 2.14



รูปที่ 2.14 แสดงข้อมูลที่รับและส่งในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด2และ3

ในขณะทำการส่งข้อมูล บิตที่ 9 จะได้จากค่าในบิต TB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON ดังในรูปที่ 2.13 บิตนี้สามารถถูกกำหนดให้มีค่าเป็น 0 หรือ 1 ก็ได้ ส่วนใหญ่ในการใช้งานจะใช้บิตนี้สำหรับตรวจสอบความถูกต้องของข้อมูลที่รับ หรือส่ง (parity bit) โดยจะนำบิต (parity) ในรีจิสเตอร์ PSW ไปไว้ในบิต TB8 ส่วนในขณะรับข้อมูลบิตที่ 9 จะไปปรากฏอยู่มีบิต RB8 ของ รีจิส

เตอร์ SCON โดยไม่สนใจบิตสุดท้ายของข้อมูล ค่าอัตราในการรับ หรือส่งข้อมูลโหมดนี้ถูกกำหนดไว้ที่ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้

ง) โหมด 3 การทำงานในโหมด 3 ข้อมูลทั้ง 11 บิตจะถูกส่งผ่านขา TXD และถูกรับเข้ามาทางขา RXD ข้อมูลทั้ง 11 บิตจะเหมือนกับโหมด 2 ทุกอย่าง แต่แตกต่างกันที่ความเร็วในการรับและส่งข้อมูลผู้ใช้สามารถกำหนดเองได้

จ) รีจิสเตอร์ใช้งานเฉพาะ SCON (SerialPort Control Register) แต่ละบิตในรีจิสเตอร์งานเฉพาะ SCON จะใช้สำหรับควบคุม และตรวจสอบการทำงานของพอร์ตสื่อสารอนุกรมดังนั้นจึงต้องทำความเข้าใจในการทำงานในแต่ละบิตของรีจิสเตอร์ใช้งานเฉพาะ SCON

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

บิต ชื่อบิต

SCON.7 SM0 บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดต่างๆ

SCON.6 SM1 บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดต่างๆ

SM0	SM1	โหมด	
0	0	โหมด 0	ทำงานเป็น shift register อัตราเร็วในการรับข้อมูล หรือส่งข้อมูลเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์
0	1	โหมด 1	8 bit UART อัตราเร็วในการรับ หรือส่งข้อมูลกำหนดเองได้
1	0	โหมด 2	9 bit UART อัตราเร็วในการรับ หรือส่งข้อมูล = 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์
1	1	โหมด 3	9 bit UART อัตราเร็วในการรับ หรือส่งข้อมูลกำหนดเองได้

SCON.5 SM2 บิตการเลือกใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 2 และ 3

SM2

1 ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมในการติดต่อระหว่างซีพียูด้วยกันเองเมื่อข้อมูลที่ 9 ที่ได้ปรับมีค่าเป็น 0 (ดาต้าไบต์) บิต RI จะไม่ถูกเซต แต่หาก ข้อมูลบิตที่ 9 มีค่าเป็น 1 (แอดเดรสไบต์) บิต RI จะถูกเซต

0 ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 และโหมด 3 ตามปกติ

ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 หากบิต SM2 ถูกเซต บิต RI จะไม่ถูกเซตจนกว่าบิตสิ้นสุดของข้อมูลจะถูกรับเข้ามา

ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 บิตนี้ควรถูกเคลียร์ให้เป็น 0

SCON.4 REN บิตควบคุมการอนุญาตให้มีการรับข้อ ดังนี้

REN

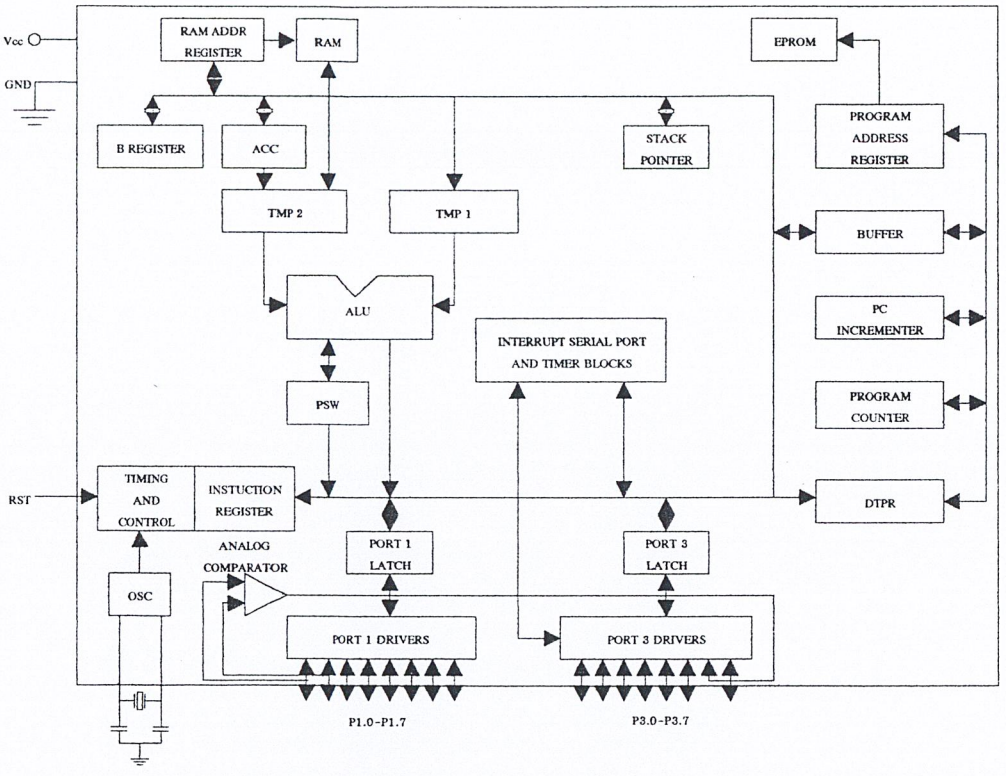
1 อนุญาตให้มีการรับข้อมูลจากภายนอกได้

0 ไม่อนุญาตให้มีการรับข้อมูลจากภายนอกได้

SCON.3	TB8	บิตข้อมูลบิตที่ 9 ซึ่งจะถูกส่งออกไปในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 การเซตหรือเคลียร์กระทำได้ด้วยคำสั่งในโปรแกรมเท่านั้น
SCON.2	RB8	บิตข้อมูลที่ 9 ที่ได้รับเข้ามาจากภายนอกในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 ส่วนในการทำงานโหมด 1 ถ้าบิต SM2=0 บิตนี้จะเป็นบิตสิ้นสุดของข้อมูลที่รับเข้ามาได้ และไม่ถูกกำหนดการใช้งานในโหมด 0
SCON.1	TI	บิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากการส่งข้อมูลถูกเซตโดยฮาร์ดแวร์เมื่อข้อมูลบิตที่ 8 ถูกส่งออกไปแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่นๆ จะถูกเซตโดยฮาร์ดแวร์เมื่อเริ่มส่งบิตสิ้นสุดของข้อมูลออกไปและจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น
SCON.0	RI	บิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากการส่งข้อมูล ถูกเซตโดยฮาร์ดแวร์เมื่อข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 หรือที่ จุดครึ่งทางของช่วงรับบิตสิ้นสุดของข้อมูลในการทำงานโหมดอื่น (มีข้อยกเว้นในกรณีใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมติดต่อระหว่างซีพียูด้วยกันเอง) และจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น

2.2.6ไมโครคอนโทรลเลอร์ AT89C2051

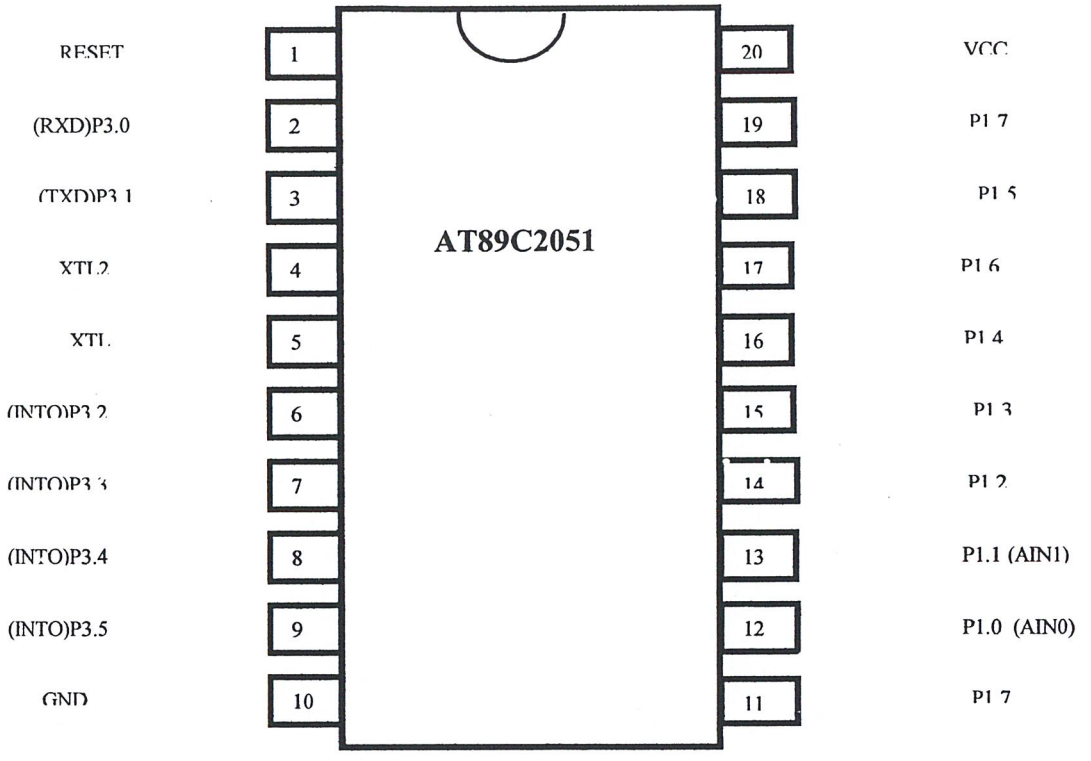
AT89C2051 เป็นไมโครคอนโทรลเลอร์ที่ผลิตขึ้นโดย ATMEL ซึ่งมีโครงสร้างและ ชุด คำสั่งเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 โดยบรรจุอยู่ในตัวถังแบบ DIP ขนาด 20 ขา นับว่าเป็นไมโครคอนโทรลเลอร์ขนาดเล็ก ตัวหนึ่ง ซึ่งมีโครงสร้างที่ง่ายต่อการศึกษาและทำความเข้าใจและยังสามารถทำการพัฒนาโปรแกรมได้โดยไม่ยุ่งยากซับซ้อนมีพอร์ต หน่วยความจำและทรัพยากรที่สำคัญอย่างครบถ้วน เหมาะแก่การนำไปใช้งานในด้านต่างๆได้เป็นอย่างดี



รูปที่ 2.15แสดงโครงสร้างของ CPU เบอร์ AT89C2051

ข้อดีทางเทคนิคของไมโครคอนโทรลเลอร์ AT89C2051

- มีโครงสร้างและ ชุดคำสั่งเหมือนกับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ดังรูป 2.15
- มีหน่วยความจำโปรแกรมชนิด Flash Memory ขนาด 2 Kbyte สามารถโปรแกรมซ้ำได้ 1000 ครั้ง
- มีหน่วยความจำแบบ Ram 8 – Bit ขนาด 128 Byte (Internal RAM)
- ทำงานได้ที่แรงดัน 2.7 – 6 โวลต์
- Run ความเร็วสัญญาณนาฬิกา 0 Hz – 24 MHz
- พอร์ต I/O 15 บิต (พอร์ต 1 และพอร์ต 3)
- พอร์ตสามารถ Sink กระแสได้ 20 mA และใช้เอาต์พุตขับ LED โดยตรง
- มีพอร์ตสื่อสารแบบอนุกรม 1 channel (UART)
- สามารถโปรแกรม ข้อมูลเพื่อป้องกันการอ่านเขียน(คัดลอกโปรแกรม) 2 ระดับ
- มีวงจรไทม์เมอร์และวงจรนับขนาด16 บิต (16 Bit Timer/Counter) จำนวน 2 Channel
- มีสัญญาณอินเตอร์รัพต์ 5 แหล่ง แบ่งระดับความสำคัญได้ 2 ระดับ
- มีวงจรเปรียบเทียบสัญญาณแอนะลอก (Analog Comparator Input) 1 Channel
- มีระบบประหยัดพลังงาน (Low Power Idel and Power Down Modes)



รูปที่ 2.16 แสดงตำแหน่งขาของไมโครคอนโทรลเลอร์ AT89C2051

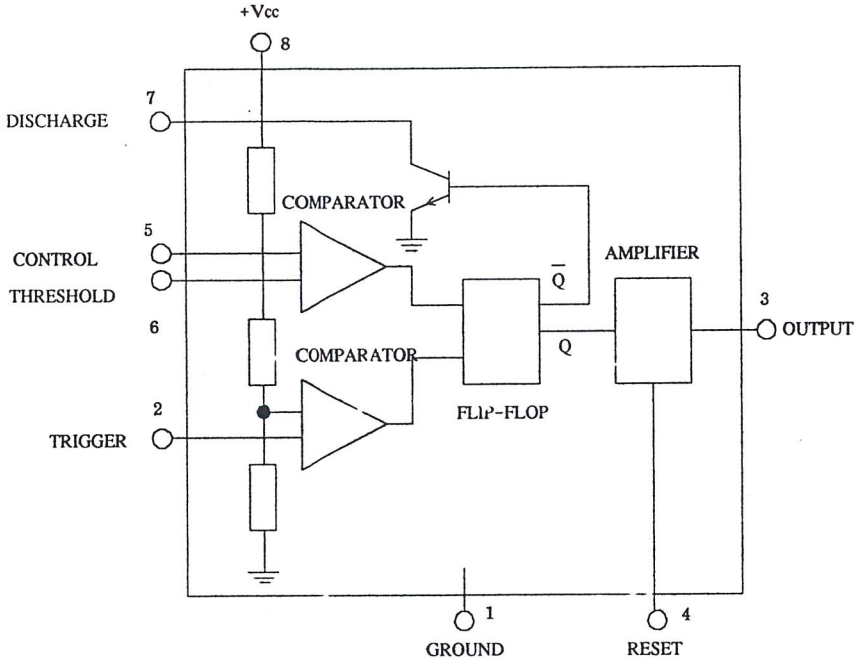
2.3 ทฤษฎีเบื้องต้นของไอซี 555

ไอซี 555 ถูกสร้างขึ้นในปี 1972 โดยถูกกำหนดลักษณะหน้าที่ที่เป็นไอซีตั้งเวลา(timer)สามารถใช้งานได้ 3 โหมดคือโมโนสเตเบิล(monostable),อะสเตเบิล(astable)หรือออสซิลเลเตอร์ และ วงจรหน่วงเวลา (time delay) ไอซี 555 ต้องการไฟเลี้ยงในย่าน 4.5-16 โวลต์

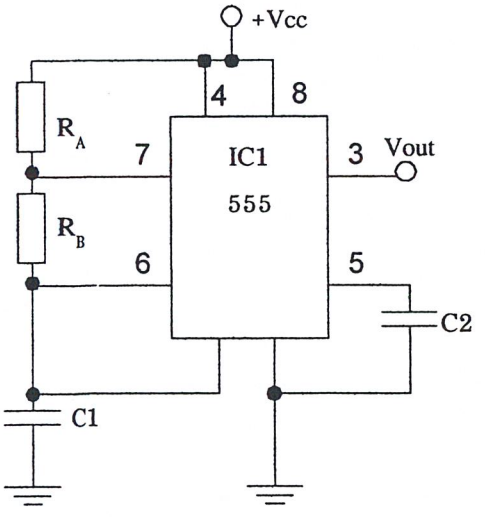
ในรูปเป็นบล็อกไดอะแกรมการทำงานภายในของไอซี 555 ในส่วนอินพุตประกอบด้วย วงจรเปรียบเทียบ 2 ชุด ซึ่งสร้างขึ้นโดยใช้อุปกรณ์จำพวกออปแอมป์ที่มีความเร็วสูงพิเศษต่ออยู่กับตัวต้านทานภายในไอซี เพื่อกำหนดแรงดันอ้างอิงที่จะใช้ในวงจรเปรียบเทียบ(หรือจุดทำงานของ วงจรเปรียบเทียบ) อยู่ที่ระดับ 2/3 Vcc หรือ 0.67 Vcc ในขณะที่ขาทริกเกอร์จะมีระดับอ้างอิงอยู่ที่ 1/3 Vcc ส่วนถัดมาเป็น R-S ฟลิปฟลอปซึ่งจะได้รับสัญญาณให้เซตและ รีเซต จากวงจรเปรียบเทียบ สัญญาณที่ได้ จาก R-S ฟลิปฟลอปจะถูกขยายให้แรงขึ้นก่อนส่งออกไปยังขา 3 หรือเอาต์พุตของ 555 ต่อไป

ที่ขาอินพุตเทรชโฮล (threshold input) จะต่อกับตัวต้านทานและตัวเก็บระจุภายนอก จัดเป็นวงจรคาบเวลา RC (RC timing circuit) ซึ่งตัวต้านทานและตัวเก็บประจุที่ต่อนี้เป็นตัวกำหนดคาบ

เวลาและความถี่ของสัญญาณเอาต์พุต ดังแสดงการต่อตัวต้านทานและตัวเก็บประจุให้แก่ไอซี 55 ตามรูปที่ 2.18



รูปที่ 2.17 แสดงบล็อกไดอะแกรมภายในไอซี 555

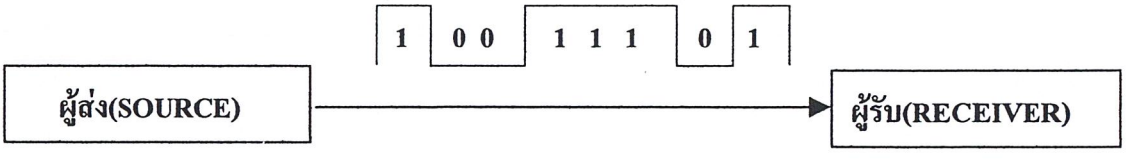


รูปที่ 2.18 แสดงการต่อตัวต้านทานและตัวเก็บประจุเพื่อกำหนดคาบเวลา และความถี่ของสัญญาณเอาต์พุตของไอซี 555

เมื่อเริ่มทำงานตัวเก็บประจุจะทำการประจุแรงดัน เมื่อแรงดันที่ขาอินพุตเทรชโฮลสูงถึง 2/3 ของไฟเลี้ยงไอซีวงจรเปรียบเทียบภายในไอซีจะทำงานส่งผลให้ฟลิปฟลอปเซต เมื่อ ฟลิปฟลอปเซต ทรานซิสเตอร์ดิสชาร์จจะทำงานแล้วทำการคายประจุตัวเก็บประจุถึงตรงจุดนี้ เอาต์พุตของ 555 จะตกลงเป็น "0" และ เมื่อตัวเก็บประจุคายประจุแรงดันที่ขาอินพุตเทรชโฮลจะลดลง เมื่อลดลงถึงระดับ 1/3 ของไฟเลี้ยงวงจรเปรียบเทียบจะหยุดทำงาน ฟลิปฟลอปก็จะ รีเซตทรานซิสเตอร์ดิสชาร์จภายในไอซีก็จะหยุดทำงานด้วย ทำให้เอาต์พุตของไอซีกลับมาเป็น "1" อีกครั้ง

2.4 รูปแบบการรับ-ส่งข้อมูลแบบอนุกรม

รูปแบบการรับ-ส่งข้อมูลในลักษณะนี้ทุกบิตที่เข้ารหัสแทนหนึ่งตัวอักษรจะถูกส่งผ่านไปตามสายส่งเรียงลำดับไปทีละบิตในสายส่งเพียงเส้นเดียวดังตัวอย่างในรูปที่ 2.19 ทำให้ประหยัดค่าใช้จ่ายในเรื่องสายส่งสัญญาณได้มากกว่าการรับ-ส่งข้อมูลแบบขนาน ดังนั้นการรับ-ส่งข้อมูลแบบอนุกรมจึงนิยมใช้กันมากในการรับ-ส่งข้อมูลในระยะไกล แต่ในด้านความเร็วการรับ-ส่ง ข้อมูลแบบอนุกรมจะทำได้ช้ากว่าการรับ-ส่งข้อมูลแบบขนานมาก



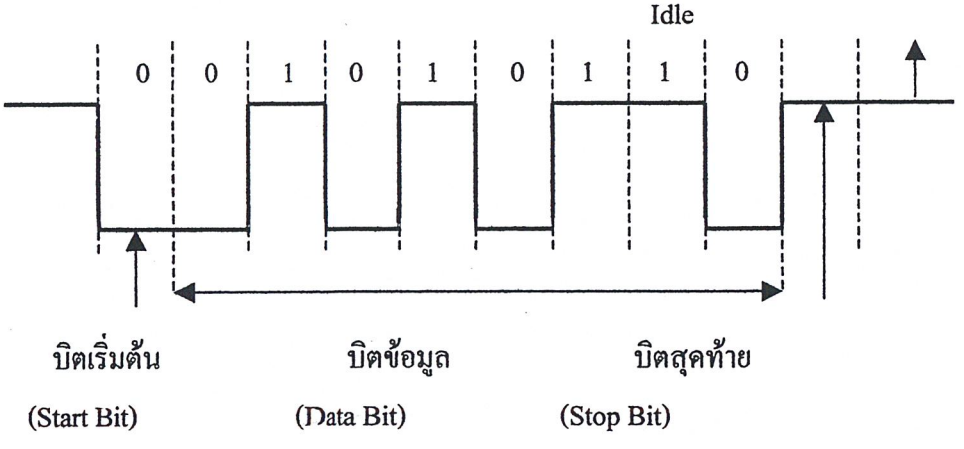
รูปที่ 2.19 การรับ-ส่งข้อมูลแบบอนุกรม

จากรูปตัวอักษรหนึ่งตัวอักษรจะประกอบด้วยบิต 8 บิตจะถูกส่งออกไปเรียงเป็นลำดับ และทางด้านรับจะรวบรวมบิตเหล่านี้ทีละบิตจนครบ 8 บิต ซึ่งถือเป็น 1 ตัวอักษร

2.4.1 การรับ-ส่งข้อมูลแบบอะซิงโครนัส (Asynchronous Transmission)

การรับ-ส่งข้อมูลแบบอะซิงโครนัส เป็นแบบในการส่งข้อมูลอักษรตัวใดตัวหนึ่งไปในทันที ทันใดซึ่งเป็นเวลาใดๆ ก็ได้โดยไม่จำเป็นต้องกำหนดความสัมพันธ์กับตัวอักษรตัวอื่นๆ อาจส่ง ตัวอักษรเรียงกันไปเลยโดยไม่มีที่ว่างระหว่างตัวอักษรก็ได้ กรณีเช่นนี้เครื่องรับปลายทางจะต้องสร้างลักษณะของซิงโครนัสขึ้นมาใหม่สำหรับตัวอักษรแต่ละตัวเครื่องรับปลายทางจะต้องรู้จักว่า ตัวอักษรที่ส่งมานั้นมีบิตแรกของตัวอักษรอยู่ที่ใด ดังนั้นหน้าตัวอักษรแต่ละตัวจะใส่บิตลักษณะพิเศษเพื่อถือเป็นจุดเริ่มต้น (Start Bit) ซึ่งเมื่อเครื่องรับพบบิตแบบนี้จะทราบได้ทันทีว่าคือ จุดเริ่มต้นของตัวอักษร

เครื่องรับสัญญาณปลายทางจะตรวจสอบพบบิตแสดงจุดเริ่มต้น (Start Bit) ได้โดยตรวจสอบลอจิกทางไฟฟ้าของสายส่งนั่นเอง คล้ายกับตัวเราเป็นผู้รับสัญญาณแล้วดูที่สายส่งว่าขณะนั้นๆ มีลอจิกเป็น "1" เป็น "0" ในขณะที่สายส่งมีลอจิกคงที่โดยไม่มีการเปลี่ยนแปลง นั่นแสดงว่าไม่มีการส่งข้อมูลมาเลย สภาวะของสายส่งในลักษณะนี้เรียกว่า Idle Line และเพื่อความสะดวกจะให้สภาวะของ Idle Line มีลอจิกทางไฟฟ้าเป็น "1" ในบางครั้งอาจเรียกสภาวะ Idle Line นี้ว่า Mark Condition สำหรับสภาวะที่ตรงข้ามกับสภาวะ Idle Line คือมี ลอจิก "0" และใน บางครั้งจะเรียกสภาวะนี้ว่า Space Condition หรือสภาวะว่างเปล่า หรืออาจเรียกอีกอย่างหนึ่งว่าสภาวะสายเปิด (Open Line) จะสังเกตได้ว่าสภาวะ Idle Line นี้เป็นสภาวะที่ตัวส่งสัญญาณส่ง ค่าบิตเป็น "1" มาติดกันโดยตลอด เมื่อต้องการส่งตัวอักษร เครื่องส่งสัญญาณ จะใส่บิตข้อมูลหนึ่งบิตมานำหน้ากลุ่มของบิตที่แทนตัวอักษรนั้น ซึ่งบิตที่ใส่นำหน้านี้จะทำหน้าที่เป็นบิตนำหน้าเพื่อบ่งบอกว่าเป็นจุดเริ่มต้นของการส่งตัวอักษร ถ้าพิจารณาในแง่ของลักษณะสัญญาณทางไฟฟ้าจะเห็นว่า เครื่องส่งสัญญาณจะสร้างบิตเริ่มต้นโดยการสับเปลี่ยนลอจิกของสายส่งจากการรักษาระดับไฟฟ้าคงที่ ซึ่งถือเป็น "1" เข้าสู่สภาวะ "0" โดยใช้เพียงบิตเดียว พร้อมกันนี้ข้อความที่ต้องการส่งก็จะตามบิตเริ่มต้นเข้าสู่เครื่องรับต่อไป



รูปที่ 2.20 รูปแบบการรับ-ส่งข้อมูลแบบอะซิงโครนัส

เมื่อเครื่องรับสัญญาณตรวจพบว่ามีการเปลี่ยนแปลงจากลอจิก "1" ไปเป็นลอจิก "0" ก็จะมีการเปรียบเทียบสัญญาณนาฬิกาทันที ภายหลังจากการผ่านพ้นไปได้ครึ่งบิตสัญญาณนาฬิกาจะเป็นตัวบอกเครื่องรับให้ตรวจสอบสภาวะของสายส่ง ถ้าตรวจสภาวะของสายส่งแล้วยังคงอยู่ในสภาวะ "0" เครื่องก็จะเริ่มตรวจสอบข้อมูลนั้น ถ้าในจุดเริ่มต้นที่มีการเปลี่ยนแปลงจากลอจิก "1" เป็น "0" นี้เป็นสัญญาณรบกวน (Noise) ช่วงสั้นๆ และก็จะหยุดตรวจสภาวะสายส่งเพราะไม่ต้องการรวมบิตเป็นบิตข้อมูลอีกต่อไป

สัญญาณนาฬิกาที่เครื่องรับจะทำให้เกิดการซิงโครนัสที่ตอนต้นของตัวอักษรที่รับเข้ามาอาจยอมให้เกิดการแตกต่างกันไปได้เล็กน้อยระหว่างความเร็วในการส่งข้อมูลกับสัญญาณนาฬิกาที่เครื่องรับใช้ ที่จุดสิ้นสุดของแต่ละตัวอักษรที่ส่งมาจะถูกส่งมายังเครื่องรับเพื่อทำให้เกิดสภาวะคงที่ก่อนที่ตัวอักษรอื่นจะตามมา กลุ่มบิตข้อมูลจะถูกปิดหัวเปิดท้ายไว้ด้วยบิตเริ่มต้นและบิตสิ้นสุดในบางครั้งจึงเรียกการรับ-ส่งข้อมูลแบบอะซิงโครนัสนี้ว่า การรับ-ส่งข้อมูลแบบเริ่มต้น/หยุด(Start/Stop Transmission)

สัญญาณหยุดหรือบิตสิ้นสุดจะมีสภาวะของสายเป็นลอจิก "1" สำหรับการรับ-ส่งข้อมูลที่มีรูปแบบเป็นรหัสแอสกี้ (ASCII) อาจใช้สัญญาณหยุดหรือบิตสิ้นสุดด้วยความยาวเป็น 1 บิตหรือ 2 บิตก็ได้ และสำหรับรหัสบอโด (BAUDO) บิตสิ้นสุดมักจะมีความยาวเท่ากับ 1.5 บิต แต่ อย่างไรก็ตามอาจมีบางกรณีใช้บิตสิ้นสุดที่มีความยาว 1.42 บิต จุดประสงค์ดั้งเดิมของสัญญาณหยุดที่ใช้ในยุคแรกใช้เพื่อบอกจุดสิ้นสุดแก่อุปกรณ์ปลายทางซึ่งเป็นอุปกรณ์ประเภทเครื่องจักร ไฟฟ้า(ไม่ใช่อุปกรณ์อิเล็กทรอนิกส์ล้วนดังปัจจุบัน) โดยใช้ตัวอักษรหนึ่งๆ ส่งมายังอุปกรณ์ เครื่องรับปลายทางจะรับข้อมูลที่ส่งเข้ามาเป็นบิตและระหว่างช่วงเวลาที่เกิดสัญญาณหยุดหรือบิตสิ้นสุดนี้เครื่องจะต้องตีความหมายด้วยว่าจะต้องทำอย่างไรกับข้อมูลเหล่านั้น (ดังเช่น ต้องเจาะรูบนเทปกระดาษ หรือ ต้องพิมพ์ข้อมูลที่รับเข้ามานั้น)

ด้วยเหตุนี้ความยาวของบิตสิ้นสุดจึงมีผลในการยืดเวลาให้เครื่องรับปลายทางด้วยว่า จะกระทำอย่างไรกับข้อมูลที่รับเข้ามาแล้วจึงจะสามารถรับข้อมูลชุดต่อไปได้ ในยุคแรกๆ อุปกรณ์รับข้อมูลปลายทางที่ใช้รูปแบบแทนตัวอักษรด้วยรหัสแอสกี้ต้องใช้บิตสิ้นสุดที่มีความยาวถึง 2 บิตจึงจะทำให้การทำงานของเครื่องเสร็จและพร้อมที่จะรับตัวอักษรตัวต่อไป แต่ในยุคปัจจุบันนี้ความเจริญทางเทคโนโลยีช่วยให้อุปกรณ์เหล่านี้มีน้ำหนักเบาและมีความเร็วในการทำงานสูงขึ้นมาก ทำให้เครื่องรับสัญญาณปลายทางตีความหมายของตัวอักษรพร้อมทั้งทำงานอื่นที่ต้องกระทำกับตัวอักษรนั้นให้เสร็จภายในระยะเวลาเพียง 1 บิตได้ ดังนั้นในปัจจุบันนี้การรับ-ส่งข้อมูลในรูปแบบของรหัสแอสกี้จึงใช้บิตสิ้นสุดเพียง 1 บิตเท่านั้น

บิตพาริตี้ใช้ในการบอกให้ส่วนรับข้อมูลทราบว่าข้อมูลที่รับเข้ามาผิดหรือไม่ บิตพาริตี้นี้จะถูกส่งออกมาพร้อมกับบิตข้อมูล ซึ่งบิตนี้จะเป็น "0" เป็น "1" นั้นขึ้นอยู่กับข้อมูลที่ส่งออกมามีจำนวนของบิตที่เป็น "1" เป็นจำนวนคู่ หรือ จำนวนคี่ และยังขึ้นอยู่กับอุปกรณ์รับ-ส่งข้อมูลด้วยว่าถูกออกแบบหรือโปรแกรมไว้ให้รับส่งบิตพาริตี้ในลักษณะของพาริตี้คู่หรือคี่อีกด้วย ในกรณีที่อุปกรณ์รับ-ส่งถูกออกแบบไว้ให้เป็นพาริตี้คู่ อุปกรณ์ส่งข้อมูลจะทำการส่งพาริตี้บิต เป็นลอจิก "1" ออกไป เมื่อเป็นจำนวนบิตที่เป็น "1" ของข้อมูลเป็นจำนวนคี่ และจะทำการส่งพาริตี้บิตเป็นลอจิก "0" เมื่อจำนวนบิตที่เป็น "1" ของข้อมูลเป็นจำนวนคู่ ในทางกลับกันเมื่ออุปกรณ์ รับ-ส่งถูกออกแบบไว้ให้เป็นพาริตี้คี่ พาริตี้บิตจะเป็น"1" ในกรณีที่จำนวนบิตที่เป็น "1" ของ ข้อมูลเป็นจำนวนคู่ และจะเป็น

