ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์วิศวกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

ปริญญานิพนธ์ คอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าภายในอาคาร

COMPUTER CONTROL ELECTRICAL DEVICES

นักศึกษา	1. นายนริศ	บุญศักดิ์เฉลิม	รหัสประจำตัว 39031214
	2. นางสาวนันทินี	แป้นแจ้ง	รหัสประจำตัว 39031216
	3. นายอานุภาพ	เสนีย์พัลย์	รหัสประจำตัว 39031245
	4. นายพันธ์ศักดิ์	ภูวัน	รหัสประจำตัว 39031246

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต สาขาวิชา วิศวกรรมโทรคมนาคม

อาจารย์ผู้ควบคุมปริญญานิพนธ์

1.อาจารย์ปิยะ ศุภวราสุวัฒน์

2. ผศ.วิสุทธิ์ อธิพรธรรม

3. อาจารย์พีระวุฒิ สุวรรณจันทร์

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1.อาจารย์ปิยะ	ศุภวราสุวัฒน์	.....
2. ผศ.วิสุทธิ์	อธิพรธรรม	.....
3. อาจารย์พีระวุฒิ	สุวรรณจันทร์	.....
4. อาจารย์ประเสริฐ	เคนพันค้อ	.....
5. อาจารย์พงษ์เกียรติ	เชษฐพิทักษ์สกุล	.....

วัน/เดือน/ปี ที่สอบ 27 เมษายน 2541 เวลา 16.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

.....

(ผศ.ดร.ธีระพล เทพหัสดิน ณ อยุธยา)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 27 เดือน มีนาคม พ.ศ. 2541

ปริญญานิพนธ์

คอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าภายในอาคาร

COMPUTER CONTROL ELECTRICAL DEVICES

นายนริศ บุญศักดิ์เฉลิม

นางสาวนันทินี แป้นแจ้ง

นายอานุภาพ เสนีย์พัลย์

นายพันธ์ศักดิ์ ภูวัน

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปริญญานิพนธ์

เรื่อง คอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าภายในอาคาร

COMPUTER CONTROL ELECTRICAL DEVICES

ผู้จัดทำ

1. นายนริศ บุญศักดิ์เฉลิม
2. นางสาวนันทินี แป้นแจ้ง
3. นายอานุภาพ เสนีย์พัลย์
4. นายพันธ์ศักดิ์ ภูวัน

อาจารย์ที่ปรึกษา

ลงนาม.....

(อาจารย์ปิยะ ศุภวราสุวัฒน์)

ลงนาม.....

(ผศ. วิสุทธิ์ อธิพรธรรม)

ลงนาม.....

(อาจารย์พีระวุฒิ สุวรรณจันทร์)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

ลงนาม.....

(ผศ.ดร.ธีระพล เทพหัสดิน ณ อยุธยา)

ปริญญานิพนธ์

เรื่อง คอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าภายในอาคาร

COMPUTER CONTROL ELECTRICAL DEVICES

วัตถุประสงค์

1. เพื่อศึกษาหลักการของการส่งข้อมูลแบบอนุกรม และหลักการควบคุมอุปกรณ์ไฟฟ้าภายในอาคารด้วยคอมพิวเตอร์
2. เพื่อออกแบบโปรแกรมคอมพิวเตอร์ และวงจรควบคุมอุปกรณ์ไฟฟ้า
3. เพื่อสร้างโปรแกรมคอมพิวเตอร์ และวงจรควบคุมอุปกรณ์ไฟฟ้า
4. เพื่อนำไปใช้ในการควบคุมอุปกรณ์ไฟฟ้าในอาคาร

ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถอธิบายหลักการของการส่งข้อมูลแบบอนุกรม และหลักการควบคุมอุปกรณ์ไฟฟ้าภายในอาคารด้วยคอมพิวเตอร์ได้
2. สามารถออกแบบโปรแกรมคอมพิวเตอร์ และวงจรควบคุมอุปกรณ์ไฟฟ้าได้
3. สามารถสร้างโปรแกรมคอมพิวเตอร์ และวงจรควบคุมอุปกรณ์ไฟฟ้าได้
4. สามารถใช้คอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าได้

คอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าภายในอาคาร

นายนริศ บุญศักดิ์เฉลิม
นางสาวนันทินี แป้นแจ้ง
นายอานุภาพ เสนีย์พัลย์
นายพันธ์ศักดิ์ ภูวัน

อาจารย์ที่ปรึกษา
อาจารย์ปิยะ ศุภวราสุวัฒน์
ผศ.วิสุทธิ์ อธิพรธรรม
อาจารย์พีระวุฒิ สุวรรณจันทร์

ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอระบบคอมพิวเตอร์ควบคุมไฟฟ้าภายในอาคาร โดยแสดงกราฟฟิกรูปบ้านจำลอง ซึ่งใช้โปรแกรมวิชวลเบสิคส่งข้อมูลการควบคุมออกทางพอร์ตอนุกรม โดยใช้มาตราฐาน RS-232C ไปยังระบบไมโครคอนโทรลเลอร์ ซึ่งทำหน้าที่ควบคุมวงจรโซลิดสเตทรีเลย์ เพื่อไปควบคุมการเปิด และปิดอุปกรณ์ไฟฟ้า ซึ่งสามารถตั้งเวลาการเปิด และปิดอุปกรณ์ไฟฟ้าด้วยโปรแกรม หรือใช้งานในลักษณะสวิตช์เปิด-ปิดธรรมดาได้

COMPUTER CONTROL ELECTRICAL DEVICES

MR.NARIS BUNSAKCHALERM
MISS NUNTINEE PAENJAENG
MR.ARNUPARP SENEEPAL
MR.PANSAK PHOOWAN

ADVISORS

MR.PIYA SUPAVARASUWAT
ASSIST PROF. WISUIT ATIPORNTUM
MR.PEERAWUT SUWANJAN

1997

ABSTRACT

This thesis presents the project of computer control electrical devices, using Visual Basic Program. The computer send data to microcontroller system via RS-232C. Microcontroller control solid-state relay circuit to on/off electrical devices.This project can turn on or turn off electrical devices by program or manual switch.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี เนื่องมาจากความอนุเคราะห์ของอาจารย์ที่ปรึกษาปริญญานิพนธ์ และอาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกๆ ท่าน ที่ได้กรุณาให้คำปรึกษา และข้อแนะนำต่างๆ ตลอดจนได้รับความอนุเคราะห์ในด้านงบประมาณในการสร้างชิ้นงาน และอุปกรณ์ทดลอง จากภาควิชาครุศาสตร์วิศวกรรม ขอขอบพระคุณ คุณพ่อคุณแม่ที่เคารพที่ให้กำลังใจ และสนับสนุนในทุกๆ สิ่งทุกๆ อย่าง

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ในปัจจุบันอุปกรณ์เครื่องใช้ไฟฟ้าภายในบ้านมีความจำเป็น และสำคัญเป็นอย่างยิ่งต่อการดำรงชีวิตประจำวัน ช่วยอำนวยความสะดวกในการประกอบกิจกรรมต่างๆ ภายในบ้าน และรวมถึงการเพิ่มคุณภาพชีวิตอีกด้วย และในยุคสมัยมีการเปลี่ยนแปลง มีการพัฒนาทางด้านเทคโนโลยีเพิ่มมากขึ้น อุปกรณ์เครื่องใช้ไฟฟ้าภายในบ้านทั่วๆ ไป จึงมีการพัฒนามากขึ้น และสอดคล้องถึงการแข่งขัน ทำให้ทุกอย่างต้องทำด้วยความรวดเร็ว และถูกต้อง ดังนั้นเมื่อกิจวัตรประจำวันจะต้องมีการใช้งานอุปกรณ์ไฟฟ้าภายในบ้านอยู่เกือบตลอดเวลา ดังนั้น เทคโนโลยีต่างๆ จึงได้เข้ามาสู่ชีวิตประจำวัน เพื่อให้เกิดความสะดวกสบาย รวดเร็ว และประหยัดเวลามากขึ้น จึงได้พัฒนาการควบคุมอุปกรณ์ไฟฟ้าภายในบ้าน ซึ่งใช้ระบบคอมพิวเตอร์เข้ามาช่วยในการควบคุมการใช้งานอุปกรณ์ไฟฟ้าต่างๆ ภายในบ้าน ช่วยในการเปิด และปิดอุปกรณ์ไฟฟ้า สามารถกำหนดช่วยเวลาการใช้งานของอุปกรณ์ ตั้งเวลาในการเปิด และปิด โดยผู้ใช้สามารถกำหนดตำแหน่งผ่านทางคอมพิวเตอร์ได้ง่าย และสะดวกยิ่งขึ้น หรืออาจนำมาประยุกต์ใช้งานในด้านอื่นๆ ได้อีกด้วย โครงงานนี้จึงได้ทำขึ้นเพื่อจุดมุ่งหมายตามที่กล่าวผ่านมาแล้ว

1.2 ขีดความสามารถของโครงงาน

1.2.1 สามารถใช้งานกับคอมพิวเตอร์ทั่วไปได้

1.2.2 สามารถควบคุมอุปกรณ์ไฟฟ้าภายในอาคารได้ 8 ตำแหน่ง

1.2.3 สามารถตั้งเวลาการเปิด - ปิดอุปกรณ์ไฟฟ้าได้

1.2.4 สามารถควบคุมอุปกรณ์ไฟฟ้าต่างๆ ได้ง่าย โดยแสดงกราฟฟิกรูปบ้านผ่านทางจอคอมพิวเตอร์

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้ แบ่งออกเป็นบทต่างๆ เพื่อความสะดวกต่อการศึกษา และทำความเข้าใจ ซึ่งในแต่ละบทจะประกอบด้วยเนื้อหาที่สำคัญดังนี้

บทที่ 2 ทฤษฎี และหลักการ ประกอบด้วยเนื้อหาในทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ทฤษฎีพื้นฐานของการสื่อสารอนุกรม การควบคุมอุปกรณ์ไฟฟ้าต่างๆ โปรแกรมวิชวลเบสิค ซึ่งส่วนต่างๆ ที่กล่าวมาเป็นพื้นฐานที่สำคัญของโครงงาน

บทที่ 3 การออกแบบ การสร้าง และการทำงานกล่าวถึงการออกแบบ และการสร้างทั้งส่วนฮาร์ดแวร์ และซอฟต์แวร์ รวมทั้งการทำงานในส่วนต่างๆ ซึ่งจะทำให้เข้าใจการทำงานทั้งหมดมากขึ้น

บทที่ 4 การทดลอง และผลการทดลอง กล่าวถึงการทดลองวงจรส่วนต่างๆ ว่ามีการทำงานเป็นไปตามที่ต้องการหรือไม่ และผลการทดลองรวมของโครงงานทั้งหมด

บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไข และพัฒนา เป็นการสรุปผลการทดลองต่างๆ ของโครงงาน แสดงถึงปัญหาต่างๆ ที่เกิดขึ้นในระหว่างการทำโครงงาน และได้เสนอแนวทางในการแก้ไข พัฒนาโครงงานให้มีประสิทธิภาพ และนำไปประยุกต์ใช้งานได้กว้างมากยิ่งขึ้น

ในภาคผนวกแสดงรายละเอียดของโปรแกรม และรายการอุปกรณ์ต่างๆ ที่ใช้จัดทำโครงงานดังนี้

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข แผนผังการทำงาน

ภาคผนวก ค โปรแกรม

ภาคผนวก ง คู่มือการใช้งาน

ภาคผนวก จ รายการแสดงคุณสมบัติของอุปกรณ์

บทที่ 2
ทฤษฎี และหลักการ

2.1 กล่าวนำ

เนื้อหาของปริญญานิพนธ์ในบทนี้เป็นทฤษฎี และหลักการที่นำมาใช้ประกอบสร้างโครงงาน โดยประกอบด้วย ทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ทฤษฎีพื้นฐานของการสื่อสารอนุกรม ทฤษฎีพื้นฐานของการควบคุมอุปกรณ์ไฟฟ้า และโปรมแกรมวิชวล-เบสิค ซึ่งมีรายละเอียดดังต่อไปนี้

2.2 ทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51

2.2.1 คุณสมบัติ

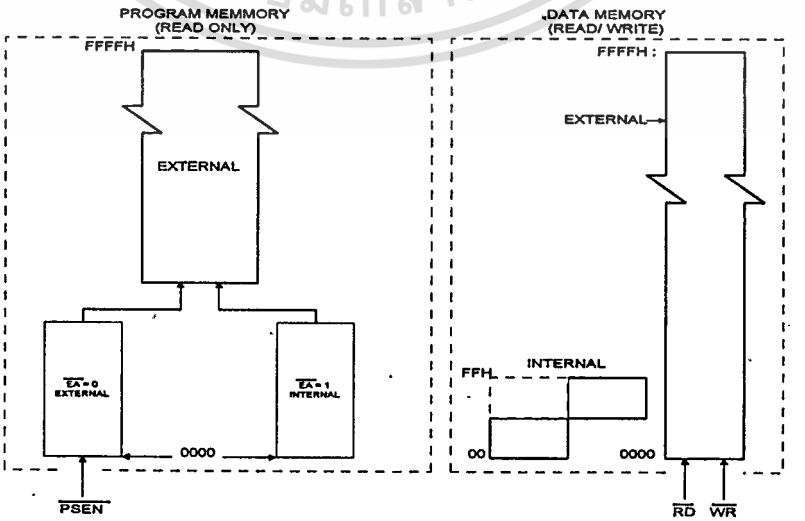
1. ซีพียู 8 บิต ที่ควบคุมได้ง่าย
2. เพิ่มการทำงานลอจิกครั้งละ 1 บิตได้
3. สายอินพุต และเอาต์พุตมีจำนวน 32 เส้น ใช้เลือกตำแหน่ง (Address) แยกต่างหากจากกันได้
4. มีแรมบรรจุไว้ภายในขนาด 128 ไบต์ หรือ 256 บิต
5. วงจรตั้งเวลา / วงจรนับมีขนาด 2, 3 หรือ 16 บิต
6. กำหนดเป็น UART (Universal Synchronous Asynchronous Receiver Transmitter) ส่งข้อมูลอนุกรมได้สองทิศทาง
7. อินเตอร์รัพต์แบ่งเป็น 2 ระดับ จาก 5 หรือ 6 แหล่ง
8. มีสัญญาณนาฬิกาอยู่ภายในตัว
9. มีหน่วยความจำสำหรับเก็บข้อมูลภายในขนาด 4 หรือ 8 กิโลไบต์ (อีพรอม 8751 และ 8752)
10. มีตำแหน่งของหน่วยความจำสำหรับเก็บโปรมแกรม จำนวนทั้งหมด 64 กิโลไบต์
11. มีตำแหน่งของหน่วยความจำสำหรับเก็บข้อมูล จำนวนทั้งหมด 64 กิโลไบต์
12. คำสั่งทั้งหมดมี 111 คำสั่ง

13. ทำงานด้วยเลขฐานสิบ และฐานสิบหก

2.2.2 หน่วยความจำ

หน่วยความจำของ 8051 แบ่งออกเป็น 2 แบบ คือ

1.Program Memory เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปรหัสภาษาเครื่อง (Machine Language) ซึ่งต้องการให้ 8051 ทำงาน เมื่อ 8051 ทำงานก็ต้องอ่านข้อมูลที่เก็บในหน่วยความจำประเภทนี้เข้าไปถอดรหัสแล้วสร้างสัญญาณควบคุมส่วนอื่นๆ ตามการทำงานของแต่ละคำสั่ง หน่วยความจำแบบนี้จะต้องเป็นแบบอ่านอย่างเดียว (Read Only Memory :ROM) และผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของหน่วยความจำเป็นรหัสภาษาเครื่องของ 8051 ตามลำดับการทำงานที่ต้องการเขียนข้อมูลลงไปบน ROM จะต้องใช้เครื่องมือพิเศษ ในระหว่างการทำงานของ 8051 ผู้ใช้จะไม่สามารถใช้คำสั่งทำการเขียนข้อมูลลงในหน่วยความจำแบบนี้ได้ จำนวนตำแหน่งสูงสุดของหน่วยความจำแบบนี้ที่ 8051 จะใช้งานได้คือ 65536 ตำแหน่ง ค่าของตำแหน่งจะเขียนเป็นเลขฐาน 16 ได้ตั้งแต่ 0000H ถึง 0FFFFH หน่วยความจำตำแหน่ง 0000H ถึง 0FFFH จำนวน 4 กิโลไบต์ นั้นผู้ใช้จะเลือกได้ว่าเป็นตำแหน่งของ ROM ที่ภายในหรือภายนอก 8051 ถ้าต้องการให้ 8051 ทำงานตามคำสั่งที่เก็บไว้ใน ROM ภายใน 8051 ก็ให้ป้อนสัญญาณสภาวะลอจิกสูง (1) เข้าที่ขา $\overline{EA}$ ของ 8051 แต่ถ้าต้องการให้ทำงานในโปรแกรมที่เก็บไว้ใน ROM ภายนอก 8051 ก็ให้ต่อลอจิกต่ำ (0) เข้าที่ขา $\overline{EA}$ ของ 8051 ส่วนหน่วยความจำ ตำแหน่ง 1FFFH ถึง 0FFFFH จะต้องต่ออยู่ภายนอก 8051 เสมอดังแสดงในแผนภูมิหน่วยความจำ (Memory Map) ในรูปที่ 2.1

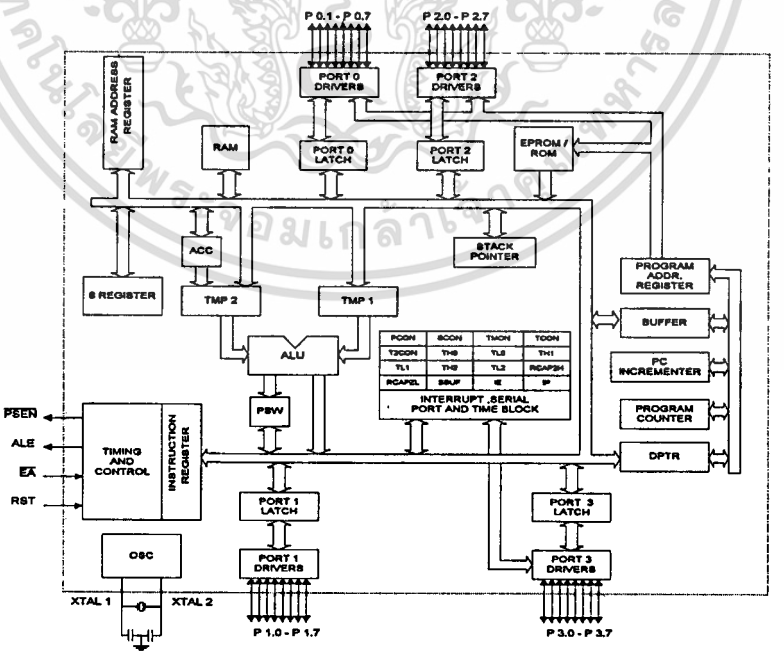


รูปที่ 2.1 แผนภูมิหน่วยความจำของ 8051

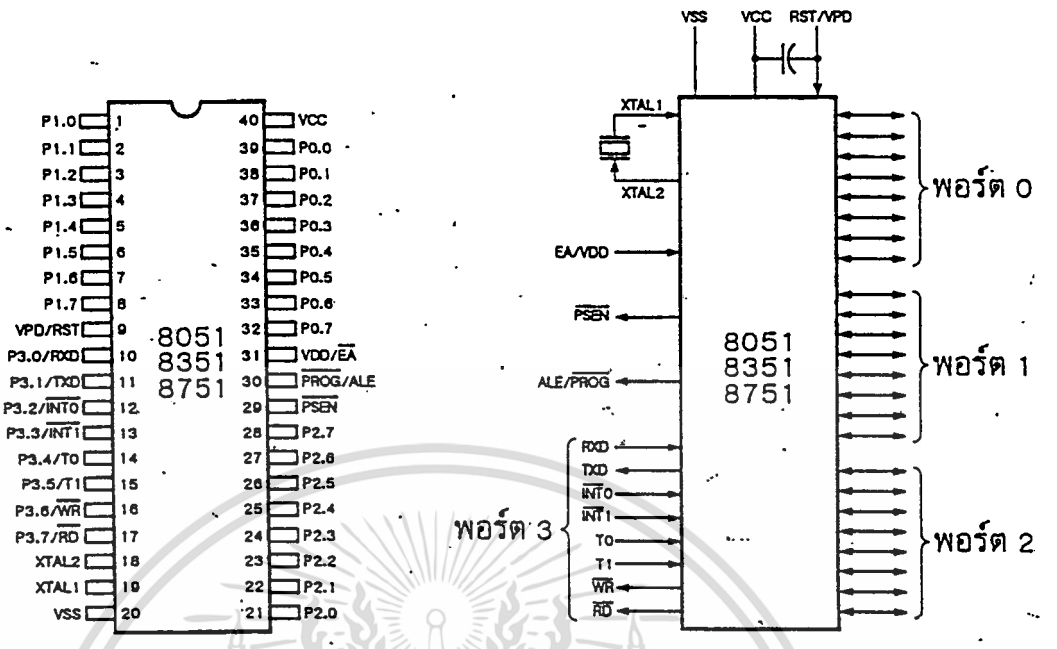
Internal Memory หมายถึง หน่วยความจำนั้นอยู่ภายใน 8051 ส่วน External Memory หมายถึง หน่วยความจำนั้นอยู่ภายนอก 8051

2. Data Memory เป็นหน่วยความจำที่ 8051 จะใช้สำหรับพัก เก็บข้อมูล แล้วเรียกมาใช้ใหม่ในระหว่างการทำงานของ 8051 การอ่าน หรือการเขียนข้อมูลจากหน่วยความจำจะกระทำโดยคำสั่งที่เก็บไว้ใน Program Memory หน่วยความจำประเภทนี้เป็นประเภทเข้าถึงข้อมูลแบบสุ่ม (Random Access Memory : RAM) ถ้ามีไฟเลี้ยงอยู่ข้อมูลที่เก็บไว้จะไม่สูญหาย หน่วยความจำแบบ Data Memory ของ 8051 จะมีอยู่ 2 ชุด ชุดหนึ่งอยู่ภายใน 8051 จำนวน 128 ไบต์ที่ตำแหน่ง 00H ถึง 7FH และอีกชุดหนึ่งจะต้องต่ออยู่ภายนอกของวงจรรวมหนึ่งจะต้องต่ออยู่ภายนอกของวงจรรวม 8051 มีได้สูงสุด 65536 ไบต์ อยู่ที่ตำแหน่ง 0000F ถึง 0FFFFH ดังแสดงในรูปที่ 2.1 หน่วยความจำแบบ Data Memory ภายใน 8051 ที่ตำแหน่ง 80H ถึง 0FFH นั้น ไม่ได้มีอยู่ทุกตำแหน่ง จะมีเฉพาะในบางตำแหน่ง ซึ่งเรียกหน่วยความจำบางตำแหน่งนี้ว่า Special Function Register (SFR) เพราะจะใช้หน่วยความจำเหล่านี้สำหรับงานพิเศษเท่านั้น อาจเป็น RAM หรือวงจรนับ วงจรตั้งเวลาก็ได้

2.2.3 สถาปัตยกรรมของ MCS-51



รูปที่ 2.2 โครงสร้างภายในของ MCS-5



รูปที่ 2.3 ตำแหน่งขาของชิพไมโครคอนโทรลเลอร์ MCS-51

หน้าที่การใช้งานแต่ละขาของชิพไมโครคอนโทรลเลอร์ MCS-51 มีดังนี้

1. ขา Vss (ขา 20) สำหรับต่อลงกราวด์

2. ขา Vcc (ขา 40) สำหรับต่อแหล่งจ่ายแรงดันกระแสไฟตรงขนาด 5 โวลต์

3. ขาพอร์ต 0 (ขา 32-39) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 0 ขนาด 8 บิต (P0.0-P0.7) แบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นพอร์ตอินพุต เอาต์พุตทั่วไปได้ และใช้ในการติดต่อหน่วยความจำสำหรับโปรแกรม และข้อมูลภายนอกชิพด้วย