"0" ในกรณีที่เป็นจำนวนคี่ เมื่อส่วนรับข้อมูลรับข้อมูลจากส่วนส่งข้อมูลไปแล้วจะทำการตรวจสอบบิตที่เป็น "1" ว่ามีจำนวนเป็นคู่หรือคี่ ถูกต้องตามที่ส่วนส่งส่งมาหรือไม่ หากตรวจสอบแล้วพาริตี้บิตไม่เป็นไปตามเงื่อนไขแล้วแสดงว่าข้อมูลที่รับเข้ามามีความผิดพลาดเกิดขึ้น

สิ่งสำคัญอีกสิ่งหนึ่งคือถ้าอุปกรณ์ส่งข้อมูลทำการส่งข้อมูลในลักษณะพาริตี้คู่หรือคี่ก็ตาม อุปกรณ์รับข้อมูลก็จะต้องการรับในลักษณะพาริตี้เดียวกันกับอุปกรณ์ส่งข้อมูลด้วย เช่น ในกรณีที่อุปกรณ์ส่งข้อมูลในลักษณะพาริตี้คู่ อุปกรณ์รับส่งข้อมูลก็จะต้องรับข้อมูลในลักษณะพาริตี้คู่ด้วยเช่นกัน

2.5 พอร์ตอนุกรม RS-232C

โดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตแบบอนุกรม เรียกชื่อกันว่า RS-232C อยู่ในตัวอยู่แล้ว แต่บางเครื่องอาจจะไม่มีก็เป็นได้ ตัวอย่างเช่น IBM PC จำเป็นจะต้องมีการ์ดที่เรียกว่า อะซิงโครนัสอะแดปเตอร์ (Asynchronous Adapter) มาเสียบใส่ พอร์ต RS-232C นี้ทำหน้าที่รับและส่งข้อมูลแบบอนุกรม เรียกว่า Universal Asynchronous Adapter เหตุที่มีชื่อเรียกว่า RS-232C ก็เนื่องจากสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์แห่งสหรัฐอเมริกา (Electronic Industries Association : EIA) ได้กำหนดมาตรฐานของอุปกรณ์การสื่อสารแบบอนุกรมเอาไว้ ภายใต้ชื่อว่า RS-232C ความจริงมาตรฐานของการส่งข้อมูลแบบอนุกรมมีหลายมาตรฐาน แต่ที่นิยมกันมากที่สุดสำหรับไมโครคอมพิวเตอร์ก็คือ RS-232C หน้าที่สำคัญของการสื่อสาร แบบอะซิงโครนัส

<u>รับสัญญาณ</u>

1. เปลี่ยนสัญญาณที่เข้ามาอนุกรมให้เป็นแบบขนาน
2. ตรวจสอบความผิดพลาดของสัญญาณที่รับเข้ามา
3. ตัดบิตสิ้นสุดและพาริตี้ออก
3. ส่งสัญญาณให้ซีพียูรู้ว่าได้รับสัญญาณเรียบร้อยแล้ว

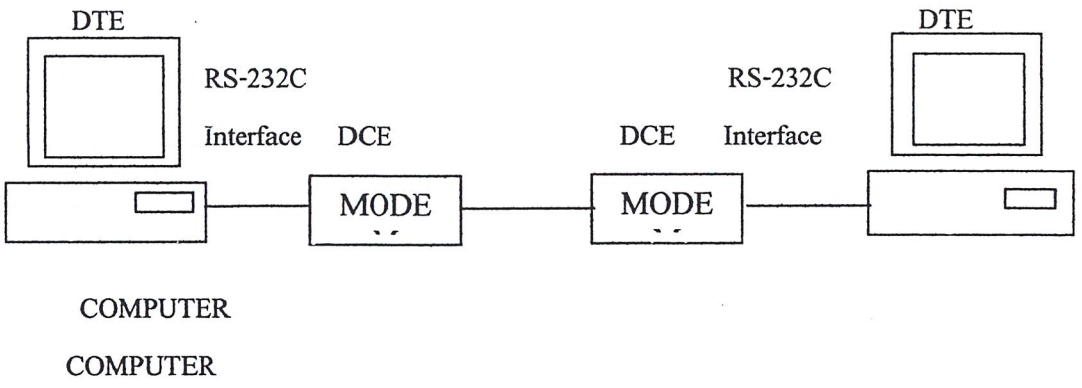
<u>ส่งสัญญาณ</u>

1. เปลี่ยนสัญญาณแบบขนานจากซีพียูแล้วทยอยส่งออกเป็นแบบอนุกรม
2. เพิ่มบิตสิ้นสุดและพาริตี้บิต
3. เพิ่มสัญญาณควบคุมโมเด็มที่เชื่อมต่ออยู่ (ถ้ามี)

2.5.1 มาตรฐานของ RS-232C

มาตรฐาน RS-232C ได้จัดทำเพิ่มขึ้นเมื่อปี ค.ศ.1969 โดยสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์แห่งสหรัฐอเมริกา RS ย่อมาจาก Recommended Standard ส่วน 232 เป็น หมายเลขบ่งบอกถึงมาตรฐานตัวนี้ C เป็นหมายเลขของฉบับท้ายสุดของมาตรฐานตัวนี้ จุดประสงค์ของมาตรฐานตัวนี้ก็เพื่อบรรยายคุณลักษณะของการเชื่อมต่ออุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment :

DTE) กับอุปกรณ์สื่อสารข้อมูล (Data Communication Equipment : DCE) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE ก็จะหมายถึงไมโครคอมพิวเตอร์ส่วน DCE ก็จะหมายถึงโมเด็มหรืออุปกรณ์อื่นๆ เช่น เครื่องพิมพ์ที่รับสัญญาณแบบอนุกรมอาจเป็นได้ทั้ง DTE และ DCE ขึ้นอยู่กับผู้ผลิต ข้อแตกต่างของ DTC เป็น DCE จะเป็นได้จากรูปต่อไปนี้



รูปที่ 2.21 อุปกรณ์ DTE และ DCE

คุณสมบัติอีกประการหนึ่งของ RS-232C คือ ความเร็วและระยะทางของการรับ-ส่งข้อมูล RS-232C สามารถเชื่อมต่อการรับ-ส่งข้อมูลได้ด้วยอัตราความเร็ว 0-20,000 บิตต่อวินาที ซึ่งเพียงพอกับการเชื่อมต่อกับไมโครคอมพิวเตอร์ที่มีอัตราเร็ว 110 ถึง 9,600 บิตต่อวินาที ความยาวของสายเชื่อมต่อสัญญาณตามมาตรฐานของ RS-232C จำกัดอยู่แค่ความยาว 50 ฟุต ซึ่งก็เพียงพอสำหรับการสื่อสารของไมโครคอมพิวเตอร์กับอุปกรณ์ต่อพ่วงรอบนอก

2.5.2 ลักษณะของสัญญาณ RS-232C

เพื่อเป็นหลักประกันว่าข้อมูลถูกส่งออกไปอย่างถูกต้องและอุปกรณ์ถูกควบคุมอย่างถูกต้องจำเป็นจะต้องมีข้อตกลงกันในเรื่องของสัญญาณที่ใช้มาตรฐาน RS-232C กำหนดย่านของแรงดันไฟฟ้าในสัญญาณเพื่อสนองจุดประสงค์ดังกล่าว สามารถแสดงได้ดังนี้

แรงดันไฟฟ้า	สถานะภาพลอจิก	สถานภาพสัญญาณ	ฟังชั่นการควบคุม
3 ถึง 25 V	“0”	สเปซ	ON
-25 ถึง-3 V	“1”	มาร์ค	OFF

ตารางที่ 2.5 ย่านของแรงดันไฟฟ้าในสัญญาณเพื่อตอบสนอง

หมายเหตุ ช่วงของระดับแรงดันระหว่าง –3V ถึง +3V จะเป็นช่วงของการเปลี่ยนแปลงลอจิกดังนั้นจึงไม่มีการระบุสถานะของสัญญาณในช่วงนี้ คอมพิวเตอร์บางเครื่องใช้แต่สัญญาณลอจิกที่ออกมาเป็นสัญญาณของ RS-232C เลย ตัวอย่างเช่น อะซิงโครนัสอะแดปเตอร์ของ IBM PC ในกรณีเช่นนี้ระยะความยาวของสายที่ใช้เชื่อมต่ออาจจะสั้นกว่า 50 ฟุต เนื่องจากระดับของกราวด์เปลี่ยนไปอันเป็นผลจากการสูญเสียของแรงดันในความต้านทานของสาย

2.5.3 คอนเน็คเตอร์ของ RS-232C

ในปัจจุบันเรามักใช้คอนเน็คเตอร์แบบ DB-25 ในการอินเตอร์เฟสตามมาตรฐาน RS-232C คอนเน็คเตอร์แบบนี้เทียบเท่าแบบมาตรฐาน ISO2113 ซึ่งเป็นมาตรฐานที่ประกาศใช้โดย Internationnal Organization for Standardzation (ISO) คอนเน็คเตอร์ตัวผู้ใช้จะใช้กับอุปกรณ์ที่เป็น DTE ส่วนคอนเน็คเตอร์ตัวเมียจะใช้กับอุปกรณ์ที่เป็น DCE

2.5.4 ขาสัญญาณต่างๆ ของคอนเน็คเตอร์แบบ DB-9

ขา 1 Carrier detect โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออน (ลอจิก “0”) ไปบอก ไมโครคอมพิวเตอร์ เมื่อได้รับสัญญาณจากโมเด็มของอีกฝ่ายหนึ่ง สัญญาณนี้จะนำไปจุด LED บอกว่าได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่งแล้วไฟ LED จะอยู่บนหน้าปัดของโมเด็มเอง

ขา 2 Receive data DTE จะรับสัญญาณจาก DCE เข้ามาทางขานี้ เมื่อไม่มีสัญญาณส่งเข้ามาสถานะของลอจิกที่ขานี้จะเป็น “1”

ขา 3 Transmit data สัญญาณของขานี้จะถูกส่งจาก DTE ไปยัง DTE หรือ DCE ตัวอื่น เช่น ไมโครคอมพิวเตอร์ตัวอื่น เครื่องพิมพ์ หรือโมเด็ม เมื่อไม่มีสัญญาณส่งออก สถานะของลอจิกที่ ขานี้จะเป็น “1”

ขา 4 Data terminal ready อุปกรณ์ DTE จะเปิดสัญญาณสายนี้ให้ออน (ลอจิก “0”) เมื่อพร้อมที่จะติดต่อกับอุปกรณ์ DCE โดยส่วนมากอุปกรณ์ DCE จะไม่รายงานสถานภาพของตัวเอง เช่น CD DSR และ CTS ให้อุปกรณ์ DTE รู้ ดังนั้นหากต้องการให้อุปกรณ์ DCE รายงาน สถานภาพตัวเองต้องเปิดสัญญาณ DTR ให้ออน

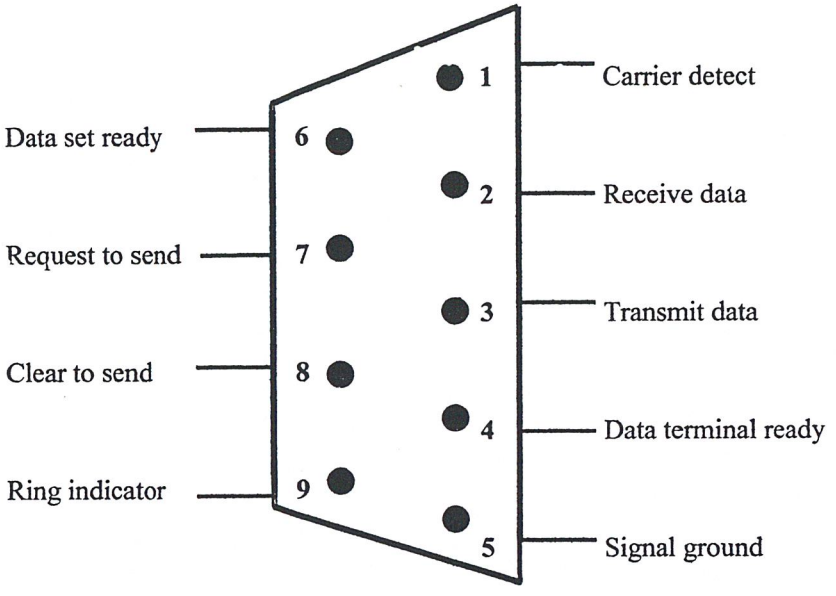
ขา 5 Signal ground ทำหน้าที่เป็นระดับแรงดันอ้างอิง สำหรับสายสัญญาณทุกๆ สายจะมีแรงดันเป็น 0 โวลต์

ขา 6 Data set ready เมื่อสัญญาณขานี้อยู่ในสถานะออน (ลอจิก “0”) เป็นการบอก DTE ว่า DCE ต่อเรียบร้อยแล้วและพร้อมที่จะส่งข้อมูลได้แล้ว

ขา 7 Request to send ใช้สำหรับส่งสัญญาณจาก DTE ไป DCE เพื่อเป็นการร้องขอที่จะส่งสัญญาณออกทางขา 2 สัญญาณนี้จะใช้ควบคู่กับ CTS หรือ Clear to send หากอุปกรณ์ DCE ได้รับ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับข้อมูลได้หรือยัง หากพร้อมที่จะรับข้อมูลแล้วจะส่งสัญญาณไปบอกขา CTS

ขา 8 Clear to send เมื่อสัญญาณนี้อยู่ในสถาวะออฟ (ลอจิก “1”) หมายความว่าอุปกรณ์รับกำลังบอกว่าพร้อมที่จะรับข้อมูลแล้ว

ขา 9 Ring indicator สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบโต้ตอบอัตโนมัติ (Auto-Answer) สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งเข้ามา และออฟระหว่างเสียงดังของกระดิ่ง



รูปที่ 2.22 คอนเน็คเตอร์ของ RS-232C แบบ DB-9

2.6 โปรแกรมวิชวลเบสิก

เป็นที่ยอมรับในวงการกันดีว่าตั้งแต่วินโดวส์ 95 และวินโดวส์ NT เป็นต้นมา วินโดวส์นั้นได้ช่วยให้คอมพิวเตอร์มีประสิทธิภาพในการทำงานสูงกว่าดอส ดังนั้นในคอมพิวเตอร์ที่ใช้วินโดวส์ ถ้าเขียนโปรแกรมสำหรับดอสเพื่อรันในดอสของวินโดวส์ก็เท่ากับไม่ได้ใช้วินโดวส์ที่มีอยู่เลย เพราะโปรแกรมจะลดความสามารถของวินโดวส์ให้เหลือเท่ากับคอมพิวเตอร์ที่ใช้ดอสเท่านั้นเพราะฉะนั้นสำหรับคอมพิวเตอร์ที่ใช้วินโดวส์ การเขียนโปรแกรมเพื่อรันในวินโดวส์จึงเป็นทางเดียวที่จะทำให้เกิดประโยชน์สูงสุด 2 ประการคือ

1. โปรแกรมสำหรับวินโดวส์จะนำความสามารถของวินโดวส์มาใช้ได้อย่างเต็มที่
2. การเขียนโปรแกรมสำหรับวินโดวส์ทำได้ง่ายกว่าการเขียนโปรแกรมสำหรับดอส

โปรแกรมหรือแอปพลิเคชั่นในวินโดวส์นั้นได้มีวิธีการเขียน 2 แบบ คือ การเขียนโปรแกรมแบบวิชวลหรือวิชวลโปรแกรมมิ่ง (Visual Programming) และการเขียนโปรแกรมแบบนอนวิชวลหรือนอนวิชวลโปรแกรมมิ่ง (None Visual Programming) "วิชวล หมายถึง มองเห็น การเขียนโปรแกรม หมายความว่า การเขียนแบบนี้เราจะมองเห็นส่วนประกอบต่างๆ ของโปรแกรมนั้น ทันทีในขณะที่กำลังเขียนโปรแกรม" ส่วนการเขียนโปรแกรมแบบนอนวิชวลเป็นการเขียนโปรแกรมด้วยอักษรและเครื่องหมาย ซึ่งยังจะไม่เห็นส่วนประกอบของโปรแกรมในทันทีแต่สามารถจะมองเห็นส่วนประกอบต่างๆ ของโปรแกรมได้ในภายหลังคือในตอนรันโปรแกรม ดังนั้นการเขียนโปรแกรมแบบวิชวลจึงช่วยให้ผู้เขียนโปรแกรมสามารถเขียนได้สะดวกและรวดเร็วกว่าในการทำงาน ด้านโปรแกรมมิ่งนั้นนับตั้งแต่เริ่มต้นในการพัฒนาโปรแกรมต่างๆ นั้นจะมีโปรแกรมภาษาต่างๆ มากมายที่ใช้ในการพัฒนางานหรือสร้างเป็นโปรแกรมประยุกต์ขึ้นมาใช้งานสำหรับการทำงานในองค์กรหรือการพัฒนาขึ้นมาใช้งานส่วนตัว แต่โดยส่วนใหญ่แล้วโปรแกรมเหล่านี้จะเป็นโปรแกรมที่ต้องใช้ความจำเป็นเลิศเกือบทั้งสิ้น เนื่องจากโปรแกรมเหล่านั้นโดยส่วนใหญ่จะเป็นโปรแกรมประเภทการเขียนโดยการใช้คำสั่งเฉพาะต่างๆ ในการสร้างหรือควบคุมการทำงานของส่วนต่างๆ ที่ต้องการโดยเรียกการใช้งานโปรแกรมเหล่านี้ว่า "การโค๊ดดิ้ง" แต่สำหรับโปรแกรม Visual Basic 6.0 นั้นโปรแกรมพัฒนาที่มีการผสมผสานกันระหว่างการโค๊ดดิ้งและ จับวาง (แดรกแอนด์ดรอป) นั่นก็คือในการทำงานนั้นสามารถที่จะกำหนดหรือสร้าง Object ต่างๆ โดยการใช้เครื่องมือต่างๆ ที่ตัวโปรแกรมมีมาให้โดยไม่ต้องเขียนคำสั่งเพื่อสร้าง Object ต่างๆ เหล่านั้นขึ้นมาใช้งานและยังสามารถที่จะเขียนคำสั่งเพื่อใช้ในการสร้างเงื่อนไขพิเศษอื่นที่ใช้ในการทำงานได้อีกด้วย

2.6.1 การเปิดโปรแกรมขึ้นมาใช้งาน

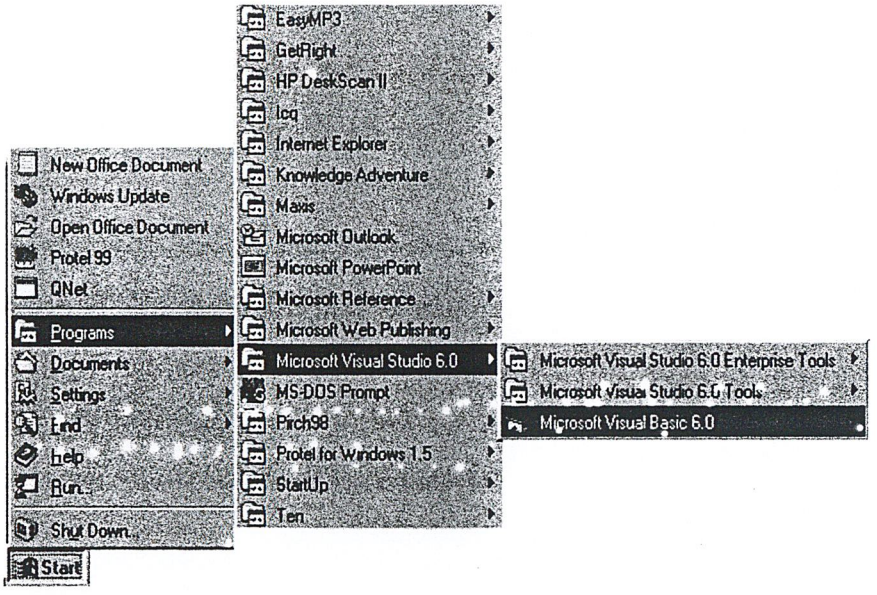
ในการเปิดโปรแกรมขึ้นมาใช้งานนั้นสามารถที่จะทำได้โดยวิธีการดังต่อไปนี้

1. คลิกปุ่ม 

2. เลือกเมนู Programes

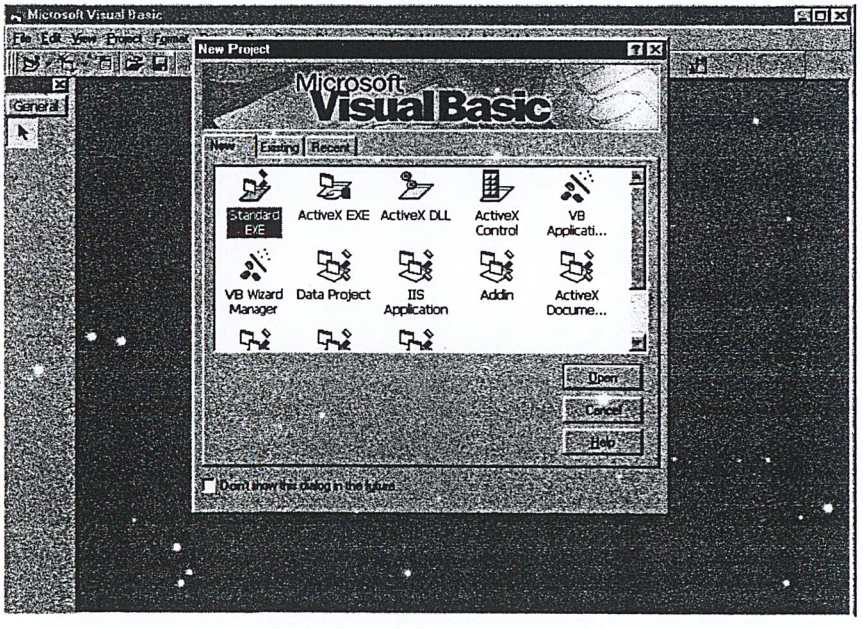
3. เลือกเมนูย่อย Microsoft Visual Studio 6.0

4. คลิกไอคอน Microsoft Visual Basic 6.0



รูปที่ 2.23 ขั้นตอนเปิดโปรแกรม Microsoft Visual Basic 6.0

เมื่อได้เปิดโปรแกรม Microsoft Visual Basic 6.0 ขึ้นมาแล้วนั้นจะปรากฏดังรูป และสำหรับส่วนประกอบในการทำงานต่างๆ ที่ใช้ในการทำงานนั้นจะได้



รูปที่ 2.24 เมื่อเริ่มเปิดโปรแกรม Microsoft Visual Basic 6.0

จากรูปนั้นจะเห็นว่าจะปรากฏไดอะล็อกบ๊อกซ์ New Project ขึ้นมาก่อนที่จะเข้าไปใช้งานโปรแกรมซึ่งในไดอะล็อกบ๊อกซ์ New Project นั้นจะประกอบด้วย 3 ส่วนด้วยกันคือ

New

ส่วนนี้เป็นส่วนที่ใช้ในการเลือกสร้างงานใหม่ขึ้นมา ซึ่งจะมีรูปแบบของการสร้างงานให้เลือกใช้หลากหลายรูปแบบด้วยกัน

Existing

ส่วนนี้จะเป็นส่วนที่ใช้ในการเปิดงานที่มีอยู่แล้วซึ่งเก็บไว้ในที่ต่างๆ ที่ต้องการเปิดขึ้นมาใช้งานในส่วนนี้สามารถที่จะเปิดงานที่ได้สร้างขึ้นจากโปรแกรม Visual Basic ในเวอร์ชั่นต่างๆ ได้ตามต้องการ

Recent

ในส่วนนี้จะเป็นการเลือกเปิดไฟล์ที่ได้เคยเปิดขึ้นมาใช้งานแล้วและปิดลงไป ซึ่งในส่วนนี้จะปรากฏชื่อไฟล์ต่างๆ ที่ได้เคยเปิดขึ้นมาใช้งานแล้ว

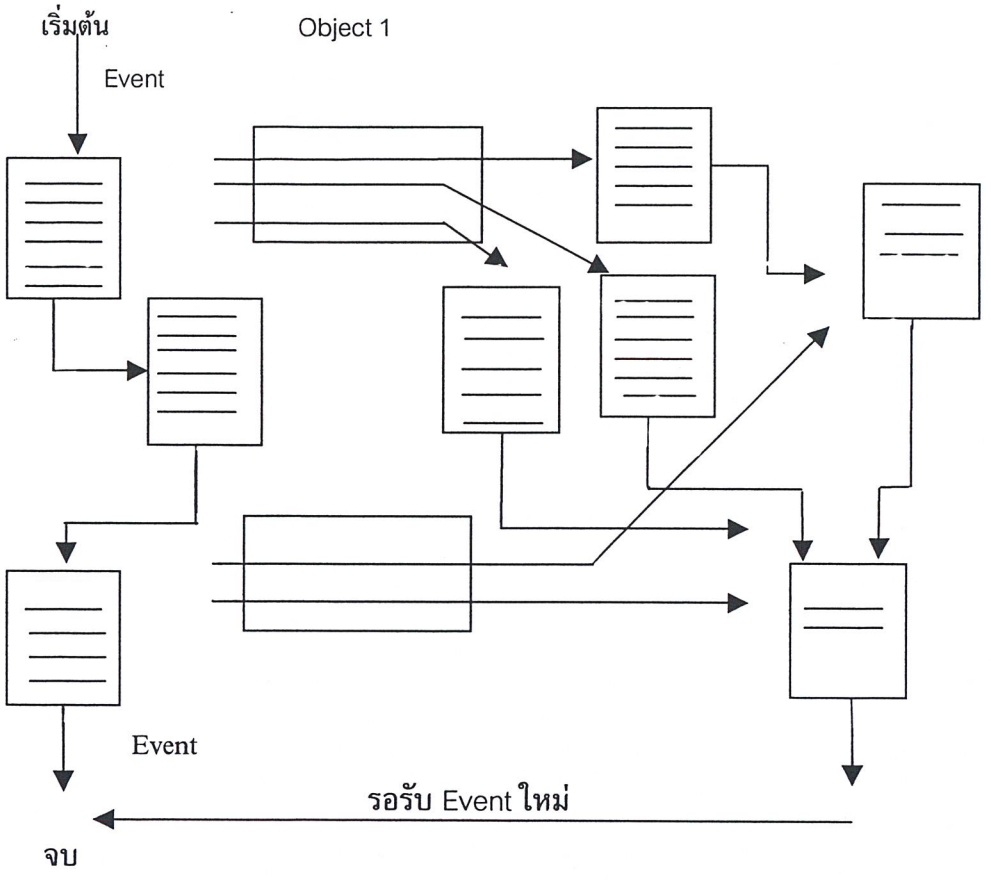
2.6.2 คุณสมบัติและข้อดีของ Visual Basic

ใน Visual Basic มีคุณสมบัติที่ดีหลายประการกล่าวคือ ในการเขียนโปรแกรมแบบเดิมนั้นจะต้องมานั่งออกแบบหน้าจอระบุตำแหน่งการแสดงผลคิดหาขั้นตอนการทำงานและอื่นๆ จากนั้นจึงทำการเขียนโปรแกรม แต่ใน Visual Basic จะใช้หลักการของภาพและการมองเห็น โดยเริ่มจากออกแบบวินโดวส์ย่อยหรือที่ใน Visual Basic เรียกว่า ฟอร์ม ในฟอร์มจะประกอบด้วยสิ่งต่างๆ ที่จะทำงานด้วยหรือเรียกว่าเป็น Object เช่น ข้อความ, ช่องรับข้อความ, Scroll Bar,หรือปุ่ม (Button) เมื่อกำหนดสิ่งต่างๆ เหล่านี้ครบตามต้องการแล้วจึงระบุว่าองค์ประกอบแต่ละอย่างทำงานอย่างไร โดยเขียนโปรแกรมย่อยๆ ปะเข้าไปกัน Object เหล่านี้ ที่ต้องทำแบบนี้ก็เพราะว่าการทำงานใน Windows เป็นแบบที่เรียกว่า Event-Driven คือ ขึ้นกับเหตุการณ์ (Even) การเขียนโปรแกรมแบบเดิมคือสั่งงานตามลำดับจะยุ่งยากมาก

Visual Basic นี้ยังคงไว้ด้วยข้อดีต่างๆ ของ Microsoft Quick Basic และยังมีการเพิ่มเติมคุณสมบัติหลากหลายที่จะสนับสนุนให้ตัวมันเองเป็นโปรแกรมพื้นฐาน สำหรับการพัฒนาโปรแกรมบน Microsoft Windows อีกด้วย ตัวอย่างเช่น กราฟฟิกเอาต์พุตที่สามารถถูกส่งออกไปยังส่วนต่างๆ ของวินโดวส์หรือแม้กระทั่งส่งไปยังเครื่องพิมพ์ สามารถเลือกสีสำหรับงานกราฟฟิกได้มากกว่า 16 ล้านเฉดสี (โดยวินโดวส์จะจัดการแสดงผลกราฟฟิกนั้นตามที่ต้องการ หรือจะลดลงมาได้เท่าที่ฮาร์ดแวร์ของเครื่องนั้นๆ จะสนับสนุนในการแสดงผลได้) โดยที่ไม่ต้องกังวลในส่วนนี้ว่าจะมีกระบวนการในการจัดการอย่างไรไม่ว่าในตอนนี้หรือต่อไปในอนาคต

ข้อดีเหนือ Quick Basic อีกอย่างก็คือ การจัดการตัวแปรใน Visual Basic มีกฎเกณฑ์ ซึ่งง่ายในการเข้าใจและจดจำ เพราะว่าถูกพัฒนาให้ง่ายและมีประสิทธิภาพโดยที่โปรแกรมของ Visual Basic

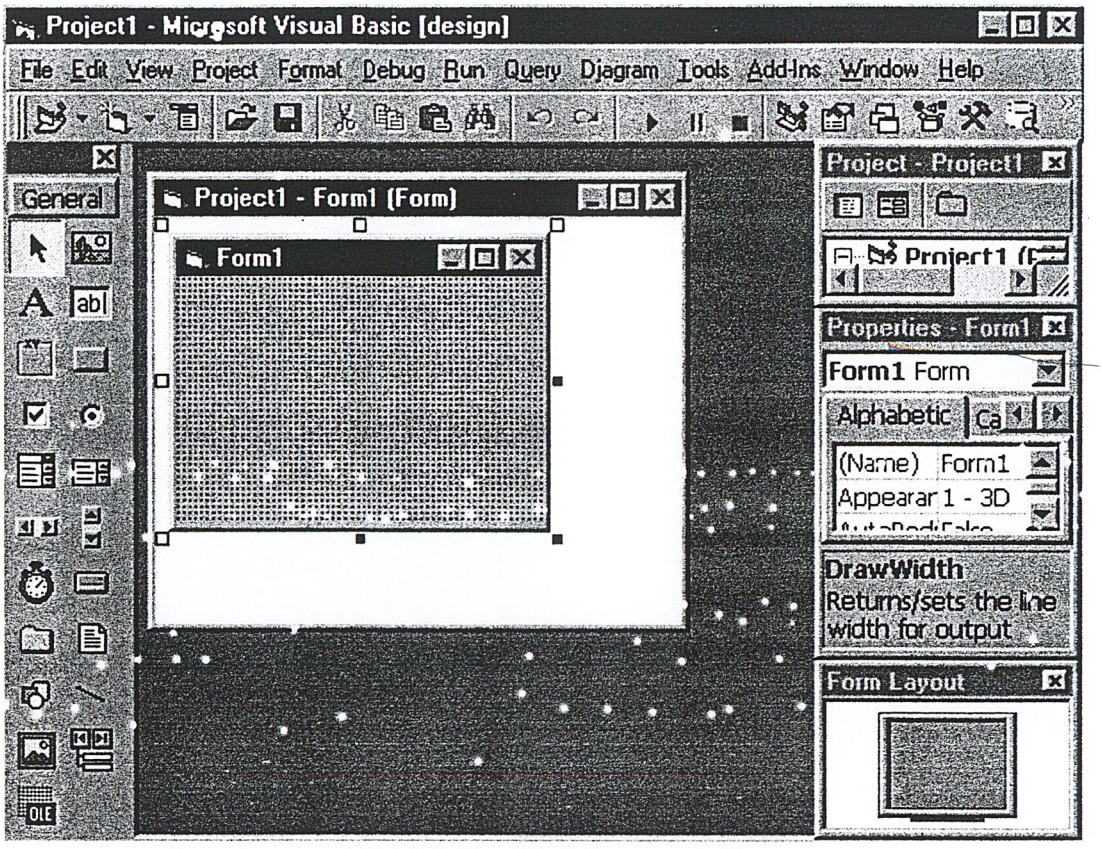
จะประกอบไปด้วยไฟล์ 2 แบบคือ From และ Module ยกเว้นเมื่อ Declare globally ที่อื่น ตัวแปรและค่าคงที่ในโปรแกรมย่อยและฟังก์ชั่นนั้นจะเป็น Local สำหรับกระบวนการที่เกิดขึ้น