4. ขาพอร์ต 1 (ขา 1-8) มี 8 ขาใช้เป็นขาสำหรับพอร์ต 1 (P1.0-P1.7) สามารถใช้เป็นพอร์ตอินพุต หรือเอาต์พุตทั่วไปได้ หากต้องการใช้งานเป็นพอร์ตอินพุต ต้องตัวภาระค่าไปยังแต่ละบิตของพอร์ตนี้

5. ขาพอร์ต 2 (ขา 21-28) มี 8 ขาใช้เป็นขาสำหรับพอร์ต 2 (P2.0-P2.7) ขนาด 8 บิต แบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นพอร์ตอินพุต เอาต์พุตทั่วไปได้

6. ขาพอร์ต 3 (ขา 10-17) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 3 (P3.0-P3.7) สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ นอกจากนี้ยังใช้งานในหน้าที่พิเศษต่างๆ ดังนี้

ขา P3.0 ใช้รับข้อมูลจากภายนอกแบบอนุกรม

ขา P3.1 ใช้ส่งข้อมูลออกไปภายนอกแบบอนุกรม

ขา P3.2 ใช้เป็นอินพุต เพื่อรับสัญญาณอินเตอร์รัพต์ชนิดที่ 0

ขา P3.3 ใช้เป็นอินพุต เพื่อรับสัญญาณอินเตอร์รัพต์ชนิดที่ 1

ขา P3.4 สัญญาณอินพุตให้ตัวนับของไทม์เมอร์ 0

ขา P3.5 สัญญาณอินพุตให้ตัวนับของไทม์เมอร์ 1

ขา P3.6 ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บภายนอกชิพ

ขา P3.7 ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลจากหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิพ

7. ขา RST (ขา 9) ใช้สำหรับการรีเซตวงจรทุกอย่างภายในชิพ เพื่อเริ่มต้นการทำงานใหม่

8. ขา ALE/PROG (ขา 30) เป็นขาสำหรับใช้ส่งสัญญาณ ออกไปภายนอก เพื่อควบคุมการแลตซ์ค่าตำแหน่งไบต์ต่ำ (Address Latch Enable) พอร์ต 0 ในการติดต่อหน่วยความจำสำหรับโปรแกรมหรือข้อมูลภายนอก

9. ขา $\overline{\text{PSEN}}$ (ขา 29) ใช้ส่งสัญญาณสโตรบเพื่ออ่านคำสั่งจากโปรแกรมที่เก็บไว้ในหน่วยความจำภายนอกชิพ (Program Strobe Enable)

10. ขา $\overline{\text{EA}}$/Vpp (ขา 31) เป็นขาสำหรับใช้เลือกให้ MCS-51 ทำงานจากโปรแกรมที่อยู่ภายในชิพหรือภายนอกชิพ หากขานี้มีสถานะเป็น 0 หมายถึงให้ใช้โปรแกรมที่อยู่ภายนอก หากขานี้มีสถานะเป็น 1 หมายถึงบังคับให้ MCS-51 ใช้โปรแกรมจากหน่วยความจำสำหรับโปรแกรมภายในชิพ ส่วน MCS-51 ที่ไม่มีหน่วยความจำโปรแกรมสำหรับเก็บโปรแกรมภายในชิพ ให้ต่อขานี้ลงกราวด์เสมอ

11. ขา XTAL1 (ขา 19) ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุตเข้าสู่วงจรออสซิสเลเตอร์

12. ขา XTAL2 (ขา 18) ใช้ต่อคริสตอลภายนอก โดยเป็นเอาต์พุตออกจากวงจรออสซิสเลเตอร์

2.3 ทฤษฎีพื้นฐานของการสื่อสารอนุกรม

2.3.1 มาตราฐาน RS-232C

โดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรมอยู่ในตัวเครื่อง ซึ่งมีชื่อเรียกว่า RS-232C เครื่องหลายเครื่องที่ไม่มีพอร์ตมากับเครื่อง อย่างเช่น IBM PC จำเป็นจะต้องมี

การ์ดที่เรียกว่า อะซิงโครนัสอะแดปเตอร์ (Asynchronous Communication Adapter) มาเสียบเพื่อนำไปใช้ในการสื่อสารข้อมูล

พอร์ต RS-232C นี้ ทำหน้าที่ในการรับ และการส่งข้อมูลในแบบอนุกรม เรียกว่า Universal Asynchronous Adapter สาเหตุที่มีชื่อเรียกว่า RS-232C เนื่องมาจากสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์ของอเมริกาหรือ EIA ได้กำหนดมาตรฐานของอุปกรณ์การสื่อสารแบบอนุกรมขึ้นมา ซึ่งคำว่า RS ย่อมาจาก Recommaned Standard ส่วน 232 เป็นหมายเลขบ่งบอกของมาตรฐานนี้ และ C เป็นหมายเลขของฉบับสุดท้ายของมาตรฐานนี้ และจุดประสงค์หลักของมาตรฐานตัวนี้ เพื่อบรรยายคุณลักษณะของการเชื่อมต่อของอุปกรณ์รับส่งข้อมูล (Data Terminal Equipment : DTE) กับอุปกรณ์ที่ใช้ในการสื่อสารข้อมูล(Data Communication Equipment : DCE) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE หมายถึงตัวไมโครคอมพิวเตอร์ และ DCE หมายถึง โมเด็ม อุปกรณ์อื่นๆ เช่น เครื่องพิมพ์ที่รับสัญญาณแบบอนุกรมอาจจะเป็นได้ทั้ง DTE และ DCE ขึ้นอยู่กับผู้ผลิต ข้อแตกต่างของ DTE และ DCE เห็นได้จากรูปที่ 2.4 จากรูปนั้นจะเห็นได้ว่า RS-232C มีส่วนสำคัญสำหรับการสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์

รูปที่ 2.4 การใช้ RS-232C เชื่อมต่ออุปกรณ์

ความจริงที่ควรทราบอีกประการหนึ่งของ RS-232C สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้จาก 0-20,000 บิตต่อวินาที ซึ่งเพียงพอสำหรับไมโครคอมพิวเตอร์ที่มีขนาดอัตราบอด 110 ถึง 9600 บอด และความยาวของสายที่ใช้ในการเชื่อมต่อตามมาตรฐาน RS232-C มีค่าจำกัดอยู่แค่ 50 ฟุต ซึ่งเพียงพอสำหรับการสื่อสารของไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอก

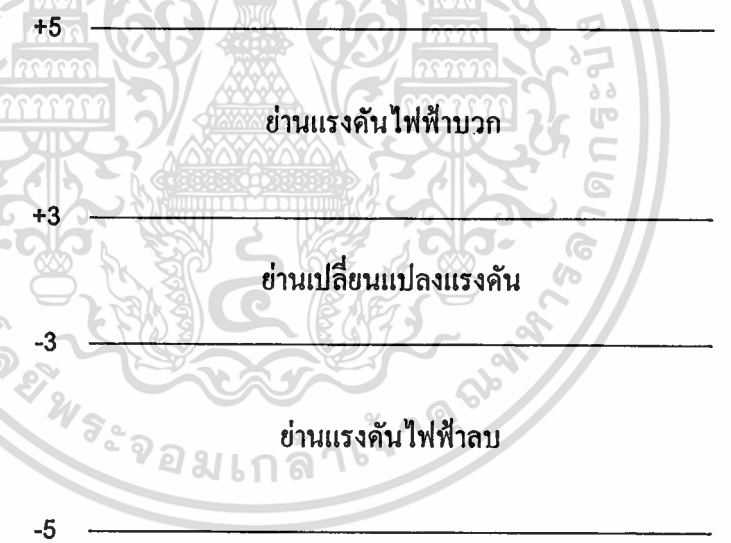
2.3.2 ลักษณะของสัญญาณ RS-232C

เพื่อเป็นการรองรับว่าข้อมูลถูกส่งออกไป และอุปกรณ์ที่ควบคุมถูกต้อง จึงจำเป็นจะต้องมีข้อตกลงกันในเรื่องของสัญญาณที่ใช้มาตรฐาน RS-232C จึงต้องมีการกำหนดย่านแรง-

ดันไฟฟ้าในสัญญาณเพื่อสนองจุดประสงค์ที่กล่าวมาแล้ว ดังแสดงในตารางที่ 2.1 และรูปที่ 2.5

ตารางที่ 2.1 การกำหนดย่านแรงดันไฟฟ้าในสัญญาณ

มาตราฐานของการใช้แรงดันไฟฟ้า			
แรงดันไฟฟ้า	สภาวะลอจิก	สภาวะของสัญญาณ	ฟังก์ชั่นในการควบคุม
บวก	0	SPACE	ON
ลบ	1	MARK	OFF



รูปที่ 2.5 ย่านของแรงดันไฟฟ้าที่ใช้ในสัญญาณ RS-232C

สำหรับไมโครคอมพิวเตอร์บางเครื่อง อาจจะใช้สัญญาณลอจิกภายในเครื่องเป็นสัญญาณของ RS-232C เช่น อะซิงโครนัสอะแดปเตอร์ของ IBM PC ในกรณีเช่นนี้ระยะทางของสายที่เชื่อมต่ออาจจะใช้ได้สั้นกว่า 50 ฟุต เนื่องจากระดับของกราวด์เปลี่ยนไป อันเนื่องจากการสูญเสียความต้านทานไปในสาย

2.3.3 การกำหนดจุดข้อต่อของ RS-232C

ในทางกายภาพมาตรฐานของ RS-232C กำหนดข้อต่อเป็นแบบ DB-25 แต่ละขาของข้อต่อได้กำหนดไว้ ดังแสดงในรูปที่ 2.6 อย่างไรก็ตาม ผู้ผลิตไมโครคอมพิวเตอร์อาจจะใช้ข้อต่อชนิดอื่น เช่น Fujitsu F-8 IBM AT, IBM Jr เป็นต้น ข้อต่อตัวเมียควรจะอยู่ที่ตัวโมเด็ม ขณะที่ข้อต่อตัวผู้ควรจะอยู่ที่อะซิงโครนัสอะแดปเตอร์ หรือที่ตัวไมโครคอมพิวเตอร์ สัญญาณต่างๆ ทำหน้าที่ ดังนี้

1.Transmit Data (TD ขาที่ 2) เป็นขาที่ใช้ส่งสัญญาณออกจากตัวไมโครคอมพิวเตอร์ไปยังโมเด็มหรือเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่นหรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออก สภาวะของลอจิกที่ขานี้จะมีค่าเท่ากับ 1 หรือเท่ากับบิตหยุด

2. Receive Data (RD ขาที่ 3) เป็นขารับสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์ เมื่อไม่มีสัญญาณรับเข้ามาขานี้จะมีสภาวะทางลอจิกเป็น 1

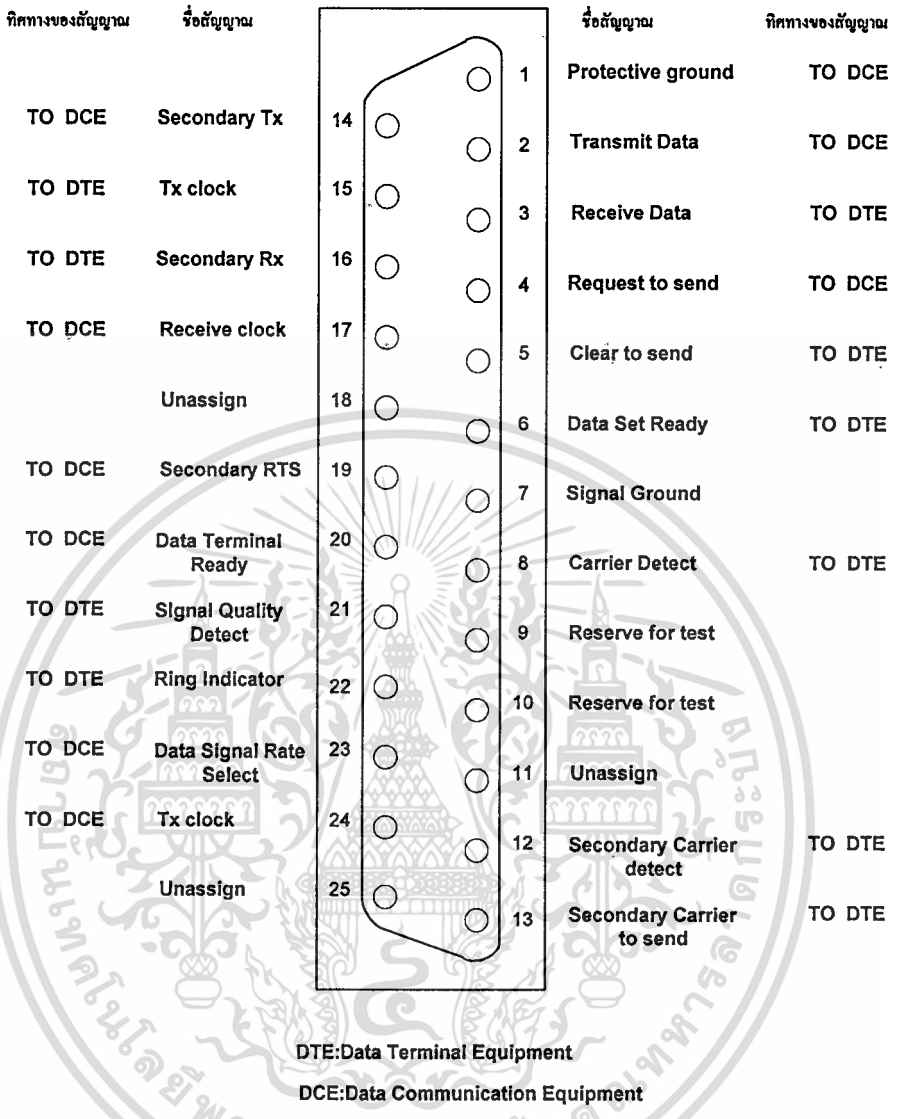
3. Request To Send (RTS ขาที่ 4) ใช้สำหรับส่งสัญญาณไปยังโมเด็มหรือเครื่องพิมพ์ เป็นการเรียกร้องที่จะส่งสัญญาณมาทางขา 2 สัญญาณนี้จะใช้คู่กับ CTS หรือ Clear to Send อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับก็จะส่งสัญญาณออกไปที่สาย CTS

4. Clear To Send (CTS ขาที่ 5) เมื่อสัญญาณนี้อยู่ในสถานะออฟ หรือลอจิกสูง แสดงว่า อุปกรณ์ทางด้านรับพร้อมที่จะรับข้อมูลแล้ว

5. Data Set Ready (DSR ขาที่ 6) เมื่อสัญญาณนี้อยู่ในสถานะออน หรือลอจิกต่ำ เป็นการบอกไมโครคอมพิวเตอร์หรือฝ่ายส่งว่าได้ต่อสัญญาณเรียบร้อยแล้ว และพร้อมที่จะส่งได้แล้ว

6. Signal Ground (SG ขาที่ 7) SG ทำหน้าที่เป็นระดับแรงดันอ้างอิง สำหรับทุกๆ สายของสัญญาณ จะมีระดับแรงดันเป็นลอจิกต่ำ เมื่อเทียบกับสัญญาณตัวอื่น

7: Carrier Detect (CD ขาที่ 8) โมเด็มจะส่งสัญญาณในสถานะออน หรือลอจิกต่ำ ไปบอกไมโครคอมพิวเตอร์ เมื่อได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่ง สัญญาณนี้จะนำไปจุดแอล-อีดี บอกว่าได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่ง แล้วไฟแอลอีดีจะอยู่บนหน้าปัทม์ของโมเด็ม



รูปที่ **2.6** การกำหนดขั้วต่อของ RS-232C

8. Data Terminal Ready (DTR ขาที่ 20) คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน หรือลอจิก 0 เมื่อพร้อมที่จะติดต่อกับโมเด็ม โมเด็มส่วนมากจะไม่รายงานสถานภาพของตัวเอง (CD, DSR และ CTS) ให้คอมพิวเตอร์รู้ หากคอมพิวเตอร์ไม่เปิดสัญญาณ DTR

9. Ring Indicator (RI ขาที่ 22) สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบได้อัตโนมัติ(Auto-answer) สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งมา และออฟระหว่างเสียงดังของกระดิ่ง

ตารางที่ 2. 2 คุณลักษณะโดยย่อของสัญญาณ RS-232C

คุณลักษณะทางไฟฟ้า	
Driver output logic levels with	15V > 0 >5V
3k to 7k load	-5V > 0 > -15V
Driver output voltage when open circuit	Vo < 25V
Driver output impedance with Power off	Ro > 300 ohms
Output Short circuit current	Io < 0.5 A
Driver slew rate	dv/dt < 30 V/s
Receiver input impedance	7k > Rin >3k
Receiver input voltage	+ 15 compatible with driver
Receiver output with open circuit input	MARK
Receiver output with +3 V input	SPACE
Receiver output with -3 V input	MARK
+15	Logic 0 = Space
+5	Control On
+5	Noise Margin
+3	
+3	Transition Region
-3	
-3	Noise Margin
-5	
-5	Logic 1 = Mark
-15	Control Off

2.4 ทฤษฎีพื้นฐานของการควบคุมอุปกรณ์ไฟฟ้า

2.4.1 การประยุกต์ใช้คอมพิวเตอร์ในการควบคุมสวิตช์

คอมพิวเตอร์เป็นอุปกรณ์ที่สามารถเก็บรวบรวม และประมวลข้อมูลได้มหาศาล ไม่ว่าข้อมูลนั้นจะเป็นตัวเลข ตัวหนังสือ หรือข้อมูลใดๆ ที่สามารถแปลงให้อยู่ในรูปเลขฐานสองได้ และนอกจากนั้น คอมพิวเตอร์ยังสามารถทำงานอื่นๆ นอกเหนือจากการเก็บรวบรวม และประมวลผลข้อมูลได้ เช่น ใช้เป็นอุปกรณ์ควบคุมการจ่ายกำลังไฟฟ้าให้ตัวภาระทางไฟฟ้าทั่วๆ ไป ตัวอย่างเช่น เมื่อต้องการที่จะเปิด หรือปิดวงจรไฟฟ้าโดยใช้คอมพิวเตอร์เป็นตัวควบคุมนั่น หมายถึงว่า วงจรไฟฟ้าที่ต่อใช้งานต้องมีทำงานด้วยแรงไฟตรง 5 โวลต์ หรือนำไปขับตัวภาระอีกทีหนึ่งโดยใช้ระดับแรงดัน 5 โวลต์ โดยในส่วนของโปรแกรมควบคุมสวิตช์นั่นต้องมีการอ่านอินพุตมาจากอุปกรณ์ตรวจจับ (Sensor) มีข้อมูลของวัน และเวลา ข้อมูลที่ป้อนจากผู้ใช้ และข้อมูลมาตรฐานอื่นๆ ที่เกี่ยวข้องมาใช้ประกอบร่วมกัน เพื่อใช้ประมวลผลว่าจะใช้สวิตช์ปิด หรือเปิดวงจรเมื่อใด

การที่จะควบคุมการตัดต่อพลังงานไฟฟ้าไปยังตัวภาระโดยใช้คอมพิวเตอร์ควบคุมนั้นต้องมีการเชื่อมต่อ หรืออินเตอร์เฟสระหว่างลอจิกเอาต์พุตของคอมพิวเตอร์กับอุปกรณ์ที่ทำการตัดต่อพลังงานไฟฟ้าไปยังตัวภาระ ซึ่งถ้าหากเป็นรีเลย์จะใช้การจ่ายแรงดันให้กับขดลวดของรีเลย์ เพื่อทำให้หน้าสัมผัสปิด หรือเปิด แต่ถ้าเป็นสวิตช์ทางอิเล็กทรอนิกส์ที่ไม่มีส่วนเคลื่อนที่ทางกล จะใช้วิธีการจ่ายกระแสควบคุมเข้าสู่อุปกรณ์สารกึ่งตัวนำที่นำมาทำเป็นสวิตช์นั้นแทน

ขั้นตอนการเลือกสวิตช์

สวิตช์โดยทั่วๆ ไปจะประกอบด้วยเทอร์มินอลหนึ่งคู่ หรือมากกว่านั้นซึ่งเทอร์มินอลดังกล่าวอาจจะเป็นหน้าสัมผัสทางกลเป็นอุปกรณ์พวกสารกึ่งตัวนำ หรือเป็นไอซีก็ได้ โดยทั่วไปสวิตช์ที่ถูกควบคุมโดยวิธีการทางอิเล็กทรอนิกส์นั้น จะใช้การจ่าย หรือตัดแรงดันคร่อมเทอร์มินอลทั้งสองของสวิตช์ เพื่อสั่งให้สวิตช์ปิด หรือเปิดวงจร

สวิตช์ในทางอุดมคตินั้น จะมีลักษณะเด่นอยู่ 3 ประการ คือ เมื่อสวิตช์เปิดจะแยกจากกันโดยสมบูรณ์ และอิมพีแดนซ์ระหว่างขั้วเทอร์มินอลทั้งสองจะมีค่าอนันต์ เมื่อสวิตช์ปิดจะหมายถึง ค่าที่เทอร์มินอลทั้งสองจะต่อถึงกันโดยสมบูรณ์ และค่าอิมพีแดนซ์ระหว่างเทอร์มินอลทั้งสองจะเป็นศูนย์ และลักษณะสุดท้ายก็คือเมื่อได้รับ หรือตอบสนองต่อสัญญาณควบคุมจะมีการปิด หรือเปิดโดยสมบูรณ์ทันทีที่ได้รับสัญญาณ โดยไม่มีการหน่วงเวลา หรือเกิดการ

บาวซน์(Bounce) แต่ความจริงแล้วจะพบว่า ไม่มีสวิตช์แบบใดเลยที่มีคุณลักษณะเป็นไปตามอุดมคติสิ่งที่ต้องทำ คือ เลือกสวิตช์ให้ได้ตามความต้องการ และดูว่าสวิตช์แบบใดที่มีพิกัดเป็นไปตามข้อกำหนดดังต่อไปนี้

1. กระแส และแรงดันควบคุม (Control Voltage and Current) จะมีการกำหนดขนาดพิกัดของกระแส และแรงดันสำหรับเทอร์มินอลควบคุม ที่จะเป็นตัวควบคุมให้สวิตช์ปิด หรือเปิดวงจรได้ ดังนั้น ระดับสัญญาณควบคุมในวงจรต้องมีขนาดตรงตามพิกัดนั้น

2. กระแสตัวภาระ (Load current) สวิตช์ที่นำมาใช้ต้องสามารถทนต่อขนาดกระแสที่สูงกว่าขนาดของกระแสสูงสุดของตัวภาระได้

3. แรงดันสวิตชิ่ง (Switching) เป็นระดับสูงสุดที่สามารถตกคร่อมเทอร์มินอลของสวิตช์ได้โดยไม่ก่อให้เกิดอันตราย ซึ่งโดยปกติแล้วระดับแรงดันนี้จะสูงกว่าระดับแรงดันทำงานของสวิตช์

4. ความเร็วในการสวิตช์ (Switching Speed) สำหรับการควบคุมอย่างง่ายนั้น ความเร็วของการสวิตช์จะขึ้นกับค่าที่ต้องการพิจารณาดูเท่านั้น แต่สำหรับในวงจรควบคุมที่มีความยุ่งยากซับซ้อนนั้น ความเร็วของการสวิตช์จะเป็นสิ่งที่สำคัญมาก เช่น ในการใช้สวิตชิ่งแหล่งจ่ายแรงดันจ่ายกระแสไฟฟ้าให้กับตัวภาระที่เป็นขดลวดเหนี่ยวนำในอัตราความถี่ 20 กิโลเฮิรตซ์ หรือมากกว่านั้น ต้องการหาค่าความเร็วสูงสุดได้ โดยการคำนวณจากค่าเวลาเปิดและปิดของสวิตช์ตามสมการ