รูปที่ 2.25 การเขียนโปรแกรมแบบธรรมดากับแบบ Evant-Driven

2.6.3 ส่วนประกอบต่างๆ ที่ใช้ในการทำงาน

เมื่อทำการเปิดโปรแกรมขึ้นมาใช้งานแล้วนั้นจะปรากฏส่วนประกอบต่างๆ ของโปรแกรมที่ควรทราบก่อนที่จะเริ่มใช้งานโปรแกรมในการสร้างงานต่างๆ เพราะจะทำให้เลือกใช้เครื่องมือหรืออุปกรณ์ในส่วนต่างๆ ได้สะดวกและรวดเร็วมากยิ่งขึ้น



รูปที่ 2.26 ส่วนประกอบต่างๆ ของโปรแกรม Visual Basic 6.0

Title Bar

จะเป็นส่วนที่แสดงชื่อของ Aplication ที่กำลังเปิดใช้งานอยู่ขณะนั้น

Minimize Button

เป็นปุ่มที่ใช้สำหรับคลิกเพื่อหดหรือซ่อนส่วนของ Window หรือหน้าต่างทำการไว้ โดยจะปรากฏหรือแสดงที่หน้าจอเฉพาะส่วนของ Title Bar ของหน้าต่างดังกล่าวไว้และหากต้องการแสดงหรือ Show หน้าต่างอีกครั้งให้ใช้เมาส์คลิกซ้ำที่ปุ่มเดิม ซึ่งปุ่มดังกล่าวจะอยู่ทางด้านขวาของ Title Bar ของหน้าต่างนั้นๆ

Maximize Button or Restore Button

จะเป็นปุ่มที่ใช้สำหรับคลิกเพื่อขยายขนาดใหญ่ขึ้นจนเต็มหน้าจอ

Close Button

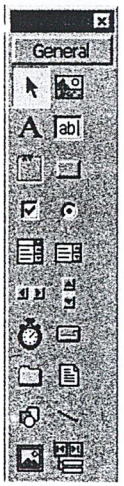
เป็นปุ่มสำหรับคลิกเพื่อปิดหรือจบการทำงานของหน้าต่างนั้นๆ

Menu Bar

เป็นแถบของเมนูหลักที่บรรจุคำสั่งย่อยหลายคำสั่งที่ใช้ในการทำงานไว้ สามารถเขียนใช้งานเมนูได้โดยการใช้เมาส์คลิกที่ชื่อของเมนูที่ต้องการบน Menu Bar หรืออาจเรียกเมนูดังกล่าวได้โดยการกดปุ่มที่คีย์บอร์ดดังนี้ กดปุ่ม Alt ค้างไว้พร้อมกับกดตัวอักษรที่ขีดเส้นใต้ของชื่อเมนูที่ต้องการเปิดดังตัวอย่างต่อไปนี้ หากต้องการใช้คำสั่งย่อยที่อยู่ในเมนู และที่ชื่อของเมนูบน จะมีเส้นขีดไว้ที่ใต้ตัวอักษร ดังนั้นในการเรียกใช้คำสั่งย่อยในเมนูดังกล่าวจำเป็นต้องเปิดเมนูก่อน โดยกดปุ่มที่คีย์บอร์ดดังนี้ Alt+F

Tool Bar

เป็นส่วนที่ปรากฏอยู่ด้านใต้ของเมนูบาร์ ซึ่งเป็นส่วนที่รวบเอาไอคอนเล็กๆ มากมายเอาไว้ ซึ่งแต่ละไอคอนเหล่านี้ก็เปรียบเสมือนคำสั่งหนึ่งๆ ของเมนู ดังนั้นไอคอนในส่วนนี้จึงถูกออกแบบเพื่อให้การเลือกใช้คำสั่งของเมนูรวดเร็วและมีลักษณะที่สื่อความหมายกับผู้ใช้มากขึ้น



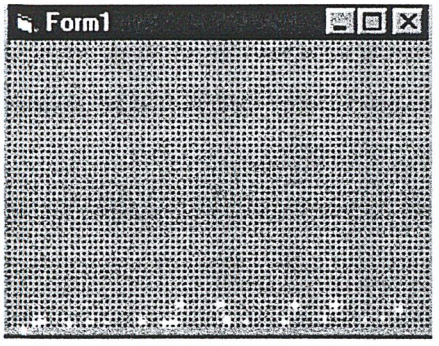
Tool Box

เป็นหน้าต่างที่รวบรวมเอาเครื่องมือต่างๆ สำหรับใช้ในขณะออกแบบฟอร์ม ซึ่งการรวมเอาเครื่องมือตัวใดให้กับฟอร์ม ผู้ใช้สามารถทำได้โดยการคลิ๊กที่คอนโทรลนั้น เพื่อเลือกแล้วนำมาวางลงในฟอร์มโดยวิธีการลากแล้ววาง หรือโดยวิธีการดับเบิลคลิ๊กที่ไอคอนคอนโทรลในทูลบ๊อกซ์ก็ได้ ซึ่ง VB ก็จะทำการวางคอนโทรลลงในฟอร์มปัจจุบันโดยอัตโนมัติ

Form

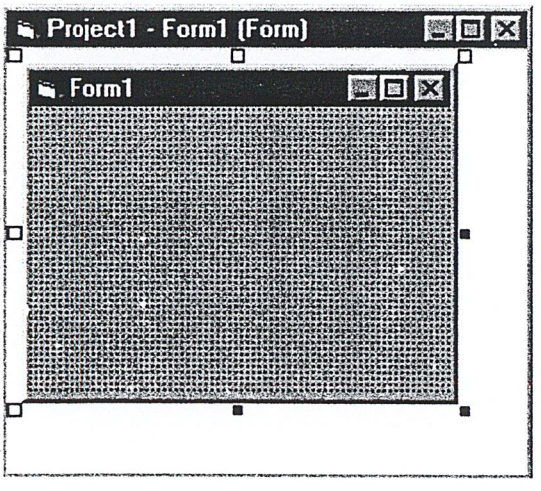
ในการแสดงหน้าต่างเพื่อที่จะสื่อการทำงานกับผู้ใช้ ทั้งนี้เนื่องจาก VB ได้รับการออกแบบให้โปรแกรมเมอร์สามารถนำมาใช้ออกแบบแอพพลิเคชั่นในลักษณะของการสื่อด้วยรูปดังนั้น การสื่อการทำงานต่างๆ ระหว่างแอพพลิเคชั่นกับผู้ใช้จะต้องกระทำผ่านฟอร์ม โดยในโปรเจ็กต์หนึ่งๆ สามารถมีได้หลายฟอร์มและภายในฟอร์มต่างๆ ก็จะถูกใช้ในการบรรจุคอนโทรลต่างๆ เช่น text

box,picture box หรือ image เอาไว้ ดังนั้นฟอร์มจึงทำหน้าที่เป็นตัวบรรจุ (container) และฟอร์มก็ยังเป็นออบเจ็กต์ตัวหนึ่งของ VB ที่อนุญาตให้ผู้ใช้แก้ไขคุณสมบัติได้ในขณะออกแบบจากหน้าต่างคุณสมบัติและสามารถควบคุมพฤติกรรมต่างๆ ของฟอร์มได้โดยผ่านทางวิธีเช่นเดียวกับออบเจ็กต์อื่นๆ



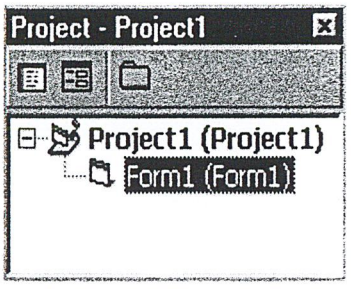
Project Container Window

คือบริเวณทั้งหมดของฉากด้านหลังที่เป็นสีขาวของ Form ใช้สำหรับเป็นพื้นที่หรือหน้าต่างๆที่ใช้บรรจุ Form หรือสร้าง Form



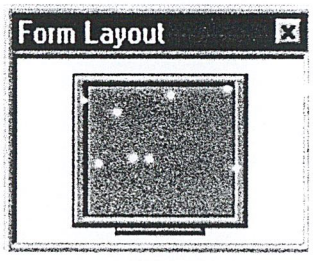
Project Explare Window

เป็นหน้าต่างที่รวบรวมรายชื่อของฟอร์ม โมดูลไฟล์ Custom Control (Class Module) หรือไฟล์ทรัพยากร (resource file) สำหรับการสร้างแอพพลิเคชั่นหนึ่งๆ ซึ่งการรวมเอาไฟล์เหล่านี้เข้าด้วยกันเพื่อสร้างแอพพลิเคชั่นภายใต้ VB/Win เรียกว่า โปรเจ็กต์ (project) หรือโครงการซึ่งโดยปกติ VB จะจัดเก็บไฟล์โปรเจ็กต์โดยมีนามสกุล .MAK โดยจะมีการจัดเก็บแยกออกจากไฟล์อื่นๆ โดยเด็ดขาด



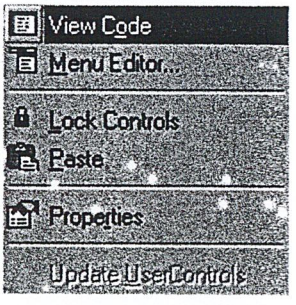
Form Layout Windows

เป็นส่วนที่ใช้กำหนดตำแหน่งของ Form ที่ต้องการแสดงบนหน้าจอและบนหน้าต่างที่ทำงาน นอกจากนี้ในการกำหนดตำแหน่งของ Form จะสามารถเลือกกำหนดได้จากการกำหนดค่าตัวแปรต่างๆ หรือคุณสมบัติต่างที่เกี่ยวกับการจัดวางตำแหน่งได้ในส่วนของ Propertites ได้เช่นกัน



Pop Up Menu or ShortCut Menu

เป็นส่วนของเมนูคำสั่งต่างๆ ที่จะปรากฏหรือแสดงเมื่อมีการกดปุ่มด้านขวาของเมาส์หรือที่เรียกว่า คลิกขวา



Tool Tips

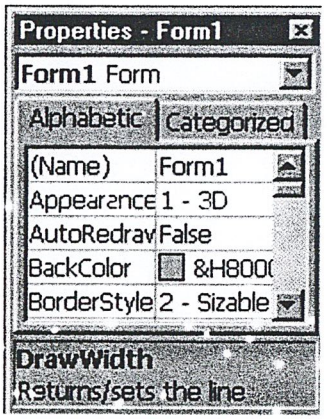
เป็นส่วนที่ใช้บอกหรือแสดงชื่อของเครื่องมือหรือปุ่มต่างๆ ที่ตำแหน่งของ Mouse Pointer ชี้อยู่ในขณะนั้น

Pointer

เป็นเครื่องมือที่ใช้ในการเลือกเมนู คำสั่ง และเครื่องมือต่างๆ

Properties Windows

จะเป็นหน้าต่างที่รวบรวมคุณสมบัติทั้งหมดของฟอร์มหรือคอนโทรลเอาไว้ ซึ่งคุณสมบัติทั้งหมดที่ปรากฏในหน้าต่างนี้จะเป็นคุณสมบัติที่ผู้ใช้สามารถกำหนดค่าได้ในขณะออกแบบ เช่น Caption,Text,Left,Top หรือ Back Color เป็นต้น เมื่อผู้ใช้ทำการแก้ไขค่าคุณสมบัติต่างๆ ในหน้าต่างคุณสมบัตินี้ก็จะส่งผลต่อคอนโทรลตัวนั้นทันที ซึ่งบางคุณสมบัติสามารถแสดงให้เห็นการเปลี่ยนแปลงคุณสมบัติได้ทันที เช่น Left,Top หรือ Fore Color เป็นต้น ส่วนคุณสมบัติบางอย่างจะแสดงผลให้เห็นก็ต่อเมื่อผู้อ่านมีการรับแอพพลิเคชั่นเท่านั้น เช่น Enabled หรือ Visible เป็นต้น



2.6.4 คอนโทรลของ VB/Win (custom control)

คอนโทรลเป็นเครื่องมืออย่างหนึ่งที่ช่วยให้การสร้างแอพพลิเคชั่นด้วย VB สามารถกระทำได้ง่ายและรวดเร็วยิ่งขึ้น ซึ่งในความเป็นจริงคอนโทรลก็คือไฟล์ไดนามิกลิงค์ไลบรารีทั่ว ๆ ไปของวินโดวส์โดยทั่วไปมักจะถูกเขียนด้วยภาษาซีหรือ Visual C++ เพียงแต่ไฟล์เหล่านี้จะมีนามสกุล .VBX หรือ .OCX ซึ่งเรียกว่า custom control เนื่องจากคอนโทรลเหล่านี้ได้รับการออกแบบเป็นพิเศษเพื่อให้สามารถใช้งานร่วมกับ VB ได้อย่างไม่มีปัญหาแล้ว ดังนั้นไฟล์เหล่านี้จะถูกจัดเก็บในไดเรคทอรีย่อย system ของวินโดวส์และไม่คอมแพตทิเบิลกับเวอร์ชั่นที่กำลังติดตั้งใหม่ ไฟล์ custom control ที่มีอยู่เดิมในไดเรคทอรีย่อย system ของวินโดวส์นั้น ก็จะถูกย้ายไปจัดเก็บไว้ในไดเรคทอรีของ VB เวอร์ชั่นนั้นๆ แทนผู้ใช้สามารถโหลดไฟล์คอนโทรลมารวมกับโปรเจ็กต์ได้ด้วยการใช้คำสั่ง Add File ในเมนู File แต่ถ้าหากโปรเจ็กต์ไม่มีการใช้คอนโทรลใดๆ ก็ให้ผู้ใช้ลบคอนโทรลตัวนั้นๆ ออกจากโปรเจ็กต์ซึ่งสามารถทำได้ด้วยการใช้คำสั่ง Remove File ในเมนู File ทั้งนี้เพื่อจะให้การโหลดโปรเจ็กต์ของ VB สามารถกระทำได้รวดเร็วยิ่งขึ้น เมื่อผู้ใช้นั้นได้มีการสร้างแอพพลิเคชั่นและแจกจ่ายให้กับผู้ใช้รายอื่นๆ ผู้ใช้ก็ต้องมีการคัดลอกไฟล์ custom control ที่ใช้แอพพลิเคชั่นลงในไดเรคทอรีย่อย system ของวินโดวส์ของเครื่องผู้ใช้ด้วย (วิธีการสร้างแผ่นดิสก์สำหรับติดตั้งแอพพลิ

เคชั่นสามารถทำได้โดยใช้ Setup Wizard หรือชุดโค้ดที่มีในไดเรคทอรีย่อย System ในไดเรคทอรีของ VB)

คอนโทรลแต่ละตัวในความเป็นจริงก็คือโค้ดชุดหนึ่ง (Library) ที่สามารถทำงานได้เฉพาะด้านตามที่กำหนดและตัวคอนโทรลก็มีคุณสมบัติเฉพาะตัวที่ผู้อ่านสามารถแก้ไขได้ นอกจากนี้เมื่อมีการกระทำใดๆ กับคอนโทรล เช่น การคลิ๊ก การเปลี่ยนค่า เป็นต้น เหตุการณ์ที่เกี่ยวข้องของคอนโทรลก็จะถูกเรียก (event-driven) ซึ่งผู้ใช้สามารถที่จะเขียนโค้ดเพื่อควบคุมการทำงานเมื่อเกิดเหตุการณ์ได้ โดยการเขียนโค้ดลงในส่วนโพรซีเดอร์เหตุการณ์ (event procedure) ซึ่งจะมีรายละเอียดคร่าวๆ ของคอนโทรลแต่ละตัวมีดังนี้

Pointer เป็นเคอร์เซอร์ที่ใช้สำหรับเลือก เลื่อนตำแหน่ง หรือขยายขนาดของฟอร์มหรือคอนโทรล ซึ่งคอนโทรลตัวนี้เป็นเครื่องมือของเอนไวรอนเมนท์ดังนั้นผู้ใช้จึงไม่สามารถนำไปใช้ในแอพพลิเคชั่น

Picture box ตัวคอนโทรลนี้ใช้สำหรับแสดงบิตแม็พ ไอคอน ไฟล์เมต้า วิธีการใช้ในแสดงผลกราฟฟิกเช่น Pset หรือ Line เป็นต้น หรือแสดงข้อความใดก็ได้ นอกจากนี้ยังสามารถใช้บรรจุคอนโทรลอื่นๆ ได้อีกด้วย (container)

Label ใช้แสดงข้อความที่ไม่สามารถแก้ไขได้โดยผู้ใช้ยกเว้นแก้ไขโดยการเขียนโค้ดเท่านั้น

Command button ปุ่มคำสั่งซึ่งผู้ใช้สามารถเลือกได้โดยการคลิ๊กที่ปุ่ม

Frame Frame ใช้เป็นตัวบรรจุ (Container) คอนโทรลอื่นๆ นอกจากนี้ยังใช้เป็นตัวรวมกลุ่ม (group) คอนโทรลอื่นๆ เช่น Option button เป็นต้น

Text box กรอบสี่เหลี่ยมสำหรับกรอกข้อความต่างๆ ซึ่งจะถูกใช้เป็นกล่องสำหรับตัวอักษรที่ถูกคีย์โดยผู้ใช้

Check box ปุ่มสำหรับเลือกสถานะถูกหรือผิดหรือสถานะใช่หรือไม่ใช่

Option button ปุ่มที่ถูกใช้รวมกันเป็นกลุ่มเพื่อให้ผู้ใช้ได้ใช้เป็นตัวเลือก (อย่างใดอย่างหนึ่ง)

Combo box คอนโทรลที่รวมเอาความสามารถของ text box และ list box เอาไว้ด้วยกัน

List box คอนโทรลที่ใช้ในการแสดงผลรายการของข้อความเพื่อให้ผู้ใช้สามารถเลือกได้

 **Horizontal scroll bar** แถบเลื่อนในแนวนอนซึ่งสามารถกำหนดค่าในช่วงที่ต้องการเลือกได้

 **Vertical scroll bar** แถบเลื่อนในแนวตั้งซึ่งผู้ใช้สามารถกำหนดค่าในช่วงที่ต้องการเลือกได้

 **Time** นาฬิกาที่นี้ถูกใช้ในการสร้างเหตุการณ์โดยที่อ้างอิงกับช่วงเวลา ซึ่งสามารถเทียบเคียงได้กับประโยค On Timer ของ Quick BASIC

2.6.5 ภาษาโปรแกรมของ Visual Basic

1. ตัวแปรและค่าคงที่

ตัวแปรใน Visual Basic มีความยาวได้ถึง 40 ตัวอักษรสามารถตั้งชื่อตัวแปรโดยผสมจากตัวอักษร ตัวเลขและ underscore (_) แต่ตัวแรกต้องเป็นอักษรเท่านั้นและห้ามตั้งชื่อตรงกับคำที่สงวนไว้ (reserved word) ซึ่งได้แก่ฟังก์ชันและคำสั่งต่างๆ ตัวแปรแบ่งออกเป็น 7 ชนิดดัง ตารางที่ 2.6

ชนิด	คำอธิบาย	ตัวอักษรบอกชนิด	ขอบเขต
Integer	2-byte integer	%	-32,768 จนถึง 32,767
Long	4-byte integer	&	-2,147,678 จนถึง 2,147,483,647
Single	4-byte floating-point Number	!	-3.402823E38 จนถึง –1.401298E-45
Double	8-byte floating-point Number	#	-1.79769313486232D308 จนถึง –4.94065645841247D-324 (ค่าลบ) 4.94065645841247D-324 จนถึง 1.79769313486232D308
Currency	8-byte number with Fixed decimal point	@	-922337203685477.5808 จนถึง 922337203685477.5807
String	String of characters	$	0 จนถึงประมาณ 65,000 ตัวอักษร
Variant	Date/time, floating-Point number, or string		Date values : วันที่ 1 มกราคม ปี ค.ศ. 0000 ถึงวันที่ 31 ธันวาคม ปี ค.ศ. 9999

ตารางที่ 2.6 ตัวแปรและค่าคงที่ในโปรแกรม Visual Basic

1.1 Integer

ไว้สำหรับเก็บค่าจำนวนเต็มในช่วง –32,768 ถึง 32,767 ตัวแปรชนิดนี้สามารถคำนวณได้เร็วมาก แต่มีข้อจำกัดด้านค่าที่เก็บได้ซึ่งอยู่ในขอบเขตแคบกว่า แบบอื่นตัวแปรชนิดนี้ระบุด้วย % เช่น A%

1.2 Long

เป็นค่าจำนวนเต็มเช่นเดียวกับชนิด Integer แต่ขยายช่วงไปเป็น –2,147,483,648 ถึง 2,147,483,647 ใช้เนื้อที่ 4 ไบต์และใช้ตัวอักษร & เป็นตัวระบุชนิด เช่น A&

1.3 Single

ใช้เนื้อที่ในการเก็บ 4 ไบต์ เก็บค่าตัวเลขที่เป็นทศนิยมซึ่งไม่ต้องการความละเอียดมากนัก (-3.402823E38 ถึง –1.401298E-45) ใช้ ! ในการระบุชนิดเช่น A!

1.4 Double

คล้ายกับชนิด Single แต่สามารถเก็บค่าได้มากกว่าและใช้เนื้อที่เก็บเป็น 2 เท่า ของ Single (8 ไบต์) อย่างไรก็ตามก็ใช้เวลาการคำนวณมากกว่าด้วย ระบุชนิดด้วย # เช่น A#

1.5 Currency

ใช้เก็บค่าทางการเงินโดยเฉพาะ เป็นแบบทศนิยมโดยมีทศนิยมคงที่ 4 ตำแหน่ง ใช้เนื้อที่ 8 ไบต์ การระบุถึงชนิดใช้ตัวอักษร @ เช่น A@

1.6 String

ใช้เก็บข้อความหรือตัวอักษรซึ่งไม่สามารถนำไปคำนวณได้ ความยาวของข้อความคือ 65,500 ตัวอักษร เนื้อที่ที่ใช้เก็บขึ้นอยู่กับขนาดของข้อความที่เก็บ ระบุชนิดด้วย $ เช่น A$

1.7 Variant

ตัวแปรชนิดนี้เพิ่งเริ่มมีใช้ใน Visual Basic 2.0 ถูกออกแบบมาใช้เก็บค่าต่างๆ กัน โดยสามารถเก็บค่าชนิดใดก็ได้ 6 ชนิดข้างต้น ไม่ว่าจะเป็นตัวเลข ตัวอักษร หรือวันที่ ข้อมูลชนิดนี้มีประโยชน์ในกรณีที่ต้องการประกาศตัวแปร ซึ่งอาจเก็บค่าได้หลายแบบโดยที่ไม่สามารถระบุได้แน่นอนว่าจะเป็นแบบใด ปัญหาคือจะทำงานช้ากว่าตัวแปรแบบอื่นเพราะ Visual Basic ต้องมีการตรวจสอบชนิดของตัวแปรประเภทนี้ทุกครั้งก่อนจะใช้งาน

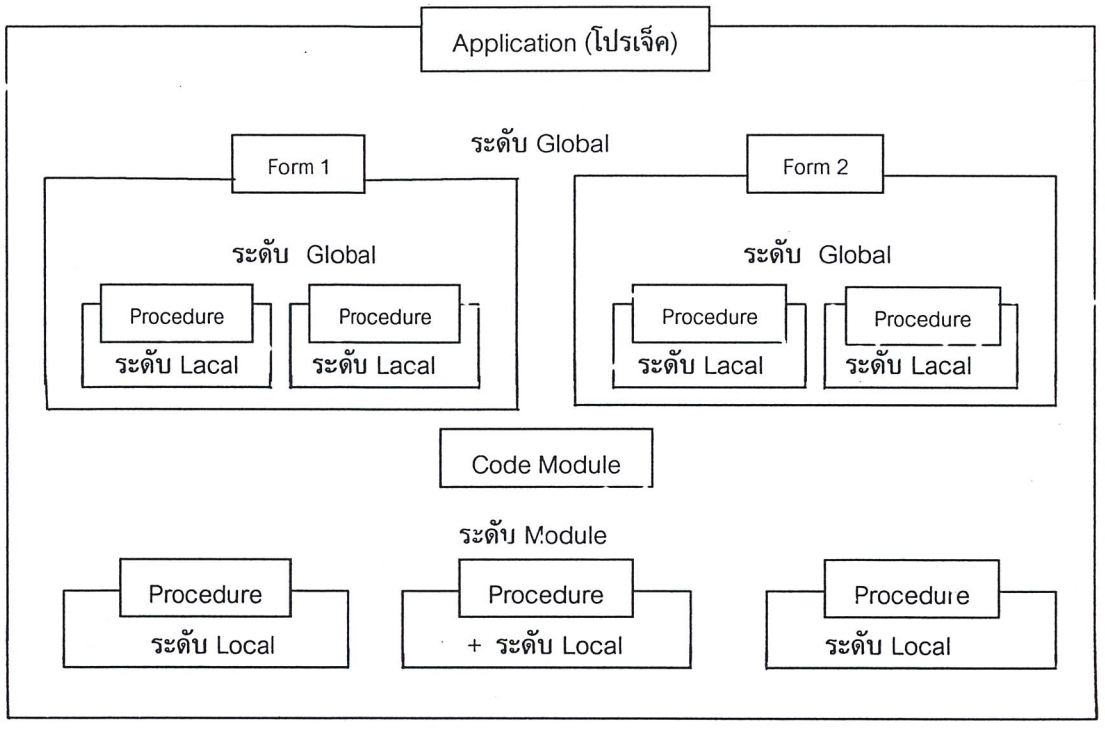
2. ขอบเขตของตัวแปร

ทั้งตัวแปรและค่าคงที่ที่ได้กล่าวถึงตอนต้น โดยปกติจะมีขอบเขตของการใช้งานหรือ ใช้งานได้ในแต่ละส่วนของโปรแกรมเท่านั้น ซึ่งขอบเขตของตัวแปร (Scope) นี้แบ่งออกได้เป็น 3 ระดับ คือ

2.1 Local

ตัวแปรชนิดนี้จะเป็นที่รู้จักกันมาก ใช้ได้ในระดับโพรซีเดอร์หรือฟังก์ชันที่ประกาศ

ใช้ตัวแปรนั่นเอง ส่วนมากมักจะใช้กับการประกาศตัวแปรสำหรับการคำนวณหรือจะใช้เก็บค่าชั่วคราวตัวแปรในระดับ Local จะถูกยกเลิกไปเมื่อการทำงานของโพซีเดอร์หรือฟังก์ชั่นนั้นสิ้นสุดลง



รูปที่ 2.27 แสดงขอบเขตของการใช้ตัวแปรและค่าคงที่ในระดับต่างๆ

2. 2 Module

ตัวแปรในระดับโมดูลสามารถอ้างอิงถึง สามารถใช้ร่วมกันได้ในหลายฟังก์ชั่นหรือหลายโพรซีเดอร์ซึ่งอยู่ในโมดูลนั้น ไม่ว่าจะเป็นในโมดูลของโปรแกรมที่ประกอบด้วยรูทีนย่อยๆ หลายรูทีนหรือในโมดูลของฟอร์มซึ่งอยู่ในรูทีนสำหรับทำงานกับ Object ในฟอร์มนั้น การกำหนดตัวแปรในระดับนี้ต้องทำที่ส่วน Declaration ของโมดูลนั้น

2.3 Global

ในตัวแปรระดับ Global นี้สามารถจะอ้างอิงได้จากจุดใดๆ ก็ตามใน Application ทุกรูทีนหรือทุกส่วนจะสามารถนำไปใช้ได้การประกาศใช้จะเหมือนกับระดับโมดูล คือประกาศค่าคงที่ในส่วน Declaration เพียงแต่ว่าจะต้องมีคำ “Global” นำหน้าชื่อตัวแปร สำหรับในการประกาศ ค่าคงที่ในระดับ Global ก็ใช้คำ “Global” นำหน้าเช่นกัน

```
Global A As Integer
Global Const VAT_Rate = 70/100
```

3. โอเปอเรเตอร์

โอเปอเรเตอร์ คือสิ่งที่จะก่อให้การกระทำเพื่อให้ได้ผลลัพธ์ออกมาพบโอเปอเรเตอร์ได้ในนิพจน์ทั่วๆ ไป ซึ่งแบ่งออกเป็นกลุ่มโอเปอเรเตอร์ทางคณิตศาสตร์ การเปรียบเทียบ และ ทางตรรกะ ดังตารางที่ 2.7 โดยสัญลักษณ์ในวงเล็บคือ โอเปอเรเตอร์ที่ใช้ในนิพจน์

ทางคณิตศาสตร์	การเปรียบเทียบ	ทางตรรกะ
ยกกำลัง (^)	เท่ากัน (=)	Not
ค่าลบ (-)	ไม่เท่ากัน (<>)	And
คูณ,หาร (*,/)	น้อยกว่า (<)	Or
หารจำนวนเต็ม (\)	มากกว่า (>)	Xor
Module (Mod)	น้อยกว่าหรือเท่ากับ (<=)	Eqv
บวก,ลบ (+,-)	มากกว่าหรือเท่ากับ (>=)	Imp
เชื่อมข้อความ (String)(&หรือ+)	เหมือน (Like)	Is

ตารางที่ 2.7 แสดงกลุ่มของโอเปอเรเตอร์

4. ประโยคคำสั่ง

เป็นประโยคที่ส่งให้เกิดการทำงานตามที่ผู้เขียนโปรแกรมต้องการ ประโยคคำสั่งหลักๆ ได้แก่ ประโยคสำหรับกำหนดค่า ประโยคหมายเหตุ ประโยคประกาศ และประโยคเพื่อควบคุมลำดับการทำงานของโปรแกรมนอกเหนือจากนี้จะเป็นประโยคเพื่อการทำงานอื่น เช่น ทำงานทางกราฟฟิก เป็นต้น

4.1 ประโยคสำหรับกำหนดค่า

ใช้สัญลักษณ์ "=" เพื่อเป็นการกำหนดค่าทางขวามือของเครื่องหมายให้กับตัวแปรหรือคุณสมบัติ (Property) ทางซ้ายมือ เช่น

```
A = 5
Text1.Text = "Hi!"
```

4.2 ประโยคหมายเหตุ

สำหรับการใส่หมายเหตุหรือคำอธิบายการทำงานของโปรแกรม ซึ่งการใส่หมายเหตุในโปรแกรมนี้ถือว่าเป็นเรื่องสำคัญ เนื่องจากสิ่งนี้จะช่วยป้องกันการหลงลืมของคนเขียนโปรแกรมเองหรือช่วยอธิบายให้ผู้อื่นซึ่งต้องมาไล่การทำงานของโปรแกรมเข้าใจโปรแกรมได้ง่ายขึ้น มิฉนั้นจะพบว่าวันหนึ่งจะมีผู้ไม่เข้าใจโปรแกรมส่วนนี้ทำงานอย่างไรมีไว้เพื่ออะไร ซึ่งอาจเกิดกับตัวผู้เขียน

โปรแกรมเองก็ได้ ประโยคหมายเหตุนี้อาจขึ้นต้นประโยคด้วยคำ “Rem” ที่ย่อมาจาก คำว่า “Remark” หรือใช้สัญลักษณ์

“ ‘ ” นำหน้าประโยคก็ได้ ดังตัวอย่าง

Rem This function is use to open database files

‘This function is use to close database files

4.3 ประโยคประกาศ

ใช้ประกาศถึงตัวแปร ค่าคงที่ และฟังก์ชันหรือรูทีนย่อย สำหรับการประกาศตัวแปรและค่าคงที่ได้กล่าวไว้แล้วในข้างต้นว่าแบ่งออกเป็น Global, Module หรือ Local ส่วนการประกาศฟังก์ชันหรือรูทีนย่อยใช้คำสั่ง “Declare” ซึ่งมีรูปแบบดังนี้