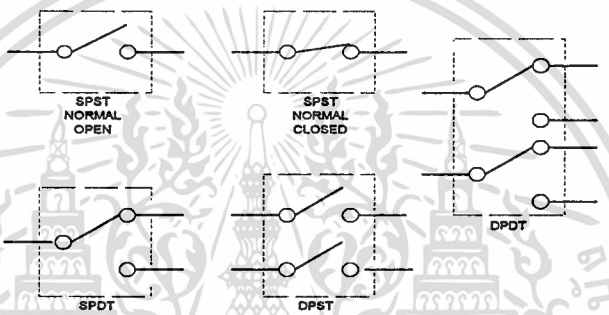
$$\text{ความเร็วสูงสุด} = \frac{1}{(T_{on} + T_{off})} \qquad (2.1)$$

เมื่อ Ton คือ ช่วงเวลาเปิดสูงสุด

Toff คือ ช่วงเวลาปิดสูงสุด

ส่วนรายละเอียดอื่นๆ ที่ต้องนำมาพิจารณาจะเป็นเรื่องราคา ขนาด และความสามารถใช้ในวงจรได้ ตัวอย่างในรูปที่ 2.7 เป็นลักษณะของการสวิตช์ทางกลที่สามารถนำมาใช้ในวงจรได้ ซึ่งหากใช้เป็นสวิตช์อิเล็กทรอนิกส์แทน ต้องให้มีการเทอร์มินอลตามจำนวนสวิตช์แต่ละแบบ ถ้าเป็นสวิตช์แบบปกติเปิด (Normal-Open) จะเปิดหน้าสัมผัสเมื่อไม่มีไฟเลี้ยงจ่ายให้กับมัน และจะปิดหน้าสัมผัสเมื่อมีไฟเลี้ยงจ่าย ส่วนสวิตช์แบบปกติปิด (Normal-Closed)

ก็จะทำงานในลักษณะที่ตรงข้ามกับแบบปกติเปิด คือ จะปิดหน้าสัมผัสเมื่อไม่มีไฟเลี้ยง และจะเปิดหน้าสัมผัส เมื่อมีไฟเลี้ยงจ่ายให้กับมัน ลักษณะการเรียกรูปแบบสวิตช์ดัง ในรูปที่ 2.8 จะพบว่า ถ้าเป็นสวิตช์แบบ Double-Throw (DT) จะมีทางออกของเอาต์พุต 2 ทาง ซึ่งการควบคุมให้เอาต์พุตออกทางไหนก็ขึ้นอยู่กับแรงดันควบคุม ส่วนแบบ Double-Pole (DP) นั้นจะมีเทอร์มินอลทางอินพุตของสวิตช์ 2 ชุด และใช้สัญญาณควบคุมตัวเดียวกัน และถ้าเป็นแบบ Double-Pole Double-Throw (DPDT) นั้น จะมีเทอร์มินอลทางอินพุต 2 ชุด และแต่ละชุดจะมีทางออกของเอาต์พุต 2 ทาง ดังรูป



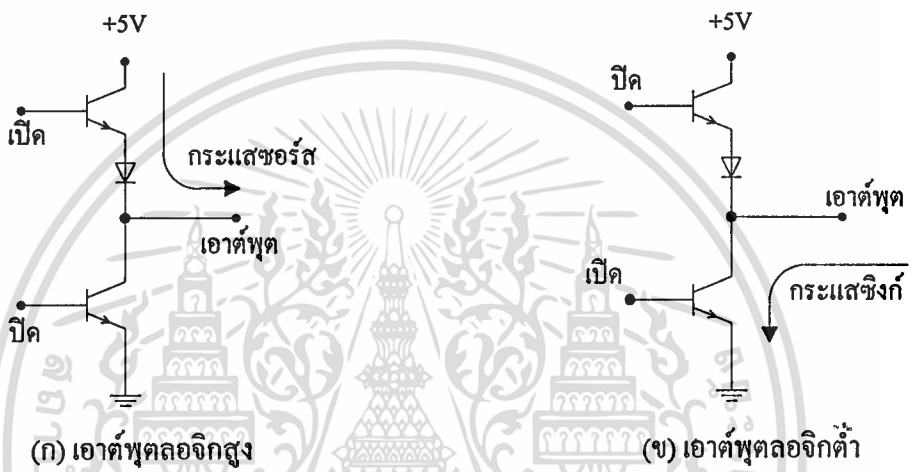
รูปที่ 2.7 ลักษณะของสวิตช์ทางกล 5 แบบ ที่สามารถเปลี่ยนไปใช้ทางอิเล็กทรอนิกส์ได้

การใช้ลอจิกเอาต์พุตเป็นสวิตช์หรือไปขับสวิตช์

กรณีการใช้งานกับตัวภาระที่ใช้กับแรงดัน และกระแสต่ำๆ สามารถที่ใช้ลอจิกเกตหรือสัญญาณทางเอาต์พุตพอร์ตใดๆ ทำหน้าที่เป็นสวิตช์ก็ได้ แต่สำหรับตัวภาระที่ใช้กับแรงดัน และกระแสสูงๆ ขึ้นไปจะต้องใช้สัญญาณลอจิกเอาต์พุตไปควบคุมทรานซิสเตอร์ให้ทำงาน แล้วให้กระแสตัวภาระไหลผ่านทรานซิสเตอร์แทน ดังนั้นคุณลักษณะของลอจิกเอาต์พุตจึงเป็นเรื่องที่ต้องทราบไว้โดยสามารถดูได้ในรายการแสดงคุณสมบัติของอุปกรณ์ของไอซีเบอร์นั้นๆ

โดยทั่วๆ ไป การใช้ลอจิกเอาต์พุตไปขับตัวภาระอื่นๆ ที่มีขนาดใหญ่กว่า หรือไม่ใช่ลอจิกอินพุตโดยตรงนั้น จำเป็นต้องคำนึงถึงค่ากระแสซอร์ส, กระแสซิงก์ และการจ่ายกำลังงานของชิพนั้นๆ ด้วย ซึ่งค่าต่างๆ มีคุณลักษณะดังแสดงในรูปที่ 2.8 โดยจะต้องทำความเข้าใจเกี่ยวกับลอจิกเอาต์พุตว่าเป็นกระแสที่ไหลเข้า หรือออกจากเอาต์พุต ซึ่งจริงๆ แล้วการที่กระแสจะไหลเข้า หรือออกจากเอาต์พุต จะขึ้นอยู่กับว่าเอาต์พุตนั้นเป็นลอจิกสูง หรือต่ำ

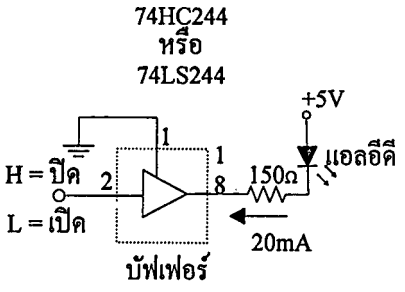
โดยทั่วๆ ไป การใช้ลอจิกเอาต์พุตไปขับตัวภาระอื่นๆ ที่มีขนาดใหญ่กว่า หรือไม่ใช่ลอจิกอินพุตโดยตรงนั้น จำเป็นต้องคำนึงถึงค่ากระแสซอร์ส, กระแสซิงก์ และการจ่ายกำลังงานของชิพนั้นๆ ด้วย ซึ่งค่าต่างๆ มีคุณลักษณะดังแสดงในรูปที่ 2.8 โดยจะต้องทำความเข้าใจเกี่ยวกับลอจิกเอาต์พุตว่าเป็นกระแสที่ไหลเข้า หรือออกจากเอาต์พุต ซึ่งจริงๆ แล้วการที่กระแสจะไหลเข้า หรือออกจากเอาต์พุต จะขึ้นอยู่กับว่าเอาต์พุตนั้นเป็นลอจิกสูง หรือต่ำ



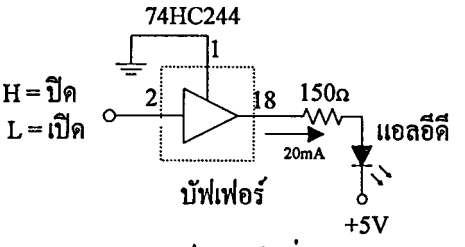
รูปที่ 2.8 ลักษณะของกระแสซอร์ส และกระแสซิงก์

ในรูปที่ 2.8 จะพบค่ากระแสซอร์สที่สภาวะเอาต์พุตลอจิกสูงนั้นจะไหลจากแหล่งจ่ายแรงดันออกมา แต่กระแสซิงก์ที่สภาวะเอาต์พุตลอจิกต่ำนั้นจะไหลลงกราวด์ ซึ่งจะมองกระแสซอร์สในรูปของกระแสที่ไหลจากเอาต์พุตลอจิกสูงผ่านตัวภาระไปยังเอาต์พุตลอจิกต่ำ ซึ่งโดยทั่วๆ ไป ตัวภาระมักเป็นลอจิกใดๆ แต่บางทีอาจเป็นวงจรอื่นๆ ที่ใช้ลอจิกเอาต์พุตไปขับก็ได้ และข้อมูลของไอซีส่วนใหญ่จะใช้เลขติดลบ เพื่อแสดงว่าเป็นกระแสซอร์ส

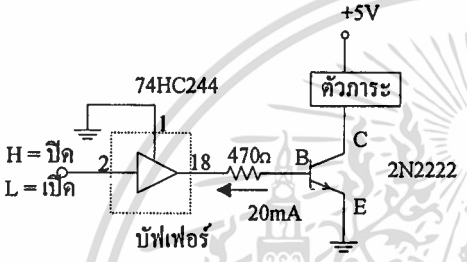
ลักษณะของกระแสซอร์ส และกระแสซิงก์ของอุปกรณ์ แต่ละตระกูลจะไม่เหมือนกัน กล่าวคือ ถ้าเป็นอุปกรณ์พวกซีมอสนั้น ลอจิกเอาต์พุตจะเป็นแบบสมมาตร ซึ่งหมายถึงกระแสซอร์ส และกระแสซิงก์มีขนาดเท่ากัน แต่ถ้าเป็นอุปกรณ์พวกทีทีแอล และเอ็นมอสจะมีเอาต์พุตที่แตกต่างไปจากซีมอส คือ จะมีกระแสซิงก์มากกว่ากระแสซอร์ส ซึ่งหากต้องการใช้เอาต์พุตของอุปกรณ์ทีทีแอล หรือเอ็นมอสไปขับตัวภาระต้องใช้ลอจิกเอาต์พุตที่เป็นลอจิกต่ำไปขับตัวภาระ



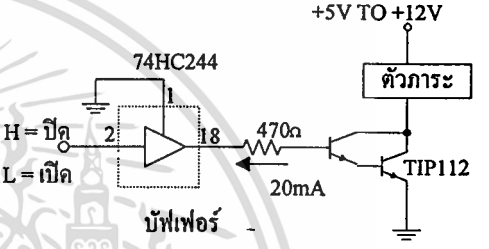
(ก) เอาต์พุตลอจิกสูง



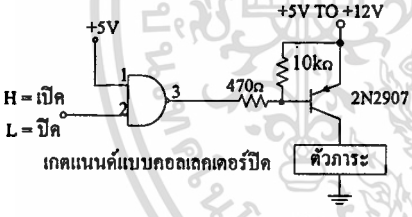
(ข) เอาต์พุตลอจิกต่ำ



(ค) ทรานซิสเตอร์เอ็นพีเอ็น



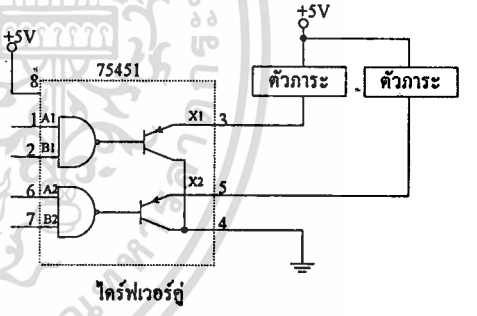
(ง) ทรานซิสเตอร์แบบดาร์ลิงตันเอ็นพีเอ็น



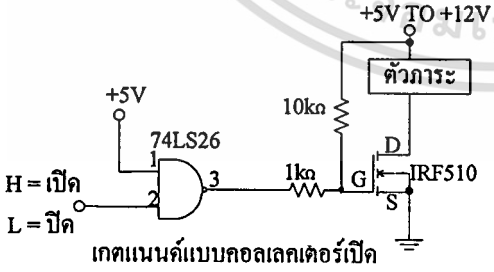
(จ) ทรานซิสเตอร์พีเอ็นพี

ตารางความจริง

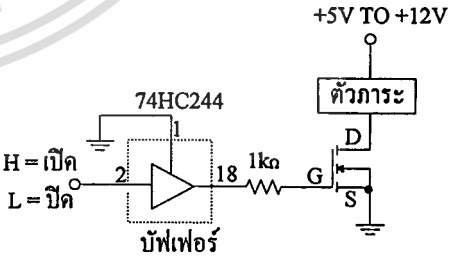
A	B	X
0	0	เปิด
0	1	ปิด
1	0	ปิด
1	1	ปิด



(ฉ) ไอซีไดร์ฟเวอร์



(ช) มอสเฟตทำงานที่แรงดัน 10 โวลต์



(ช) มอสเฟตทำงานที่แรงดัน 5 โวลต์

รูปที่ 2.9 วงจรที่ใช้ลอจิกเอาต์พุตไปควบคุมสวิตช์จ่ายกำลังให้ตัวภาระ

สำหรับกรณีการใช้ไมโครคอนโทรลเลอร์ ซึ่งมีความซับซ้อนมากขึ้น การพิจารณากระแสเอาต์พุตสูงสุดนั้น บางทีอาจต้องพิจารณาค่ากระแสทั้งหมดของแต่ละพอร์ต แล้วหาค่ากระแสรวมจากทุกพอร์ตด้วย เช่น ที่พอร์ตเอาต์พุตของ PIC16C5X จะมีค่ากระแสซิงก์ถึง 25 มิลลิแอมป์ และกระแสซอร์สถึง 20 มิลลิแอมป์ แต่ค่ากระแสรวมที่แต่ละพอร์ตนั้น จะต้องไม่เกิน 50 มิลลิแอมป์ สำหรับกระแสซิงก์ และ 40 มิลลิแอมป์สำหรับกระแสซอร์ส ซึ่งค่าดังกล่าวนี้เป็นค่าอัตราส่วนสัมประสิทธิ์สูงสุดของชิพแล้ว ดังนั้นการออกแบบต้องออกแบบให้ทำงานได้ดีภายในย่านพิกัดนี้

ถ้าหากออกแบบให้วงจรมีกระแสเอาต์พุตเพียงไม่กี่มิลลิแอมป์การเลือกใช้การเลือกใช้ไมโครคอนโทรลเลอร์เป็นทางเลือกที่ดีอย่างหนึ่ง แต่ถ้าหากต้องการให้มีกระแสเอาต์พุตสูงขึ้นก็สามารถใช้เอาต์พุตพอร์ตไปขับบัฟเฟอร์ เช่น 74HC244 ก่อนได้ เพราะบัฟเฟอร์จะช่วยแยกส่วนวงจรของคอมพิวเตอร์กับส่วนของวงจรตัวภาระออกจากกันด้วย และเมื่อเกิดผิดปกติขึ้นในวงจรใดวงจรหนึ่งก็จะทำให้เสียเพียงบัฟเฟอร์เท่านั้น

การใช้เอาต์พุตจากอุปกรณ์บัฟเฟอร์ เช่น 74HC244 หรือ 74HC374 สามารถที่จะส่งไปขับตัวภาระโดยผ่านสายเคเบิลได้ในระยะทางประมาณ 10 ฟุต เช่น ใน PC ที่ส่งเอจต์พุตแบบขนานไปยังเครื่องคอมพิวเตอร์จะใช้เอาต์พุตจาก 74LS374 ผ่านสายไฟยังเครื่องพิมพ์ ถ้าใช้เอาต์พุตจากพอร์ตแบบขนานของเครื่องพีซีเป็นสัญญาณควบคุมแล้วใช้บัฟเฟอร์พวก 74HCT ที่ปลายทางของสายส่งสัญญาณแล้วจะทำให้แน่ใจได้ว่าสัญญาณที่ได้ทางปลายสายจะเป็นสัญญาณที่สมบูรณ์ไม่มีสัญญาณรบกวน หรือไม่ผิดเพี้ยน

การใช้ไบโพลาร์ทรานซิสเตอร์เป็นสวิตช์

ถ้าหากตัวภาระในวงจรต้องการใช้กระแส หรือแรงดันในระดับที่สูงกว่าที่ได้มาจากลอจิกเอาต์พุตจะสามารถเอาลอจิกเอาต์พุตนี้ไปควบคุมทรานซิสเตอร์ก่อนได้ ซึ่งไบโพลาร์ทรานซิสเตอร์ปีนตัวเลือกหนึ่งที่มีราคาไม่แพง และการใช้งานเป็นวงจรขยายกระแสได้ง่าย

ถึงแม้ว่าทรานซิสเตอร์จะมีมากมายหลายแบบซึ่งทำให้ยากต่อการเลือก สามารถใช้ทรานซิสเตอร์อย่างง่ายแบบใช้งานทั่วๆ ไปแบบใดมาใช้งานก็ได้ ที่มีพิกัดขนาดแรงดัน และกระแสเป็นไปตามที่ต้องการ ดังแสดงในรูปที่ 2.9 (ค) เป็นวงจรที่ใช้ทรานซิสเตอร์แบบธรรมดาเบอร์ 2N2222 โดยใช้สัญญาณลอจิกสูงมาเป็นตัวขับให้ทรานซิสเตอร์ทำงานใน

สภาวะทำงาน ซึ่งมีกระแสเพียงเล็กน้อยไหลผ่านจากขาเบสไปอิมิตเตอร์ ทำให้ค่าความต้านทาน ทรานซิสเตอร์ไปยังตัวภาระ และไหลลงกราวด์ได้ และในสภาวะที่ทรานซิสเตอร์ทำงานจะมีแรงดันตกคร่อมระหว่างขาคอลเล็กเตอร์ กับอิมิตเตอร์ประมาณ 0.3 โวลต์ ดังนั้นแรงดันทั้งหมดจากแหล่งจ่ายแรงดันจะไม่ตกคร่อมทั้งหมด

ค่าความต้านทานที่ต่อจำกัดกระแสเบสของทรานซิสเตอร์ไม่จำเป็นก็ได้ เพราะโดยทั่วไปจะใช้ค่าอยู่ระหว่าง 200-1,000 โอห์ม การเลือกค่าความต้านทานนี้ควรให้มีค่าต่ำพอที่จะมีกระแสไหลผ่านไปทำให้ทรานซิสเตอร์นำกระแสได้ และในขณะเดียวกันก็ต้องเป็นค่ากระแสสูงพอที่จะป้องกันกระแสจากลอจิกเอาต์พุตที่สูงเกินไปไม่ให้ไหลเข้าทำความเสียหายแก่ทรานซิสเตอร์ได้

ขนาดของแหล่งจ่ายแรงดันที่จะจ่ายกำลังไฟฟ้าตรงให้กับตัวภาระนั้นควรมีค่าตั้งแต่5โวลต์ขึ้นไป แต่ถ้าหากมีขนาดเกิน 12 โวลต์แล้ว ต้องทำการตรวจสอบแรงดันพังทลายระหว่างขาคอลเล็กเตอร์กับอิมิตเตอร์ของทรานซิสเตอร์ด้วย โดยต้องแน่ใจว่าแรงดันนี้มีขนาดที่สูงกว่าแรงดันที่จ่ายมาจากแหล่งจ่ายแรงดันที่ตกคร่อมทรานซิสเตอร์ในขณะที่มันไม่ทำงาน

สำหรับกรณีที่กระแสตัวภาระมีค่ามากๆ ก็สามารถใช้ทรานซิสเตอร์แบบดาร์ลิงตันก็ได้ ซึ่งมีลักษณะดังรูปที่ 2.9 (ง) จะพบว่าแรงดันที่ไหลผ่านทรานซิสเตอร์ตัวแรกจะเป็นกระแสเบสให้กับทรานซิสเตอร์ตัวที่สอง ทำให้สามารถรับกระแสตัวภาระสูงๆได้ เพราะอัตราการขยายทั้งหมดจะมีค่าเท่ากับค่าอัตราการขยายของทรานซิสเตอร์สองตัวคูณกัน โดยทั่วไปอัตราขยายของทรานซิสเตอร์แบบดาร์ลิงตันนี้จะมีค่าเป็น 1,000 ซึ่งตัวอย่างของ ทรานซิสเตอร์แบบดาร์ลิงตัน เช่น TIA 112 ซึ่งบรรจุอยู่ในตัวถังแบบ TO-220 โดยมีขนาดพิกัดกระแสคอลเล็กเตอร์ถึง 2 แอมป์ แรงดันระหว่างขาคอลเล็กเตอร์กับขาอิมิตเตอร์ในขณะไม่ทำงานมีขนาดถึง 100 โวลต์ และในขณะทำงาน มีขนาด 1 โวลต์ ซึ่งจะเห็นว่ามีขนาดสูงกว่าการใช้ทรานซิสเตอร์ตัวเดียว

จากตัวอย่างที่กล่าวมาทั้งหมดข้างต้นจะเป็นการใช้ทรานซิสเตอร์แบบเอ็นพีเอ็นที่ต้องใช้เอาต์พุตลอจิกสูงมาเป็นตัวควบคุมให้ทรานซิสเตอร์ทำงานแต่ถ้าต้องการใช้เอาต์พุตลอจิกต่ำในการควบคุมก็สามารถทำได้ โดยเปลี่ยนมาใช้ทรานซิสเตอร์แบบพีเอ็นพีแทนดังแสดงในรูปที่ 2.9 (จ) ซึ่งตามวงจรเอาต์พุตลอจิกต่ำจะเป็นตัวจ่ายไบอัสให้ทรานซิสเตอร์ทำงาน และการควบคุมไม่ให้ทรานซิสเตอร์ทำงานก็สามารถทำได้โดยการจ่ายแรงดันเท่ากับแรงดันจากแหล่งจ่ายแรงดันให้กับทรานซิสเตอร์

การควบคุมการจ่ายพลังงานให้ตัวภาระอีกวิธีหนึ่ง อาจทำได้โดยใช้ไอซีไดรฟ์เวอร์ เช่นเบอร์ 7545X ดังแสดงในรูปที่ 2.9 (ฉ) ซึ่งในชิพแต่ละตัวจะประกอบด้วยเกต 2 ตัวที่แยกอิสระจากกันโดยสัญญาณเอาต์พุตจากเกตแต่ละตัวจะเป็นตัวสั่งให้ทรานซิสเตอร์ที่ต่ออยู่กับเกตแต่ละตัวทำงาน

ชิพตระกูล 7545X นั้นจะมีใช้อยู่ 4 รุ่นด้วยกันคือ 75451 เป็นแบบมีแอนด์เกต 2 ตัวอยู่ภายใน, 75452 เป็นแบบมีแนนด์เกต 2 ตัวอยู่ภายใน, 75453 เป็นแบบออร์เกต 2 ตัวอยู่ภายใน, 75454 เป็นแบบนอร์เกต 2 ตัวอยู่ภายใน ซึ่งเอาต์พุตของแต่ละตัวมีกระแสซิงก์ต่ำสุด 300 มิลลิแอมป์ที่แรงดัน 0.7 โวลต์

การใช้มอสเฟต (MOSFET) เป็นสวิตช์

อีกทางเลือกหนึ่งแทนการใช้ไบโพลาร์ทรานซิสเตอร์ คือ การใช้มอสเฟตซึ่งที่นิยมใช้เป็นชนิด N-Channel และนำมาใช้ร่วมกับเกตเพื่อสั่งให้มอสเฟตทำงาน โดยแรงดันที่เป็นบวกที่ได้จากเกตไบอัสให้ความต้านทานระหว่างขาเดรน และซอร์สลดลงจนสามารถนำกระแสได้ดังรูปที่ 2.9 (ช) ยังมีมอสเฟตชนิด P-Channel ที่ปรับปรุงมาจากชนิด N-Channel คล้ายๆ การปรับปรุงใช้ทรานซิสเตอร์พีเอ็นพี แทนชนิด P-Channel นี้จะทำงานได้เมื่อแรงดันที่ขาเกตเป็นลบ มากกว่าที่ขาซอร์ส และการที่จะหยุด หรือไม่ทำงานมอสเฟตนั้น (ไม่ว่าจะเป็นชนิด P หรือ N-Channel) ส่วนใหญ่จะเป็นการจ่ายแรงดันเข้าที่ขาเกต แล้วทำให้มอสเฟตทำงานเป็นแบบสวิตช์เปิดวงจร มากกว่าจะใช้เป็นแรงดันที่ทำให้มอสเฟตทำงานเป็นสวิตช์ปิดวงจร

การใช้มอสเฟตเป็นสวิตช์จะไม่เหมือนกับการใช้ไบโพลาร์ทรานซิสเตอร์เป็นสวิตช์ เพราะการใช้ไบโพลาร์ทรานซิสเตอร์เป็นสวิตช์จะมีกระแสเบสไหลอยู่เล็กน้อย แต่การใช้มอสเฟตที่มีความต้านทานที่ขาเกตสูงมากจะทำให้มีกระแสเกตต่ำมากๆ จนเกือบไม่มีเลยก็ว่าได้ และในเรื่องของแรงดัน หากใช้ไบโพลาร์ทรานซิสเตอร์จะมีแรงดันระหว่างขาเบสกับอิมิตเตอร์ในขณะทำงานเพียง 0.7 โวลต์ แต่การใช้มอสเฟตจะต้องการแรงดันตกคร่อมที่ขาเกตและซอร์สประมาณ 5 หรืออาจถึง 10 โวลต์ ถึงทำให้มอสเฟตทำงาน โดยสมบูรณ์ และวิธีการสร้างแรงดันคร่อมมอสเฟตจากแรงดันลอจิก 5 โวลต์ให้ได้อย่างต่ำ 10 โวลต์นั้น สามารถทำได้ตามลักษณะดังแสดงในรูปที่ 2.9 (ซ) หรืออีกวิธีการหนึ่งที่สามารถใช้ได้ คือ การเลือกใช้อุปกรณ์ที่ใช้แรงดันตกคร่อมตัวมันต่ำกว่านี้ เช่น ZVN4603A ของบริษัท Zeter ที่สามารถทำงานเป็นสวิตช์ได้ในระดับกระแสถึง 1.5 แอมป์แปร์ แต่ใช้แรงดันตกคร่อมเพียงแค่ +5 โวลต์เท่านั้นดังรูปที่ 2.9 (ซ)

เนื่องจากมอสเฟตจะมีความต้านทานต่ำเมื่อทำงานดังนั้นจึงมีแรงดันตกคร่อมขาเดรนและซอร์สต่ำเพียงไม่ถึง 1 โวลต์เท่านั้น โดยค่าความต้านทานของมอสเฟตในขณะทำงาน มีค่าเพียง 0.45 โอห์มที่กระแส 1.5 แอมป์ทำให้มีแรงดันตกคร่อมตัวมันเพียง 0.7 โวลต์ และที่กระแสต่ำกว่านี้ค่าความต้านทาน และแรงดันคร่อมมอสเฟตจะต่ำลง

การใช้โซลิดสเตทรีเลย์ (Solid State Relay) เป็นสวิตช์

อีกวิธีการหนึ่งที่ควบคุมการจ่ายกำลังไฟฟ้าให้ตัวภาระอาจทำได้โดยผ่านทางโวลิดสเตทรีเลย์ ซึ่งเป็นรีเลย์แบบแยกแรงดันไฟฟ้าด้านแรงดันต่ำและแรงดันสูงด้วยแสง (Optical-Isolatate) โดยการทำงานจะใช้แสงเป็นตัวควบคุมการทำงานของมอสเฟต และบรรจุไว้ในตัวถังเดียวกันดังแสดงในรูปที่ 2.10 (ก) ซึ่งการทำงานจะเกิดขึ้นเมื่อจ่ายแรงดันให้กับชุดควบคุมอินพุต จะทำให้มีกระแสไหลผ่านแอลอีดี (เพราะว่าแอลอีดีประกอบอยู่ในตัวถังปิดทำให้เราไม่สามารถมองเห็นเอาต์พุตของแอลอีดีได้) แล้วพลังงานแสงจากแอลอีดีจะเป็นตัวสั่งให้โฟโตไดโอดทำงาน และแรงดันที่ตกคร่อมโฟโตไดโอดจะเป็นแรงดันที่จ่ายให้กับขาเกตของมอสเฟต ทำให้มอสเฟตสามารถทำงานอยู่ในสภาวะทำงานได้ และการที่จะไม่ทำงานสวิตช์ก็สามารถทำได้โดยการตัดแรงดันควบคุมออกมาให้แอลอีดีดับ และสวิตช์ก็ไม่นำกระแสต่อไป

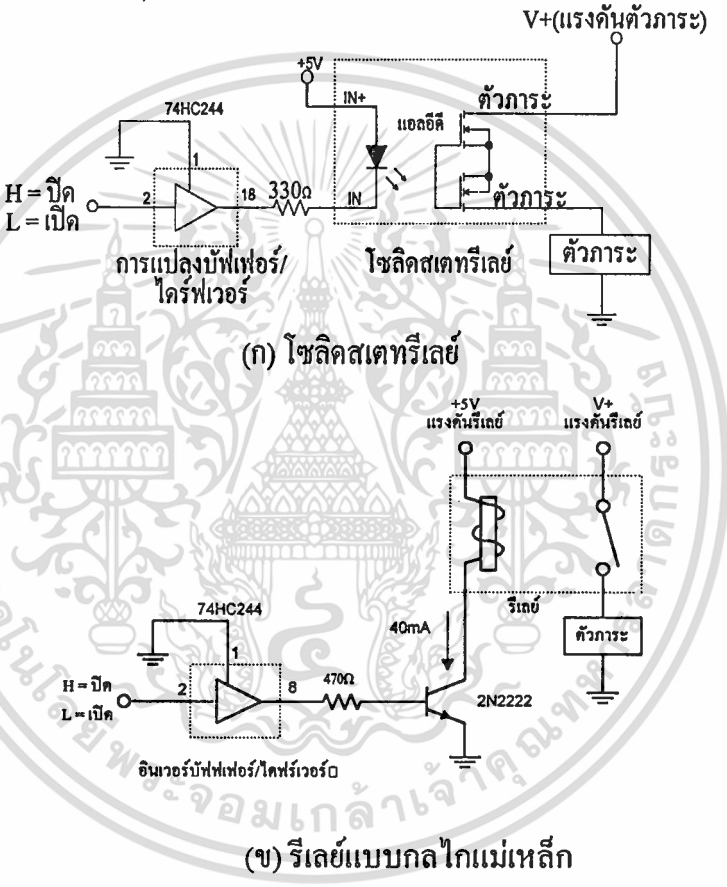
ในปัจจุบันโวลิดสเตทรีเลย์สามารถใช้เป็นสวิตช์จ่ายกำลังไฟฟ้ากระแสสลับได้โดยใช้สัญญาณลอจิกไปควบคุมการปิด-เปิด ส่วนใหญ่จะใช้เอสซีอาร์ หรือ ไตรแอค ซึ่งเป็นอุปกรณ์ที่ไม่มีแรงดันตกคร่อมตัวมัน (Zero-Voltage-Switch) ทำให้เกิดการรบกวนทางไฟฟ้าน้อย เพราะมันจะทำการสวิตชิ่ง เมื่อแรงดันไฟฟ้ากระแสสลับมีค่าเข้าใกล้ศูนย์เท่านั้น

การใช้รีเลย์แบบกลไกแม่เหล็ก

รีเลย์แบบทำงานด้วยกลไกแม่เหล็กเป็นรีเลย์ที่มีขึ้นมานานแล้ว แต่ยังมีใช้อยู่จนถึงปัจจุบันภายในตัวมันจะประกอบด้วยขดลวด และหน้าสัมผัสหนึ่งชุดหรือมากกว่านั้นติดอยู่กับอาร์เมเจอร์ (Armature) ดังแสดงในรูปที่ 2.10 (ข) การทำงานจะเกิดขึ้นเมื่อจ่ายแรงดันให้กับขดลวด ก็จะมีกระแสไหลผ่านขดลวด แล้วทำให้เกิดสนามแม่เหล็กมาบังคับให้อาร์เมเจอร์เคลื่อนที่เป็นผลทำให้หน้าสัมผัสของรีเลย์เปิด หรือปิด และการตัดแรงดันที่จ่ายให้กับขดลวดจะทำให้สนามแม่เหล็กหมดไปทำให้อาร์เมเจอร์ และหน้าสัมผัสกลับมาอยู่ในสภาวะเดิมก่อนที่จะมีกระแสไฟฟ้าไปเลี้ยงขดลวด

ไดโอดที่ต่อคร่อมขดลวดของรีเลย์จะเป็นตัวป้องกันทรานเซียนต์ หรือเรียกว่า สไปค์ (Spike) ที่จะปรากฏที่ขดลวดในขณะที่หน้าสัมผัสเปิดด้วย ซึ่งความจริงแล้วการต่อไดโอด

สามารถต่อคร่อมตัวภาระแบบขดลวดใดๆก็ได้ เช่น ขดลวดของมอเตอร์ แต่ลักษณะการต่อควรให้มีทิศทางดังรูปที่ 2.10 ส่วนถ้าเป็นตัวภาระกระแสสลับจะใช้วาริสเตอร์ต่อคร่อมขดลวดแทนปัญหา หรืออุปสรรคของการใช้รีเลย์แบบกลไกแม่เหล็ก คือ มีขนาดใหญ่ ใช้กระแสจำนวนมากไปเลี้ยงขดลวด (ประมาณ 50-200 มิลลิแอมป์) ความเร็วในการสวิตชิ่งต่ำ และอาจต้องมีการซ่อมแซม หรือเปลี่ยนสัมผัสเมื่อใช้เป็นเวลานาน

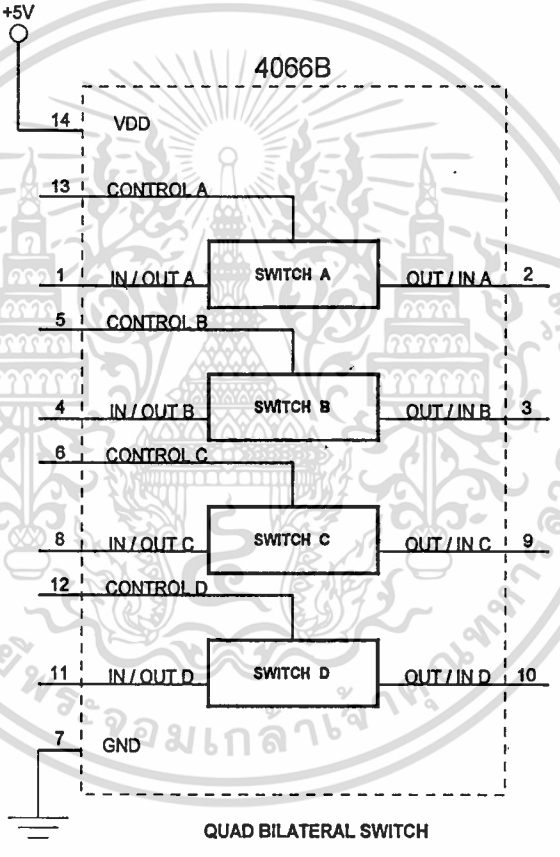


รูปที่2.10 ลักษณะของโซลิดสเตทรีเลย์ และรีเลย์แบบกลไกลแม่เหล็ก

การใช้แอนะลอกสวิตช์ (Analog Switch)

แอนะลอกสวิตช์เป็นสวิตช์ที่แตกต่างที่ได้กล่าวมาแล้ว คือ จะเป็นสวิตช์แบบซีมอสแอนะลอกสวิตช์สามารถใช้เป็นตัวตัดต่อกำลังไฟฟ้าไปยังตัวภาระโดยสามารถนำเอาแอนะลอกสวิตช์ไปควบคุมสัญญาณเสียง หรือสัญญาณภาพได้ ตัวอย่างของสวิตช์แบบที่นิยมใช้บ่อย คือ เบอร์ 4066B สามารถใช้ควบคุมการส่งผ่านสัญญาณความถี่ต่ำๆได้มีลักษณะดังรูปที่ 2.11 ตัว

สวิตช์จะประกอบอยู่ใช้ชิพ โดยมีสัญญาณควบคุม 4 สัญญาณ แต่ละสัญญาณจะควบคุมอินพุต-เอาต์พุต 2 ทาง ซึ่งถ้าสัญญาณควบคุมเป็นลอจิกสูงก็จะทำการต่อสวิตช์แต่ถ้าสัญญาณเป็นลอจิกต่ำก็จะทำการตัดสวิตช์ หรือหยุดต่อวงจรชิพดังกล่าวนี้ใช้ไฟเลี้ยง 5 โวลต์ ซึ่งมีความต้านทานของสวิตช์ในขณะทำงานประมาณ 270 โอห์ม ทำให้มีแรงดันตกคร่อมความต้านทานนี้ในประมาณที่สูงแต่ก็ยังมีบางรุ่นที่มีความต้านทานต่ำกว่านี้ เช่น ชิพเบอร์ 74FC4066 ที่มีความต้านทานขณะทำงานเพียง 100 โอห์ม



รูปที่ 2.11 แอนะลอกสวิตช์เบอร์ 4066

2.5 โปรมแกรมวิชวลเบสิค

2.5.1 ความเป็นมา

วิชวลเบสิคเวอร์ชันแรกที่ออกมานั้นเป็นเวอร์ชันบนวินโดวส์ออกสู่สายตาผู้ใช้เมื่อปี 1991 เป็นเพียงเครื่องมืออย่างง่ายสำหรับการสร้างแอพพลิเคชั่นบนวินโดวส์มากกว่า

องค์ประกอบ หรือ Object ที่ใช้ได้ก็มีแค่องค์ประกอบพื้นฐานของวินโดวส์ เช่น Text Box, List Box เท่านั้น

ในอนาคตคาดว่าวิชวลเบสิคจะมีบทบาทต่อผู้ใช้วินโดวส์มากขึ้น เพราะต่อไปแอพพลิเคชั่นของ Micrcsoft ทุกตัวบนวินโดวส์ จะมีภาษามาโครเดียวกันหมด คือ Visual Basic For Application (VBA) นั่นคือ แทนที่ผู้ใช้จะต้องเรียนรู้ภาษามาโครของ Word For Windows, Excel หรือโปรแกรมอื่นของ Microsoft แต่ละตัวก็จะเป็นการเรียนรู้ VBA เพียงอย่างเดียว ทั้งนี้เนื่องจากความง่ายของภาษาประสิทธิภาพ และความนิยมของผู้ใช้ รวมทั้งยังเป็นการสร้างมาตรฐานของภาษาสำหรับแอพพลิเคชั่นบนวินโดวส์ อีกด้วย ในลักษณะเดียวกันกับภาษา REXX ของ OS/2 นอกจากนี้ยังมีข่าวการพัฒนาวิชวลเบสิคบนแพลตฟอร์มอื่นออกมาเป็นระยะๆ ด้วย ฉะนั้นถือได้ว่าการเรียนรู้วิชวลเบสิค ทำให้เราก้าวไปสู่การใช้แอพพลิเคชั่นสำเร็จรูปอย่างมีประสิทธิภาพขึ้น นอกเหนือจากการพัฒนาแอพพลิเคชั่นขึ้นมาใช้เอง

2.5.2 ภาพรวมของวิชวลเบสิค

วิชวลเบสิค มีสภาพแวดล้อมสำหรับการพัฒนาโปรแกรมบนวินโดวส์ ประกอบด้วยเครื่องมือต่างๆ ครบถ้วนไม่ว่าจะเป็นส่วนของการออกแบบ User Interface, ส่วนออกแบบเมนู(Menu Designer) , การสร้างรายงาน (Report Writer), อิดิเตอร์สำหรับป้อนโปรแกรม และ Debugger เพื่อการตรวจหาข้อผิดพลาดในโปรแกรมองค์ประกอบเหล่านี้นับว่าเอื้ออำนวยต่อการทำงานของโปรมแกรมเมอร์เป็นอย่างมาก

ในด้านตัวภาษาวิชวลเบสิคได้นำไวยากรณ์ของ Basic และ GW-Basic มาใช้โดยสนับสนุนความสามารถเดิมเกือบทั้งหมดนอกจากนี้ยังได้เพิ่มการโปรแกรมแบบมีโครงสร้างของ Quick Basic ซึ่งคล้ายกับในภาษาที่มีโครงสร้าง เช่น Pascal หรือ C เข้าไปด้วย

นอกจากนี้ยังมีการเพิ่มคำสั่ง และฟังก์ชั่นเกี่ยวกับ Object และการเรียกฟังก์ชั่นของระบบปฏิบัติการ (API) เพื่อให้การทำงานกว้างขวางขึ้นรวมทั้งสนับสนุนความสามารถของระบบ เช่น OLD, DDE และการใช้คลิปบอร์ด เป็นต้น

ด้วยความสามารถของภาษาวิชวลเบสิค ผู้ใช้สามารถสร้างแอพพลิเคชั่นได้หลายประเภท ไม่วาจะเป็นโปรแกรมวาดภาพ, การคำนวณทางการเงิน หรือแม้แต่โปรแกรม Cardfile ซึ่งเป็นโปรแกรมมาตรฐานในวินโดวส์ โดยไม่ต้องใช้ชุด SDK เลย

วิชวลเบสิคแต่ละเวอร์ชันจะมี 2 Edition คือ Standard และ Professional Editionซึ่งข้อแตกต่าง คือ ในชุด Professional นั้นจะมี Custom Control (Object ที่สามารถนำมาใช้ในฟอร์ม)

มากกว่า และจะมีเครื่องมืออื่นๆ เช่น Help Complier สำหรับการสร้างข้อความอธิบายการใช้ Setup Kit เพื่อทำส่วนของการติดตั้งแอพพลิเคชั่น (Installing) และ Report Writer พร้อมข้อมูลเพิ่มเติมอื่น สำหรับผู้พัฒนา แต่ในด้านความสามารถของภาษาจะเหมือนกันทั้ง 2 Edition

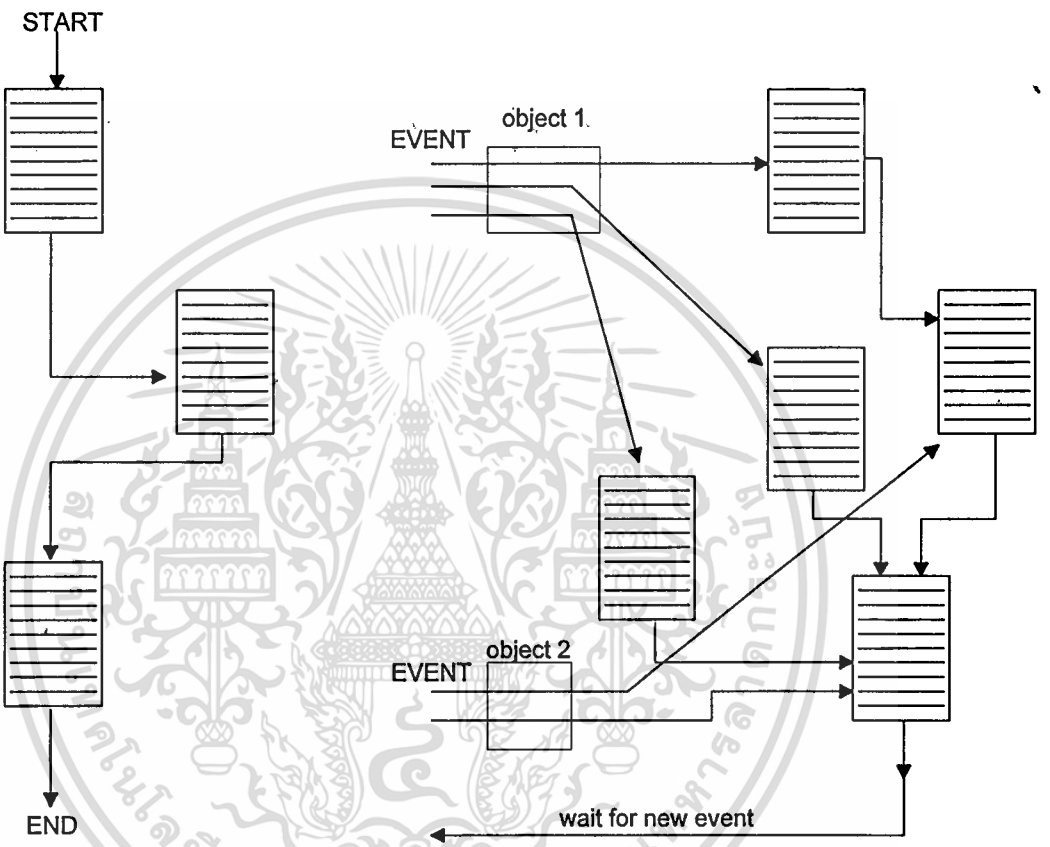
2.5.3 หลักการโปรแกรมเชิงภาพของวิชวลเบสิค

ในวิชวลเบสิคนั้น การพัฒนา และการเขียนโปรแกรมจะเป็นไปในอีกรูปแบบหนึ่ง กล่าวคือ ในการเขียนโปรแกรมแบบเดิมนั้น เราจะต้องมานั่งออกแบบหน้าจอ ระบุตำแหน่งการแสดงผล คิดหาขั้นตอนการทำงาน และอื่นๆ จากนั้นจึงทำการเขียนโปรแกรม โปรแกรมที่จะได้อธิบายและสั่งงานคอมพิวเตอร์เป็นลำดับไป แต่ในวิชวลเบสิค จะใช้หลักของภาพ และการมองเห็นโดยเริ่มจากออกแบบวินโดว์ย่อย หรือที่ในวิชวลเบสิค เรียกว่า *ฟอร์ม* ในฟอร์มจะประกอบด้วยสิ่งต่างๆ ที่เราจะทำงานด้วย หรือเรียกว่าเป็น Object เช่น ข้อความ, ช่องรับ ข้อความ, Scroll Bar หรือปุ่ม เมื่อกำหนดสิ่งเหล่านี้ครบตามต้องการแล้ว จึงระบุว่าองค์ประกอบแต่ละอย่างจะทำงานอย่างไร โดยเขียนโปรแกรมย่อยๆ ปะเข้าไปกับ Object เหล่านี้ ที่ต้องทำแบบนี้ก็เพราะว่าการทำงานในวินโดวส์เป็นแบบที่ เรียกว่า อีเว้นท์-ไดร์ฟเว้น คือ ขึ้นกับเหตุการณ์ (Event) การเขียนโปรแกรมแบบเดิม คือ การสั่งงานตามลำดับจะยุ่งยากมาก หรือบางกรณีอาจทำไม่ได้เลย เพราะอย่าลืมว่าในขณะใดขณะหนึ่งนั้น ในระบบไม่ใช่จะมีเพียงโปรแกรมประยุกต์ของเราเท่านั้นที่ทำงาน วินโดวส์ จะต้องจัดการกับทุกโปรแกรมที่ทำงานในขณะนั้นทั้งหมดไปพร้อมๆ กัน ในขณะที่โปรแกรมแสดงหน้าจอสำหรับรับอินพุต อาจพิมพ์ข้อมูลเข้าไป ใช้เมาส์เลื่อนไปคลิ๊กตรงนั้นตรงนี้ได้โดยอิสระ ทำให้ยากที่จะเขียนโปรแกรมธรรมดาให้คอยดักเส้นทางการทำงาน ในการรับอินพุตว่าจะเกิดอะไรขึ้นตรงไหนได้ จึงต้องใช้รูปแบบการโปรแกรมในลักษณะ อีเว้นท์-ไดร์ฟเว้น ดังกล่าว ซึ่ง Object แต่ละตัวก็จะมีเหตุการณ์เกิดขึ้นได้หลายอย่าง