Declare Sub globalname Lib “libname” [Alias “aliasname”]

[([[ByVal] variable [As type] [,ByVal] variable [As type]]…])]

Declare Function globalname Lib “libname” [As Type] [,ByVal]

Variable [As type]]…])] [As Type]

5. ประโยคควบคุมลำดับการทำงาน

แบ่งออกเป็นหลายกลุ่มย่อยมีทั้งประโยคตัดสินใจ ประโยคการทำงานวนรอบ (loop) การกระโดดแบบมีเงื่อนไข รวมถึงคำสั่งจบการทำงานของโปรแกรมทั้งแบบชั่วคราวและถาวร ซึ่งจะขอกล่าวเรียงตามลำดับดังนี้

5.1 If…Then…Else

เป็นคำสั่งตัดสินใจว่าจะทำงานหรือไม่ตามเงื่อนไขที่กำหนด สามารถกำหนดเพิ่มได้ว่าหากไม่เป็นไปตามเงื่อนไขให้ทำงานใด ซึ่งอยู่หลังคำ “Else” อีกทั้งยังกำหนดให้ตรวจสอบหลายเงื่อนไขได้ด้วย “Else”

If condition1 Then

[statementblock-1]

[Else If condition2 then

[statementblock-2]]…

[Else

[statementblock-n]]

End If

5.2 Select Case

ในบางครั้งการเขียนประโยคตัดสินใจด้วย If…Then…Else จะทำให้โปรแกรมเยิ่นเย้อและอ่านยาก ประโยค Select Case จะทำหน้าที่คล้ายกับ If…Then…Else แต่ให้ความกระชับว่าโดยใช้

กับค่าต่างๆ ของนิพจน์เดียวเท่านั้น

```
Select Case testexpression
    [Case exp1
      [statementblock-1]
    [Case exp2
      [statementblock-2]]...
    [Case Else
      [statementblock-n]
End Select
```

5.3 For...Next

เพื่อกำหนดให้มีการทำงานซ้ำๆ กันเป็นจำนวนครั้งที่ต้องการ รูปแบบคำสั่งได้แก่

```
For counter = start to end [Step increment]
    [Statements]
Next [counter]
```

ทำการกำหนดค่าเริ่มต้น (Start) ให้กับ counter เพื่อเป็นตัวนับให้มีการทำงานในส่วน Statements ไปเรื่อยๆ จนกว่าค่า end โดยในการทำงานแต่ละรอบจะเพิ่มค่า Counter ขึ้นตามค่าใน Increment

5.4 Do...Loop

ใช้เมื่อต้องการให้ทำงานวนรอบแบบมีเงื่อนไข ประโยค Do...Loop นี้สามารถกำหนดให้ตรวจสอบเงื่อนไขก่อนหรือหลังการทำงานในแต่ละรอบก็ได้โดยใส่ค่า "While" หรือ "Until" ข้างหลัง Do หรือ Loop ตามลำดับ การใช้ while จะหมายถึงให้ทำงานวนรอบเมื่อนิพจน์ที่ตามมาให้ค่าจริง (True) คือทำงานจนกว่านิพจน์จะกลายเป็นเท็จ (False) หรือพูดอีกนัยหนึ่งคือทำงานไปจนกว่านิพจน์จะเป็นจริงนั่นเอง

```
Do [{While/Until} condition]
    [Statementblock]
Loop
```

หรือ

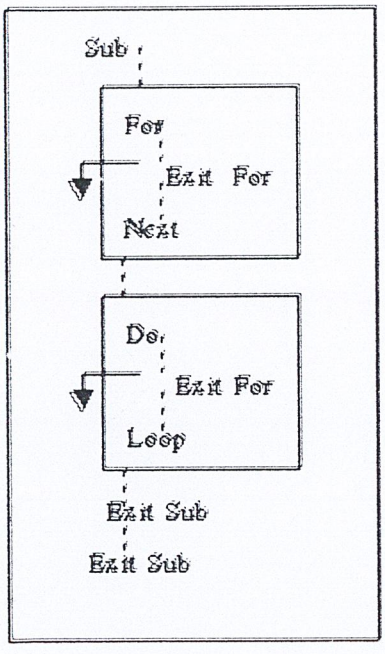
```
Do
    [Statementblock]
Loop [{While/Unit} condition]
```

5.5 On…GoSub, On…Goto

เป็นการกระโดดการทำงานไปยังรูทีนในส่วนอื่น ซึ่งมีทั้งแบบย้อนกลับ (GoSub) หรือไม่ย้อนกลับ (Goto) โดยมีการคำนวณค่านิพจน์ที่กำหนด ซึ่งจะได้ค่าจาก 0 ถึง 255 และจะมีการกระโดดไปยังส่วนของโปรแกรมตามรายการลาเบลที่ระบุหลัง GoSub หรือ Goto ซึ่งจะไปที่ตำแหน่งใดก็ขึ้นอยู่กับค่าของนิพจน์ เช่น ได้ค่า 1 จะไปที่ลาเบลแรก, 2 ไปลาเบลที่ 2,... ตามลำดับ ซึ่งมีรูปแบบดังนี้

On expression {GoSub/GoTo} destinationlist

6.หยุดการทำงานของโปรแกรม



รูปที่ 2.28 แสดงการออกจากส่วนของโปรแกรมแบบต่างๆ ด้วยคำสั่ง Exit

ใน Visual Basic มีทั้งคำสั่งในการหยุดทำงานแบบชั่วคราวและถาวร รวมทั้งจบการทำงานในวนรอบ (Loop) ในฟังก์ชันหรือโพรซีเดอร์ด้วย การหยุดทำงานชั่วคราวใช้คำสั่ง Stop ซึ่งจะเป็นคำสั่งโดดไม่ต้องมีพารามิเตอร์ประกอบส่วนการหยุดโปรแกรมอย่างถาวรใช้คำสั่ง End ข้อแตกต่างของ 2 คำสั่ง นี้คือ Stop จะไม่ยกเลิกตัวแปรและจะยังไม่ปิดไฟล์ที่เปิดค้างอยู่ให้ อีกทั้งยังสามารถสั่งให้ทำงานต่อไปภายหลังได้โดยเลือกข้อ Continue จากเมนู Run หรือหัวข้อ Single Step ในเมนู Debug ส่วน End จะยกเลิกตัวแปรและปิดไฟล์ทั้งหมดที่เปิดค้างอยู่ในโปรแกรมนั้น

2.6 Winsock Contrrol

โหมดการทำงานของ Winsock Control กับโพรโตคอล TCP/IP นี้แบ่งการทำงานออกเป2 โหมด คือ TCP กับ UDP

-TCP (Transmisson Control Protocal)

TCP เป็นการทำงานในลักษณะ คอนเนคชั่นเบส เปรียบเหทือนกับการทำงานของโททรศัพท์ที่จะมีการตรวจสอบการทำงานจากทั้งสองฝั่งมีการโต้ตอบระหว่างกัน ในโหมดนี้จะเหมาะกับการส่งข้อมูลที่ต้องการการทำงานที่ถูกต้องแม่นยำของข้อมูล เช่นการส่งภาพหรือเสียงผ่านเครือข่าย มีการใช้ทรัพยากรของระบบค่อนข้างสูง

-UDE (User Datagram Protocol)

UDP เป็นการทำงานในลักษณะตรงกันข้ามกับ TCP นั่นคือมีการทำงานในลักษณะคอนเนคชั่นเลส เปรียบเหมือนกับการกระเสียงวิทยุ จะเหมาะกับการส่งข้อมูลที่ไม่สำคัญนัก เช่น ส่งข้อมูลแสดงสถานะการทำงาน ซึ่งทำให้ใช้ทรัพยากรค่อนข้างน้อย

2.6.1 พรอพเพอร์ตี้สำคัญของวินซอค

-Protocol เป็นการเลือกโพรโตคอลสำหรับการทำงาน

-LocalPort เป็นการกำหนดหมายเลขพอร์ตของคอมพิวเตอร์ที่จะใช้งานกับ วินซอค

-RemoteHost เป็นการกำหนดชื่อของคอมพิวเตอร์ที่เราจะติดต่อด้วยอาจจะเป็นไอพีแอดเดรส หรือ เป็นชื่อคอมพพิวเตอร์ที่เป็นชื่อที่ง่ายต่อการจดจำ

-RemotePort เป็นการกำหนดหมายเลขพอร์ตของคอมพิวเตอร์ที่เราต้องการจะติดต่อด้วย

-ByteReceive เป็นจำนวนข้อมูลที่รับเข้ามาเก็บในบัฟเฟอร์

2.6.2 เมธอดสำคัญของวินซอค

-Listen เป็นเมธอดที่ใช้สร้างซอคเกตทำให้คอมพิวเตอร์เครื่องอื่นๆสามารถติดต่อเข้ามาได้

-Connect เป็นเมธอดที่ใช้รับ รีเควสจากคอมพิวเตอร์ที่ติดต่อเข้ามา

-SendData เป็นเมธอดที่ใช้ส่งข้อมูลไปยังคอมพิวเตอร์เครื่องอื่นที่เราติดต่อแบบซอคเกต

-GetData เป็นเมธอดที่ใช้รับข้อมูลจากบัฟเฟอร์เข้ามาเก็บในตัวแปรที่เรากำหนดให้โดยสามารถกำหนดชนิดตัวแปรและความยามข้อมูลที่จะนำมาเก็บได้

-Close เป็นเมธอดที่ใช้ยกเลิกการติดต่อแบบซอคเกต

2.6.3 อีเวนต์สำคัญของวินซอค

-ConnectionRequest เป็นเหตุการณ์ที่เกิดขึ้นเมื่ คอมพิวเตอร์เครื่องอื่นมรการรีเควสเข้ามา ซึ่งมีการกำหนด ไอดี ให้กับแต่ละรีเควสที่เข้ามา

-DataArrival	เป็นเหตุการณ์ที่เกิดขึ้นเมื่อ คอมพิวเตอร์เครื่องอื่นมีการรีเควสเข้ามา ซึ่งจะมีการกำหนด ไอดีกับแต่ละรีเควสที่เข้ามา
-SendProgess	เป็นเหตุการณ์ที่เกิดเกิดเมื่อมีข้อมูลชุดใหม่เข้ามาเก็บในบัฟเฟอร์ ซึ่งเราสามารถตรวจสอบขนาดข้อมูลได้จากพรอพเพอร์ตี้ไบท์รีซีฟ
-SendComplete	เป็นเหตุการณ์ทีเกิดเมื่อการส่งข้อเสร็จสิ้นสมบูรณ์
-Error	เป็นเหตุการณ์ที่เกิดเมื่อมีความผิดพลาดเกิดขึ้น ซึ่งจะสามาถแสดงหมายเลขของความผิดพลาด คำอธิบาย และรายละเอียดอื่นๆ สำหรับการจัดการกับข้อผิดพลาดที่เกิดขึ้น

บทที่3

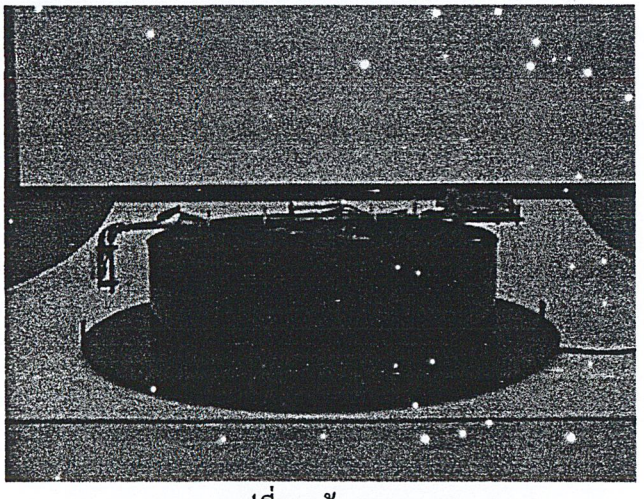
การออกแบบและการสร้าง

ในการดำเนินการสร้างหลอดไฟแสดงผลแบบหมุนควบคุมผ่านอินฟราเรด สามารถแบ่งขั้นตอนการดำเนินงานได้ดังนี้

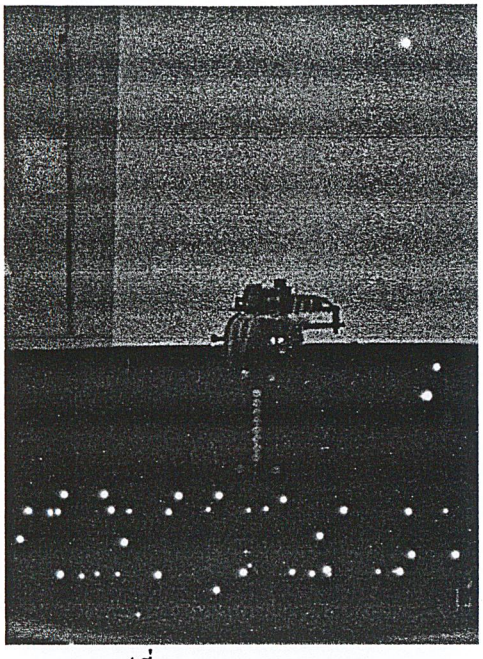
1. ออกแบบและสร้างโครงสร้างหลักของก้านหมุนหลอดแสดงผล
2. ออกแบบส่วนภาครับและภาคแสดงผลข้อมูลแบ่งเป็น
   2.1 ส่วนไมโครคอนโทรลเลอร์
   2.2 ส่วนขยายบัฟเฟอร์
   2.3 ส่วนภาครับสัญญาณอินฟราเรด
   2.4 ส่วนควบคุมการแสดงผล
3. ออกแบบส่วนภาคส่งข้อมูล
   3.1วงจรภาคส่ง
   3.2โปรแกรมควบคุมการทำงาน

3.1 การออกแบบและสร้างโครงสร้างหลักของก้านหมุนหลอดแสดงผล

ในการออกแบบโครงสร้างของก้านหมุนแสดงผล ต้องคำนึงถึงความสมดุลของแกนหมุน ซึ่งจะต้องมีแผงวงจรวางอยู่บนก้านหมุน โดยจะต้องออกแบบให้เกิดความสมดุลของแกนหมุนทั้งสองข้าง ก้านหมุนมีรัศมีในการหมุนอยู่ประมาณ 1 ฟุต และมีแผ่นวงจรหลอดไฟ LED ติดอยู่ที่ปลายก้าน และส่วนที่สำคัญก็คือการออกแบบส่วนส่งจ่ายกระแสไฟเพื่อจ่ายไฟให้แผงวงจรที่จะต้องหมุนอยู่กับแกนหมุนตลอดเวลา โดยจะส่งกระแสไฟผ่านแปลงถ่านที่สัมผัสกับวงแหวนทองแดงเพื่อจ่ายไฟไปจากแหล่งจ่ายไปแผงวงจรที่หมุนอยู่ด้านบน



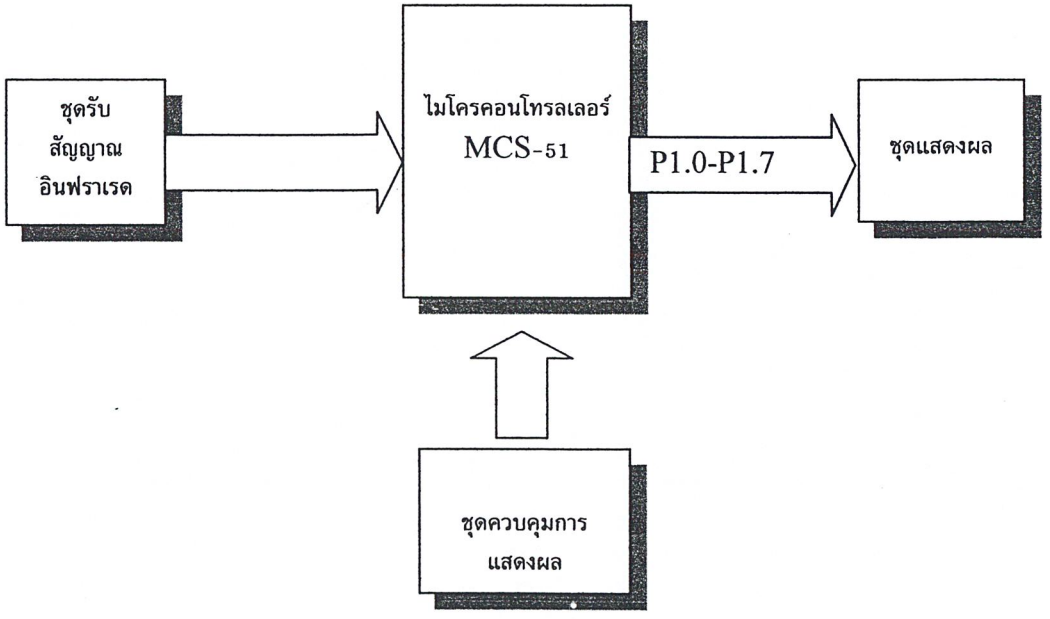
รูปที่3.1 ก้านหมุนและวงจร



รูปที่3.2 หลอดแสดงผล

3.2 การออกแบบส่วนภาครับและภาคแสดงผลข้อมูล

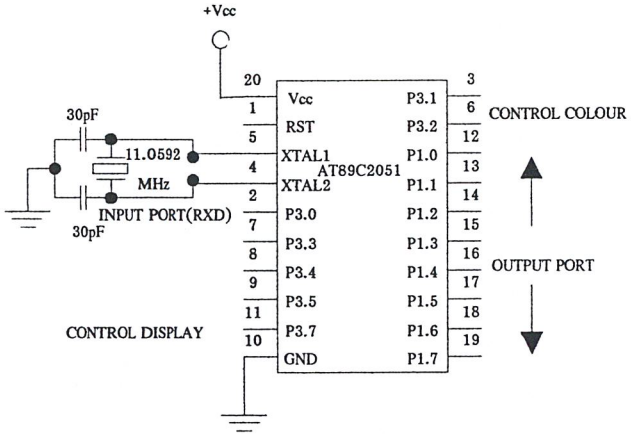
ในการออกแบบส่วนของการแสดงผลสามารถแบ่งออกเป็นส่วนต่างๆ ดังนี้



รูปที่ 3.3 บล็อกไดอะแกรมภาครับข้อมูล

3.2.1 ส่วนไมโครคอนโทรลเลอร์

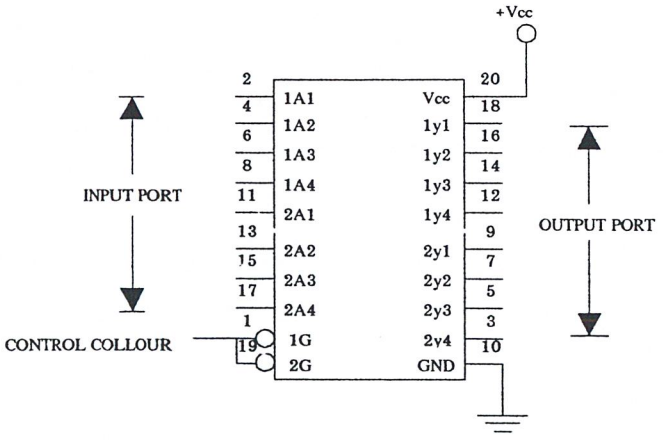
ในส่วนของวงจรไมโครคอนโทรลเลอร์จะใช้ไอซีเบอร์ AT89C2051 ซึ่งมีโครงสร้างและ ชุดคำสั่งเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 โดยบรรจุอยู่ในตัวถังแบบ DIP ขนาด 20 ขา มีหน่วยความจำโปรแกรมชนิด Flash Memory ขนาด 2 กิโลไบต์ สามารถโปรแกรมซ้ำได้ 1000 ครั้ง ทำงานด้วยแรงดัน 2.7-6 โวลต์ มีพอร์ต I/O 16 บิต (พอร์ต1และพอร์ต3) ซึ่งเหมาะสมกับการใช้เป็นวงจรควบคุมการแสดงผล โดยใช้พอร์ต RXD เป็นพอร์ตรับข้อมูลจาก TFMS 5330 (โฟโตโมดูล) และพอร์ต P3.7 ต่อกับ ออปโต ทรานซิสเตอร์ เป็นตัวควบคุมการแสดงผล ใช้พอร์ตเอาต์พุต P1.0-P1.7 เป็นพอร์ตส่งข้อมูลไปวงจรขยายบัฟเฟอร์ ส่วนการเลือกสีเราจะใช้พอร์ต P3.1 และ P3.2 เป็นการกำหนดสีแดง สีเขียว หรือส้ม



รูปที่ 3.4 ส่วนไมโครคอนโทรลเลอร์

3.2.2 ส่วนบัฟเฟอร์

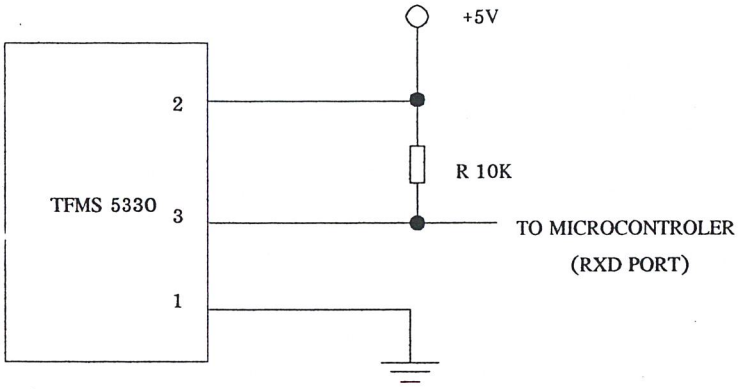
ในส่วนบัฟเฟอร์เป็นตัวขยายแรงดันเพื่อให้มีแรงดันพอที่จะจ่ายให้กับ หลอด LED ซึ่งใช้ไอซีเบอร์ 74LS244 จำนวน 2 ตัวเป็นตัวควบคุมสีในการทำงาน



รูปที่ 3.5 ส่วนบัฟเฟอร์ 74LS244

3.2.3 ส่วนรับสัญญาณอินฟราเรด

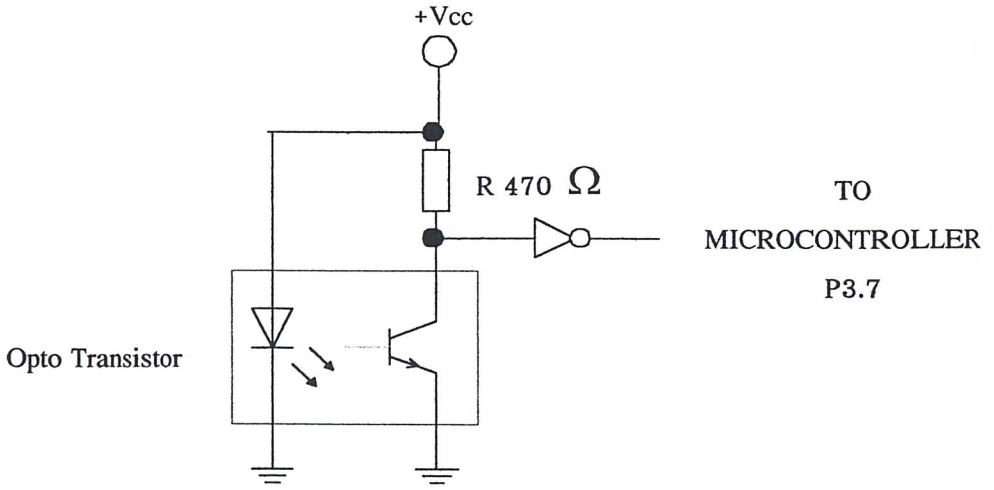
ในส่วนรับสัญญาณอินฟราเรดจะใช้ โฟโตโมดูลเบอร์ TFMS5330 ใช้ความถี่ 33 กิโลเฮิร์ต เป็นตัวรับสัญญาณอินฟราเรดและแปลงสัญญาณออกมาเป็นสัญญาณดิจิตอล โดยจะต่อเข้าพอร์ตอินพุต RXD ให้กับไมโครคอนโทรลเลอร์ โดยมี R 10K ทำหน้าที่เป็นพูลอัพ



รูปที่ 3.6 ส่วนรับสัญญาณอินฟราเรด

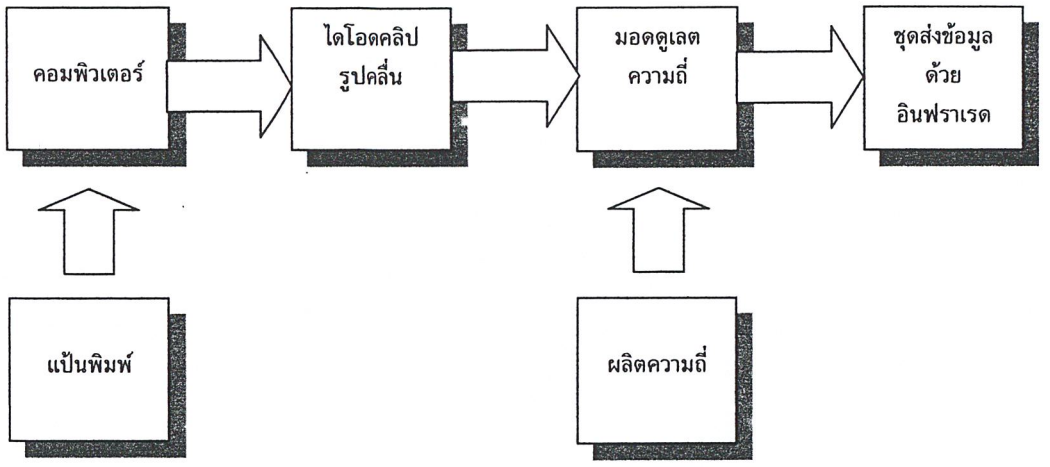
3.2.4 ส่วนควบคุมการแสดงผล

ในส่วนควบคุมการแสดงผลจะใช้ออปโต้ทรานซิสเตอร์เป็นตัวควบคุม โดยออปโตทรานซิสเตอร์จะติดอยู่บนก้านหมุน เมื่อก้านหมุนหมุนไปรอบๆ ออปโต้ทรานซิสเตอร์จะหมุนไปด้วย โดยจะมีแผ่นกั้นแสงเพื่อกั้นแสงทำให้เกิดการทริกสัญญาณและนำสัญญาณที่ได้มาเข้า NOT GATE เพื่อส่งไปยังไมโครคอนโทรลเลอร์นำไปควบคุมการแสดงผล

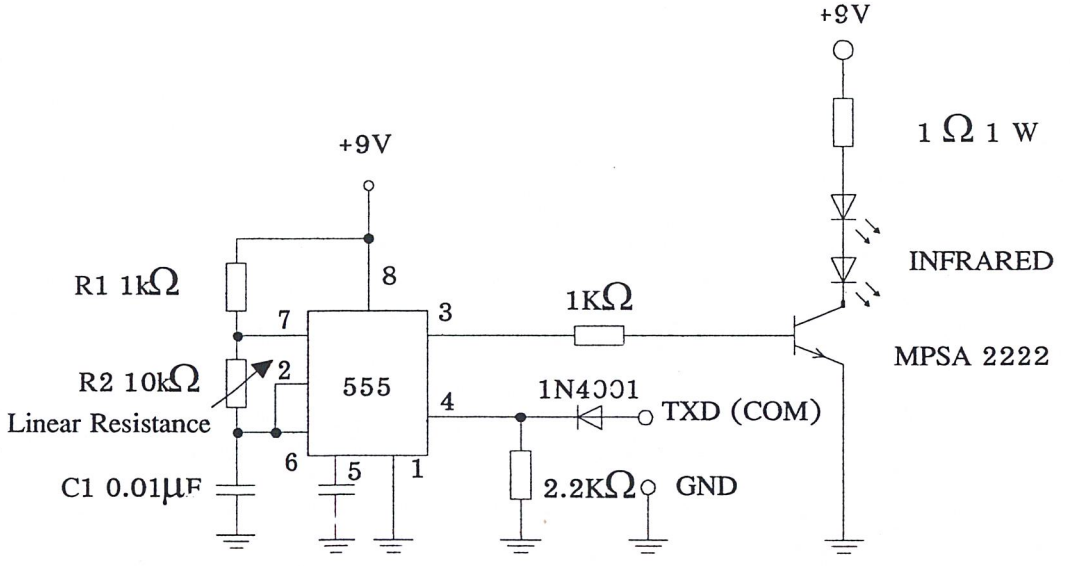


รูปที่ 3.7 วงจรควบคุมการแสดงผล

3.2.5 การออกแบบส่วนภาคส่งข้อมูล



รูปที่ 3.8 บล็อกไดอะแกรมภาคส่งข้อมูล



รูปที่ 3.9 วงจรภาคส่งข้อมูล

การออกแบบวงจรในภาคส่งโดยวงจรจะมีไอซี 555 เป็นองค์ประกอบหลักในการทำงาน การออกแบบวงจรจะเป็นวงจรอะสเตเบิลมัลติไวเบรเตอร์ ซึ่งจะต้องมีการการคำนวณค่า R_1 , R_2 C_1 ที่เหมาะสม ค่า $t_1 = 0.693\ C_1(R_1+R_2)$ และ $t_2 = 0.693\ C_1.R_2$ การที่จะกำหนดให้ Duty cycle ของ Vo มีค่า 50% คือต้องให้ $t_1 = t_2$ เมื่อดูจากสมการทั้งสองแล้วจะเห็นว่าไม่มี โอกาสเป็นไปได้เพราะ R_1 ไม่มีค่าเป็นศูนย์ ดังนั้นจึงใช้เทคนิคการต่อไดโอดปิดกั้นทางเดินของกระแสที่ C_1 ใช้ในการเก็บประจุและคายประจุ

เมื่อ C_1 เก็บประจุกระแสจะผ่าน R_1 , D_1 , C_1 ดังนั้นสมการ t_1 คือ

$$t_1 = 0.693 .C_1.R_1$$

และเมื่อ C_1 คายประจุกระแสจะผ่าน D_2 , R_2 ขา 7 (discharge) ของ ไอซี 555 ดังนั้นสมการ t_2 คือ

$$t_2 = 0.693 .C_1.R_2$$

เมื่อ $R_1 = R_2$ ดังนั้น $t_1 = t_2$ ค่า Duty cycle = 50%

การคำนวณค่า RC ในวงจร

จากส่วนรับข้อมูล TFMS 5330 จะใช้ความถี่ในการรับข้อมูล 33 กิโลเฮิร์ท และใช้แรงดันแหล่งจ่าย Vcc = + 9 V

$$T = t_1+t_2 = 1/f = 1/PRF = 1/33 \text{ kHz} = 30.30 \ \mu S$$

$$t_1 = t_2 = 30.30 \ \mu S/2 = 15.15 \ \mu S$$

เลือกใช้ค่า $C_1 = 0.01 \ \mu F$

$$R_1 = R_2 = t_1/0.693 \times C_1 = 15.15 \ \mu S/0.693 \times 0.01 \mu F$$

$$= 2.186 \ K\Omega \text{ (เลือกค่ามาตรฐาน 2 } K\Omega)$$

3.3 การทำงาน

การทำงานจะอาศัยแรงหมุนจากตัวมอเตอร์ซึ่งจะหมุนด้วยความเร็วประมาณ 1200 รอบต่อนาที และช่วงเวลาในการติดดับของตัว LED เอง ทำให้เกิดภาพซ้ำๆ กันอย่างน้อยประมาณ 16 ภาพต่อวินาที ซึ่งดวงตาของมนุษย์ยังคงจำภาพเดิมได้ทำให้เกิดเป็นภาพตัวอักษรขึ้นมาได้ โดยมีการทำงานของภาคต่าง ๆ ดังนี้