นอกจาก Object จะมีการตอบสนองต่อเหตุการณ์ต่างๆ ที่กำหนดแล้ว ยังมีอีกเรื่องหนึ่งที่จะขอกล่าวคือ ทุก Object จะมีคุณลักษณะ หรือคุณสมบัติ (Property) ของตัวเอง เช่น ช่องรับข้อความ (Text Box) จะมีชื่อ, ข้อความในนั้น, ความกว้าง, ความสูง, สี โดยเราสามารถอ้างถึง หรือเปลี่ยนคุณสมบัติเหล่านี้ได้ขณะที่โปรแกรมทำงานอยู่ เป็นต้นว่า หากไม่มีการป้อนข้อมูลจะแสดงด้วยสีหนึ่ง หรืออาจไม่ต้องแสดงบนจอภาพเลย

ในการที่จะกระทำสิ่งหนึ่งสิ่งใดกับ Object นั้น จะมีสิ่งที่เรียกว่า Method ซึ่งเปรียบเทียบเป็นกระบวนการทำงานของ Object ซึ่ง Object แต่ละแบบก็อาจจะมี Method ที่แตกต่าง

กันออกไปเช่น ถ้าต้องการสั่งให้เลื่อนตำแหน่งของข้อความ (Label) ก็จะมีกระบวนการ หรือ Method ชื่อ Move ของ Label เพื่อทำงานนี้โดยสั่งว่า *Label.Move = ตำแหน่งที่จะย้ายไป* หรือการสั่งพิมพ์ก็มี Method ชื่อ Print เป็นต้น ถ้าจะพูดไปแล้ว Method นี้ก็คล้ายๆกับคำสั่งที่ใช้ได้กับ Object จะมีคุณลักษณะเฉพาะที่สามารถเปลี่ยนแปลงได้ของตัวเอง



รูปที่ 2.12 การเขียนโปรแกรมแบบธรรมดากับแบบอีเว้นท์-ไดร์ฟเว้น

การทำงานกับ Object ไม่ว่าจะเป็นการอ้างถึงคุณสมบัติ หรือใช้ Method ในบางครั้งเราสามารถจะอ้างถึง Object ได้ (ถ้าหาก Visual Basic เข้าใจได้ว่าจะทำงานกับ Object ใด) เช่น คำสั่ง Print " Hi!" Visual Basic จะเข้าใจเองว่าเป็นการใช้ Method Print กับฟอร์มที่กำลังทำงานอยู่อย่างไรก็ตามเราควรระบุชื่อ Object ทุกครั้งที่ทำงานด้วย ไม่ว่าจะเป็นการอ้างถึงคุณสมบัติหรือใช้ Method เพื่อไม่ให้เกิดความคลุมเครือ

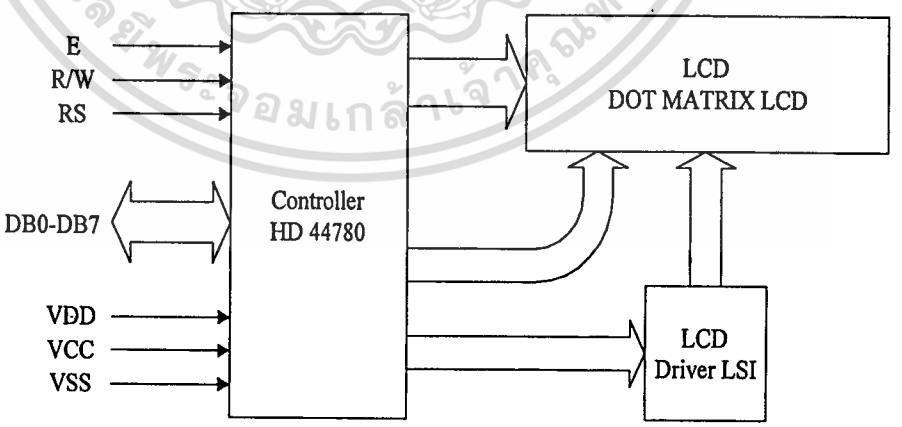
2.6 จอแสดงผลแบบผลึกเหลว (Liquid Crystal Display : LCD)

อุปกรณ์ในปัจจุบันมักมีส่วนแสดงผลเพื่อติดต่อกับผู้ใช้ให้สามารถควบคุม และใช้งานได้สะดวกขึ้น เช่น ใช้บอกสถานะการทำงาน บอกข้อผิดพลาดที่เกิดขึ้น ระหว่างการทำงาน ส่วนแสดงผลอาจจะเป็นไฟบอกสถานะอย่างง่ายๆ หรืออาจเป็นจอแสดงผลแสดงข้อความเป็นตัวอักษรได้ จอแสดงผลแบบหลังมีด้วยกันหลายประเภทขึ้นกับเทคโนโลยีที่ใช้ เช่นใช้ แอลอีดี (Light Emitting Diode : LED) หรือ จอแสดงผลแบบผลึกเหลว (Liquid Crystal Display : LCD) จอแสดงผลประเภทที่กำลังเป็นที่นิยมมากที่สุดในปัจจุบัน ได้แก่แบบที่ใช้จอแสดงผลแบบผลึกเหลวเนื่องจากใช้พลังงานน้อย และมีความละเอียดสูง สามารถแสดงตัวอักษร และรูปภาพได้หลายแบบ บริษัทผู้ผลิตสินค้าส่วนใหญ่จึงนิยมใช้จอแสดงผลแบบนี้เป็นส่วนหนึ่งของผลิตภัณฑ์เพื่อสร้างภาพพจน์ของสินค้า เช่น เครื่องเล่นคอมแพกดิสก์ วิทยุซาวด์อเบาท์ เครื่องเล่นวีดีโอ หรือเลเซอร์ดิสก์ เป็นต้น

ในที่นี้กล่าวถึงการประยุกต์ใช้งานจอแสดงผลแบบจอแสดงผลแบบผลึกเหลวซึ่งสามารถแสดงรายละเอียดได้สูงแต่จะมีความซับซ้อน และใช้งานยากพอสมควร

2.6.1 การประยุกต์ใช้ส่วนของจอแสดงผลแบบผลึกเหลว กับ MCS - 51

ปัจจุบันจอแสดงผลแบบผลึกเหลวที่มีขายในท้องตลาดส่วนใหญ่จะประกอบเป็นโมดูลเพื่อให้สะดวกในการใช้งาน โดยมีส่วนประกอบทั่วไปดังในรูปที่ 2.13



รูปที่ 2.13 โครงสร้างทั่วไปของจอแสดงผลแบบผลึกเหลว

จอแสดงผลแบบผลึกเหลวที่จะกล่าวต่อไปนี้ จะกล่าวถึงเฉพาะคุณสมบัติของจอแสดงผลแบบผลึกเหลวซึ่งจะเรียก ย่อ ๆ ว่า LCM โดยมีส่วนประกอบที่สำคัญดังนี้

1. Dot Matrix LCD : เป็นส่วนที่ทำหน้าที่แสดงผล ซึ่งใช้หลักการหักเหของแสงผ่านผลึกโดยจะประกอบไปด้วยจุด (pixel) จำนวนมากที่สามารถบังคับให้ติดหรือดับได้ทุกจุด

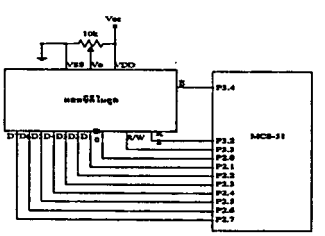
2.Driver:เป็นวงจรที่ใช้ขับจอแสดงผลแบบผลึกเหลวส่วนใหญ่จะใช้ชิพเบอร์ HD44110H

3. Controller : เป็นส่วนที่ใช้ควบคุมการทำงานทั้งหมดของจอแสดงผลแบบผลึกเหลว โดยจะรับข้อมูลจากภายนอกมาจัดการให้จอแสดงผลแบบผลึกเหลว แสดงผลในรูปแบบต่าง ๆ ส่วนใหญ่จะใช้ชิพเบอร์ HD44780 ซึ่งมีใช้งานในแบบคุณสมบัติจอแสดงผลแบบผลึกเหลว

การใช้จอแสดงผลแบบผลึกเหลวผู้ใช้เพียงแต่ศึกษาและทำความเข้าใจส่วนคอนโทรลเลอร์ LCM เท่านั้น เพราะส่วนนี้เป็นส่วนที่รับข้อมูลที่ต้องการแสดงผลจากวงจรภายนอก และควบคุมการทำงานทั้งหมดของ LCM โดยจะกล่าวถึงเฉพาะชิพที่เป็นคอนโทรลเลอร์เบอร์ HD44780 เท่านั้น ส่วนชิพคอนโทรลเลอร์เบอร์อื่นส่วนใหญ่จะมีการใช้งานที่คล้ายกับเบอร์นี้มาก

ชิพคอนโทรลเลอร์เบอร์ HD44780 เป็นชิพของบริษัท HITACHI สามารถต่อใช้งานเพื่อควบคุม LCM กับไมโครคอนโทรลเลอร์หรือไมโครโปรเซสเซอร์ได้ทั้งแบบ 4 bit 2 operation หรือแบบ 8 bit 1 operation ดังนั้นชิพเบอร์นี้สามารถอินเตอร์เฟสกับไมโครโปรเซสเซอร์ได้ทั้งแบบ 4 บิต และ 8 บิต

เนื่องจากไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีข้อมูลขนาด 8 บิต ดังนั้น เราจะกล่าวถึงเฉพาะการติดต่อในแบบ 8 bit 1 operation เท่านั้น ตัวอย่างวงจรการอินเตอร์เฟส MCS-51 กับ LCM มีดังแสดงในรูปที่2.14



รูปที่ 2.14 ตัวอย่างการอินเตอร์เฟส MCS - 51 กับ LCM

จากในรูปจะเห็นว่า LCM ติดต่อกับ MCS - 51 โดย

- ใช้ขา P1.0 - P1.7 เป็นข้อมูล (DB0 - DB7) ในการติดต่อ
- ใช้ขา P3.2 เป็นสัญญาณ RS
- ใช้ขา P3.3 เป็นสัญญาณ R/W
- ใช้ขา P3.4 เป็นสัญญาณ EN (E)

การทำความเข้าใจใช้งาน LCM จำเป็นต้องทราบรายละเอียดดังต่อไปนี้เสียก่อน คือ

1. LCM มีหลายขนาด แต่ทุกขนาดจะมีคำสั่งในการควบคุมเหมือนกัน ต่างกันเพียงขนาดของหน่วยความจำในการแสดงผล หรือ DDRAM (Data Display RAM) เท่านั้น

2. แผนผังเวลาในการติดต่อกับ LCM

3. คำสั่งในการควบคุม LCM

ในช่วงที่สัญญาณ RS เป็น 1 จะมีรายละเอียดของแต่ละสัญญาณมีดังนี้

1. RS : เนื่องจากในชิพคอนโทรลเลอร์มีรีจิสเตอร์อยู่ 2 ประเภท คือ รีจิสเตอร์คำสั่ง (Command Register หรือ Intruction Register) และ รีจิสเตอร์ข้อมูล (Data Register)โดยรีจีส-เตอร์ทั้งสองจะถูกเลือกโดยสัญญาณRS ดังนี้

สัญญาณ RS = 0 หมายถึงเลือกใช้รีจิสเตอร์ข้อมูล

สัญญาณ RS = 1 หมายถึงเลือกใช้รีจิสเตอร์คำสั่ง

2. R/W (Read/Write) เป็นสัญญาณที่ใช้เลือกว่าจะทำการเขียนหรืออ่านข้อมูลจาก LCMโดย

สัญญาณ R/W = 0 หมายถึงต้องการอ่านข้อมูลจาก LCM

สัญญาณ R/W = 1 หมายถึงต้องการเขียนข้อมูลไปยัง LCM

3. E (Enable) มีรายละเอียดดังแสดงในตารางที่ 2.3

ตารางที่ 2.3 การทำงานของสัญญาณ E

RS	R/W	E	OPERATION
0	0	ขอบขาลง	write instruction code
0	1	พัลส์บวก	read busy flag and address counter
1	0	ขอบขาลง	write data
1	1	พัลส์บวก	read data

จากแผนผังเวลาในการตรวจสอบ Busy Flag และจากตารางจะเห็นได้ว่าในการเขียนรหัสคำสั่ง (Intruction Code) ทุกครั้ง

- RS และ RW ต้องมีค่าเป็นลอจิกต่ำ และส่งข้อมูลไปในขณะที่สัญญาณ E เปลี่ยนจากลอจิกสูง เป็นลอจิกต่ำในการเขียนข้อมูลทุกครั้ง

- RS = 1 และ RW = 0 และส่งข้อมูลไปขณะที่สัญญาณ E เปลี่ยนจากลอจิกสูง เป็นลอจิกต่ำ ในการอ่าน Busy Flag และ Address counter ทุกครั้ง

- RS = 0 , RW = 1 และรับข้อมูลเข้ามาในขณะที่สัญญาณ E เป็นลอจิกสูงในการอ่านข้อมูลทุกครั้ง

- RS = 1 , RW = 1 และรับข้อมูลเข้ามาในขณะที่สัญญาณ E เป็นลอจิกสูง

2.6.2 เทคโนโลยีของจอแสดงผลแบบผลึกเหลว

แผงของจอแสดงผลแบบผลึกเหลวจะประกอบด้วยเซกเมนต์แสดงผลขนาดเล็กจำนวนมากในเซกเมนต์จะบรรจุชั้นของเหลวเป็นแผ่นบาง ๆ อยู่ระหว่างชั้นของแก้ว ของเหลวนี้เป็นสารประกอบทำงาน โดยอาศัยพลังงานไฟฟ้ามาควบคุมการทำงาน หรือการแสดงผลของจอแสดงผลแบบผลึกเหลวเกิดขึ้น เนื่องจากการควบคุมแรงดันที่ตกคร่วมตัวมัน เช่น ถ้าให้แรงดันตกคร่อมเซกเมนต์ ก็จะเกิดสีดำหรือทึบแสง แต่ถ้าเอาแรงดันนั้นออกเซกเมนต์นั้นก็จะสว่างหรือโปร่งแสง ด้วยวิธีการจ่ายแรงดัน และงดจ่ายแรงดันนี้ก็จะเพียงพอที่จะควบคุมการแสดงตัวเลข ตัวอักษร และสัญลักษณ์ต่างๆ ได้ และจากสาเหตุที่จอแสดงผลแบบผลึกเหลวใช้แรงดันควบคุม ดังนั้นจึงกินกำลังงานต่ำ และขนาดเล็กบาง

โมดูลของจอแสดงผลแบบผลึกเหลว บางรุ่นอาจจะมี 1 แถว หรือมากกว่า การแสดงผลของจอแสดงผลแบบผลึกเหลวจะอยู่ในรูปเมตริกซ์ เช่น บางรุ่นแสดงเมตริกซ์ที่มีขนาดกว้าง 5 เซกเมนต์ สูง 8 เซกเมนต์ และสำหรับรุ่น HD44780 สามารถควบคุมการแสดงผลได้สูงถึง 11 เซกเมนต์ ซึ่งเป็นผลดีกับการแสดงตัวอักษรบางตัว เช่น g , p และ q

ตัวอักษรจะถูกสร้างโดยการปรับตำแหน่งของแต่ละเซกเมนต์ให้เหมาะสม เช่น ตัวอักษร L จะสร้างจากแนวตั้ง 1 แถว และแนวนอน 1 แถว

ตารางที่ 2.4 แสดงขาสัญญาณต่าง ๆ ที่ใช้ในการเชื่อมต่อกับไมโครคอนโทรลเลอร์ เนื่องจากว่าการควบคุมจอแสดงผลแบบผลึกเหลวต้องการเวลา เพื่อรอทำงานตามคำสั่ง หรือรับสัญญาณ ดังนั้น ถ้าใช้คอมพิวเตอร์จะต้องพิจารณาเรื่องเวลาด้วย แต่การเชื่อมต่อกับไมโคร-

คอนโทรลเลอร์ หรือไมโครโปรเซสเซอร์สามารถต่อโดยตรงได้ไม่ต้องมีอุปกรณ์อื่นมาต่อเพิ่ม หรือถ้ามีก็เพียงเล็กน้อยเท่านั้น

จอแสดงผลแบบผลึกเหลวนั้น มีให้เลือกใช้หลายขนาด แต่ที่นิยมใช้กันมากก็เป็นแบบ 1x16 (1แถว 16 ตัวอักษร) , 2x16 (2 แถว 16 ตัวอักษร) และ 2x20 (2 แถว 20 ตัวอักษร) ส่วนถ้าเป็นจอแสดงผลขนาดใหญ่สามารถแสดงตัวอักษรได้ถึง 80 ตัวอักษรต้องมีวงจรขับหรือชิพคอนโทรลเลอร์เพิ่มขึ้น เพื่อใช้ร่วมกัน HD44780 ที่ต่อสายสัญญาณ 14 เส้นได้

ตารางที่ 2.4 ตำแหน่งขาต่าง ๆ ที่ใช้เชื่อมต่อกับจอแสดงผลแบบผลึกเหลว

ขา	สัญลักษณ์	ความหมาย
1	Vss	กราวด์
2	Vdd	+5 โวลต์
3	Vo	ปรับความสว่างด้วยแรงดัน (0-5 โวลต์)
4	RS	เลือกรีจีสเตอร์(0=รีจีสเตอร์คำสั่งหรือแฟลกแสดงสภาวะการทำงานและตัวนับแอดเดรส ; 1 = รีจีสเตอร์ดาต้า)
5	R/W	เลือกการอ่านหรือเขียน (0= เขียน ; 1= อ่าน)
6	E	อีนาเบิลการอ่านหรือเขียน LCD
7	D0	ดาต้าอินพุต/เอาต์พุตบิตต่ำสุด
8	D1	ดาต้าอินพุต/เอาต์พุตบิตที่ 2
9	D2	ดาต้าอินพุต/เอาต์พุตบิตที่ 3
10	D3	ดาต้าอินพุต/เอาต์พุตบิตที่ 4
11	D4	ดาต้าอินพุต/เอาต์พุตบิตที่ 5
12	D5	ดาต้าอินพุต/เอาต์พุตบิตที่ 6
13	D6	ดาต้าอินพุต/เอาต์พุตบิตที่ 7
14	D7	ดาต้าอินพุต/เอาต์พุตบิตสูงสุด

2.6.3 แหล่งจ่ายไฟเลี้ยงสำหรับจอแสดงผลแบบผลึกเหลว

จอแสดงผลแบบผลึกเหลวจะใช้ไฟเลี้ยง +5 โวลต์ป้อนให้ที่ขา 2 ซึ่งตัวมันกินกระแสเพียงไม่กี่มิลลิแอมป์แปร์ ส่วนขา 3 ต่อเพื่อปรับมุมมองการแสดงผลให้เหมาะสม ทั้งนี้ก็ขึ้นอยู่กับผลของแสงในขณะนั้นด้วยรวมไปถึงตำแหน่งการติดตั้ง และอุณหภูมิ

เมื่อเปรียบเทียบคุณสมบัติระหว่างแอลอีดี กับ จอแสดงผลแบบผลึกเหลว จะพบว่าในที่ที่มีแสงสว่างค่อนข้างสูงแอลอีดีเกือบจะมองไม่เห็น ส่วนจอแสดงผลแบบผลึกเหลวสามารถอ่านในที่ที่มีแสงสว่างได้ เนื่องจากว่าการทำงานของแอลอีดี นั้นจะปล่อยพลังงานแสงออกมา ส่วนจอแสดงผลแบบผลึกเหลวนั่นจะใช้การหักเหแสง โดยใช้แสงส่งผ่านตัวมัน ซึ่งบางสภาวะในที่มีแสงสว่างน้อยก็ไม่สามารถอ่านค่าจอแสดงผลแบบผลึกเหลวได้ วิธีการแก้ก็ คือ การใช้จอแสดงผลแบบผลึกเหลวที่มีแบ็กไลต์จึงเป็นการใช้ฉาก Electrolumine (EL) ซึ่งมีความสามารถในการเรืองแสงได้นำไปติดตั้งไว้ด้านหลัง ทำให้จอแสดงผลแบบผลึกเหลว มีความสว่าง และทำให้เรามองเห็นได้

การที่จะนำสารเรืองแสง EL มาใช้งานนั้น ที่ชุดโมดูลจอแสดงผลแบบผลึกเหลวต้องมีแผง EL และชุดแปลงแรงดันเป็นสัญญาณไฟสลับแรงดันสูง ซึ่งจะเป็นอุปกรณ์แรงดันไฟตรง 5 โวลต์ ถึง 100 โวลต์ที่ความถี่ 400 เฮิรตซ์ อุปกรณ์แปลงแรงดันที่ต้องใช้กระแสหลายมิลลิแอมป์แปร์ในการทำงาน จึงทำให้เป็นข้อเสียเปรียบของอุปกรณ์ตัวนี้

โมดูลของจอแสดงผลแบบผลึกเหลว แบ่งออกเป็นแบบสะท้อนกลับแบบนี้จะไม่ใช้แหล่งกำเนิดแสงทางด้านหลัง ส่วนอีกแบบหนึ่ง คือ แบบส่งผ่านแบบนี้จะใช้แหล่งกำเนิดแสงด้านหลัง หรือไม่ใช้ก็ได้ โดยสามารถต่อสวิตช์เข้ากับแหล่งกำเนิดแสง เวลาจะใช้แหล่งกำเนิดแสงเปิด หรือถ้าไม่ต้องการใช้ก็ปิดตามต้องการ

2.6.4 คอนโทรลเลอร์และการควบคุม

การที่จะใช้โมดูลของจอแสดงผลแบบผลึกเหลว ในงานหนึ่งงานใดนั้น จะต้องทำความเข้าใจกับตัวควบคุมก่อน HD44780 เป็นตัวควบคุมขนาดเล็กที่คล้ายกับคอมพิวเตอร์โดยจะทำงานทั้งหมด 11 คำสั่ง เพื่อควบคุมการทำงานต่าง ๆ เช่น เคลียร์หน้าจอแสดงผล, เขียนตัวอักษร, เลือกตำแหน่ง และอ่านข้อมูลจากจอแสดงผล

หน่วยความจำภายใน HD44780 มี 2 ชนิด คือ Character Generator (CG) ROM และ Character-Generator (CG) RAM

CGROM ใช้สำหรับเก็บตัวอักษรเกือบ 200 รูปแบบ เช่น ตัวอักษรภาษาอังกฤษ, ตัวเลขเครื่องหมายทางคณิตศาสตร์, สัญลักษณ์พิเศษ และอักษรญี่ปุ่น ซึ่งจะกำหนดลงในรอมไว้แล้วไม่สามารถแก้ไข หรือเปลี่ยนแปลงได้

CGRAM ใช้เก็บตัวอักษรที่ผู้ใช้สามารถออกแบบขึ้นเองได้ เช่น โลโก้, สัญลักษณ์พิเศษ อักษรกราฟิคง่าย ๆ ที่สามารถออกแบบบนเมตริกซ์ขนาด 5x8 ได้ จะเขียนขึ้นนี้จะเขียนครั้งละ 5 บิต หลาย ๆ คำ แต่ละคำจะแทนรูปแบบเซกเมนซ์ 1 แถว แล้วเก็บไว้ใน CGRAM รูปแบบอักษรนี้จะหายไปเมื่อปิดเครื่อง และเมื่อจะใช้งานต้องเรียกข้อมูลมาใหม่ หลังจากเปิดเครื่อง

อักษรใน CGROM และ CGRAM เป็นอักษรขนาด 8 บิต(0 ถึง FFH) ซึ่งบางตำแหน่งก็ไม่ได้ใช้ ตำแหน่งแอดเดรสที่ใช้กันมากจากช่วง 21H ถึง 7DH ซึ่งจะตรงกับตำแหน่งรหัสแอสกี้บนคอมพิวเตอร์ เช่น “A” จะถูกเก็บไว้ที่ตำแหน่ง 41H และ “B” จะเก็บไว้ที่ตำแหน่ง 42H เป็นต้น โดยตำแหน่งแอดเดรสจะถูกเก็บอยู่ในเลขฐาน 16

ไอซี HD44780 มีรีจิสเตอร์ 2 ตัวคือ รีจิสเตอร์คำสั่ง ซึ่งใช้สำหรับเก็บรหัสของคำสั่ง และ รีจิสเตอร์ข้อมูล ซึ่งใช้สำหรับเก็บรหัสตัวอักษรเมื่อต้องการเขียน หรืออ่านข้อมูลจากไอซี จะต้องเลือกใช้รีจิสเตอร์ให้เหมาะสม เพื่อนำไปต่อกับขา 4 ของจอแสดงผลแบบผลึกเหลวตามหน้าที่การทำงานที่เราต้องการ

หน่วยความจำแรมเก็บข้อมูลแสดงผล Display Data (DD) RAM จะเก็บรหัสตัวอักษรขนาด 8 บิตได้มากกว่า 8 ตัวอักษร ไว้ในตำแหน่งแต่ละตำแหน่ง โดยรหัสของตัวอักษรที่เก็บไว้ใน DDRAM จะกำหนดว่าให้แสดงอักษรที่ตำแหน่งไหน

ในกรณีที่จอแสดงผลแบบผลึกเหลว มี 2 แถว (ถ้าต้องการลบแถวแรก) การทำงานของมันจะทำการเลื่อนตำแหน่งซ้ายสุดของแถวบน ซึ่งเป็นตำแหน่งสูงก่อน และตำแหน่งต่อไปก็จะเลื่อนตามมา ตามลำดับ จนกว่าจะหมดแถวที่ 1 แล้วแถวที่ 2 จะตามมา เช่น สมมุติว่าแถวที่ 1 มีตำแหน่งเริ่มต้นที่ 0-39H พอหมดแถวที่ 1 แล้วแถวที่ 2 จะถูกลบต่อไปเป็น 40H, 41H, 42H, 43H และต่อๆ ไปจนครบ

แต่ละครั้งที่ทำการเขียนอักษรลงในตำแหน่งของ DDRAM แต่ละตำแหน่งจะเพิ่มตำแหน่งขึ้นโดยอัตโนมัติตามลำดับของการเขียนตัวอักษร ยกเว้นกรณีที่ผู้ใช้กำหนดตำแหน่งที่จะเขียนข้อมูลเองในบางครั้ง

อย่างไรก็ตามเนื่องจากว่าแถวที่สอง เริ่มที่ตำแหน่งแอดเดรส 40H ดังนั้นถ้าหากผู้ใช้ต้องการให้แสดงที่แถวที่สอง จะต้องอ้างตำแหน่งให้ถูกต้องด้วย เช่น ถ้าจอแสดงผลมี 16

ตำแหน่ง แถวที่ 1 จะสิ้นสุดที่ตำแหน่ง 0F H และแถวที่ 2 จะเริ่มต้นที่ 40 H ดังนั้นถ้าต้องการย้ายอักษรจากตำแหน่งขวาสุดของแถวที่ 1 ไปยังตำแหน่งซ้ายสุดของแถวที่ 2 ผู้ใช้จะต้องอ้างตำแหน่ง ไปที่ 40 H

นอกจากนี้จอแสดงผลบางรุ่น จะมีแถวแสดงอักษรเป็นแบบ ฟิสิคอล 1 แถว และแบบลอจิคอลอีก 2 แถว ในกรณีที่เป็นจอแสดงผลแบบ 2 แถว ในกรณีที่เป็น 16 ตัวอักษร 8 ตัว อักษรแรกจะเริ่มที่แอดเดรส 0 ถึง 7 และ 8 ตัวอักษรหลังจะเริ่มที่แอดเดรส 40H ถึง 47H ดังนั้นถ้าจะเขียนข้อมูลไปที่ตำแหน่ง 8 ตัวหลังต้องกำหนดตำแหน่งไปที่ 40H

อย่างไรก็ตาม ในจอแสดงผลขนาดเล็กจะไม่มี DDRAM เนื่องจากว่า DDRAM เป็น RAM ที่ใช้สำรองไว้ให้ผู้ใช้เลือกใช้ตามวัตถุประสงค์

2.6.5 การอ่านและการเขียนข้อมูล

ในการอ่านและเขียนข้อมูลกับจอแสดงผลแบบผลึกเหลว ขั้นตอนของการเขียนเริ่มจากมีสัญญาณ RS เข้ามา และให้สัญญาณ R/W มีสถานะเป็นลอจิกต่ำ หลังจากนั้น ประมาณ 140 นาโนวินาที สัญญาณอีนาเบิลจะมีสถานะเป็นลอจิกสูง สถานะอยู่อย่างน้อย 450 นาโนวินาที เพื่อที่จะให้ขาที่ดาต้า D0-D7 ส่งข้อมูลอย่างน้อย 195 นาโนวินาที ก่อนที่สัญญาณอีนา-เบิลจะเป็นลอจิกต่ำอีกครั้ง

ส่วนขั้นตอนในการอ่านข้อมูลจะคล้ายกับการเขียนแต่สัญญาณ R/W จะเป็นลอจิกสูง ส่วนสัญญาณข้อมูล D0-D7 จะทำงานหลังจากสัญญาณอีนาเบิลเป็นลอจิกสูง แล้วประมาณ 320 นาโนวินาที

ไอซี HD44780 จะไม่ทำคำสั่งใหม่ที่เข้ามาจนกว่าจะทำคำสั่งที่ทำงานอยู่ขณะนั้นเสร็จก่อน ซึ่งในกรอบแยกที่ 1 จะแสดงเวลาที่มากที่สุดที่แต่ละคำสั่งใช้ในการประมวลผล แต่ถ้าใช้ภาษาเบสิก หรือภาษาระดับสูงในการโปรแกรมค่าเวลาเหล่านี้อาจไม่ต้องใส่ใจกับมันมากนัก เพราะว่าตัวโปรแกรมจะเข้าถึงคำสั่งโดยอัตโนมัติอยู่แล้ว ถ้าหากต้องการใช้ชุดโมดูลจอแสดงผลแบบผลึกเหลวรับคำสั่งต่อมาอาจทำได้โดยเขียนโปรแกรมหน่วงเวลาหลังจากทำคำสั่งเหล่านั้น หรืออาจจะอ่านแฟล็กว่าง (บิต 7 ก็ได้)

โดยส่วนใหญ่จะใช้การเชื่อมต่อข้อมูลขนาด 8 บิต (D0-D7) แต่ถ้าต้องการจะใช้แค่ข้อมูลขนาด 4 บิตเท่านั้นก็ทำได้ ซึ่งบางครั้งถ้าเรามีไมโครคอนโทรลเลอร์ขนาด 8 บิต เราสามารถที่จะต่อใช้ข้อมูลเพียง 4 บิตก็ได้เพื่อเป็นการประหยัดฮาร์ดแวร์ การใช้ข้อมูลเพียง 4บิตนี้จะใช้สัญญาณในการส่งข้อมูลทั้งหมดเพียง 7 เส้น (D4-D7 , RS ,R/W และอีนาเบิล) หรืออาจ

จะใช้เพียง 6 เส้น ถ้าให้สัญญาณ R/W เป็นลอจิกต่ำตลอด ก็จะสามารถอ่านข้อมูล D0-D7 แสดงบนจอแสดงผลได้

อย่างไรก็ตามการที่จะส่งคำสั่งเพื่อใช้กับข้อมูลขนาด 8 บิตนั้น คุณจะต้องส่งข้อมูลบิต D4-D7 ไปพร้อมกับสัญญาณ RS และ R/W โดยไม่ต้องใช้บิต D0-D3

ตัวอย่างการเขียนตัวอักษร "Z" (SAH) ให้ออกจอแสดงผลกรณีที่ใช้บิตข้อมูลขนาด 8 บิตจะใช้คำสั่งดังนี้ คือ

RA	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	1	1	0	1	0

ส่วนกรณีที่ใช้แบบ 4 บิต จะใช้คำสั่งเหมือนกันแต่จะต้องป้อนออกไป 2 ครั้งดังนี้ คือ

RS	R/W	D7	D6	D5	D4
1	0	0	1	0	1
1	0	1	0	1	0

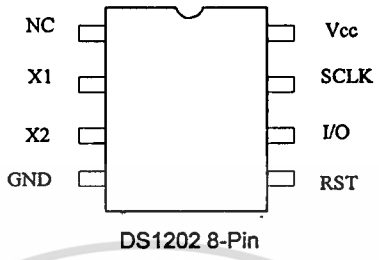
การใช้งานแบบ 4 บิตทำงานช้าและโปรแกรมยาวกว่าแบบ 8 บิต แต่ข้อ คือ การใช้การต่อวงจร หรือขาสัญญาณน้อยกว่าแบบ 8 บิต

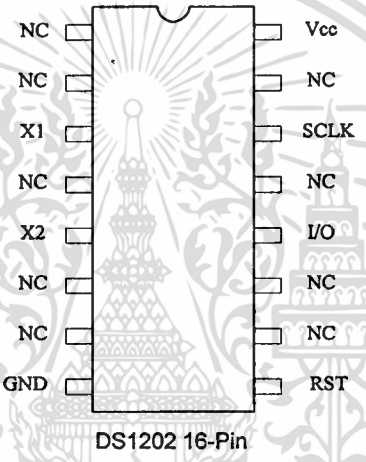
2.7 วิธีสร้างฐานเวลาจริงให้แก่ MCS-51

ส่วนใหญ่จะใช้ชิพทำหน้าที่เป็นนาฬิกาที่สามารถส่งข้อมูลเวลาในขณะใดๆให้แก่ไมโครคอนโทรลเลอร์ได้ เวลาที่ได้จากวิธีนี้จะเป็นเวลาจริงๆ ที่เดินอยู่ตลอดเวลาอย่างเที่ยงตรงชิพที่ทำหน้าที่ดังกล่าวนี้ปัจจุบันมีอยู่ด้วยกันหลายเบอร์ แต่เบอร์ที่ใช้งานได้ง่ายและสะดวกที่สุดคือชิพ RTC ของบริษัท Dallas Semiconductor เบอร์ DS1202 "Serial Time keeper Chip" ซึ่งจะได้กล่าวต่อไป

ชิพ RTC จริงๆ มีอยู่ด้วยกันหลายชนิด บางเบอร์สามารถอินเตอร์รัพต์ ไมโครคอนโทรลเลอร์ภายในช่วงเวลามีกำหนดได้ ชิพ RTC เบอร์ที่เราจะศึกษาต่อไปนี้ทำได้เพียงแค่ให้ข้อมูลที่เป็นเวลาในขณะใดๆ แก่ไมโครคอนโทรลเลอร์เท่านั้น ไม่สามารถอินเตอร์รัพต์

ซีพียูได้ การเลือกใช้งานชิพ RTC ประเภทใดขึ้นอยู่กับผู้ออกแบบว่าต้องการความสามารถมากน้อยเพียงใด





รูปที่ 2.15 ลักษณะรูปร่างทั้งสองแบบของไอซีเบอร์ DS1202

ชิพ RTC เบอร์ DS1202 ที่จะได้กล่าวถึงนี้มีความเที่ยงตรงในการทำงานสูงมาก สามารถนำมาต่อร่วมกับระบบ เพื่อบอกเวลาให้แก่ไมโครคอนโทรลเลอร์ได้สะดวก เพราะใช้จำนวนสายในการติดต่อระหว่างตัวชิพเองกับไมโครคอนโทรลเลอร์เพียง 3 เส้น เท่านั้น เนื่องจากชิพ RTC เบอร์นี้ใช้ในการติดต่อรับส่งข้อมูลแบบอนุกรม คุณสมบัติของชิพ RTC เบอร์ DS1202 มีดังนี้

1. ทำหน้าที่นับวินาที นาที ชั่วโมง วันที่ของเดือน เดือน ปี รวมทั้งคำนวณปีอธิกสุรทินให้เองโดยอัตโนมัติ

2. มีหน่วยความจำขนาด 24 ไบต์สำหรับเก็บข้อมูลทั่วๆไป ส่วนใหญ่ไว้เก็บข้อมูลที่ต้อง การสำรองในขณะที่ไม่มีพลังงานจ่ายให้แก่ระบบ เช่น รหัสผ่านที่เปลี่ยนค่าได้ เวลาที่ต้องการให้เครื่องจักรทำงาน ทำให้ไม่จำเป็นต้องสำรองหน่วยความจำทั้งระบบนั้นเอง

3. ใช้การติดต่อแบบอนุกรม จึงใช้จำนวนสายในการเชื่อมต่อกับระบบเพียง 3 เส้นเท่านั้น

4. ใช้แรงดันไฟฟ้าเพียง 2.0 ถึง 5.5 โวลต์ และใช้กระแสเพียง 300 นาโนแอมป์ที่ระดับแรงดัน 2.0 โวลต์

5. การโอนย้ายข้อมูลสามารถกระทำได้ทั้งในแบบครั้งละ 1 ไบต์ (Single Byte) หรือครั้งละหลายๆ ไบต์ (Multiple byte หรือ Burst mode) ไม่ว่าจะเป็นการเขียน หรืออ่านข้อมูล

6. ตัวชิพเองมีให้เลือกทั้งแบบ 8 PIN DIP หรือ 16 PIN SOIC เพื่อใช้สำหรับแผ่นวงจรชนิด Surface Mount

7. ใช้ได้กับระดับสัญญาณทีทีแอล (Vcc = 5 โวลต์)

8. ช่วงอุณหภูมิในการใช้งานกว้างมาก ระหว่าง -40 องศาเซลเซียส ± 88 องศาเซลเซียส

2.7.1 รายละเอียด ชิพ RTC เบอร์ DS1202

ชิพ RTC เบอร์ DS1202 มี Real Time Clock/Calender และ Static RAM ขนาด 24 ไบต์ ใช้สายเพียง3 เส้นในการเชื่อมต่อกับไมโครคอนโทรลเลอร์เพื่อรับส่งข้อมูลเกี่ยวกับเวลา ข้อมูลที่ชิพ RTC DS1202 มีให้ประกอบด้วย วินาที , นาที , ชั่วโมง , วันที่ , เดือน , ปี

วันที่ในวันสุดท้ายของเดือนจะถูกปรับโดยอัตโนมัติสำหรับเดือนที่มีจำนวนวันน้อยกว่า 31 วัน และมีการคำนวณจำนวนวันของเดือนกุมภาพันธ์ในปีอธิกสุรทินให้เอง ข้อมูลที่ส่งให้แก่ไมโครคอนโทรลเลอร์สามารถเลือกรูปแบบได้ทั้งแบบ 24 ชั่วโมง (0.00-23.59 นาฬิกา) หรือแบบ 12 ชั่วโมง (0.00-12.00 นาฬิกา โดยมีข้อมูลเพิ่ม เพื่อบอกให้ทราบว่าเป็นเวลาในช่วงกลางวันหรือกลางคืน)

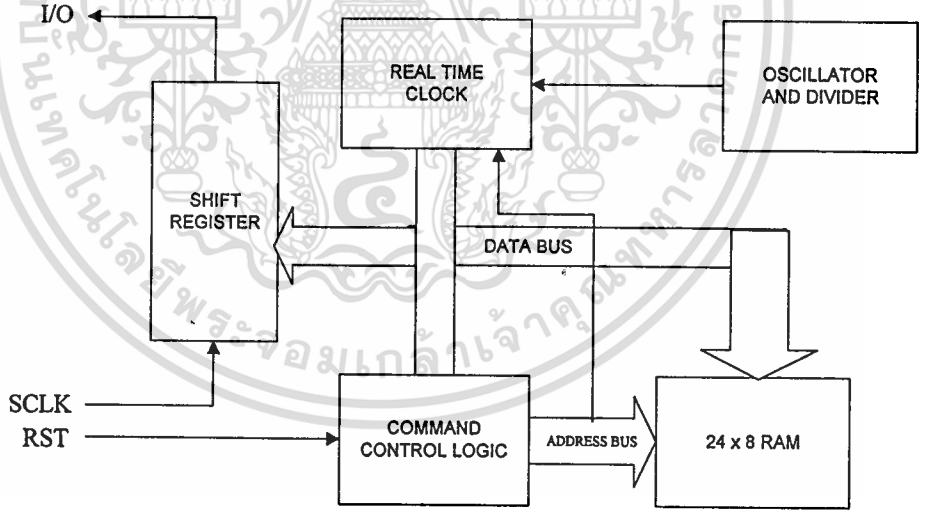
การเชื่อมต่อชิพ RTC DS1202 ใช้การติดต่อแบบอนุกรมชนิด Synchoronous Serial Communication ขาที่ต้องการใช้ในการรับส่งข้อมูลทั้งสาม คือ

- RTC (reset)
- I/O (data line)
- SCLK (serial clock)

เนื่องจากในชิพ RTC DS1202 มีนาฬิกาหรือเวลาที่เดินอยู่ตลอดเวลา รวมทั้งมีหน่วยความจำจำนวนหนึ่ง ดังนั้นในการติดต่อกับชิพ RTC DS1202 ผู้ใช้สามารถเลือกได้ว่าต้องการข้อมูลจากนาฬิกา หรือจากหน่วยความจำภายในชิพ การรับส่งข้อมูลสามารถกระทำได้ทั้งแบบ

ทีละไบต์ หรือรับส่งกันคราวละหลายไบต์ดังจะได้กล่าวต่อไป นอกจากนี้ RTC DS1202 ยังถูกออกแบบให้ใช้พลังงานน้อยมาก และสิ้นเปลืองพลังงานจากแบตเตอรี่น้อยที่สุด เพื่อความสะดวกในการสำรองพลังงาน โดยชิพตัวนี้สามารถเก็บรักษาข้อมูลในหน่วยความจำ และเวลาที่เดินอยู่ตลอดเวลาได้ที่กำลังไฟฟ้าน้อยกว่า 1 ไมโครวัตต์

โครงสร้างภายในของ RTC DS 1202 ประกอบไปด้วยส่วนที่สำคัญๆ ดังแสดงในรูปที่2.16 ในการรับหรือส่งข้อมูลใดๆ ให้แก่ RTC DS1202 ไมโครคอนโทรลเลอร์หรือไมโครโปรเซสเซอร์ที่ต้องการติดต่อด้วยจะต้องส่งข้อมูลที่เป็นคำสั่งควบคุมการติดต่อ ซึ่งมีขนาด 8 บิตเสียก่อน โดยเริ่มต้นด้วยการให้ขา RST มีสถานะเป็นลอจิกสูง (อยู่ในช่วงการติดต่อ) จากนั้นส่งข้อมูลจำนวน 8 บิตเข้าไปไว้ในชิฟต์รีจิสเตอร์ของ RTC ข้อมูลขนาด 8 บิตจะประกอบด้วยคำสั่งในการควบคุมชิพ RTC และตำแหน่งของหน่วยความจำที่ต้องการติดต่อ (address/command byte) ในแต่ละครั้ง การรับข้อมูลจากไมโครโปรเซสเซอร์แต่ละบิตจะกระทำที่ช่วงขอบขาขึ้นของสัญญาณที่ขา SCLK (serial clock)



รูปที่ 2.16 โครงสร้างภายในของ RTC DS1202

ภายในชิพ RTC เบอร์ DS1202 ประกอบด้วยหน่วยความจำขนาด 24 ไบต์ และรีจิสเตอร์ที่ทำหน้าที่เก็บเวลาของชิพในขณะปัจจุบันจำนวน 8 ตัว รีจิสเตอร์ทั้ง 8 ตัวนี้สามารถเข้าถึงได้เสมือนเป็นหน่วยความจำตำแหน่งหนึ่ง ดังนั้นต่อไปเราจะมองว่าชิพ RTC เบอร์นี้มี

หน่วยความจำรวมทั้งสิ้น 32 ตำแหน่ง โดยประกอบขึ้นจากรีจิสเตอร์ 8 ตำแหน่ง และหน่วยความจำ 24 ตำแหน่ง

ข้อมูลขนาด 8 บิตแรกจะระบุตำแหน่งหน่วยความจำที่ต้องการติดต่อ (ทั้งตำแหน่งของหน่วยความจำทั่วไป และตำแหน่งของรีจิสเตอร์สำหรับเก็บเวลา) และบอกว่าเป็นการเขียนหรืออ่านข้อมูลจากหน่วยความจำตำแหน่งนั้นๆ รวมทั้งระบุว่าให้การรับส่งข้อมูลเป็นแบบครั้งละ 1 ไบต์ หรือครั้งละหลายๆ ไบต์

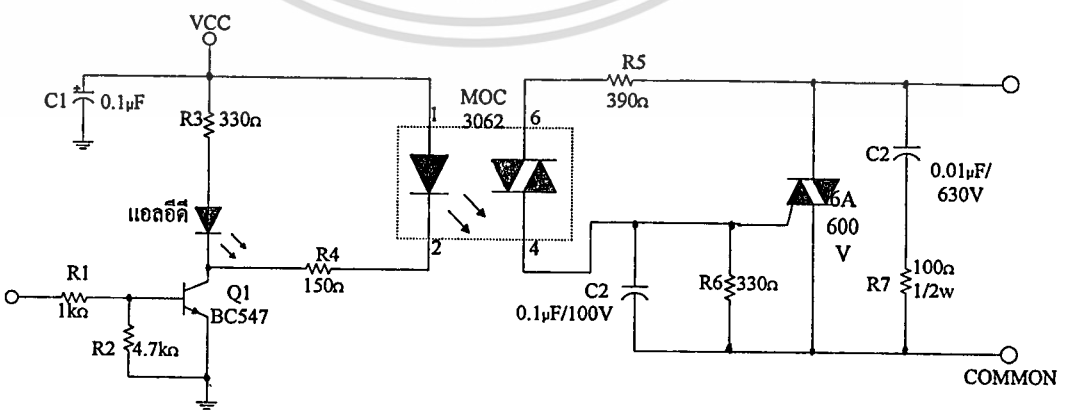
หลังจากมีสัญญาณนาฬิกาเกิดขึ้น 8 ครั้ง (สัญญาณ SCLK)ในระหว่างการเขียนข้อมูล 8 บิตแรกเข้าไปในชิฟต์รีจิสเตอร์สัญญาณนาฬิกาต่อไปที่จะเกิดขึ้นจะเป็นการนำข้อมูลออกจากชิพ RTC สำหรับการอ่านข้อมูล หรือนำข้อมูลเข้าไปยังชิพ RTC สำหรับการเขียนข้อมูล จำนวนของสัญญาณนาฬิกาทั้งหมดที่เกิดขึ้นในการติดต่อครั้งหนึ่งๆ จึงเท่ากับ 8+8 ในการส่งข้อมูลครั้งละ 1 ไบต์ หรือมากที่สุด 192 (8x24) สำหรับการส่งข้อมูลครั้งละหลายๆ ไบต์