3.3.1ภาคส่งข้อมูล

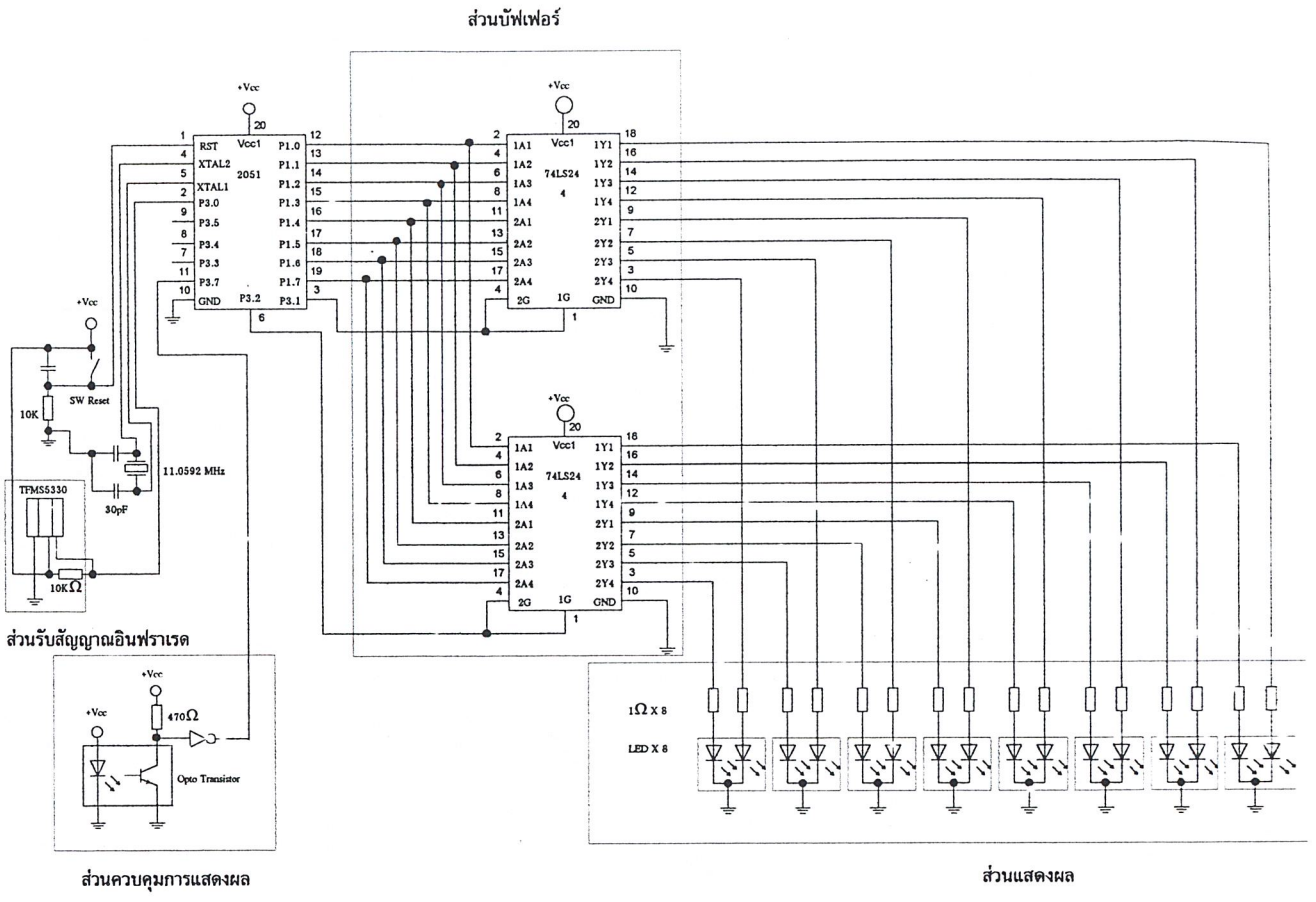
คอมพิวเตอร์จะทำการรับข้อมูล โดยป้อนข้อมูลจากคีบอร์ดของคอมพิวเตอร์ส่งออกที่ Port TXD.COM มาเข้าที่ขา 4 ของ Ic 555 ซึ่งเป็นขาที่ควบคุมการสร้างสัญญาณพัลส์ออกที่ ขา 3 ของ Ic 555 ไปควบคุมการทำงานของทรานซิสเตอร์ Q1 ทำงานโดยจะมีกระแสไหลผ่านหลอดอินฟราเรด ทำให้หลอดอินฟราเรดทำการส่งข้อมูลตามเอาต์พุตที่ออกมาจาก Port ของคอมพิวเตอร์ โดยคอมพิวเตอร์จะส่งข้อมูลออกมาครั้งละ 10 บิต โดยมีบิตเริ่มต้น 1บิต บิตข้อมูล 8 บิต และบิตสิ้นสุด 1 บิต ซึ่งใช้ความเร็วในการส่ง 2400 บิตต่อวินาที

3.3.2 การทำงานภาครับข้อมูล

จะใช้ ไอซีไมโครคอนโทรลเลอร์ AT89C2051 เป็นตัวประมวลผล โดยที่หลอดอินฟราเรดจะส่งสัญญาณมาในรูปของสัญญาณแสงอินฟราเรด ซึ่งไม่สามารถมองเห็นได้ด้วยตาเปล่า มาเข้าที่ โ ฟโต โมดูล ซึ่งทำหน้าที่เปลี่ยนจากพลังงานแสงที่รับเข้ามาเป็นพลังงานไฟฟ้าเข้าที่ขา RXD ของไมโครคอนโทรลเลอร์ AT 89C2051 ซึ่งขา RXD นี้เป็นขาที่ใช้สำหรับรับข้อมูลแบบอนุกรมเข้ามาเก็บอยู่ในไมโครคอนโทรลเลอร์ แล้วทำการเปรียบเทียบข้อมูลที่รับมากับหน่วยความจำของโปรแกรม เช่น ถ้ารับข้อมูลเป็นเลขฐาน 16 จำนวน 8 บิต เข้ามา เช่น "A" มีค่าเท่ากับ 41H ถ้าข้อมูลที่รับเข้ามามีค่าเท่ากับ 41H ก็จะทำการส่งค่าที่เป็นรหัสของอักษร "A" ออกที่ Port P1.0-P1.7 ของไมโครคอนโทรลเลอร์

3.3.3 การทำงานภาคแสดงผล

จะใช้ Ic 74LS244 ซึ่งทำหน้าที่เป็น Buffer 2 ชุด ต่อขนานกันทั้งทางด้านอินพุต และทางเอาต์พุต โดยชุดแสดงผลนี้ จะอาศัยการหมุนของมอเตอร์กระแสตรง เป็นองค์ประกอบในการแสดงผล คือ ไมโครคอนโทรลเลอร์จะทำการตรวจสอบก่อนว่า ตอนนี้แขนของชุดแสดงผลหมุนอยู่ในตำแหน่งที่เรากำหนดหรือยัง โดยใช้ออปโต้ทรานซิสเตอร์ คือถ้าออปโต้ทรานซิสเตอร์รับแสงจะทำให้เอาต์พุตเป็น "0" จากนั้นจึงนำส่งสัญญาณไปผ่าน NOT GATE เพื่อให้เอาต์พุตที่ออกมาเป็น "1" ให้กับไมโครคอนโทรลเลอร์ทำให้ไม่มีการส่งข้อมูลออกจากพอร์ต P1.0-P1.7 เมื่อออปโต้ทรานซิสเตอร์ไม่ได้รับแสงจะทำให้เอาต์พุตของออปโตเป็น "1" จากนั้นจึงนำส่งสัญญาณไปผ่าน NOT GATEเพื่อให้เอาต์พุตที่ออกมา "0" ให้กับไมโครคอนโทรลเลอร์ทำให้มีการส่ง ข้อมูลออกจากพอร์ต P1.0-P1.7 ไปเข้าอินพุตของ 74LS244 ทั้ง 2 ชุดพร้อมกัน โดยมีพอร์ต P3.1 และ P3.2 เป็นตัวกำหนดการทำงานของ Ic 74LS244 เป็นตัวควบคุมว่าจะให้ LED ติดเป็นสีที่ระบุตาม Port โดยการป้อนลอจิก "0" ให้ขาเกทของ Ic 74LS244 ชุดนั้นจะทำงานถ้าป้อนลอจิก"1" จะเป็น High Impedance Ic 74LS244 จะ ไม่ทำงาน ได้เพื่อให้เอาต์พุตที่ออกมา "0" ให้กับไมโครคอนโทรลเลอร์ทำให้มีการส่ง ข้อมูลออกจากพอร์ต P1.0-P1.7 ไปเข้าอินพุตของ 74LS244 ทั้ง 2 ชุดพร้อมกัน โดยมีพอร์ต P3.1 และ P3.2 เป็นตัวกำหนดการทำงานของ Ic 74LS244 เป็นตัวควบคุมว่าจะให้ LED ติดเป็นสีที่ระบุตาม Port โดยการป้อนลอจิก "0" ให้ขาเกทของ Ic 74LS244 ชุดนั้นจะทำงานถ้าป้อนลอจิก"1" จะเป็น High Impedance Ic 74LS244 จะ ไม่ทำงาน



รูปที่3.10 วงจรภาครับและภาคแสดงผล

บทที่4
บทสรุป

4.1 การทดลอง

เมื่อทำการทดลองในส่วนของวงจรในแต่ละส่วนแล้ว นำวงจรแต่ละภาคมาต่อร่วมกันจะได้เป็นหลอดไฟแสดงผลควบคุมผ่านอินฟราเรด และเมื่อทำการทดลองการทำงานของโครงงานจะได้ผลดังนี้

4.1.1 ผลการสร้างโครงสร้าง

ในการสร้างโครงสร้างเราเลือกใช้มอเตอร์กระแสตรงเป็นตัวหมุนก้านหมุน ส่วนของการส่งกระแสไฟฟ้าจากแหล่งจ่ายด้านล่างไปส่วนวงจรการแสดงผลเราเลือกใช้แปลงถ่าน 2 อัน โดยมี วงแหวนทองแดงทำหน้าสัมผัสกับแปลงถ่านเพื่อนำกระแสไฟไปเลี้ยงวงจรที่หมุนอยู่ จากการทดลองหมุนพบว่า เมื่อจ่ายกระแสไฟผ่านแปลงถ่านขณะที่มอเตอร์หมุนอยู่วงจรแสดงผลยังคงแสดงผลได้

การแสดงผลตัวอักษรเริ่มจากต่อวงจรเพื่อทดสอบการแสดงผลให้หลอดไฟ LED ทั้ง 8 หลอดติด เมื่อเริ่มจ่ายไฟให้มอเตอร์โดยเริ่มจาก 0 โวลต์ ทำให้ก้านแสดงผลหมุนไปตัดกับแผ่นทึบแสงที่ติดตั้งไว้เพื่อกั้น ออปโต้ ทรานซิสเตอร์ ให้มีการเช็คสวิตช์ทุกๆรอบที่หมุน ทำให้เมื่อมีการอินเตอร์รัปต์หลอด LED จะติดสว่าง จากนั้นจึงปรับแหล่งจ่ายไฟเพิ่มขึ้นเลื่อยๆ จนกว่าจะสังเกตเห็นหลอด LED ติดเป็นเส้นตรงชัดเจน

ผลการทดลองด้านโครงสร้าง

แรงดันที่จ่ายให้มอเตอร์ในการหมุนจนหลอดไฟ LED ติดเป็นเส้นตรงอยู่ในช่วงประมาณ 9-10 โวลต์

4.1.2 ผลการทดลองของส่วนแสดงผล

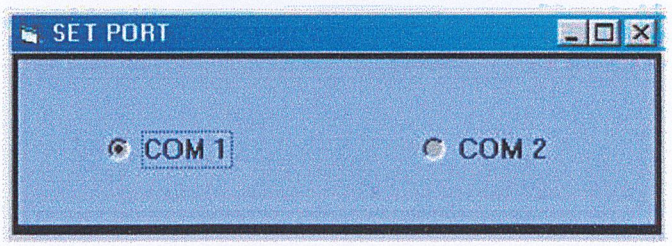
เมื่อทำการเปิดเครื่องข้อมูลที่เก็บอยู่ในหน่วยความจำของ MCS-51 จะแสดงข้อมูลทุกครั้งเมื่อยังไม่มีการส่งข้อมูลจากคอมพิวเตอร์

4.1.3 การทดลองการทำงานของโปรแกรมวิชวลเบสิก

เมื่อทำการรันโปรแกรมวิชวลเบสิกจะพบหน้าจอแสดงดังรูปที่ 4.1 โดยการทำงานของโปรแกรมจะต้องมีการเลือกพอร์ตในการใช้งานในการต่อสายส่งไปยังภาคส่งข้อมูลที่จะเลือกใช้พอร์ตอนุกรมพอร์ตใด โดยทำการคลิกไปที่ตัวอักษร Set Port ตรงมุมขวาล่างของรูปที่ 4.1

รูปที่ 4.1 แสดงหน้าจอโปรแกรมวิชวลเบสิก

เมื่อคลิกแล้วจะพบรูปที่ 4.2จากนั้นจึงเลือกคลิกตามพอร์ตที่เราต่อตัวส่งข้อมูล โดย ดับเบิ้ลคลิกตามพอร์ตที่ต้องการ



รูปที่ 4.2 แสดงการเลือกใช้พอร์ตในการส่งข้อมูล

จากนั้นจึงทำการควบคุมการทำงานของหลอดไฟแสดงผลแบบหมุนได้ การทำงานจะแบ่งประเภทการควบคุมการแสดงตัวอักษรได้ 3 ประเภทคือ แบบตัวอักษรวิ่ง แบบตัวอักษรหยุด และแบบหยุดเริ่มต้น โดยผู้ใช้จะต้องเลือกการควบคุมอย่างใดอย่างหนึ่ง การเลือกสีในการแสดงผลจะมีอยู่ 3 สีคือ สีแดง สีส้ม และสีเขียว โดยการส่งข้อความตัวอักษรในแต่ละครั้งจะเลือกสีในการแสดงผลได้เพียงสีเดียว การแสดงเวลานั้นจะต้องคลิกที่แสดงเวลาโดยเมื่อคลิกไปแล้วจะไม่สามารถกดปุ่มใดๆได้ โดยโปรแกรมจะแสดงเวลาไปตลอดจนกว่าจะคลิกที่ปุ่มแสดงเวลาอีกครั้ง จึงจะสามารถส่งข้อความได้ดังเดิม การส่งข้อความตัวอักษร จะสามารถส่งตัวอักษรภาษาอังกฤษบนคีย์บอร์ดได้ทุก

ตัว โดยในการส่งแต่ละครั้งจะส่งตัวอักษรได้ความยาวไม่เกิน 100 ตัวอักษร การส่งข้อความจะต้องคลิกที่บล็อกข้อความและพิมพ์ตัวอักษรในบล็อกนั้นจากนั้นเมื่อจะส่งข้อความก็คลิกที่ปุ่มส่งข้อความข้อความก็จะถูกส่งไปยังส่วนแสดงผล

สรุปผลการทดลองการทำงานของโปรแกรมวิชวลเบสิก

จากการทดลองการทำงาน พบว่าการควบคุมการแสดงผลผ่านโปรแกรมวิชวลเบสิก เมื่อทดลองส่งข้อมูลผ่านภาคส่งข้อมูลไปยังภาคแสดงผล ภาคแสดงผลสามารถรับข้อมูลและแสดงผลได้ตรงตามที่ทำการทดลองโดยสามารถเปลี่ยน ข้อความ เลือกสีในการแสดงผล และเลือกลักษณะการแสดงผลให้วิ่งหรือหยุดนิ่งได้ตรงตามที่ทดลอง

4.1.4 ผลการทดลองการติดตั้งตัวรับ TFMS 5330

จากการทดลองในการรับข้อมูลโดยมีการติดตั้งตัวรับข้อมูล TFMS 5330 ไว้ 2 แบบคือ แบบตั้ง และแบบนอนซึ่งผลที่ได้ต่างกัน จึงต้องทำการทดลองเพื่อหาการติดตั้งที่ดีที่สุดในการติดตัวรับโดยได้ผลการทดลองดังนี้

การทดลองติดตั้งตัวรับแบบตั้ง

จากการทดลองในการส่งข้อมูล เมื่อทำการส่งข้อมูลไปยังส่วนแสดงผล โดยเราทำการติดตั้งตัวรับแบบตั้งพบว่า

ระยะในการส่ง (เมตร)	ผลการทดลอง	เปอร์เซ็นต์ผิดพลาดการส่งข้อความ	เปอร์เซ็นต์ผิดพลาดการเปลี่ยนสี	เปอร์เซ็นต์ผิดพลาดการส่งลักษณะการแสดงผล
1	ได้	1%	0%	0%
2	ได้	3%	0%	0%
3	ได้	5%	0%	0%
4	ได้	7%	0%	0%
5	ได้	8%	0%	0%
6	ได้	9%	0%	0%
7	ได้	11%	0%	0%
8	ไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้
9	ไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้

ตารางที่ 4.1 แสดงผลการทดลองการติดตั้งตัวรับ TFMS 5330 แบบตั้ง

จากการทดลองการส่งข้อมูลพบว่าการติดตั้งตัวรับข้อมูลแบบนี้เกิดความผิดพลาดขึ้น เมื่อระยะในการรับส่งข้อมูลเพิ่มขึ้นความผิดพลาดของข้อมูลที่แสดงผลก็มีเพิ่มตามด้วย

การทดลองติดตั้งตัวรับแบบนอน

จากการทดลองในการส่งข้อมูล เมื่อทำการส่งข้อมูลไปยังส่วนแสดงผล โดยเราทำการติดตั้งตัวรับแบบนอนพบว่า

ระยะในการส่ง (เมตร)	ผลการทดลอง	เปอร์เซ็นต์ผิดพลาดการส่งข้อความ	เปอร์เซ็นต์ผิดพลาดการเปลี่ยนสี	เปอร์เซ็นต์ผิดพลาดการส่งลักษณะการแสดงผล
1	ได้	0%	0%	0%
2	ได้	0%	0%	0%
3	ได้	0%	0%	0%
4	ได้	0%	0%	0%
5	ได้	0%	0%	0%
6	ไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้
7	ไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้
8	ไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้
9	ไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้	รับข้อมูลไม่ได้

ตารางที่ 4.2 แสดงผลการทดลองการติดตั้งตัวรับ TFMS 5330 แบบนอน

จากการทดลองการส่งข้อมูลพบว่าการติดตั้งตัวรับข้อมูลแบบนี้เกิดความผิดพลาดน้อยกว่าการติดตั้งตัวรับแบบตั้ง เมื่อระยะในการรับส่งข้อมูลอยู่ในระยะ 3 เมตร จะเห็นว่าการรับส่งข้อมูลไม่เกิดความผิดพลาดขึ้นเลย แต่ในการติดตั้งตัวรับแบบนี้ระยะทางในการรับส่งจะลดลง

สรุปผลการทดลอง TFMS 5330

จากการทดลองพบว่าการติดตั้งตัวรับทั้ง 2 แบบ สามารถรับข้อมูลได้ ซึ่งจากการทดลองเราเลือกการติดตั้งแบบนอนเพราะแม้ระยะทางในการรับส่งข้อมูลจะไม่มากนัก แต่ความผิดพลาดของข้อมูลในการรับมีน้อยมากเราจึงเลือกการติดตั้งตัวรับ TFMS 5330 แบบนอน

4.2 บทสรุป

จากที่ได้ทำโครงงานหลอดไฟแสดงผลแบบหมุนควบคุมผ่านอินฟราเรด และนำมาทดลองใช้งานสามารถสรุปผลของโครงงานได้ดังนี้

โครงงานหลอดไฟแสดงผลแบบหมุนควบคุมผ่านอินฟราเรด สามารถทำงานได้ทั้งแบบแสดงข้อความอัตโนมัติเมื่อเริ่มเปิดเครื่องและควบคุมการทำงานผ่านโปรแกรมวิชวลเบสิก โดยสามารถเลือกสีในการแสดงผล เลือกลักษณะการแสดงผลให้ตัวอักษรวิ่งหรือหยุดนิ่ง แสดงเวลา และเปลี่ยนข้อความแสดงผลได้โดยการควบคุมผ่านคีย์บอร์ดจากคอมพิวเตอร์ ซึ่งการส่งข้อมูลจะใช้การรับส่งข้อมูลแบบอนุกรมผ่านพอร์ตอนุกรมของคอมพิวเตอร์ โดยเลือกใช้ชุดรับส่งอินฟราเรดในการรับส่งข้อมูล การแสดงผลจะอาศัยการหมุนของมอเตอร์และการกระพริบของหลอดไฟทำให้เกิดเป็นตัวอักษร ซึ่งเมื่อทำโครงงานสำเร็จและทดลองใช้เครื่องสามารถแสดงผลได้ดีและรับส่งข้อมูลถูกต้อง

4.3 ปัญหาและวิธีแก้ไข

จากที่ได้ทำโครงงานและนำมาทดลองใช้งานได้พบปัญหาหลายประการสามารถสรุปปัญหาของโครงงานได้ดังนี้

1.การส่งกระแสไฟไปเลี้ยงวงจรหลักด้านบนนั้นทำให้มีแรงดันไฟที่แปลงจากส่วนล่าง 5 โวลต์ เมื่อผ่านไปถึงวงจรแล้วเหลือ 3.5 โวลต์ ทำให้ไม่พอไปเลี้ยงวงจรหลักด้านบนเนื่องจากมีการสูญเสียระหว่างทาง เพราะฉนั้นควรที่จะส่งกระไฟไปมากกว่าที่วงจรใช้จริง เช่นส่งกระแสไฟไป 12 โวลต์ และทำชุดแปลงแรงดันไฟ 5 โวลต์ ไว้แปลงแรงดันไฟก่อนส่งเข้าไปเลี้ยงวงจรหลัก ทำเช่นนี้จะทำให้กระแสไฟที่ให้กับวงจรคงที่และเป็นการป้องกันไฟกระตุกได้

2.ในส่วนของการส่งกระแสไฟไปเลี้ยงวงจรที่หมุนอยู่ตลอดเวลา เมื่อมอเตอร์หมุนในความเร็วรอบที่สูงทำให้ไม่มีกระแสไฟไปเลี้ยงวงจรหลักด้านบน ดังนั้นควรที่จะใช้แปลงถ่าน เพราะถ้าใช้วัสดุอื่น เมื่อมอเตอร์หมุนในความเร็วรอบที่สูงอาจทำให้ไม่สามารถส่งกระแสไฟไปเลี้ยงวงจรหลักได้

3.การติดตั้งตัวรับ TFMS 5330 ในแนวตั้งทำให้ข้อมูลที่รับเกิดความผิดพลาดมาก เพราะฉนั้นควรที่จะติดตั้งแบบแนวนอน เพราะจะทำให้การผิดพลาดของข้อมูลในการรับส่งข้อมูลมีน้อย

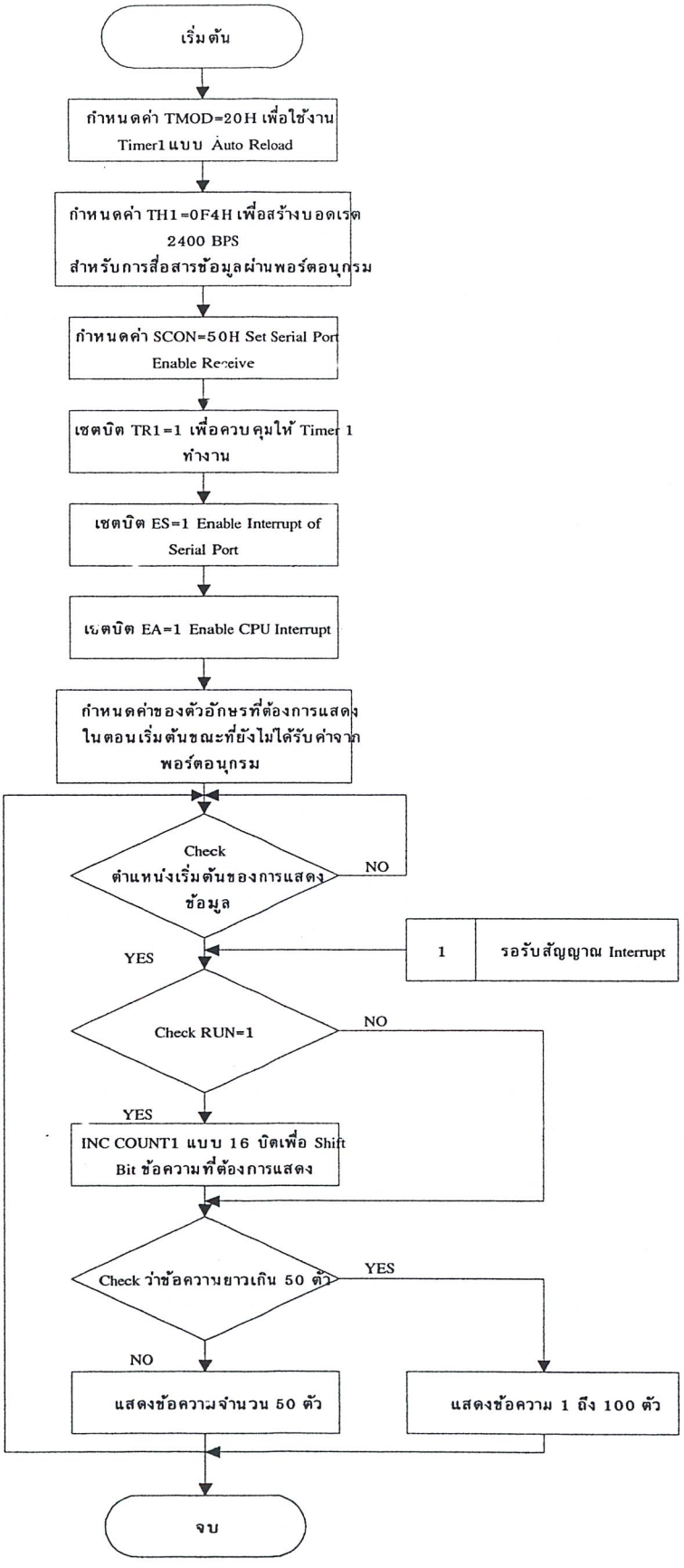
4.ไม่สามารถรับส่งข้อมูลไปยังภาครับได้เมื่อมอเตอร์หมุนในความเร็วรอบสูง เนื่องจาก ภาคส่งข้อมูลมีกำลังส่งน้อยและความถี่ของภาครับส่งข้อมูลไม่ตรงกัน เพราะฉนั้นควรใช้อุปกรณ์ที่มีความเที่ยงตรงสูง โดยผู้จัดทำใช้ Linear Resistance เพื่อปรับความถี่ให้ตรงกับภาครับทำให้สามารถการรับส่งข้อมูลได้ระยะไกลและเกิดความผิดพลาดของข้อมูลน้อย

4.4 ประโยชน์ในการนำไปใช้งาน

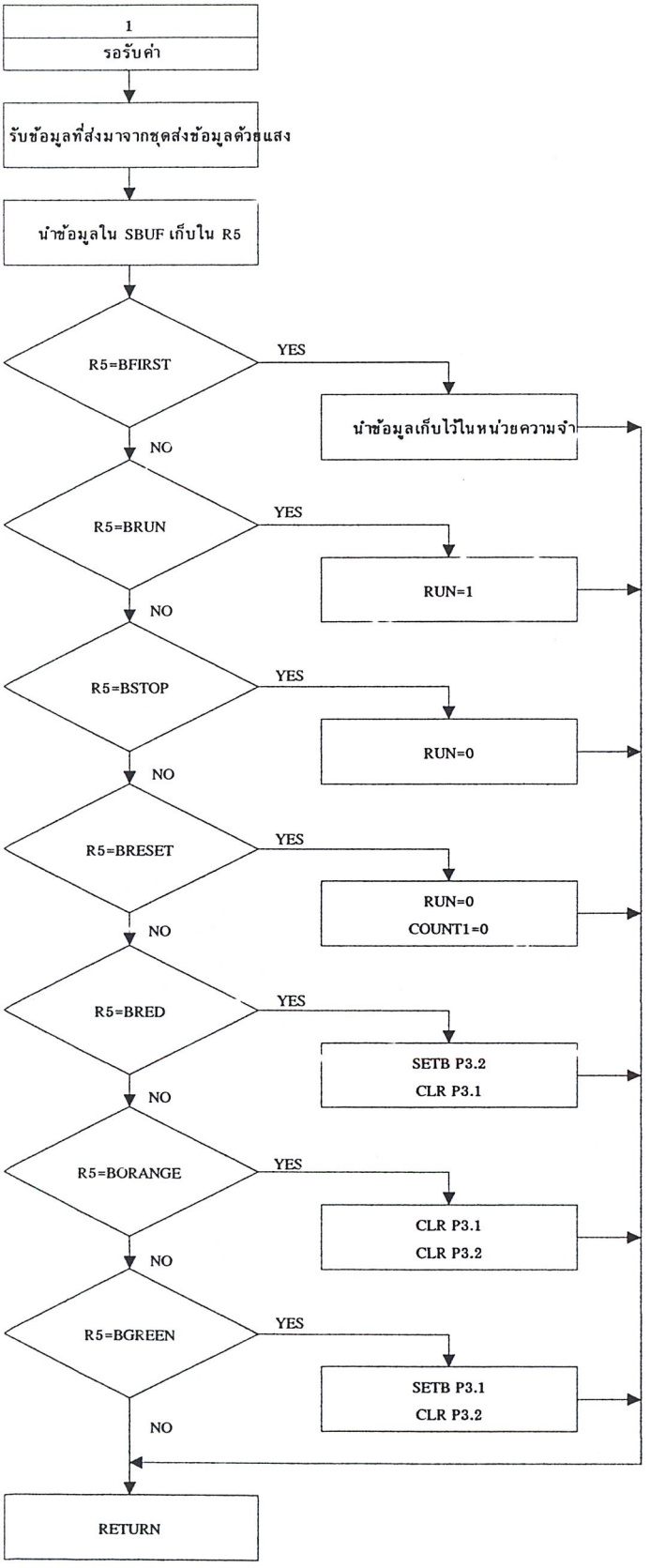
สามารถนำไปใช้งานในด้านการประชาสัมพันธ์นำเสนอข้อมูลข่าวสารต่างสู่สาธารณะชนซึ่งมีประสิทธิภาพใกล้เคียงกับสื่อที่ใช้ในปัจจุบันแต่จะมีข้อได้เปรียบในการประหยัดพลังงานได้อย่างมากและยังสามารถดึงความสนใจของผู้พบเห็นได้อีกด้วย

ภาคผนวก

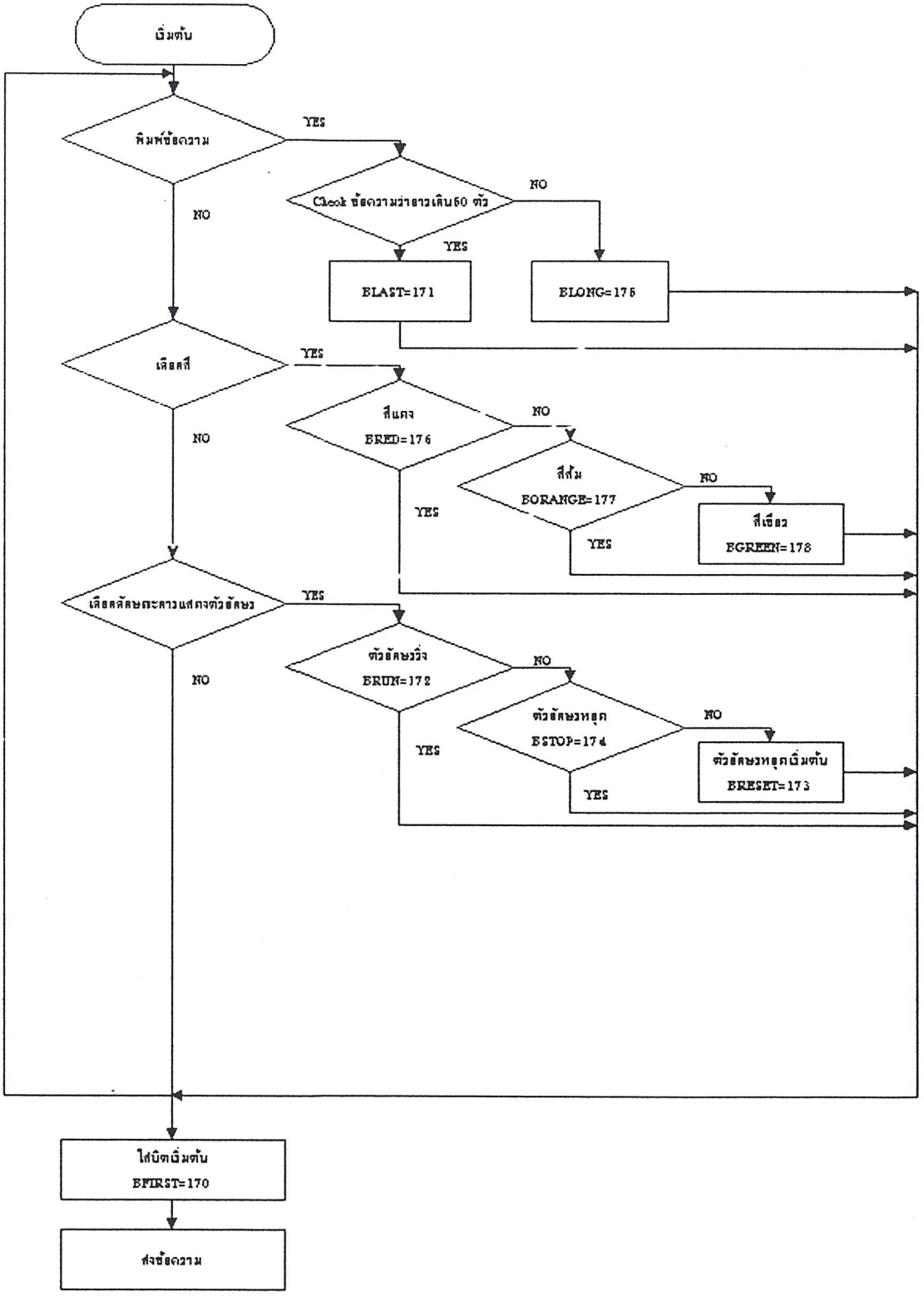
Flowchart การทำงานของโปรแกรมหลัก MCS-51



Flowchart การทำงานของโปรแกรมย่อยในการรับข้อความ MCS-51



Flowchart การทำงานของโปรแกรมวิชวลเบสิก



โปรแกรมการทำงานของ MCS-51

```
N  EQU 0FH
        LONG EQU 13H
      COLUMN EQU 14H
     COUNT10 EQU 15H
     COUNT11 EQU 16H
     COUNT20 EQU 17H
     COUNT21 EQU 18H
     COUNT30 EQU 19H
     COUNT31 EQU 1AH
         RUN EQU 1BH
        TXT1 EQU 1CH
      TXT100 EQU 80H
             ORG 0000H
             JMP MAIN
             ORG 0023H
             JMP S_INT
             ORG 0050H
MAIN:          CLR P3.1          ;****ORANGE****;
               CLR P3.2          ;****ORANGE****;
               MOV TMOD,#20H      ;SET MODE TIME 1 AUTO RELOAD 8 BIT
               MOV TH1,#0F4H      ;BAUD RATE =2400
               MOV SCON,#50H      ;SERIAL MODE 1
               SETB EA           ;OPEN INTERUPT
               SETB ES           ;ENABLE SERIAL INTERUPT
               SETB TR1          ;START TIMER
               MOV COUNT10,#00H    ;CLEAR COUNT1
               MOV COUNT11,#00H    ;CLEAR COUNT1
               MOV COUNT20,#00H    ;CLEAR COUNT2
               MOV COUNT21,#00H    ;CLEAR COUNT2
               MOV COUNT30,#00H    ;CLEAR COUNT3
               MOV COUNT31,#00H    ;CLEAR COUNT3
               MOV N,#00H        ;CLEAR N
               MOV RUN,#0        ;CLEAR RUN
               MOV FIRST,#00H     ;CLEAR FIRST
               MOV R1,#TXT1       ;SET FIRST ADDRESS WORD
```

```
MOV  LONG,#00          ;CLEAR LONG
MOV  1CH,#53H          ;R
MOV  1DH,#50H          ;O
MOV  1EH,#45H          ;T
MOV  1FH,#43H          ;A
MOV  20H,#41H          ;T
MOV  21H,#49H          ;A
MOV  22H,#4CH          ;B
MOV  23H,#20H          ;L
MOV  24H,#50H          ;L
MOV  25H,#52H          ;
MOV  26H,#4FH          ;1
MOV  27H,#4AH          ;
MOV  28H,#45H          ;C
MOV  29H,#43H          ; O
MOV  2AH,#54H          ; L
MOV  2BH,#20H          ;U
MOV  2CH,#4FH          ; M
MOV  2DH,#46H          ;N
MOV  2EH,#20H
MOV  2FH,#4BH          ;K
MOV  30H,#49H          ;I
MOV  31H,#4EH          ;N
MOV  32H,#47H          ;G
MOV  33H,#20H
MOV  34H,#4DH          ;M
MOV  35H,#4FH          ;O
MOV  36H,#4EH          ;N
MOV  37H,#47H          ;G
MOV  38H,#4BH          ;K
MOV  39H,#55H          ;U
MOV  3AH,#54H          ;T
MOV  3BH,#27H          ;'
MOV  3CH,#53H          ;S
MOV  3DH,#20H
MOV  3EH,#49H          ;I
```

```
MOV 3FH,#4EH        ;N
MOV 40H,#53H        ;S
MOV 41H,#54H        ;T
MOV 42H,#49H        ;I
MOV 43H,#54H        ;T
MOV 44H,#55H        ;U
MOV 45H,#54H        ;T
MOV 46H,#45H        ;E
MOV 47H,#20H
MOV 48H,#4FH        ;O
MOV 49H,#46H        ;F
MOV 4AH,#20H
MOV 4BH,#54H        ;T
MOV 4CH,#45H        ;E
MOV 4DH,#43H        ;C
MOV 4EH,#48H        ;H
MOV 4FH,#4EH        ;N
MOV 50H,#4FH        ;O
MOV 51H,#4CH        ;L
MOV 52H,#4FH        ;O
MOV 53H,#47H        ;G
MOV 54H,#59H        ;Y
MOV 55H,#20H
MOV 56H,#4EH        ;L
MOV 57H,#4FH        ;A
MOV 58H,#52H        ;D
MOV 59H,#54H        ;K
MOV 5AH,#48H        ;R
MOV 5BH,#20H        ;A
MOV 5CH,#42H        ;B
MOV 5DH,#41H        ;A
MOV 5EH,#4EH        ;N
MOV 5FH,#47H        ;G
MOV 60H,#4BH        ;
MOV 61H,#4FH        ;B
MOV 62H,#4BH        ;Y
```

```
                MOV 63H,#20H            ;
                MOV 64H,#42H            ;3
                MOV 65H,#59H            ;R
                MOV 66H,#20H            ; 1
                MOV 67H,#4DH            ;*
                MOV R0,#7CH
ABCD:            MOV @R0,#32            ;SPACE
                INC R0
                CJNE R0,#TXT100,ABCD    ;CHECK 100 WORD
                MOV R0,#TXT1
LOOP0:           MOV A,P3                       ;CHECK START POSITION
                JB P3.7,LOOP0
LOOP1:           MOV A,P3                       ;CHECK START POSITION
                JNB P3.7,LOOP1
                INC N                   ;COUNT ROUND OF MOTER
                MOV A,N
                CJNE A,#1,SUB0          ;CHECK ROUND OF MOTER TO BEGIN SHIFT BIT
                MOV N,#00H              ;CLEAR N
                MOV A,RUN
                CJNE A,#1,SUB0          ;CHECK MODE RUN
                MOV A, COUNT10          ;**********************;
                ADD A,#1                ; INC COUNT1 16BIT  ;
                MOV COUNT10,A           ;                   ;
                MOV A,COUNT11           ;   TO SHIF BIT     ;
                ADDC A,#0               ;                   ;
                MOV COUNT11,A           ;**********************;
SUB0:            MOV A,LONG
                CJNE A,#0,LOOP9          ; CHECK 100 DATA JUMP
                JMP L1
LOOP9:           JMP LONG0
L1:             MOV A,COUNT11            ;**********************  ;
                CJNE A,#2,SUB1          ; CHECK COUNT1 16 BIT ;
                MOV A,COUNT10            ;        = 600        ;
                CJNE A,#58H,SUB1        ;********************** ;
                MOV COUNT10,#00H         ;*****CLEAR COUNT ****;
                MOV COUNT11,#00H         ;******** 16 BIT   ******;
```

```
SUB1:    MOV A,COUNT10        ;************************;
         MOV COUNT20,A        ; COUNT1=COUNT2      ;
         MOV A,COUNT11        ;                    ;
         MOV COUNT21,A        ;************************;
         MOV COUNT30,#00H     ;****** AR COUNT3 *****;
         MOV COUNT31,#00H     ;******  16 BIT   *******;
SUB2:    MOV A,COUNT21        ;*************************;
         SWAP A               ;                    ;
         MOV 12H,A            ;                    ;
         MOV A,COUNT20        ;                    ;
         ANL A,#111!0000B     ;                    ;
         SWAP A               ;                    ;
         ADD A,12H            ;                    ;
         MOV B,#6             ; DIV COUNT2 16BIT   ;
         DIV AB               ;
         SWAP A               ;        6           ;
         MOV 11H,A            ;                    ;
         MOV A,B              ;                    ;
         SWAP A               ;                    ;
         MOV 12H,A            ;                    ;
         MOV A,COUNT20        ;                    ;
         ANL A,#00001111B     ;                    ;
         ADD A,12H            ;                    ;
         MOV B,#6             ;                    ;
         DIV AB               ;                    ;
         ADD A,11H            ;*************************;
         ADD A,#TXT1
         MOV COLUMN,B         ;COLUMN = COUNT2 16BIT/6
         MOV R0,A             ;R0 = #TXT1+( COUNT2 16 BIT /6)
         MOV A,COUNT20        ;*************************;
         ADD A,#1             ;                    ;
         MOV COUNT20,A        ;                    ;
         MOV A,COUNT21        ; INC COUNT2 16 BIT  ;
         ADDC A,#0            ;                    ;
         MOV COUNT21,A        ;*************************;
         MOV A,COUNT30        ;*************************;
```

```
        ADD A,#1                ;                       ;
        MOV COUNT30,A           ;                       ;
        MOV A,COUNT31           ;  INC COUNT3 16 BIT    ;
        ADDC A,#0               ;                       ;
        MOV COUNT31,A           ;************************;
        CALL SHOW               ; SHOW DATA
        MOV A,COUNT21           ;************************;
        CJNE A,#2,SUB3          ; CHECK COUNT2 16 BIT ;
        MOV A,COUNT20           ;        =600           ;
        CJNE A,#58H,SUB3        ;************************;
        MOV COUNT20,#00H
        MOV COUNT21,#00H
SUB3:   MOV A,COUNT31           ;************************;
        CJNE A,#1,SUB2          ; CHECK COUNT3 16 BIT  ;
        MOV A,COUNT30           ;        =300           ;
        CJNE A,#2CH,SUB2        ;************************;
        MOV P1,#00H
        JMP LOOP0
LONG0:  MOV A,COUNT11           ;************************;
        CJNE A,#2,LONG1         ; CHECK COUNT1 16 BIT ;
        MOV A,COUNT10           ;        = 600          ;
        CJNE A,#58H,LONG1       ;************************;
        MOV COUNT10,#00H
        MOV COUNT11,#00H
LONG1:  MOV A,COUNT10           ;************************;
        MOV COUNT20,A           ; COUNT1=COUNT2        ;
        MOV A,COUNT11           ;                       ;
        MOV COUNT21,A           ;************************;
        MOV COUNT30,#00H
        MOV COUNT31,#00H
LONG2:  MOV A,COUNT21           ;************************;
        SWAP A                  ;                       ;
        MOV 12H,A               ;                       ;
        MOV A,COUNT20           ;                       ;
        ANL A,#11110000B        ;                       ;
        SWAP A                  ;                       ;
```

```
        ADD A,12H               ;                       ;
        MOV B,#6                ; COUNT2 16BIT          ;
        DIV AB                  ;          DIV          ;
        SWAP A                  ;           6           ;
        MOV 11H,A               ;                       ;
        MOV A,B                 ;                       ;
        SWAP A                  ;                       ;
        MOV 12H,A               ;                       ;
        MOV A,COUNT20           ;                       ;
        ANL A,#00001111B        ;                       ;
        ADD A,12H               ;                       ;
        MOV B,#6                ;                       ;
        ADD A,11H               ;**********************;
        ADD A,#TXT1
        MOV COLUMN,B            ;COLUMN =COUNT2 16BIT / 2
        MOV R0,A                ;R0 = #TXT1+( COUNT2 16 BIT /6)
        MOV A,COUNT20           ;**************************;
       ADD A,#1                 ;                       ;
        MOV COUNT20,A           ; INC COUNT2 16BIT;
        MOV A,COUNT21           ;                       ;
        ADDC A,#0               ;                       ;
        MOV COUNT21,A           ;**************************;
        MOV A,COUNT30           ;**************************;
        ADD A,#1                ;                       ;
        MOV COUNT30,A           ; INC COUNT3 16BIT      ;
        MOV A,COUNT31           ;                       ;
        ADDC A,#0               ;                       ;
        MOV COUNT31,A           ;**************************;
        CALL SHOW               ;SHOW DATA
        MOV A,COUNT21           ;**************************;
        CJNE A,#2,LONG3         ; CHECK COUNT2 16BIT    ;
        MOV A,COUNT20           ;          =600         ;
        CJNE A,#58H,LONG3       ;**************************;
        MOV COUNT20,#00H
        MOV COUNT21,#00H
LONG3:  MOV A,COUNT31           ;**************************;
```

```
        CJNE A,#2,LONG2          ; CHECK COUNT3 16BIT  ;
        MOV A,COUNT30            ;          =600       ;
        CJNE A,#58H,LONG2        ;************************;
        MOV P1,#00H
        JMP LOOP0
SHOW:   MOV DPTR,#TEXT
        MOV B,#5
        MOV A,@R0
        MUL AB                   ;DATA MUL 5
        ADD A,DPL                ;************************;
        MOV DPL,A                ;     DATA MUL 5       ;
        MOV A, B                 :       +TEXT          ;
        ADDC A,DPH               ;                      ;
        MOV DPH,A                ;************************;
        MOV A,COLUMN
        CJNE A,#5,S1             ;CHECK COLUMN=5
        JMP S2
S1:     MOVC A,@A+DPTR
        MOV P1,A
        CALL WAIT
        RET
S2:     MOV P1,#00H
        CALL WAIT
        RET
WAIT:   MOV R6,#140
W1:     DJNZ R6,$
        RET
        FIRST EQU 10H            ;ADDRESS OF FIRST
        BFIRST EQU 170           ;BYTE FOR FIRST STRING
        BLAST EQU 171            ;BYTE FOR LAST STRING
        BRUN EQU 172             ;BYTE FOR COMMAND RUN
        BRESET EQU 173           ;BYTE FOR COMMAND RESET
        BSTOP EQU 174            ;BYTE FOR COMMAND STOP
        BLONG EQU 175            ;BYTE FOR COMMAND 40 WORD
        BRAD EQU 176             ;BYTE FOR COLOR RAD
        BORANGE EQU 177          ;BYTE  FOR COLOR ORANGE
```

```
            BGREEN  EQU 178          ;BYTE FOR COLOR GREEN
S_INT:      MOV R5,FIRST
            CJNE R5,#1,CMD           ;CHECK FIRST == 1
            JMP INT2
CMD:        MOV R5,SBUF              ;GST DATA FROM SERIAL
            CJNE R5,#BFIRST,CRUN     ;CHECK DATA == BYTE FIRST
            MOV FIRST,#1             ;FIRST = 1
            JMP EX_SINT              ;EXIT INTERRUPT
CRUN:       CJNE R5,#BRUN,CRESET     ;CHECK DATA == COMMAND RUN
            MOV RUN,#1               ;SET COMMAND RUN
            JMP EX_SINT              ;EXIT INTERRUPT
CRESET:     CJNE R5,#BRESET,CSTOP    ;CHECK DATA== COMMAND RESET
            MOV COUNT10,#00H         ;RESET
            MOV COUNT11,#00H
            MOV RUN,#00H
            JMP EX_SINT              ;EXIT INTERRUPT
CSTOP:      CJNE R5,#BSTOP,CRAD      ;CHECK DATA == COMMAND STOP
            MOV RUN,#0               ;CLEAR COMMAND RUN
            JMP EX_SINT              ;EXIT INTERRUPT
CRAD:       CJNE R5,#BRAD,CORANGE    ;CHECK DATA ==COMMAND RAD
            CLR P3.1
            SETB P3.2
            JMP EX_SINT
CORANGE:    CJNE R5,#BORANGE,CGREEN  ;CHECK DATA == COMMAND ORANGE
            CLR P3.1
            CLR P3. 2
            JMP EX_SINT
CGREEN:     CJNE R5,#BGREEN,EX_SINT  ;CHECK DATA ==COMMAND GREEN
            SETB P3.1
            CLR P3.2
            JMP EX_SINT
EX_SINT:    CLR RI                   ;CLEAR RX FLAG
            RETI                     ;RETURN INTERRUPT
INT2:       MOV R5,SBUF              ;GET DATA FROM SERIAL
            CJNE R5,#BLAST,CLONG     ;CHECK DATA == BYTE LAST
            MOV LONG,#0              ;DATA WORD<=50
```

```
                CALL  CLRMEM                ;CALL FUNCTION CLEAR STRING
                JMP  EX_SINT                ;EXIT INTERRUPT
CLONG:          CJNE  R5,#BLONG,INDATA      ;CHECK DATA == BLONG
                MOV  LONG,#1                ;DATA WORD > 50
                CALL  CLRMEM                ;CALL FUNCTION CLREAR STRING
                JMP  EX_SINT                ;EXIT INTERRUPT
INDATA:         CALL  IN_MEM                ;SEND WORD TO MEMORY
                JMP  EX_SINT                ;EXIT INTERRUPT
IN_MEM:         XCH  A,R5
                MOV  @R1,A                  ;INPUT R5 TO MEMMORY
                XCH  A,R5
                INC  R1                     ;INC ADDRESS POINTER
                CJNE  R1,#TXT100,EX_INMEM   ;ADDRESS == BYTE LAST
                MOV  FIRST,#0
EX_INMEM:       RET                         ;RETURN FUNCTION
CLRMEM:         MOV  R5,#32                 ;R5 == SPACE
                CALL  IN_MEM                ;CALL FUNCTION INPUT DATA TO MEMORY
                MOV  R5,FIRST
                CJNE  R5,#0,CLRMEM          ;CHECK FIRST == 0
                MOV  R1,#TXT1               ;RESET POINTER TO POINT FIRST WORD
                RET                         ; RETURN FUNCTION
                        ORG   760
                TEXT:
                        ORG   920
AS32: DB 00000000B  AS33: DB 00000000B  AS34: DB 00000000B  AS35: DB 00100100B  AS36: DB 00110010B
      DB 00000000B      DB 01110000B      DB 11100000B      DB 11111111B      DB 01001001B
      DB 00000000B      DB 11111101B      DB 00000000B      DB 00100100B      DB 11111111B
      DB 00000000B      DB 01110000B      DB 11100000B      DB 11111111B      DB 01001001B
      DB 00000000B      DB 00000000B      DB 00000000B      DB 00100100B      DB 00100110B

AS37: DB 01100011B  AS38: DB 00000110B  AS39: DB 00000000B  AS40: DB 00111100B AS41: DB 00000000B
      DB 01100100B      DB 01101001B      DB 00000000B      DB 01000010B      DB 10000001B
      DB 00001000B      DB 10011001B      DB 10100000B      DB 10000001B      DB 10000001B
      DB 00010011B      DB 01100110B      DB 11000000B      DB 10000001B      DB 01000010B
      DB 01100011B      DB 00000001B      DB 00000000B      DB 00000000B      DB 00111100B

AS42: DB 01010100B  AS43: DB 00001000B AS44: DB 00000000B  AS45: DB 00001000B  AS46: DB 00000000B
      DB 00111000B      DB 00001000B      DB 00000000B      DB 00001000B      DB 00000000B
      DB 11111110B      DB 00111110B      DB 00000101B      DB 00001000B      DB 00000011B
```

DB 00111000B DB 00001000B DB 00000110B DB 00001000B DB 00000000B

DB 01010100B DB 00001000B DB 00000000B DB 00001000B DB 00000000B

AS47: DB 00000011B AS48: DB 01111110B AS49: DB 00000000B AS50: DB 01000011B AS51: DB 01000010B

DB 00001100B DB 10000111B DB 01000001B DB 10000101B DB 10000001B

DB 00110000B DB 10011001B DB 11111111B DB 10001001B DB 10010001B

DB 11000000B DB 11100001B DB 00000001B DB 10010001B DB 10010001B

DB 00000000B DB 01111110B DB 00000000B DB 01100001B DB 01101110B

AS52: DB 00111000B AS53: DB 11110010B AS54: DB 01111110B AS55: DB 11000000B AS56: DB 01101110B

DB 01001000B DB 10010001B DB 10010001B DB 10000000B DB 10010001B

DB 10001000B DB 10010001B DB 10010001B DB 10001111B DB 10010001B

DB 11111111B DB 10010001B DB 10010001B DB 10010000B DB 10010001B

DB 00001000B DB 10001110B DB 01001110B DB 11100000B DB 01101110B

AS57: DB 01100010B AS58: DB 00000000B AS59: DB 00000000B AS60: DB 00001000B AS61: DB 00010100B

DB 10010001B DB 01100011B DB 01100101B DB 00010100B DB 00010100B

DB 10010001B DB 01100011B DB 01100110B DB 00100010B DB 00010100B

DB 10010001B DB 00000000B DB 00000000B DB 01000001B DB 00010100B

DB 01111110B DB 00000000B DB 00000000B DB 00000000B DB 00000000B

AS62: DB 00000000B AS63: DB 01100000B AS64: DB 00100110B AS65: DB 00111111B AS66: DB 11111111B

DB 01000001B DB 10000000B DB 01000101B DB 01001000B DB 10010001B

DB 00100010B DB 10001101B DB 01000111B DB 10001000B DB 10010001B

DB 00010100B DB 10010000B DB 01000001B DB 01001000B DB 10010001B

DB 00001000B DB 01100000B DB 00111110B DB 01001000B DB 01101110B

AS67: DB 01111110B AS68: DB 11111111B AS69: DB 11111111B AS70: DB 11111111B AS71: DB 01111110B

DB 10000001B DB 10000001B DB 10010001B DB 10010000B DB 10000001B

DB 10000001B DB 10000001B DB 10010001B DB 10010000B DB 10000001B

DB 10000001B DB 01000010B DB 10010001B DB 10010000B DB 10001001B

DB 01000010B DB 00111100B DB 10010001B DB 10000000B DB 01001110B

AS72: DB 11111111B AS73: DB 00000000B AS74: DB 00000110B AS75: DB 11111111B AS76: DB 11111111B

DB 00010000B DB 10000001B DB 00000001B DB 00011000B DB 00000001B

DB 00010000B DB 11111111B DB 10000001B DB 00100100B DB 00000001B

DB 00010000B DB 10000001B DB 11111110B DB 01000010B DB 00000001B

DB 11111111B DB 00000000B DB 10000000B DB 10000001B DB 00000001B

AS77: DB 11111111B AS78: DB 11111111B AS79: DB 01111110B AS80: DB 11111111B AS81: DB 01111110B

DB 01000000B DB 01100000B DB 10000001B DB 10010000B DB 10000001B

DB 00100000B DB 00011000B DB 10000001B DB 10010000B DB 10000111B

DB 01000000B DB 00000110B DB 10000001B DB 10010000B DB 01111110B

DB 11111111B DB 00000110B DB 01111110B DB 01100000B DB 00000011B

AS82: DB 11111111B AS83: DB 01100010B AS84: DB 10000000B AS85: DB 11111110B AS86: DB 11111100B

DB 10010000B	DB 10010001B	DB 10000000B	DB 00000001B	DB 00000010B
DB 10011000B	DB 10010001B	DB 11111111B	DB 00000001B	DB 00000001B
DB 10010110B	DB 10010001B	DB 10000000B	DB 00000001B	DB 00000010B
DB 01100001B	DB 01001110B	DB 10000000B	DB 11111110B	DB 11111100B

AS87: DB 11111111B AS88: DB 11000111B AS89: DB 11100000B AS90: DB 11000011B AS91: DB 00000000B

DB 00000010B	DB 00101000B	DB 00010000B	DB 10000101B	DB 11111111B
DB 00000100B	DB 00010000B	DB 00001111B	DB 10001001B	DB 10000001B
DB 00000010B	DB 00101000B	DB 00010000B	DB 10010001B	DB 10000001B
DB 11111111B	DB 11000111B	DB 11100000B	DB 11100011B	DB 00000000B

AS92: DB 11000000B AS93: DB 00000000B AS94: DB 00100000B AS95: DB 00000001B AS96: DB 00000000B

DB 00110000B	DB 10000001B	DB 01000000B	DB 00000001B	DB 00100000B
DB 00C01100B	DB 10000001B	DB 10000000B	DB 00000001B	DB 01000000B
DB 00000011B	DB 11111111B	DB 01000000B	DB 00000001B	DB 10000000B
DB 00000000B	DB 00000000B	DB 00100000B	DB 00000001B	DB 00000000B

AS97: DB 00000010B AS98: DB 11111111B AS99: DB 00001110B AS100: DB 00001110B AS101: DB 00001110B

DB 00010101B	DB 00010001B	DB 00010001B	DB 00010001B	DB 00010101B
DB 00010101B	DB 00010001B	DB 00010001B	DB 00010001B	DB 00010101B
DB 00010101B	DB 00010001B	DB 00010001B	DB 00010001B	DB 00010101B
DB 00001110B	DB 00001110B	DB 00000000B	DB 11111111B	DB 00001100B

AS102:DB 00000000B AS103:DB 00001000B AS104:DB 11111111B AS105:DB 00000000B AS106:DB 00000000B

DB 00010000B	DB 00010101B	DB 00010000B	DB 00000000B	DB 00000010B
DB 01111111B	DB 00010101B	DB 00010000B	DB 00101111B	DB 00000001B
DB 10010000B	DB 00010101B	DB 00010000B	DB 00000000B	DB 01011110B
DB 01000000B	DB 00001110B	DB 00001111B	DB00000000B	DB 00000000B

AS107:DB 11111111B AS108:DB 00000000B AS109:DB 00011111B AS110:DB 00011111B AS111:DB 00001110B

DB 00000100B	DB 00000000B	DB 00010000B	DB 00001000B	DB 00010001B
DB 00001010B	DB 11111111B	DB 00011111B	DB 00010000B	DB 00010001B
DB 00010001B	DB 00000001B	DB 00010000B	DB 00010000B	DB 00010001B
DB 00000000B	DB 00000000B	DB 00001111B	DB 00001111B	DB 00001110B

AS112:DB 00011111B AS113:DB 00001000B AS114:DB 00011111B AS115:DB 00001001B AS116:DB 00000000B

DB 00010100B	DB 00010100B	DB 00001000B	DB 00010101B	DB 00010000B
DB 00010100B	DB 00010100B	DB 00010000B	DB 00010101B	DB 01111111B
DB 00001000B	DB 00011111B	DB 00010000B	DB 00010010B	DB 00010001B
DB 00000000B	DB 00000000B	DB 00000000B	DB 00000000B	DB 00000000B

```
AS117:DB 00011110B AS118:DB 00011100B AS119:DB 00011110B AS120:DB 00010001B AS121:DB 00000000B
      DB 00000001B      DB 00000010B      DB 00000001B      DB 00001010B      DB 00011001B
      DB 00000001B      DB 00000001B      DB 00001110B      DB 00000100B      DB 00000101B
      DB 00000001B      DB 00000010B      DB 00000001B      DB 00001010B      DB 00000101B
      DB 00011110B      DB 00011100B      DB 00011110B      DB 00010001B      DB 00011110B

AS122:DB 00010001B AS123:DB 00000000B AS124:DB 00000000B AS125:DB 00000000B AS126:DB 00010000B
      DB 00010011B      DB 00011000B      DB 00000000B      DB 10000001B      DB 00100000B
      DB 00010101B      DB 11100111B      DB 11111111B      DB 11100111B      DB 00010000B
      DB 00011001B      DB 11100111B      DB 00000000B      DB 00011000B      DB 00010000B
      DB 00010001B      DB 00000000B      DB 00000000B      DB 00000000B      DB 00100000B
END
```

โปรแกรมภาษาวิชวลเบสิก

```
VERSION 6.00
Begin VB.Form Form2
Private Sub Form_Load()
   Form1.Enabled = False
   Option1(P - 1).Value = True
   Form1.MSComm1.PortOpen = False
End Sub
Private Sub Form_Unload(Cancel As Integer)
   Form1.Enabled = True
   Form1.InitPort (P)
End Sub
Private Sub Option1_Click(Index As Integer)
   P = Index + 1
End Sub
```

```
VERSION 5.00
Begin VB.Form Form1
Dim i, ascii As Byte
Dim Word As String
Dim chek As Byte
```

```
Dim begin As Single
Dim iTime As String
Private Const comSETTING = "2400,N,8,1"
Private Const BFIRST As Byte = 170
Private Const BLAST As Byte = 171
Private Const BRUN As Byte = 172
Private Const BRESET As Byte = 173
Private Const BSTOP As Byte = 174
Private Const BLONG As Byte = 175
Private Const Tdelay As Single = 0.02
Private Const coMaxData = 100
Private Const coHalfMaxData = coMaxData / 2
Enum coPicture
   coTriangle = 127
   coSquare = 128
   coCircle = 129
End Enum
Enum coCOLOR
   RED = 176
   ORANGE = 177
   GREEN = 178
End Enum
Private iColor As Integer
Private iPicture As coPicture
Private NowDate As Date
Private NowTime As Date
Private Msg As String
Private Function CheckData(ByVal data As String) As Integer '0 เป็นจริง ผ่าน
Dim iCheckData%
On Error GoTo ErrHandle
   For i = 1 To Len(data)
      ascii = Asc(Mid(data, i, 1))
```

```
    If ascii < 32 Or ascii > 129 Then
      iCheckData = 1
      GoTo DSP_ERROR
    End If
  Next i
    Word = Trim(data)
  If InStr(1, Word, "fuck", 1) <> 0 Then
    iCheckData = 2
    GoTo DSP_ERROR
  ElseIf InStr(1, Word, "kuay", 1) <> 0 Then
    iCheckData = 2
    GoTo DSP_ERROR
  ElseIf InStr(1, Word, "pennis", 1) <> 0 Then
    iCheckData = 2
    GoTo DSP_ERROR
  End If
  CheckData = 0
  Exit Function
  DSP_ERROR:
  If iCheckData = 1 Then
    MsgBox "กรุณาพิมพ์ภาษาอังกฤษเท่านั้น", vbCritical + vbApplicationModal, "Error"
  ElseIf iCheckData = 2 Then
     MsgBox "กรุณาใช้คำสุภาพ", vbCritical + vbApplicationModal, "Error"
  End If
  CheckData = iCheckData
  Exit Function
 ErrHandle:
  CheckData = -1
End Function
Private Sub Delay(t As Single)
  begin = Timer
  While Timer < begin + t
```

```
        'DoEvents
    Wend
End Sub
Private Sub SetColor(ByVal Flag)
    iColor = Flag
End Sub
'Private Sub SendPicture(ByVal iPic As Integer)
'On Error Resume Next
'   SendData Chr(iPic)
'End Sub
Private Sub SendData(data As String)
    MSComm1.Output = Chr(BFIRST)
    Delay (Tdelay)
   For i = 1 To Len(data)
        MSComm1.Output = Mid(data, i, 1)
        Delay (Tdelay)
    Next
        If Len(data) >= 51 Then 'If Len(data) > 100 Then
       MSComm1.Output = Chr(BLONG)
    Else
        MSComm1.Output = Chr(BLAST)
    End If
    Delay (Tdelay)
        If Option1(0).Value = True Then
        MSComm1.Output = Chr(BRUN)
    ElseIf Option1(1).Value = True Then
        MSComm1.Output = Chr(BSTOP)
    ElseIf Option1(2).Value = True Then
        MSComm1.Output = Chr(BRESET)
    End If
End Sub
Public Sub InitPort(Port As Byte)
```

```
On Error GoTo ErrHandle
   MSComm1.CommPort = Port
   MSComm1.Settings = comSETTING ' 2400 baud, no parity, 8 data, and 1 stop bit.
   MSComm1.InputLen = 0    ' Tell the control to read entire buffer when Input     ' is used.
   MSComm1.PortOpen = True    ' Open the port.
   Exit Sub
  ErrHandle:
   MsgBox "ไม่สามารถ ใช้พอร์ทดังกล่าวได้" & vbNewLine & "เนื่องจากพอร์ทดังกล่าวกำลังถูกใช้งาน อยู่ หรือ ไม่สามารถใช้ได้" & vbNewLine & "No." & Err.Number & " : " & Err.Description, vbApplicationModal + vbCritical, "ERROR"
End Sub
Private Sub cbColor_Click()
On Error Resume Next
   iColor = cbColor.ListIndex + RED
   Call Send1ByteData(iColor)
End Sub
Private Sub cbPicture_Click()
On Error Resume Next
   Call SendData(Chr(cbPicture.ListIndex + coTriangle))
End Sub
Function Send1ByteData(ByVal iData As Integer)
   MSComm1.Output = Chr(BSTOP)
   Delay (Tdelay)
   MSComm1.Output = Chr(iData)
   Delay (Tdelay)
End Function
Private Sub Check1_Click(Index As Integer)
Dim NowDate, NowTime As Date
Static bFlag As Boolean
      If bFlag = True Then Exit Sub
   bFlag = True
   Select Case Index
```

```
Case 0 'Send Clock
   If Check1(0).Value = 1 Then
      Check1(1).Value = 0
      cbPicture.ListIndex = -1
      Msg = Text1.Text
      Command1.Enabled = False
      Text1.Enabled = False
      lbSetPort.Enabled = False
      'fraData.Enabled = False
      Timer1.Enabled = True
   End If
Case 1 'Send Picture
   If Check1(1).Value = 1 Then
      Check1(0).Value = 0
      Timer1.Enabled = False
      Msg = Text1.Text
      Command1.Enabled = False
      Text1.Enabled = False
      lbSetPort.Enabled = False
      fraData.Enabled = False
      cbPicture.ListIndex = 0
      cbPicture.SetFocus
   End If
End Select
If Check1(Index).Value = 0 Then
   Command1.Enabled = True
   Text1.Enabled = True
   lbSetPort.Enabled = True
   fraData.Enabled = True
   Timer1.Enabled = False
   Text1.Text = Msg
   SendData (Msg)
```

```
    End If
    bFlag = False
End Sub
Private Sub cmdSetDelay_Click()
Dim A
On Error Resume Next
    A = Tdelay
    A = InputBox("ตั้ง Delay ของ Tdelay  ", "ตั้งเวลา", Tdelay)
    If A > 0 Then
        Timer1.Interval = Tdelay
    End If
End Sub
Private Sub cmdSetTime_Click()
Dim A
On Error Resume Next
    A = Timer1.Interval
    A = InputBox("ตั้ง Interval ของ Timer  ", "ตั้งเวลา", Timer1.Interval)
    If A > 0 Then
        Timer1.Interval = A
    End If
End Sub
Private Sub Command1_Click()
    If CheckData(Text1.Text) = 0 Then
        Command1.Enabled = False
        Text1.Enabled = False
        Call SendData(Text1.Text)
        Command1.Enabled = True
        Text1.Enabled = True
    Else
        Text1.SetFocus
    End If
End Sub
```

```
Private Sub Form_Load()
   'Timer1.Interval = Tdelay * 1000
   Caption = lbTitle & " Version " & Version & " ( K M I T 'N B )"
   iColor = RED
   cbColor.ListIndex = 0
   P = 1
   InitPort (P)
End Sub
Private Sub Form_Unload(Cancel As Integer)
On Error Resume Next
   MSComm1.PortOpen = False
End Sub
Private Sub lbSetPort_Click()
   Form2.Show
End Sub
Private Sub mnuFileSub_Click(Index As Integer)
   Select Case Index
      Case 0
         Call Option1_Click(2)
         End
         'Unload Me
   End Select
End Sub
Private Sub mnuHelpSub_Click(Index As Integer)
   Select Case Index
      Case 0
         frmAbout.Show vbModal
   End Select
End Sub
Private Sub Option1_Click(Index As Integer)
   Select Case Index
   Case 0
```

```
      MSComm1.Output = Chr(BRUN)
    Case 1
      MSComm1.Output = Chr(BSTOP)
    Case 2
      MSComm1.Output = Chr(BRESET)
    End Select
End Sub
 Private Sub Text1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
      Command1.Enabled = False
      Text1.Enabled = False
      SendData (Text1.Text)
      Command1.Enabled = True
      Text1.Enabled = True
    End If
End Sub
Private Sub Timer1_Timer()
Static A
    A = A + 1
    Debug.Print "A=" & A
    iTime = Format(Time, " HH:MM:SS ")
    'Call SendClock(" ")
    Call SendClock(iTime)
    Text1.Text = iTime
End Sub
Function SendClock(ByVal data$)
    MSComm1.Output = Chr(BSTOP)
    Delay (Tdelay)
      MSComm1.Output = Chr(BFIRST)
    Delay (Tdelay)
    For i = 1 To Len(data)
      MSComm1.Output = Mid(data, i, 1)
```

```
    Delay (Tdelay)
  Next
    MSComm1.Output = Chr(BLAST)
  Delay (Tdelay)
    'CLEAR
  MSComm1.Output = Chr(BFIRST)
  Delay (Tdelay)
  MSComm1.Output = Chr(32)
  Delay (Tdelay)
  MSComm1.Output = Chr(BLAST)
  Delay (Tdelay)
End Function
```