บทที่ 3
การออกแบบ การสร้าง และการทำงาน

ในบทนี้จะกล่าวถึงการออกแบบ,การสร้าง และ การทำงาน ของโครงงานโดยสามารถแบ่งออกเป็น 2 ส่วน แยกเป็นส่วนฮาร์ดแวร์ และซอฟต์แวร์ ส่วนของฮาร์ดแวร์จะประกอบด้วยวงจรทั้งหมดของโครงงานซึ่งจะกล่าวโดยละเอียดต่อไป และส่วนของซอฟต์แวร์จะมี 2 ส่วน คือ ซอฟต์แวร์ของโปรแกรมวิชวลเบสิค และซอฟต์แวร์ของไมโครคอนโทรลเลอร์ MCS-51

3.1 วงจรโซลิดสเตทรีเลย์

จากรูปที่ 3.1 เป็นวงจรที่ใช้ในการควบคุมการเปิด และปิดของอุปกรณ์ไฟฟ้า โดยมีอุปกรณ์โซลิดสเตทรีเลย์เป็นหัวใจสำคัญในการทำงาน ที่อินพุตของวงจรจะรับค่าลอจิก "0" หรือ "1" มาจากภาคคอนโทรลเลอร์ ซึ่งลอจิก "0" หมายถึง การปิดอุปกรณ์ และลอจิก "1" หมายถึง การเปิดอุปกรณ์ จากวงจรประกอบด้วย 2 ส่วน คือ ส่วนของวงจรทรานซิสเตอร์สวิตช์ที่ใช้ควบคุมการทำงานของโซลิดสเตทรีเลย์ และส่วนของวงจรที่ใช้ในการควบคุมอุปกรณ์ไฟฟ้า ซึ่งใช้ไตรแอคในการควบคุม โดยทั้ง 2 ส่วน มีโซลิดสเตทรีเลย์เป็นตัวเชื่อมต่อการทำงาน



รูปที่ 3.1 วงจรโซลิดสเตทรีเลย์

วงจรทรานซิสเตอร์สวิตช์ที่ใช้ควบคุมโซลิดสเตทรีเลย์ ประกอบด้วย R_1 , R_2 , R_3 และ Q_1 โดยการทำงานใน 2 ลักษณะ คือ ถ้าอินพุตเป็นลอจิก "1" แล้ว Q_1 จะทำงานอยู่ในสภาวะอิ่มตัวทำให้ที่ขาคอลเลคเตอร์ของ Q_1 เปรียบเสมือนกราวด์ จึงส่งผลทำให้โซลิดสเตทรีเลย์ ซึ่งขาหนึ่งต่ออยู่กับแหล่งจ่ายแรงดัน และขา 2 ต่อกับ R4 ผ่านมายังขาคอลเลคเตอร์ของ Q_1 ซึ่งขณะนี้เป็นกราวด์ จึงทำให้โซลิดสเตทรีเลย์มีแรงดันไบอัสที่ถูกต้อง แล้วโซลิดสเตทรีเลย์ แล้วอยู่ในสภาวะสวิตช์ปิด จึงจะทำให้ส่วนเอาต์พุตทำงาน แต่ถ้าอินพุตเป็นลอจิกต่ำแล้ว Q_1 จะทำงานอยู่ในสภาวะไม่ทำงานก็จะทำให้ทำงานกลับกับภาวะอิ่มตัว โดยที่ขาคอลเลคเตอร์จะมีแรงดันสูงจนเปรียบเสมือนเป็นแหล่งจ่ายแรงดัน ทำให้โซลิดสเตทรีเลย์อยู่ในสภาวะสวิตช์ปิด ซึ่งจะทำให้ส่วนเอาต์พุตไม่ทำงานตามไปด้วย และในส่วนของวงจรทรานซิสเตอร์สวิตช์นี้จะมีแอลอีดี เพื่อแสดงว่าวงจรอยู่ในสภาวะเปิด หรือปิดอีกด้วย

ส่วนวงจรเอาต์พุตที่ใช้ในการควบคุมไฟฟ้าจะเป็นวงจรที่ใช้ไตรแอคในการเปิด และปิดอุปกรณ์ไฟฟ้า โดยไตรแอคเป็นอุปกรณ์ที่สามารถใช้กับแรงดันไฟสลับได้ และการควบคุมการทำงานของไตรแอคจะทำได้โดยควบคุมแรงดันที่ขาเกตของไตรแอค ซึ่งก็จะมาจากส่วนเอาต์พุตของโซลิดสเตทรีเลย์ที่ขา 4 และ 6 โดยโครงสร้างภายในของโซลิดสเตทรีเลย์ที่ขา 4 และขา 6 จะเป็นไตรแอค ซึ่งสามารถใช้กับแรงดันไฟฟ้าสลับได้เช่นกัน

ในขณะที่โซลิดสเตทรีเลย์อยู่ในสภาวะไม่ทำงานจะไม่มีกระแสไหลผ่านส่วนของวงจรนี้ เนื่องจากไตรแอค ในส่วนของโซลิดสเตทรีเลย์ไม่นำกระแส จึงทำให้แรงดันที่ขาเกตของไตรแอคเป็น 0 ก็จะทำให้ไตรแอคไม่นำกระแสเช่นกัน ดังนั้นถ้านำอุปกรณ์ไฟฟ้านั้นๆ และถ้าโซลิดสเตทรีเลย์อยู่ในสภาวะทำงานก็จะมีกระแสไหลผ่านไตรแอคภายในโซลิดสเตท-รีเลย์ จึงทำให้มีแรงดันตกคร่อมที่ R_6 ซึ่งต่ออยู่กับขาเกตของไตรแอค จึงทำให้ไตรแอคนำกระแส ดังนั้นถ้านำอุปกรณ์ไฟฟ้ามาต่อก็จะเป็นการเปิดอุปกรณ์ไฟฟ้านั้นๆ

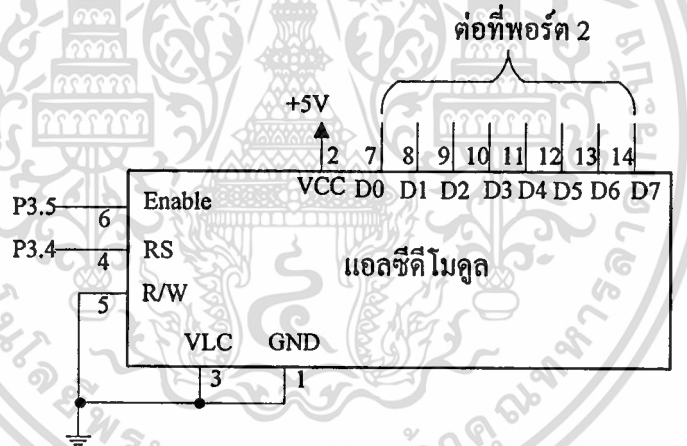
ที่ส่วนของเอาต์พุตจะมี R_1 และ C_2 ซึ่งทำหน้าที่ในลักษณะค่าคงตัวเวลา RC เพื่อป้องกันไฟกระชากในขณะเปิดอุปกรณ์ไฟฟ้าใดๆ ทำให้ไม่เกิดความเสียหายกับโซลิดสเตท-รีเลย์ และไตรแอค โดยกระแสไฟฟ้าจะไหลผ่าน C_2 และ R_7 นั่นเอง

ส่วนของความถี่ในการสวิตช์ของชุดทรานซิสเตอร์สวิตช์ ไม่ต้องคำนึงถึง เนื่องจากในการนำไปใช้งานจริงนั้น การเปิด ปิดอุปกรณ์ไฟฟ้าต่างๆ มักจะเปิดเป็นเวลานาน ซึ่งไม่มีการเปิดปิดด้วยความเร็วเท่าใดนัก จึงไม่ต้องคำนึงถึงในส่วนนี้

3.2 ภาคแสดงผลแบบผลึกเหลว

ในส่วนของการแสดงผลของตัวเครื่องที่ใช้ควบคุมวงจรโซลิดสเตทรีเลย์ทั้ง 8 ช่องจะใช้จอแสดงผลแบบผลึกเหลว ซึ่งจะใช้แสดงค่าของเวลาในการเปิด และปิดอุปกรณ์ไฟฟ้าต่างๆ แสดงหน้าที่การทำงานต่างๆ รวมถึงแสดงการเปิด และปิดอุปกรณ์ไฟฟ้าต่างๆ โดยตรง

โดยในส่วนตัวแสดงแบบผลึกเหลวนี้จะใช้ชนิดการแสดงผลแบบตัวอักษร ซึ่งมีขนาด 16 ตัวอักษร 4 บรรทัด รุ่น DMC 164 ซึ่งมีคอนโทรลเลอร์เบอร์ HD44780 เป็นไอซี LSI ตัวหนึ่งใช้ควบคุมตัวแสดงผลแบบผลึกเหลว โดยแสดงผลในรูปตัวอักษร หรือสัญลักษณ์ต่างๆ ตัวมันเองสามารถต่อใช้งานแบบ 4 บิต หรือ 8 บิต ก็ได้ โดยถ้าเราต่อแบบ 4 บิต จะต่อใช้งานที่ DB7-DB4 เท่านั้น โดยข้อมูลครั้งแรกที่ส่งนั้น HD44780 จะถือเป็นข้อมูล 4 บิตบน และข้อมูลที่ส่งต่อมานั้นเป็นข้อมูล 4 บิตล่าง



รูปที่ 3.2 ภาคแสดงผลแบบผลึกเหลว

จากรูปที่ 3.2 เป็นการต่อ 8951 ให้เข้ากับตัวแสดงผลแบบผลึกเหลว โดยจะจำลองสัญญาณต่างๆ ขึ้นมา โดยใช้สัญญาณจากพอร์ต 2 และพอร์ต 3 โดยพอร์ต 2 นั้นจะใช้เป็นพอร์ตข้อมูล และเป็นสัญญาณควบคุมใช้เมื่อเปิดไฟป้อนให้กับ HD44780 นั้นก็จะทำการรีเซตตัวมันเอง โดยใช้เวลาประมาณ 10 มิลลิวินาที หลังจากไฟ VDD ถึง 4.5 โวลต์ แล้ว โดยจะเซตตัวมันเอง

3.2.1 ขาต่างๆ ในการต่อใช้งาน

1. RS (Registor Selection) จะเป็นขาเลือกรีจิสเตอร์ภายในซึ่งมีอยู่ 2 ตัว คือ รีจิสเตอร์คำสั่ง (IR) และรีจิสเตอร์ข้อมูล (DR) โดยถ้าเป็น 1 จะเลือกรีจิสเตอร์ข้อมูล และถ้าเป็น 0 จะเป็นการเลือกรีจิสเตอร์คำสั่ง

2. R/W (Read / Write) เป็นตัวเลือกว่าจะอ่าน หรือเขียนข้อมูลจากตัวไอซีโดยอ่านข้อมูล = 1 , เขียนข้อมูล = 0

3. E (Enable Signal) เป็นขากำหนดสภาพการรับเขียนอ่านข้อมูล

ตารางที่ 3.1 การเลือกรีจิสเตอร์

RS	R / W	ENABLE	OPERATION
0	0	falling egde	IR write as internal operation (Display clear, atc)
0	1	positive pulse	Read busy flag (DB7) and address counter (DB0-DB6)
1	0	falling egde	DR write as internal operation (DR to DD or CG RAM)
1	1	positive pulse	DR read as internal operation (DD or CG RAM to DR)

4. DB0-DB7 เป็นขารับส่งข้อมูลจากตัวไอซี

5. VDD ไฟเลี้ยงตัววงจร +5V

6. VSS เป็นขากราวด์

7. VLC เป็นขารับแรงดันในการขับตัวแสดงผลแบบผลึกเหลวให้สว่าง หรือมืด

3.2.2 การเขียนโปรแกรมควบคุม

1. เมื่อจ่ายไฟเลี้ยงให้กับตัวแสดงผลแบบผลึกเหลวครั้งแรก ภายในจะมีการรีเซตระบบโดยอัตโนมัติซึ่งจะใช้เวลา 10 มิลลิวินาที หลังจากที่ระบบแรงดันไฟขึ้นถึง 4.5 โวลต์ แล้วทั้งนี้ระบบรีเซตดังกล่าวจะกระทำสิ่งต่างๆ ต่อไปนี้

- ทำการล้างจอภาพทั้งหมด (Clear Display)

- กำหนดคุณสมบัติด้วยคำสั่ง Function set คือ DL= 1 (ติดต่อกับไมโครคอนโทรลเลอร์ในแบบ 8 บิต) , N = 0 (แสดงข้อมูล 1 บรรทัด) , F = 0 (กำหนดตัวอักษรแบบ 5x7 Dots)

- กำหนดคุณสมบัติด้วยคำสั่ง Display ON/OFF คือ D = 0 (ไม่แสดงข้อมูล) , C = 0 (Cursor OFF) , B = 0 (Blank OFF)

- กำหนดคุณสมบัติด้วยคำสั่ง Entry mode set คือ I/D = 1 (Increment) , S = 0 (No shift)

การใช้งานตัวแสดงผลแบบผลึกเหลวต้องรอให้ขบวนการรีเซตภายในทำงานเรียบร้อยเสียก่อนซึ่งจะตรวจสอบได้ด้วย BF หรืออาจจะใช้การหน่วงเวลาก็ได้

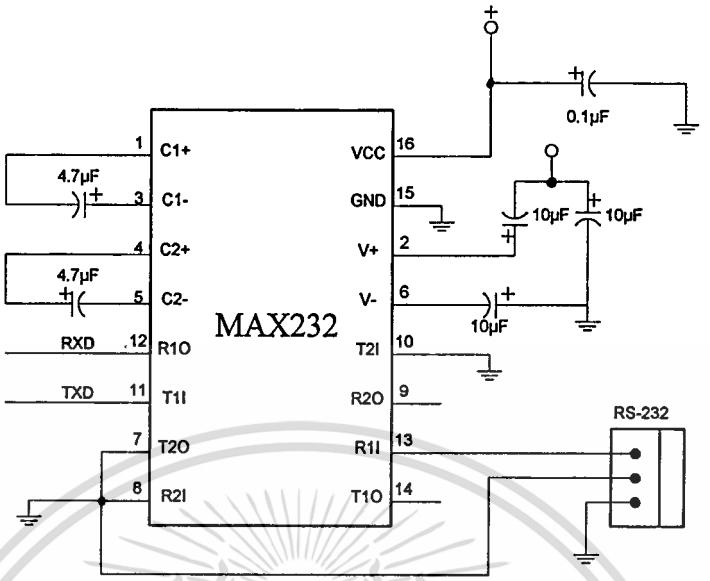
2. การใช้งานตัวแสดงผลแบบผลึกเหลวจะต้องเกี่ยวข้องกับทางด้านโปรแกรมเป็นส่วนใหญ่ ชุดคำสั่งต่างๆ รวมถึงการอ่าน หรือเขียนข้อมูลนั้น จะถูกกำหนดด้วยขาสัญญาณทั้งหมดที่มีอยู่ ปกติโปรแกรมจะต้องกำหนดคุณสมบัติต่างๆ ที่ต้องการไว้ที่ส่วนต้น จากนั้นจะเป็นการอ่าน หรือเขียนข้อมูลลงใน DDRAM ซึ่งก็คือข้อความที่จะให้แสดงผลนั่นเอง

3.3 วงจร RS-232C

ในการติดต่อระหว่างคอมพิวเตอร์กับภาคไมโครคอนโทรลเลอร์ เพื่อที่จะสั่งงานจากคอมพิวเตอร์มายังภาคไมโครคอนโทรลเลอร์ แล้วภาคไมโครคอนโทรลเลอร์ก็จะไปควบคุมชุดโซลิดสเตทรีเลย์อีกทีนั้น การติดต่อจะใช้มาตรฐาน RS-232C โดยจะใช้ไอซี MAX232 เป็นตัวกลางในการแปลงระดับสัญญาณจาก RS-232C ให้เป็นทีทีแอล เพื่อภาคไมโครคอนโทรลเลอร์จะรับข้อมูล และนำไปประมวลผลได้

จากรูปที่ 3.3 จะเป็นการต่อใช้งานของ MAX232 ซึ่งจะใช้เพียง 3 เส้นสัญญาณ ในลักษณะข้อมูลแบบอนุกรม คือ TD,RD และ GND ข้อมูลจาก DB-2 จะเป็นสัญญาณมาตรฐาน RS-232C ซึ่งจะมาต่อเข้ากับ MAX232 ดังรูป คือ TD ต่อกับขา T20 (ขา7) และ RD ต่อกับขา R1I (ขา13)

เมื่อรับข้อมูลจากคอมพิวเตอร์แล้ว MAX232 จะแปลงระดับสัญญาณ RD เป็นระดับสัญญาณแบบทีทีแอล โดยสัญญาณที่แปลง แล้วของสัญญาณ RD จะออกมาเป็นระดับสัญญาณทีทีแอล (ขา 11) แล้วทั้งสองสัญญาณก็จะส่งไปยังภาคไมโครคอนโทรลเลอร์ ซึ่งจะส่งไปยังขา RXD (ขา 10) และขา TXD (ขา 11) ของ 8951 ตามลำดับ แล้วจากนั้นไมโคลคอนโทรลเลอร์ก็จะนำไปประมวลผลตามคำสั่งนั้นๆ



รูปที่ 3.3 การต่อใช้งานไอซี MAX232

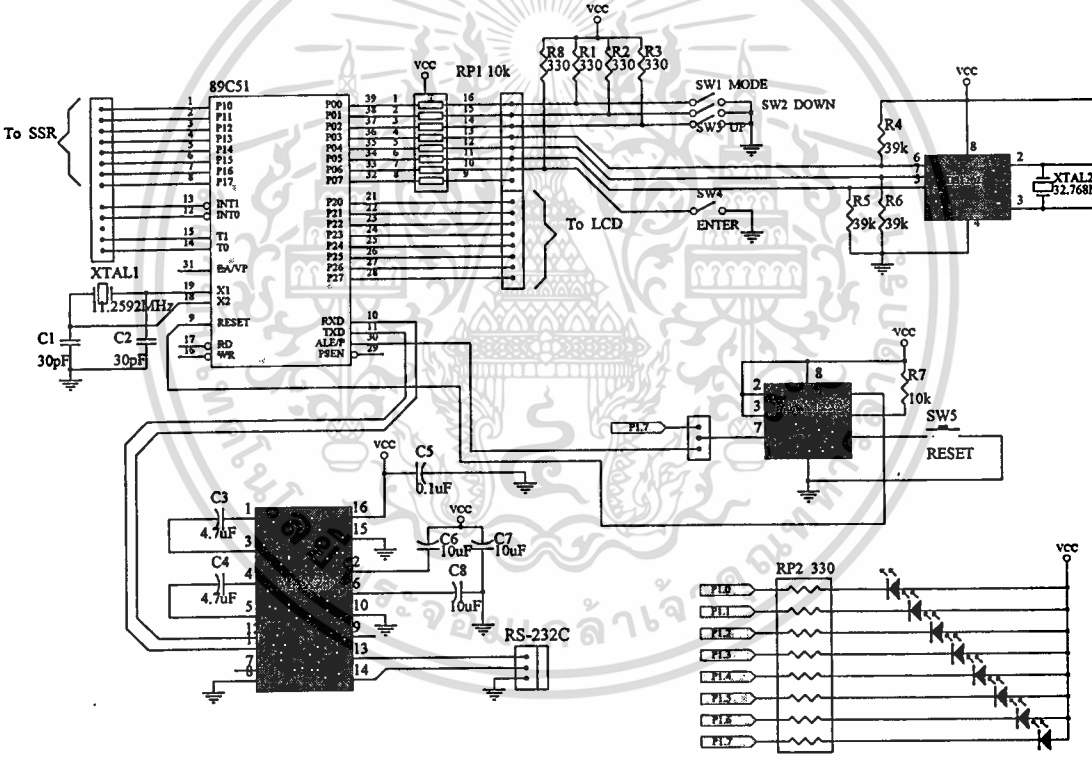
3.4 วงจรไมโครคอนโทรลเลอร์

ในโครงงานนี้จะใช้ชิพเบอร์ 89C51 ของบริษัท ATMEL ซึ่งโครงสร้างพื้นฐานภายในจะเหมือนกับ 8751 (ตระกูล MCS51 ของบริษัท INTEL) ทุกประการ โดยมีหน่วยความจำภายในเป็นแบบแฟรช ความจุ 4 กิโลไบต์ ซึ่งสามารถ โปรแกรมได้ถึง 1,000 ครั้ง

จากรูปที่ 3.4 เป็นวงจรทั้งหมดของภาคไมโครคอนโทรลเลอร์ ซึ่งจะประกอบด้วยไมโครคอนโทรลเลอร์เบอร์ 8951 , วงจรรีเซต และ Watch Dog เบอร์ MAX1232 และส่วนของวงจรที่ใช้เป็นฐานเวลา ซึ่งใช้เบอร์ DS1202 ไมโครคอนโทรลเลอร์จะทำหน้าที่ในการรับข้อมูลแบบอนุกรมมาจาก MAX232 จากนั้นจะนำข้อมูลที่รับมาไปทำการประมวลผลว่าต้องการให้ทำอะไร ที่พอร์ต 1 ของไมโครคอนโทรลเลอร์จะใช้ในการควบคุมวงจรโซลิดสเตท-รีเลย์ พอร์ต 2 , P3.5 และ P3.4 จะใช้ในควบคุมการแสดงผลแบบผลึกเหลว พอร์ต 0 จะใช้ในการรับคีย์ที่กดสั่งงานเข้า และเป็นตัวรับค่าเวลาจาก DS1202 ดังรูปที่ 3.4

ในส่วนของการรีเซตนั้นจะใช้ชิพเบอร์ MAX1232 โดยจะใช้สำหรับการรีเซตระบบรวมทั้งวงจร Watch Dog พร้อมกันไปด้วย ซึ่งการรีเซตแบบนี้จะให้ผลดีกว่าการรีเซต ด้วยวงจร RC ทั่วๆ ไป โดยจะทำการรีเซต ทั้งช่วง Power up และ Down จึงต้องแน่ใจได้ว่าระบบจะไม่เกิดสภาพรวนจากการรีเซตอย่างแน่นอน การรีเซตจากMAX1232นี้จะตั้งระดับไว้ที่ 10 % ของ VCC คือระบบจะทำงานทันที เมื่อไฟ VCC ต่ำกว่า 4.5 V

ส่วนวงจร Watch Dog ก็จะยิ่งช่วยให้ระบบเกิดความเสถียรมากยิ่งขึ้น โดยสามารถเลือก Disable หรือ Enable ได้ด้วย Jumper ในกรณีที่ตั้งไว้ที่ Disable ระบบจะต่ออยู่กับขา ST ของ MAX1232 เข้ากับ ALE ของซีพียู หมายถึงว่าจะมีพัลส์ส่งมามาตลอดเวลา แต่ถ้าตั้งไว้ที่ Enable ระบบจะต่อเข้ากับ P1.7 ของซีพียู เพราะฉะนั้นโปรแกรมที่เขียนขึ้นจะต้องส่งสัญญาณมากระตุ้นที่ P1.7 เสมอภายในเวลา 1.2 วินาที ในกรณีที่สัญญาณขาดหายไป ซึ่งอาจจะเกิดจากการ Hang ของระบบตัว MAX 1232 ก็จะทำการ Reset ระบบทันทีจะมีทำให้ระบบทั้งหมดกลับมาทำงานได้ตามปกติ



รูปที่ 3.4 วงจรทั้งหมดของภาคไมโครคอนโทรลเลอร์

ในการใช้งานชิพเบอร์ DS1202 เพื่อเป็นตัวเก็บค่าเวลาจริง จะมีการใช้งานคือ จะต้องให้ขา RST และ SCLK เป็นลอจิกต่ำทั้งคู่ก่อนหลังจากนั้น RST จะต้องเป็นลอจิกสูง เพื่อให้