Using a Personal Computer to Program the AT89C51/C52/LV51/LV52/C1051/C2051

Introduction

This application note describes a personal computer-based programmer for the AT89C51/C52/LV51/LV52/C1051/C2051 Flash-based Microcontrollers. The programmer supports all flash memory microcontroller functions, including code read, code write, chip erase, signature read, and lock bit write. When used with the AT89C51/C52/ LV51/LV52, code write, chip erase, and lock bit write may be performed at either five or twelve volts, as required by the device.

Devices sporting a "-5" suffix are intended for operation at five volts, while devices lacking the suffix operate at the standard twelve volts.

The programmer connects to an IBM PC-compatible host computer through one of the host's parallel ports. Required operating voltages are produced by an integral power supply and external, wall-mounted transformer.

Software

Software for the programmer is available by downloading it from the Atmel BBS at 408-436-4309.

The programmer is controlled by software running on the host. The AT89C51/C52 and C1051/C2051 have dedicated control programs, which were written in Microsoft C. Programs dedicated to the AT89LV51/LV52 do not exist; these devices are supported by the programs for the AT89C51/C52, respectively. In the text below, all references to the AT89C51/C52 may be assumed to apply to the AT89LV51/LV52 as well.

All programmer control programs are invoked from the DOS command line by entering the program name followed by "LPT1" or "LPT2" to specify parallel port one or two, respectively. If the parallel port is not specified, the program will respond with an error message. The control programs are menu-driven, and provide the following functions:

Chip Erase

Clear code memory to all ones. The successful operation of this function is not automatically verified.

Program from File

Write the contents of the specified file into device memory. The user is prompted for the file name, which may require path and extension.

The file is expected to contain binary data; hex files are not accepted. The first byte in the file is programmed into the first location in the device. Successive bytes are programmed into successive locations until the last location in the device has been programmed or until the data in the file has been exhausted.

Programming occurs regardless of the existing contents of device memory; a blank check is not automatically performed. After programming, the contents of device memory are not automatically verified against the file data.

Each programmed location in the device receives the maximum programming time specified in the data sheet. This is done because timing is enforced by software; the programming status information provided by $\overline{\text{DATA}}$ polling and RDY/$\overline{\text{BSY}}$ is not utilized.

The control program provides no visual indication that programming is in progress. The main menu is redisplayed when programming is complete.

ATMEL

8-Bit Microcontroller with Flash

Application Note

0285D-B–12/97

Verify against File
Compare the contents of code memory against the contents of the specified file. The user is prompted for the file name, which may require path and extension.

The file is expected to contain binary data; hex files are not accepted. The first byte in the file is compared to the first location in the device. Successive bytes are compared to successive locations until the last location in the device has been compared or until the data in the file has been exhausted.

Locations which fail to compare are displayed by address, with the expected and actual byte contents. If there are no compare failures, nothing is displayed.

Save to File
Copy the contents of device memory to the specified file. The user is prompted for the file name, which may require path and extension. The number of bytes in the resulting file is the same as the number of memory locations in the device.

Blank Check
Verify that the contents of device memory are all ones. Only pass or fail is reported; the addresses and contents of failing locations are not displayed.

Read Signature
Read and display the contents of the signature bytes. The number of signature bytes and their expected contents varies between devices. Refer to the device data sheet for additional information.

Write Lock Bit 1
Write Lock Bit 2
Write Lock Bit 3
Set the indicated lock bit. Note that the AT89C1051/C2051 contain only two lock bits, while the AT89C51/LV51 and AT89C52/LV52 contain three lock bits. The state of the lock bits cannot be verified by direct observation.

Exit
Quit the programmer control program.

System Dependency

The control programs for the AT89C51 and AT89C52 come in two flavors: host system-dependent and host system-independent. System-dependency results from the use of software timing loops to enforce required delays, the duration of which will vary between host systems running at different speeds. The code provided was tested on an 80386-based system running at 33 MHz, and may require modification for use on other systems. This method was chosen for its simplicity.

Host system-independence is achieved by using the Programmable Interval Timer embedded in the system hardware to enforce time delays independent of system speed. The timer is reconfigured when the control program is invoked and restored to its original state before the program terminates. In order to guarantee that the program is not exited before the timer configuration is restored, the CTRL-C and CTRL-BREAK keys are disabled. This means that the program cannot be aborted except by specifying the exit option at the main menu or by rebooting the system.

The timer control code is provided as an 8086 assembly language module, which is linked with the compiled control program. The granularity of the timer is 0.838 microseconds, but the minimum practical delay is system- and software-dependent. The timer code ensures that the delay produced will not be of shorter duration than requested.

The control programs provided for the AT89C1051/C2051 are system independent.

Programmer

The programmer circuitry (see Figures 1 and 2) consists of the host interface and switchable power supplies. The signal sequencing and timing required for programming is generated by the host under software control. A 40-pin ZIF socket is provided for programming the AT89C51/C52; the 20-pin ZIF socket accommodates the AT89C1051/C2051. Note that the power and ground connections and bypass capacitors required by the TTL devices are not shown on the schematic.

Power for the programmer circuitry and the AT89C51/C52/C1051/C2051 is provided by a fixed five volt supply. A second supply provides either five or twelve volts, selectable, for use during programming. The addition of a transistor to the output of the variable supply provides a third level, ground, for use when programming the AT89C1051/C2051.

The resistor values utilized in the variable power supply circuit were determined using the equations presented in the LM317 voltage regulator data sheet. Power supply ramp rates are accommodated by the host software. For 5 V-VPP programming, the devices must be ordered from the factory as an AT89CX-XX-5 (not available with the AT89C1051/2051).

The programmer is connected to the host with a 25-conductor ribbon cable. To minimize the effect on signal integrity, the length of the cable should be as short as possible, preferably not exceeding three feet.

Parallel Interface

The original parallel interface provided by IBM was probably not intended to support bidirectional data transfers. However, due to the way in which the interface was implemented, bidirectional transfers are possible. Over the years, many products have appeared which exploit this capability.

Unfortunately, many system and interface card manufacturers have not faithfully cloned the IBM design, resulting in bus contention when the peripheral attempts to drive return

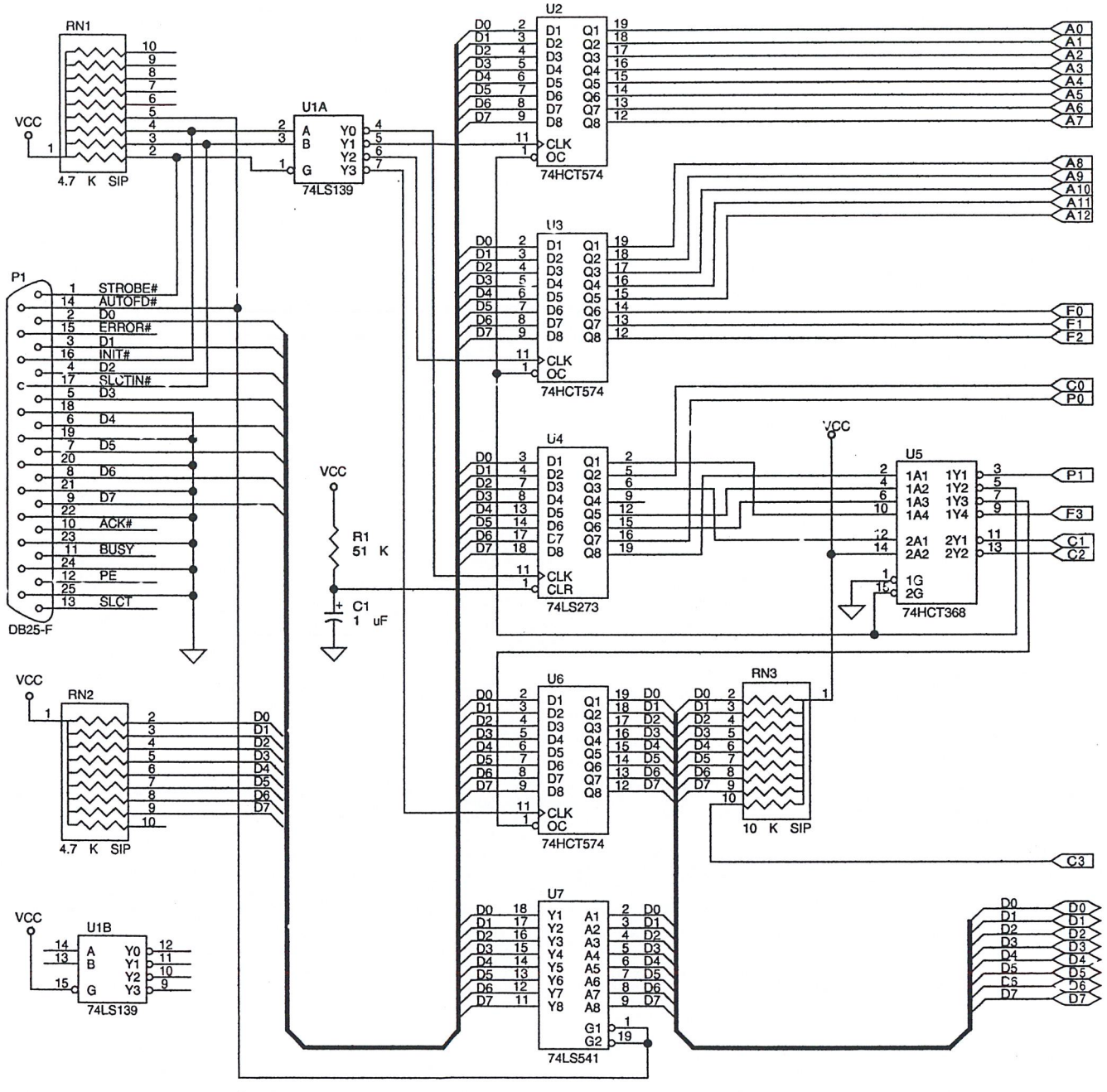
data into the interface. Usually the peripheral drivers can overpower the interface drivers and the peripheral works, though this is not considered a good design practice.

Most parallel interfaces are now implemented in a single chip, such as the 82C411 or 16C452. These chips allow their output drivers to be disabled under software control, providing true bidirectional operation. The programmer software automatically enables bidirectional operation when used with parallel interfaces utilizing the 82C411, 16C452, or similar chips.

Note that these chips also possess a mode control pin which must be at the correct level to enable the directional control feature. As a result, parallel interfaces utilizing these chips cannot be assumed to be bidirectional.

If the programmer writes devices, but fails to verify, or the signal levels at the interface don't meet TTL specifications, the parallel interface may be incompatible with the programmer. A design is provided (see Figure 4 and Figure 5) for a parallel interface which supports bidirectional operation and is compatible with the programmer. This design is simple, requiring only six ICs. The interface can be strapped to appear as LPT1 (addresses 378-37F hex) or LPT2 (278-27F hex) and will be recognized by the POST when the host system is powered up. Due to its simplicity, the parallel interface CANNOT be used as a printer interface.

Figure 1. AT89 Series Programmer Interface



Note: 0.1 µF bypass caps on all ICs

Figure 2. Power Supply for AT89 Series Programmer

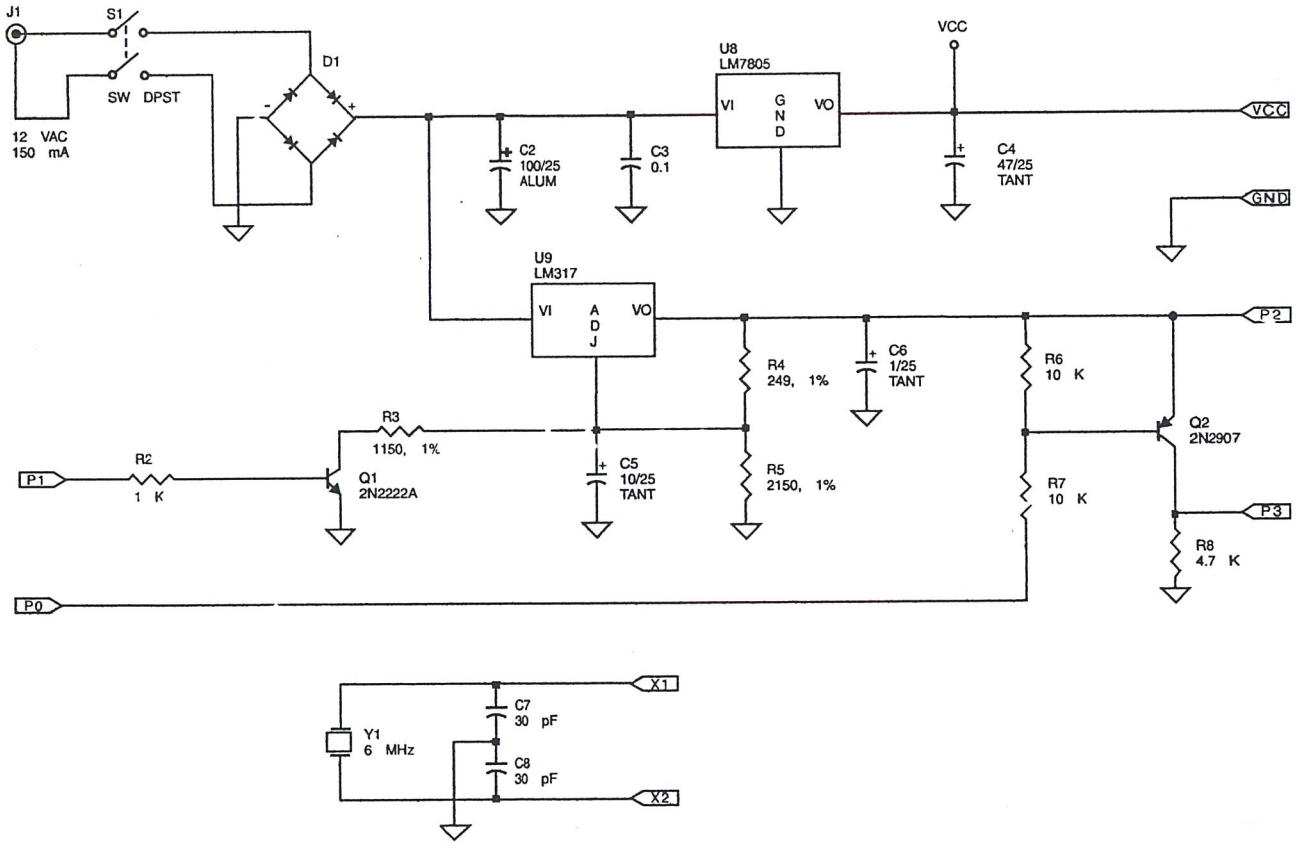


Figure 3. AT89 Series Programmer Socket Wiring

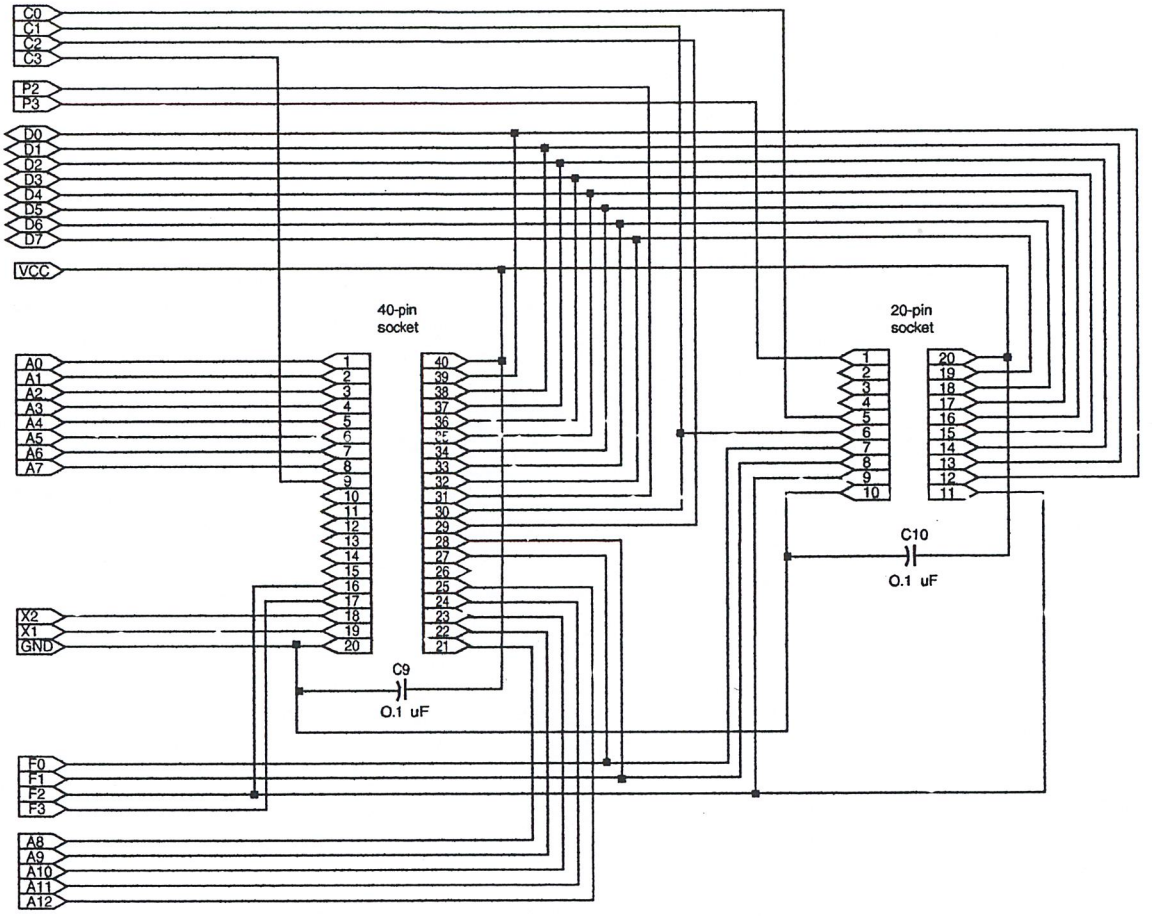


Figure 4. A Parallel Interface Supporting Bidirectional Operation

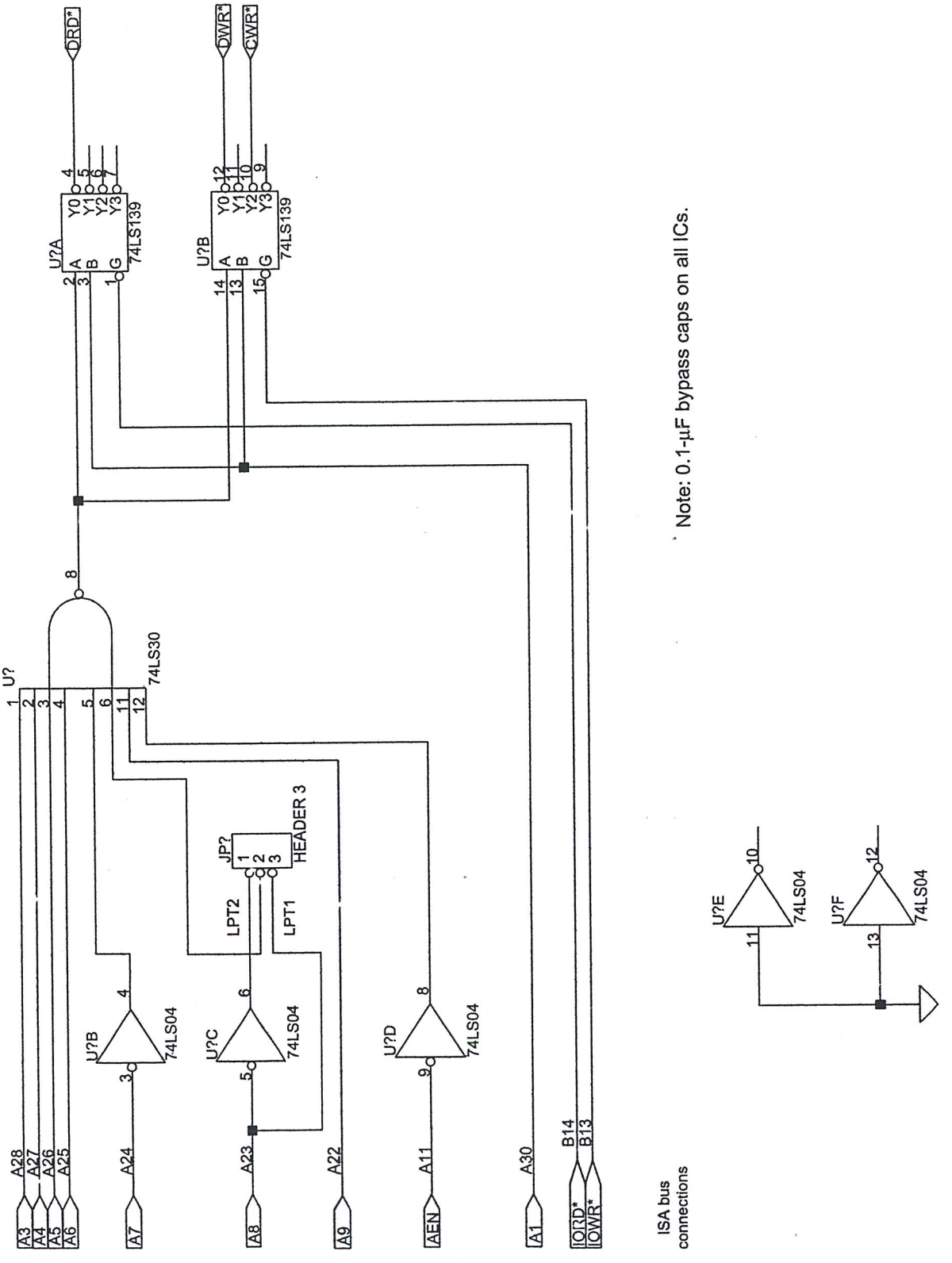
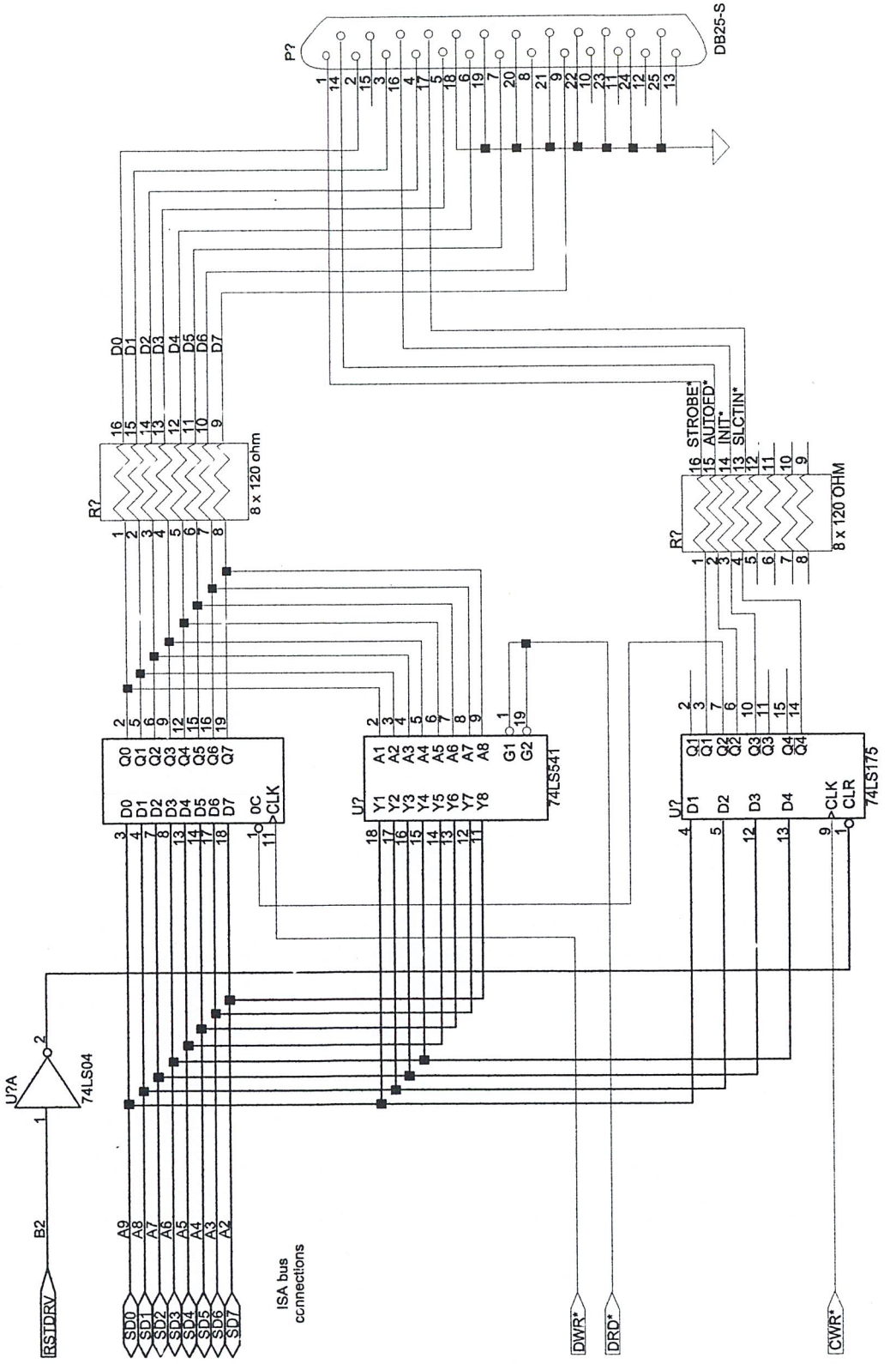


Figure 5.


P?
DB25-S
R?
8 x 120 ohm
R?
8 x 120 OHM
STROBE*
AUTOFD*
INIT*
SLCTIN*
D0
D1
D2
D3
D4
D5
D6
D7
U?
74LS541
U?
74LS175
U?A
74LS04
B2
RSTDRV
SD0
SD1
SD2
SD3
SD4
SD5
SD6
SD7
ISA bus connections
DWR*
DRD*
CWR*
Note: 0.1-µF bypass caps on all ICs.

MOTOROLA

OCTAL BUFFER/LINE DRIVER WITH 3-STATE OUTPUTS

SN54/74LS240
SN54/74LS241
SN54/74LS244

OCTAL BUFFER/LINE DRIVER WITH 3-STATE OUTPUTS

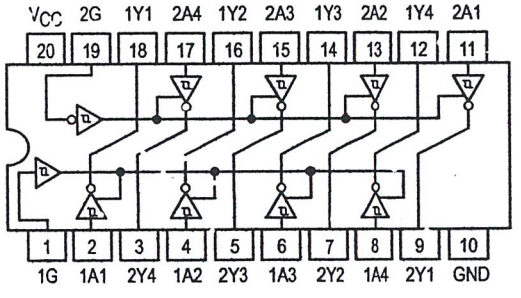
LOW POWER SCHOTTKY

The SN54/74LS240, 241 and 244 are Octal Buffers and Line Drivers designed to be employed as memory address drivers, clock drivers and bus-oriented transmitters/receivers which provide improved PC board density.

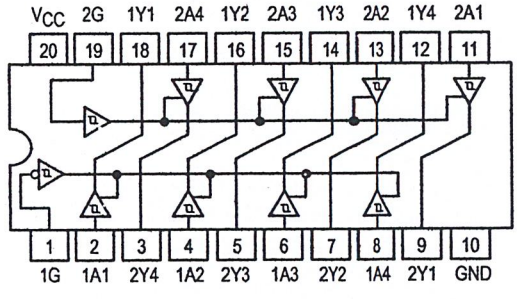
- Hysteresis at Inputs to Improve Noise Margins
- 3-State Outputs Drive Bus Lines or Buffer Memory Address Registers
- Input Clamp Diodes Limit High-Speed Termination Effects

LOGIC AND CONNECTION DIAGRAMS DIP (TOP VIEW)

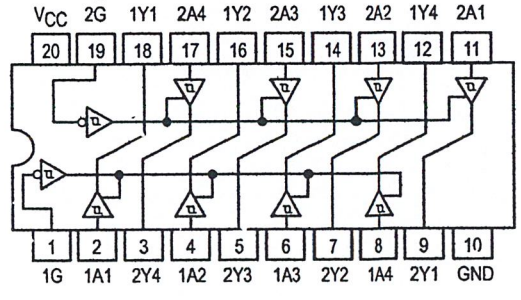
SN54/74LS240

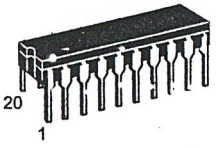


SN54/74LS241

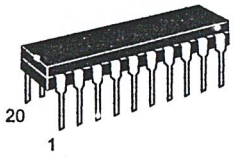


SN54/74LS244

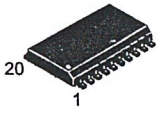




J SUFFIX
CERAMIC
CASE 732-03



N SUFFIX
PLASTIC
CASE 738-03



DW SUFFIX
SOIC
CASE 751D-03

ORDERING INFORMATION

SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXDW	SOIC

SN54/74LS240 • SN54/74LS241 • SN54/74LS244

TRUTH TABLES

SN54/74LS240

INPUTS		OUTPUT
$\overline{1G}$, $\overline{2G}$	D	
L	L	H
L	H	L
H	X	(Z)

SN54/74LS244

INPUTS		OUTPUT
$\overline{1G}$, $\overline{2G}$	D	
L	L	L
L	H	H
H	X	(Z)

SN54/74LS241

INPUTS		OUTPUT	INPUTS		OUTPUT
$\overline{1G}$	D		$\overline{2G}$	D	
L	L	L	H	L	L
L	H	H	H	H	H
H	X	(Z)	L	X	(Z)

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial
Z = HIGH Impedance

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54, 74			−3.0	mA
		54 74			−12 −15	mA
I_{OL}	Output Current — Low	54 74			12 24	mA

SN54/74LS240 • SN54/74LS241 • SN54/74LS244

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Typ	Max	Unit	Test Conditions	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
$V_{T+}-V_{T-}$	Hysteresis		0.2	0.4		V	V_{CC} = MIN	
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54, 74	2.4	3.4		V	V_{CC} = MIN, I_{OH} = −3.0 mA	
		54, 74	2.0			V	V_{CC} = MIN, I_{OH} = MAX	
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA	
I_{OZH}	Output Off Current HIGH				20	µA	V_{CC} = MAX, V_{OUT} = 2.7 V	
I_{OZL}	Output Off Current LOW				−20	µA	V_{CC} = MAX, V_{OUT} = 0.4 V	
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current				−0.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 1)		−40		−225	mA	V_{CC} = MAX	
I_{CC}	Power Supply Current Total, Output HIGH				27	mA	V_{CC} = MAX	
	Total, Output LOW	LS240			44			
		LS241/244			46			
	Total at HIGH Z	LS240			50			
		LS241/244			54			

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C, V_{CC} = 5.0 V)

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH} t_{PHL}	Propagation Delay, Data to Output LS240		9.0 12	14 18	ns	C_L = 45 pF, R_L = 667 Ω
t_{PLH} t_{PHL}	Propagation Delay, Data to Output LS241/244		12 12	18 18	ns	
t_{PZH}	Output Enable Time to HIGH Level		15	23	ns	
t_{PZL}	Output Enable Time to LOW Level		20	30	ns	
t_{PLZ}	Output Disable Time from LOW Level		15	25	ns	C_L = 5.0 pF, R_L = 667 Ω
t_{PHZ}	Output Disable Time from HIGH Level		10	18	ns	

AC WAVEFORMS

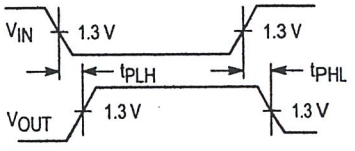


Figure 1

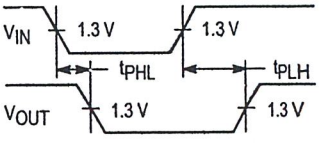


Figure 2

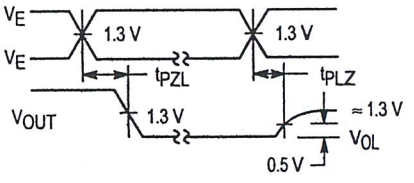


Figure 3

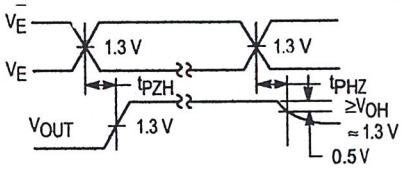
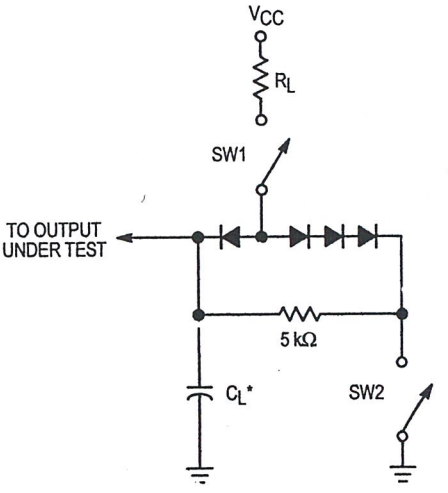


Figure 4



SWITCH POSITIONS

SYMBOL	SW1	SW2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_{PLZ}	Closed	Closed
t_{PHZ}	Closed	Closed

Figure 5

Timer

NE/SA/SE555/SE555C

DESCRIPTION

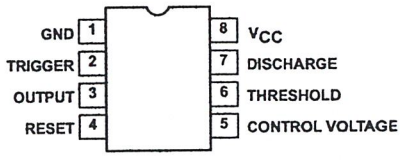
The 555 monolithic timing circuit is a highly stable controller capable of producing accurate time delays, or oscillation. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For a stable operation as an oscillator, the free running frequency and the duty cycle are both accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output structure can source or sink up to 200mA.

FEATURES

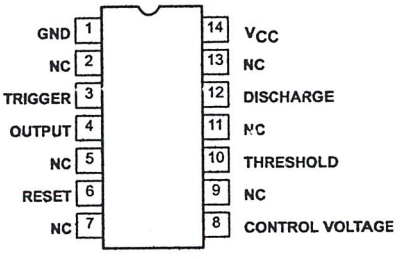
- Turn-off time less than 2µs
- Max. operating frequency greater than 500kHz
- Timing from microseconds to hours
- Operates in both astable and monostable modes
- High output current
- Adjustable duty cycle
- TTL compatible
- Temperature stability of 0.005% per °C

PIN CONFIGURATIONS

D, N, FE Packages



F Package



TOP VIEW

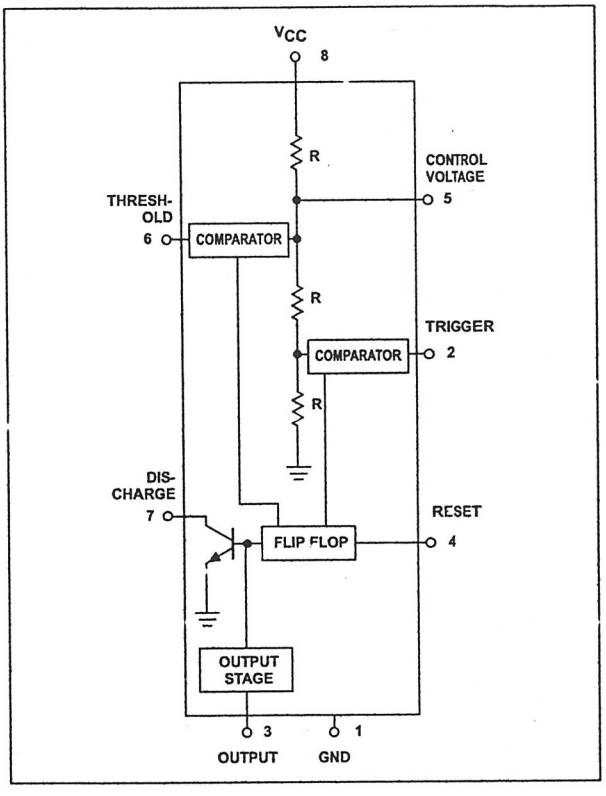
APPLICATIONS

- Precision timing
- Pulse generation
- Sequential timing
- Time delay generation
- Pulse width modulation

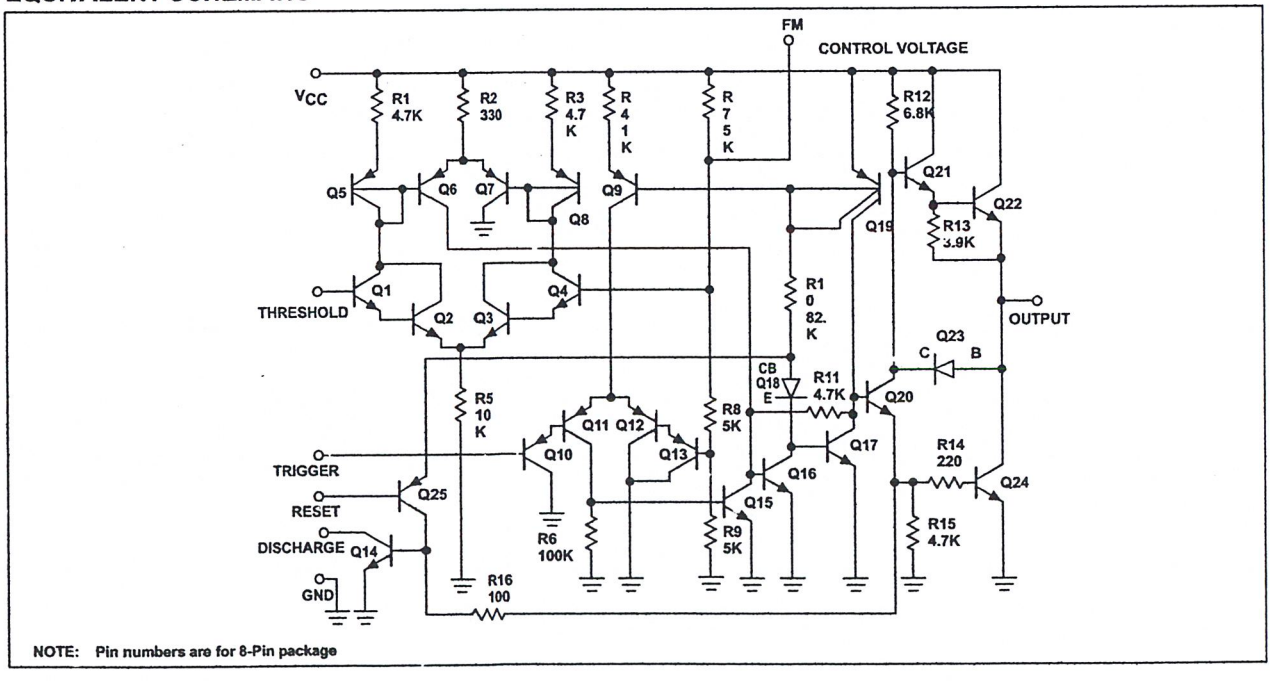
ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
8-Pin Plastic Small Outline (SO) Package	0 to +70°C	NE555D	0174C
8-Pin Plastic Dual In-Line Package (DIP)	0 to +70°C	NE555N	0404B
8-Pin Plastic Dual In-Line Package (DIP)	-40°C to +85°C	SA555N	0404B
8-Pin Plastic Small Outline (SO) Package	-40°C to +85°C	SA555D	0174C
8-Pin Hermetic Ceramic Dual In-Line Package (CERDIP)	-55°C to +125°C	SE555CFE	
8-Pin Plastic Dual In-Line Package (DIP)	-55°C to +125°C	SE555CN	0404B
14-Pin Plastic Dual In-Line Package (DIP)	-55°C to +125°C	SE555N	0405B
8-Pin Hermetic Cerdip	-55°C to +125°C	SE555FE	
14-Pin Ceramic Dual In-Line Package (CERDIP)	0 to +70°C	NE555F	0581B
14-Pin Ceramic Dual In-Line Package (CERDIP)	-55°C to +125°C	SE555F	0581B
14-Pin Ceramic Dual In-Line Package (CERDIP)	-55°C to +125°C	SE555CF	0581B

BLOCK DIAGRAM



EQUIVALENT SCHEMATIC



NOTE: Pin numbers are for 8-Pin package

ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER	RATING	UNIT
V_{CC}	Supply voltage		
	SE555	+18	V
	NE555, SE555C, SA555	+16	V
P_D	Maximum allowable power dissipation[1]	600	mW
T_A	Operating ambient temperature range		
	NE555	0 to +70	°C
	SA555	-40 to +85	°C
	SE555, SE555C	-55 to +125	°C
T_{STG}	Storage temperature range	-65 to +150	°C
T_{SOLD}	Lead soldering temperature (10sec max)	+300	°C

NOTES:

1. The junction temperature must be kept below 125°C for the D package and below 150°C for the FE, N and F packages. At ambient temperatures above 25°C, where this limit would be derated by the following factors:
   D package 160°C/W
   FE package 150°C/W
   N package 100°C/W
   F package 105°C/W

DC AND AC ELECTRICAL CHARACTERISTICS

T_A = 25°C, V_{CC} = +5V to +15 unless otherwise specified.

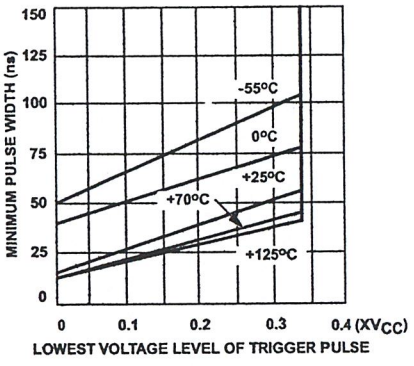
SYMBOL	PARAMETER	TEST CONDITIONS	SE555			NE555/SE555C			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{CC}	Supply voltage		4.5		18	4.5		16	V
I_{CC}	Supply current (low state)[1]	V_{CC}=5V, R_L=∞		3	5		3	6	mA
		V_{CC}=15V, R_L=∞		10	12		10	15	mA
	Timing error (monostable)	R_A=2kΩ to 100kΩ							
t_M	Initial accuracy[2]	C=0.1µF		0.5	2.0		1.0	3.0	%
$\Delta t_M/\Delta T$	Drift with temperature			30	100		50	150	ppm/°C
$\Delta t_M/\Delta V_S$	Drift with supply voltage			0.05	0.2		0.1	0.5	%/V
	Timing error (astable)	R_A, R_B=1kΩ to 100kΩ							
t_A	Initial accuracy[2]	C=0.1µF		4	6		5	13	%
$\Delta t_A/\Delta T$	Drift with temperature	V_{CC}=15V			500			500	ppm/°C
$\Delta t_A/\Delta V_S$	Drift with supply voltage			0.15	0.6		0.3	1	%/V
V_C	Control voltage level	V_{CC}=15V	9.6	10.0	10.4	9.0	10.0	11.0	V
		V_{CC}=5V	2.9	3.33	3.8	2.6	3.33	4.0	V
V_{TH}	Threshold voltage	V_{CC}=15V	9.4	10.0	10.6	8.8	10.0	11.2	V
		V_{CC}=5V	2.7	3.33	4.0	2.4	3.33	4.2	V
I_{TH}	Threshold current[3]			0.1	0.25		0.1	0.25	µA
V_{TRIG}	Trigger voltage	V_{CC}=15V	4.8	5.0	5.2	4.5	5.0	5.6	V
		V_{CC}=5V	1.45	1.67	1.9	1.1	1.67	2.2	V
I_{TRIG}	Trigger current	V_{TRIG}=0V		0.5	0.9		0.5	2.0	µA
V_{RESET}	Reset voltage[4]	V_{CC}=15V, V_{TH} =10.5V	0.3		1.0	0.3		1.0	V
I_{RESET}	Reset current	V_{RESET}=0.4V		0.1	0.4		0.1	0.4	mA
	Reset current	V_{RESET}=0V		0.4	1.0		0.4	1.5	mA
V_{OL}	Output voltage (low)	V_{CC}=15V							
		I_{SINK}=10mA		0.1	0.15		0.1	0.25	V
		I_{SINK}=50mA		0.4	0.5		0.4	0.75	V
		I_{SINK}=100mA		2.0	2.2		2.0	2.5	V
		I_{SINK}=200mA		2.5			2.5		V
		V_{CC}=5V							
		I_{SINK}=8mA		0.1	0.25		0.3	0.4	V
		I_{SINK}=5mA		0.05	0.2		0.25	0.35	V
V_{OH}	Output voltage (high)	V_{CC}=15V							
		I_{SOURCE}=200mA		12.5			12.5		V
		I_{SOURCE}=100mA	13.0	13.3		12.75	13.3		V
		V_{CC}=5V							
		I_{SOURCE}=100mA	3.0	3.3		2.75	3.3		V
t_{OFF}	Turn-off time[5]	V_{RESET}=V_{CC}		0.5	2.0		0.5	2.0	µs
t_R	Rise time of output			100	200		100	300	ns
t_F	Fall time of output			100	200		100	300	ns
	Discharge leakage current			20	100		20	100	nA

NOTES:

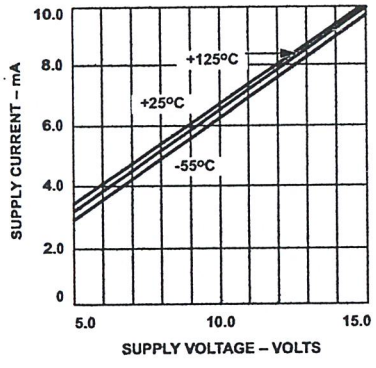
1. Supply current when output high typically 1mA less.
2. Tested at V_{CC}=5V and V_{CC}=15V.
3. This will determine the max value of R_A+R_B, for 15V operation, the max total R=10MΩ, and for 5V operation, the max. total R=3.4MΩ.
4. Specified with trigger input high.
5. Time measured from a positive going input pulse from 0 to 0.8xV_{CC} into the threshold to the drop from high to low of the output. Trigger is tied to threshold.

TYPICAL PERFORMANCE CHARACTERISTICS

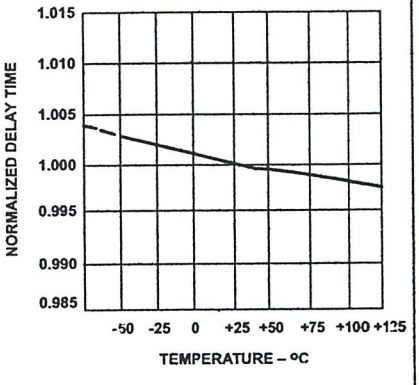
Minimum Pulse Width Required for Triggering



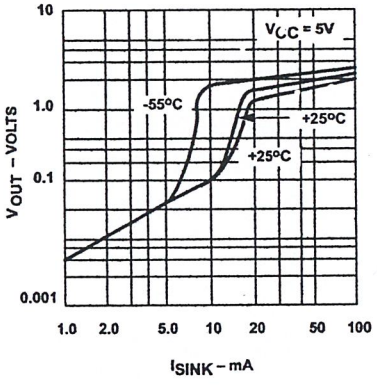
Supply Current vs Supply Voltage



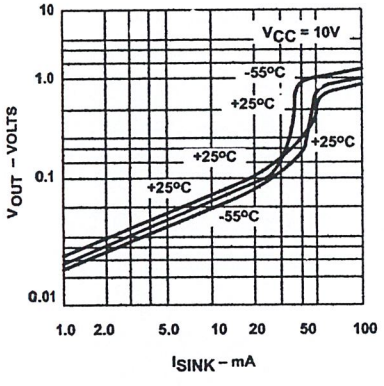
Delay Time vs Temperature



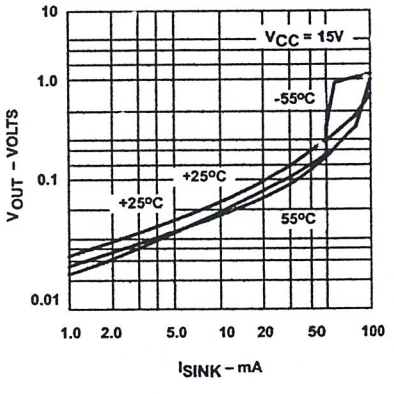
Low Output Voltage vs Output Sink Current



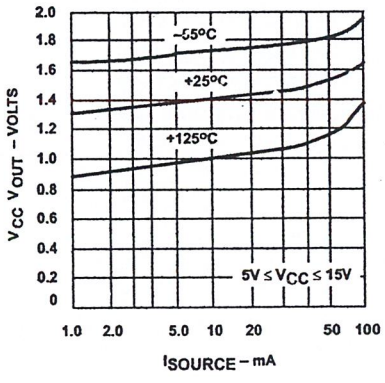
Low Output Voltage vs Output Sink Current



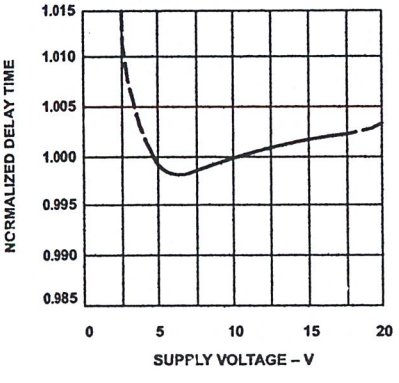
Low Output Voltage vs Output Sink Current



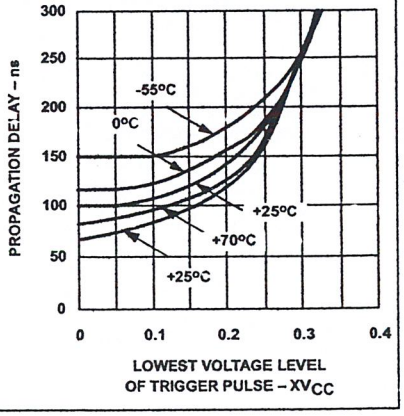
High Output Voltage Drop vs Output Source Current



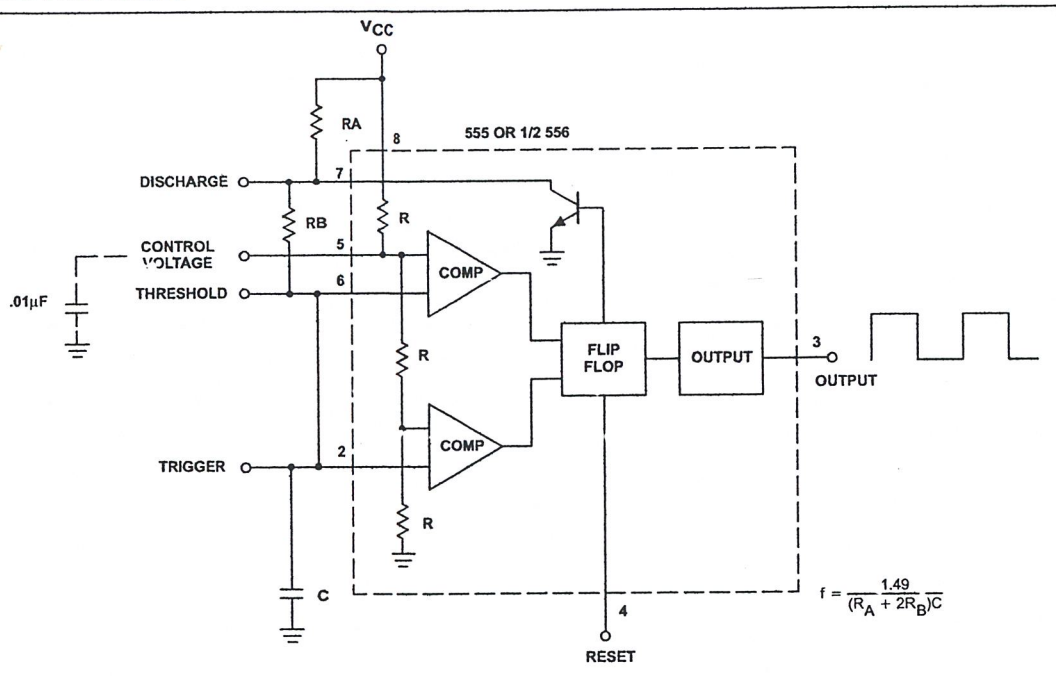
Delay Time vs Supply Voltage



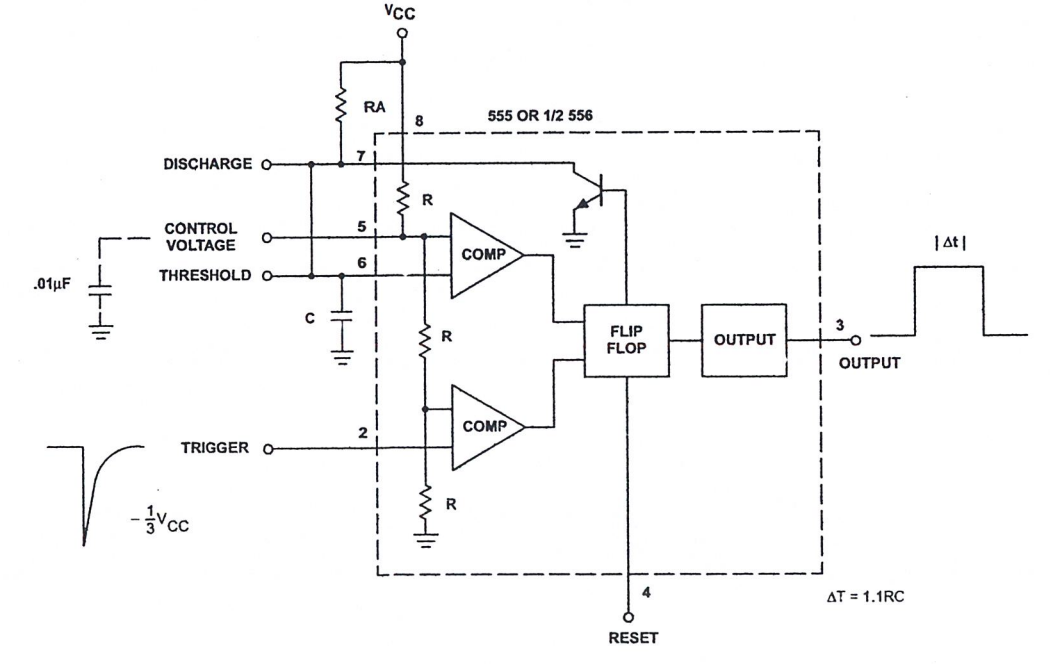
Propagation Delay vs Voltage Level of Trigger Pulse



TYPICAL APPLICATIONS



Astable Operation



Monostable Operation

TYPICAL APPLICATIONS

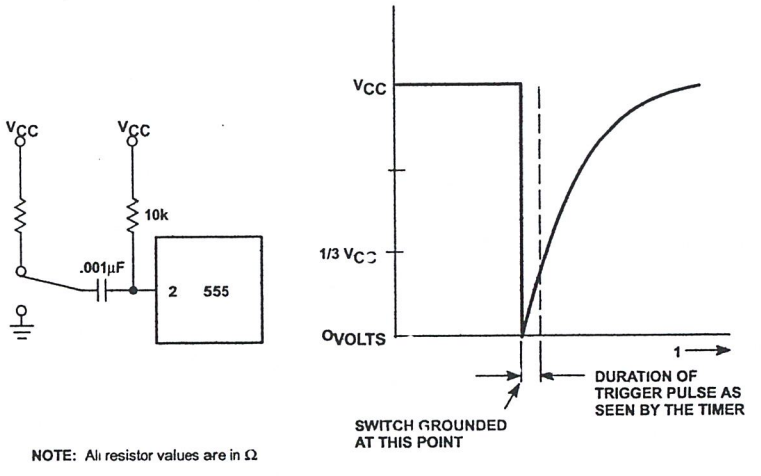


Figure 1. AC Coupling of the Trigger Pulse

Trigger Pulse Width Requirements and Time Delays

Due to the nature of the trigger circuitry, the timer will trigger on the negative going edge of the input pulse. For the device to time out properly, it is necessary that the trigger voltage level be returned to some voltage greater than one third of the supply before the time out period. This can be achieved by making either the trigger pulse sufficiently short or by AC coupling into the trigger. By AC coupling the trigger, see Figure 1, a short negative going pulse is achieved when the trigger signal goes to ground. AC coupling is most frequently used in conjunction with a switch or a signal that goes to ground which initiates the timing cycle. Should the trigger be held low, without AC coupling, for a longer duration than the timing cycle the output will remain in a high state for the duration of the low trigger signal, without regard to the threshold comparator state. This is due to the predominance of Q_{15} on the base of Q_{16}, controlling the state of the bi-stable flip-flop. When the trigger signal then returns to a high level, the output will fall immediately. Thus, the output signal will follow the trigger signal in this case.

Another consideration is the "turn-off time". This is the measurement of the amount of time required after the threshold reaches 2/3 V_{CC} to turn the output low. To explain further, Q_1 at the threshold input turns on after reaching 2/3 V_{CC}, which then turns on Q_5, which turns on Q_6. Current from Q_6 turns on Q_{16} which turns Q_{17} off. This allows current from Q_{19} to turn on Q_{20} and Q_{24} to given an output low. These steps cause the 2µs max. delay as stated in the data sheet.

Also, a delay comparable to the turn-off time is the trigger release time. When the trigger is low, Q_{10} is on and turns on Q_{11} which turns on Q_{15}. Q_{15} turns off Q_{16} and allows Q_{17} to turn on. This turns off current to Q_{20} and Q_{24}, which results in output high. When the trigger is released, Q_{10} and Q_{11} shut off, Q_{15} turns off, Q_{16} turns on and the circuit then follows the same path and time delay explained as "turn off time". This trigger release time is very important in designing the trigger pulse width so as not to interfere with the output signal as explained previously.

กิตติกรรมประกาศ

ขอขอบพระคุณอาจารย์ที่ปรึกษา อ.อิทธิภูมิ บุญพิคำ เป็นอย่างสูงที่ได้ให้แนวทางความรู้ ความคิดริเริ่ม คำปรึกษา แนวทางปรับปรุงแก้ไข เครื่องมืออุปกรณ์และเอกสารอันเป็นประโยชน์แก่ผู้จัดทำ และขอขอบพระคุณทุกๆท่านที่ให้การสนับสนุนและให้การช่วยเหลือในด้านต่างๆ จนโครงงานปริญญานิพนธ์นี้สามารถสำเร็จลุล่วงไปได้ด้วยดี

หนังสืออ้างอิง

1. กรมวิชาการ กระทวงศึกษาธิการ. แสงที่มองไม่เห็น หนังสืออ่านเพิ่มเติมชุดวิทยาศาสตร์ กรุงเทพมหานคร : โรงพิมพ์คุรุสภา 2520.
2. กัมพล ทองเรือง. ทฤษฎีและการออกแบบวงจรพัลส์ กรุงเทพมหานคร : โรงพิมพ์สกายบุ๊กส์ 2539.
3. จิระ จริงจิต. เรียนลัดวิชวลแบสิก ครั้งที่ 4 กรึงเทพมหานคร : โรงพิมพ์ลริษัทซีเดยูเคชั่น จำกัด (มหาชน) 2543.
4. ฉัททวุฒิ พืชผล และพิชิต สัติกุลานนท์. คู่มือเรียนวิชวลเบสิก 6 ครั้งที่ 3 กรุงเทพมหานคร : โรงพิมพ์บริษัทซีอีดยูเคชั่น จำกัด (มหาชน) 2543.
5. นภัทร วัจนเทพินทร์. คู่มือการออกแบบทดลองวงจรพัลส์ กรึงเทพมหานคร: โรงพิมพ์สกายบุ๊กส์ 2540.
6. ปรเมษฐ์ ประนยานันท์ และปิยพงศ์ เผ่าวณิช. คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51 กรุงเทพมหานคร : โรงพิมพ์บริษัทซีเอ็ดยูเคชั่น จำกัด(มหาชน) 2536.
7. สุเจตน์ จันทรังษ์. ไมโครคอนโทรลเลอร์ 8051 กรุงเทพมหานคร : โรงพิมพ์มหาวิทยาลัยเทคโนโลยีมหานคร 2535.
8. วัฒนา ถาวร. การส่องสว่าง ครั้งที่3 กรุงเทพมหานคร : โรงพิมพ์บริษัทควงกมลสมัย จำกัด 2536.