ติดต่อกับรีจิสเตอร์ภายใน DS1202 ได้ และระหว่างติดต่อกับ DS1202 ขา RST จะต้องเป็นลอจิกสูง โดยการอินพุตข้อมูลเข้า DS1202 จะเกิดขึ้นที่ขาขึ้นของ SCLK และการเอาต์-พุตข้อมูลจาก DS1202 จะเกิดขึ้นที่ขาลงของ SCLK และหลังจากเลิกติดต่อกับ DS1202 จะต้องให้ RST เป็นลอจิกต่ำอีก เพราะฉะนั้นการอ่าน และเขียนค่าให้กับ DS1202 จะอยู่ในส่วนของโปรแกรมที่จะต้องเขียนขึ้น

นอกจากการสั่งงานมาจากคอมพิวเตอร์ แล้วยังจะสามารถสั่งงาน และควบคุมอุปกรณ์ไฟฟ้าทั้ง 8 ตัวได้จากการกดคีย์ที่ภาคไมโครคอนโทรลเลอร์ได้อีกด้วย โดยมีทั้งหมด 4 ปุ่ม คือ ปุ่มโหมด , ปุ่มขึ้น , ปุ่มลง และปุ่ม Enter ในส่วนของโหมดการทำงานจะมีทั้งหมด 4 โหมด คือ โหมดตั้งเวลา ,โหมดเปิด และปิดอุปกรณ์ไฟฟ้าโดยตรง , โหมดตั้งเวลาเปิด และโหมดตั้งเวลา ปิด เมื่อกดปุ่มโหมดแล้วที่จอแสดงผลจากแสดงทั้ง 4 โหมดดังกล่าวให้เลือกโดยการกดปุ่มขึ้น หรือปุ่มลงไปยังโหมดที่ต้องการเลือกแล้วก็กดปุ่ม Enter เป็นการแสดงว่าต้องการเลือกโหมดนั้นๆ

บทที่ 4

การทดลอง และผลการทดลอง

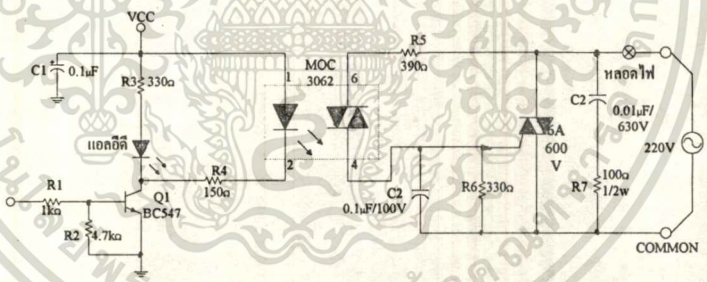
4.1 การทดลองวงจรโซลิดสเตทรีเลย์

4.1.1 ลำดับขั้นการทดลอง

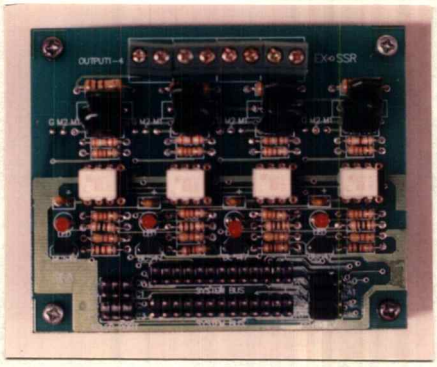
1. ต่อหลอดไฟฟ้าที่จุดต่อบนบอร์ดโซลิดสเตทรีเลย์ซึ่งมีทั้งหมด 8 ช่อง เพื่อแสดงการเปิด-ปิดหลอดไฟฟ้า

2. ป้อนแรงดันไฟฟ้ากระแสตรง 5 โวลต์ และแรงดันไฟฟ้ากระแสสลับ 220 โวลต์เข้าที่บอร์ดวงจรโซลิดสเตทรีเลย์ เพื่อเป็นแหล่งจ่ายไฟให้กับวงจร

3. ป้อนสัญญาณลอจิก 1 เข้าที่อินพุตของแต่ละช่อง เพื่อไบอัสให้วงจรโซลิดสเตทรีเลย์ทำงาน



รูปที่ 4.1 วงจรโซลิดสเตทรีเลย์



รูปที่ 4.2 แผงวงจรโซลิดสเตทรีเลย์

4.1.2 ผลการทดลอง

จากการทดลองปรากฏว่าเมื่อป้อนสัญญาณลอจิก 1 เข้าที่อินพุตช่องใดช่องหนึ่งใน 8 ช่องหรือทั้ง 8 ช่องพร้อมกันทั้งหมดนั้น แอลอีดีที่ใช้แสดงผลบนบอร์ดในช่องนั้นจะสว่าง และหลอดไฟฟ้าที่ต่ออยู่จะสว่างด้วย

ตารางที่ 4.1 ผลการทดลองวงจรโซลิดสเตทรีเลย์

อินพุต	เอาต์พุต	หลอดไฟ
0 V	0 V	ไม่ติด
5 V	220 V	ติด

4.2 การทดลองของจอแสดงผลแบบผลึกเหลว

4.2.1 ลำดับขั้นการทดลอง

1. ต่อจอแอลซีดีเข้ากับบอร์ดเอ็มซีเอส 51 โดยต่อที่พอร์ต 2
2. เขียนข้อมูลภาษาแอสเซมบลี และทำการบันทึกข้อมูลลงใน 89C51
3. นำ 89C51 เสียบลงบอร์ด
4. ป้อนแรงดันกระแสตรง 5 โวลต์ ให้แก่ระบบ

4.2.2 ผลการทดลอง

เมื่อทำการป้อนไฟให้แก่ระบบ ผลการทดลองปรากฏว่าจอแสดงผลแบบผลึกเหลวทำการแสดงผลตรงกับโปรแกรมที่เขียน โดยจะปรากฏชื่อของปริญญานิพนธ์ วัน เดือน ปี และเวลา

รูปที่ 4.3 การแสดงผลของแอลซีดี

4.3 การทดลองวงจร RS-232C

4.3.1 ลำดับขั้นการทดลอง

1. นำเอาบอร์ดต่างๆ เชื่อมต่อเข้าด้วยกัน

2. ป้อนแรงดันไฟฟ้ากระแสตรง 5 โวลต์ให้แก่บอร์ดต่างๆ

3.นำสายคอนเนคเตอร์เชื่อมต่อระหว่างไมโครคอมพิวเตอร์ไปยังพอร์ต RS-232C ที่อยู่บนบอร์ด

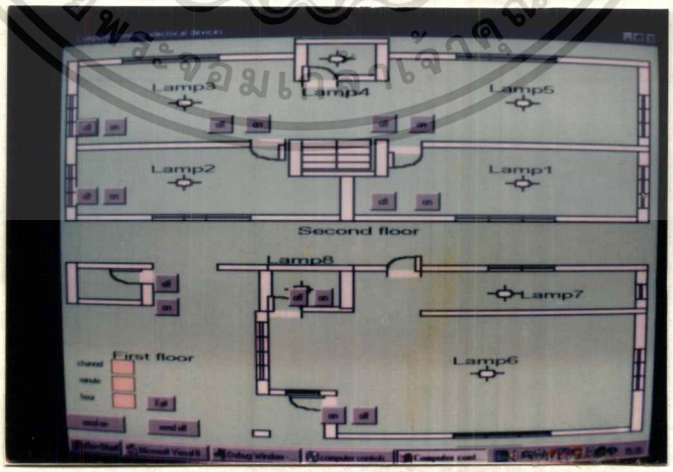
4. เปิดเครื่องคอมพิวเตอร์ แล้วเรียกโปรแกรมวิชวลเบสิค แล้วเปิดแฟ้มข้อมูลชื่อ Project1.Mak

5. ทดลองเปิด - ปิดหลอดไฟผ่านทางคอมพิวเตอร์

6. ทดลองตั้งเวลา เพื่อเปิด - ปิด หลอดไฟผ่านทางคอมพิวเตอร์

4.3.2 ผลการทดลอง

เมื่อเรียกโปรแกรมขึ้นมาจะปรากฏผังของแบบจำลอง และสวิตช์ที่ใช้เปิด- ปิด หลอดไฟ จากการทดลองตามข้อ 5 ใช้เม้าส์คลิกที่สวิตช์ ผลปรากฏว่าหลอดไฟที่ทำการทดลองจะติดเมื่อทำการสั่งเปิด (On) และจะดับเมื่อทำการสั่งปิด (Off) ในข้อ 6 ทำการตั้งเวลาในการเปิด-ปิดสวิตช์ โดยตั้งเวลาที่ต้องการ ผลปรากฏว่าหลอดไฟติด และดับตามเวลาที่กำหนด



รูปที่ 4.4 โปรแกรมวิชวลเบสิค

ตารางที่ 4.2 ผลการทดลองโปรแกรมวิชวลเบสิค

อินพุต	สภาวะของหลอดไฟ
Off	ไม่ติด
On	ติด

4.4 การทดลองวงจรไมโครคอนโทรลเลอร์

4.4.1 ลำดับขั้นการทดลอง

1. นำเอาบอร์ดต่างๆ จอแสดงผลแบบผลึกเหลว และ หลอดไฟเชื่อมต่อเข้าด้วยกัน
2. ป้อนแรงดันไฟฟ้ากระแสตรง 5 โวลต์ให้แก่บอร์ดต่างๆ และป้อนแรงดันไฟฟ้ากระแสสลับ 220 โวลต์ ให้แก่วงจรโซลิดสเตทรีเลย์ด้วย
3. กดโหมด เพื่อเข้าสู่เมนูหลัก
4. ทดลองตั้งวัน เดือน ปี และเวลา
5. ทดลองเปิด - ปิดหลอดไฟ
6. ทดลองตั้งเวลาเปิดหลอดไฟ
7. ทดลองตั้งเวลาปิดหลอดไฟ

4.4.2 ผลการทดลอง

เมื่อปฏิบัติตามข้อ 1 และข้อ 2 แล้วจอแสดงผลแบบผลึกเหลวจะแสดงชื่อปริญญา-นิพนธ์ วัน เดือน ปี และ เวลา เมื่อกดโหมดจะเป็นการเข้าสู่เมนูหลักได้แก่โหมดการตั้งเวลา โหมดการเปิด- ปิดหลอดไฟโดยตรง โหมดการตั้งเวลาเปิด โหมดการตั้งเวลาปิด จากนั้นทดลองตามข้อ 4 5 6 และ 7 ต่อไป

ผลการทดลองข้อ 4 ปรากฏว่า วัน เดือน ปี และ เวลา ที่ได้ตรงตามที่ตั้งไว้

รูปที่ 4.5 การแสดงผลของจอแอลซีดี เมื่อทำการตั้งเวลา

ผลการทดลองข้อ 5 เมื่อทำการตั้งเปิด- ปิด ผลปรากฏว่าหลอดไฟที่ทำการทดลองจะติด เมื่อทำการกด สวิตช์ Up และหลอดไฟจะดับเมื่อทำการกดสวิตช์ Down

ตารางที่ 4.3 ผลการทดลองวงจรไมโครคอนโทรลเลอร์

อินพุต	สภาวะของหลอดไฟ
Down	ไม่ติด
Up	ติด

ผลการทดลองข้อ 6 เมื่อทำการตั้งเวลาเปิดให้ทั้ง 8 ช่อง ปรากฏว่าหลอดไฟที่ต่ออยู่ในแต่ละช่องจะติด เมื่อถึงเวลาที่กำหนด

ผลการทดลองข้อ 7 เมื่อทำการตั้งเวลาปิดให้ทั้ง 8 ช่อง ปรากฏว่าหลอดไฟที่ต่ออยู่ในแต่ละช่องจะดับ เมื่อถึงเวลาที่กำหนด

4.5 สรุปผลการทดลอง

การทดลองในบทที่ 4 นี้ เป็นการทดลองการทำงานของแต่ละส่วน และทดลองการทำงาน เมื่อรวมแต่ละส่วนเข้าด้วยกัน ซึ่งทำให้ทราบถึงระบบการทำงานของเครื่องต้นแบบ และโปรแกรมว่าสามารถทำงานได้ตามที่กำหนด และสอดคล้องกันหรือไม่ และผลการทดลองที่ได้นั้นได้ผลดีเป็นที่น่าพอใจ

บทที่ 5

บทสรุป ปัญหา แนวทางแก้ไข และพัฒนา

5.1 บทสรุป

ขอบเขตของการออกแบบโครงงานคอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าภายในอาคารที่เสนอขออนุมัติหัวข้อปริญญานิพนธ์ โดยลักษณะการใช้งานจะแบ่งออกเป็น 2 ลักษณะ คือ

1. สามารถควบคุมการทำงานจากตัวเครื่องต้นแบบที่ต่อกับอุปกรณ์ไฟฟ้าโดยตรง
2. สามารถควบคุมการทำงานจากไมโครคอมพิวเตอร์

โดยระหว่างไมโครคอมพิวเตอร์กับตัวเครื่องต้นแบบจะรับส่งข้อมูลแบบอนุกรมด้วยมาตรฐาน RS-232C

5.1.1 การควบคุมอุปกรณ์ด้วยตัวเครื่องต้นแบบ

การทำงานในส่วนเครื่องต้นแบบนี้สามารถทำงานได้ โดยที่ไม่ต้องอาศัยไมโครคอมพิวเตอร์ หรือสามารถทำงานในลักษณะเครื่องเดี่ยว (Stand Alone) สามารถสั่งให้เปิด หรือปิดอุปกรณ์ไฟฟ้าทั้ง 8 ตัวได้โดยตรง ตั้งเวลาเปิด และปิดอุปกรณ์ไฟฟ้าได้ การสั่งงานควบคุมดังกล่าวจะมีปุ่มที่ใช้กดควบคุม โดยมีจอแสดงผลแบบผลึกแสดงเป็นเมนูควบคุมการทำงาน รวมทั้งยังมีแอลอีดี 8 ตัว เป็นตัวแสดงผล ซึ่งจะแสดงถึงอุปกรณ์ทั้ง 8 ตัว ว่าอุปกรณ์ตัวใดเปิดหรือปิดอยู่ในขณะนั้น

ในส่วนของระบบฐานเวลาก็จะมีการแสดงผลอยู่บนจอแสดงผลแบบผลึกเหลว ทั้งเวลา และวัน เดือน ปี ปัจจุบัน และเป็นฐานเวลาจริงสำหรับการควบคุมการทำงานของอุปกรณ์ไฟฟ้าทั้ง 8 ตัว รวมทั้งสามารถตั้งค่า วัน และเวลาได้อีกด้วย

การทำงานในส่วนนี้ จัดได้ว่ามีการทำงานที่สมบูรณ์แบบตามวัตถุประสงค์ที่ได้เสนอไว้ในการขอนุมัติหัวข้อปริญญานิพนธ์

5.1.2 การควบคุมอุปกรณ์ไฟฟ้าด้วยไมโครคอมพิวเตอร์

การทำงานในส่วนของไมโครคอมพิวเตอร์ จะใช้โปรแกรมวิชวลเบสิค สร้างแบบบ้านแล้วสามารถใช้เมาส์คลิ๊กสั่งให้เปิด หรือปิดอุปกรณ์ไฟฟ้าทั้ง 8 ตัวได้ โดยแบบบ้านที่แสดงหน้าจอจะเหมือนกับบ้านจำลองที่ใช้เป็นบ้านต้นแบบของโครงงาน

เมื่อเราสั่งงานจากไมโครคอมพิวเตอร์ที่หน้าจอแล้ว ข้อมูลคำสั่งจะถูกส่งแบบอนุกรมไปยังเครื่องต้นแบบด้วยมาตรฐาน RS-232C แล้วจากนั้นตัวเครื่องต้นแบบจะนำไปประมวลและควบคุมการเปิด และปิดอุปกรณ์ไฟฟ้าทั้ง 8 ตัว รวมทั้งตั้งเวลาเปิด และปิดด้วย

5.2 ปัญหา และแนวทางการแก้ไข

5.2.1 ปัญหา

1. เนื้อหาโปรแกรมวิชวลเบสิค ที่กล่าวถึงการส่งข้อมูลแบบอนุกรมที่เป็นภาษาไทยมีน้อย ส่วนใหญ่จะมีแต่หนังสือภาษาอังกฤษ

2. การเขียนโปแกรมวิชวลเบสิค และ การรันโปรแกรม รวมทั้งการแก้ไขเป็นไปด้วยความล่าช้ามาก

3. การส่งข้อมูลแบบอนุกรมจากไมโครคอมพิวเตอร์ มายังชุดไมโครคอนโทรลเลอร์มีความล่าช้า และบางครั้งการส่งเกิดความผิดพลาด เนื่องจากอัตราเร็วในการส่งที่ไม่สัมพันธ์กัน

4. อุปกรณ์ในส่วนของวงจรโซลิดสเตทรีเลย์มีราคาค่อนข้างสูง

5. ถ้านำไปควบคุมอุปกรณ์ที่ใช้กำลังไฟฟ้าสูงๆ เช่น ตู้เย็น หรือ เครื่องปรับอากาศ อุปกรณ์ไตรแอคในวงจรโซลิดสเตทรีเลย์อาจพังเสียหายได้

6. ในส่วนของไมโครคอนโทรลเลอร์ ที่จะต้องอัดโปรแกรมเข้าไป เมื่อมีการแก้ไขต้องถอดออกจากวงจรมาอัดโปรแกรมใหม่ แล้วนำไปใส่ในวงจรเพื่อทดลอง ซึ่งขั้นตอนนี้จะมีความล่าช้ามาก

5.2.2 แนวทางแก้ไข

1. เขียนโปรแกรมให้กะทัดรัด และใช้เครื่องไมโครคอมพิวเตอร์ที่มีการทำงานเร็ว

2.ปรับปรุงอัตราเร็วในการส่งให้สัมพันธ์กันทั้งส่วนของชุดไมโครคอนโทรลเลอร์และไมโครคอมพิวเตอร์

3. ใช้อุปกรณ์ที่มีคุณสมบัติใกล้เคียงกัน แต่ราคาลดลง

4. ใช้อุปกรณ์ไตรแอคที่มีอัตราทนกำลังสูงๆ

5. ใช้เครื่องอัดโปรแกรมที่สามารถใช้งานเป็นอีมูเลเตอร์ (Emulator) ได้ในตัวเดียวกัน

5.3 แนวทางในการพัฒนา

1. สามารถเพิ่มจำนวนของอุปกรณ์ไฟฟ้าที่ควบคุมให้มีจำนวนมากขึ้น

2. พัฒนาโปรแกรมวิชวลเบสิค ให้ได้แปลนแบบบ้านที่มีความสวยงามมากขึ้น

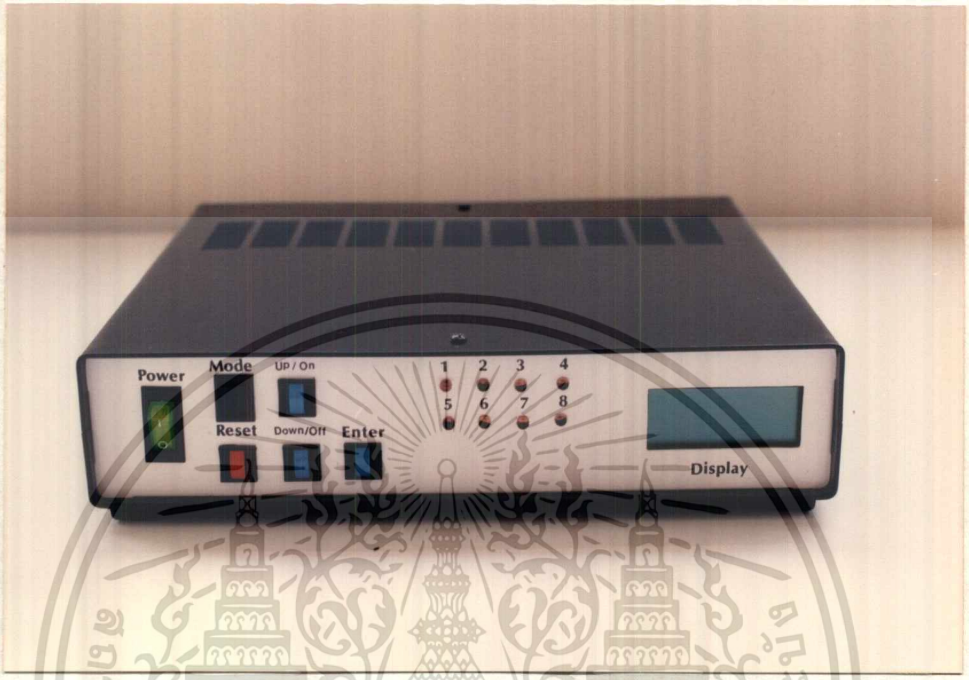
3.พัฒนาให้ใช้ได้กับอุปกรณ์ที่ใช้กำลังไฟฟ้าสูงๆ เพื่อพัฒนาให้สามารถนำไปใช้ในโรงงานอุตสาหกรรมได้

4. สามารถนำไปใช้งานในชีวิตประจำวันได้อย่างมีประสิทธิภาพ

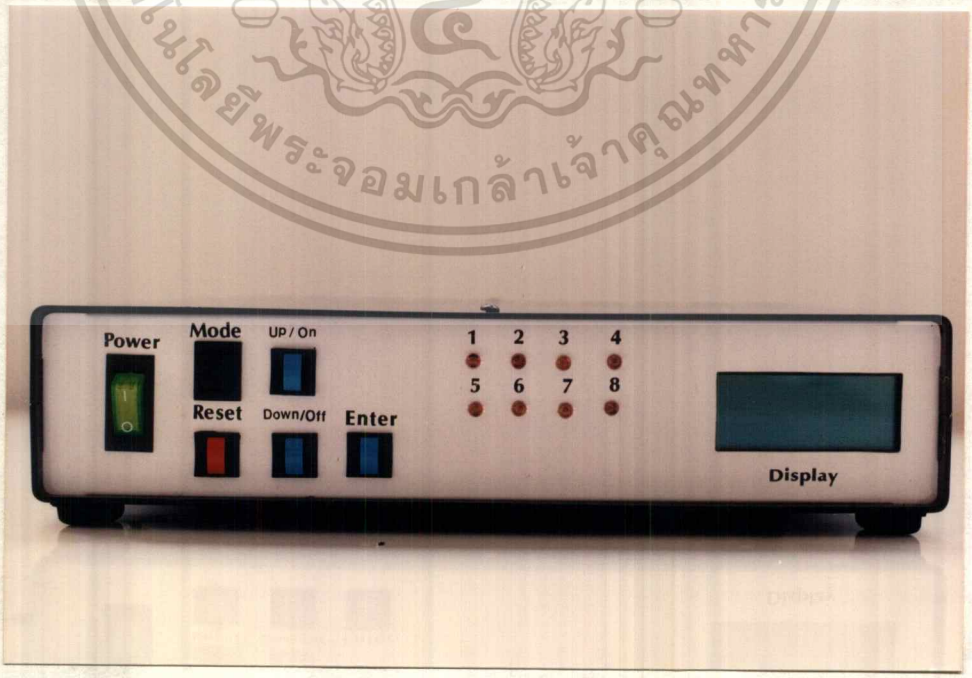
5. อาจเพิ่มเติมประสิทธิภาพของการทำงานเข้าไป นอกเหนือจากการตั้งเวลาเปิด และปิด เช่น มีสัญญาณเตือนเมื่ออุปกรณ์ที่ควบคุมเกิดไม่ทำงาน หรือเกิดความเสียหาย , สามารถควบคุมการหรี่ไฟของอุปกรณ์ที่ส่องสว่างได้ เป็นต้น

ภาคผนวก ก
เครื่องต้นแบบ

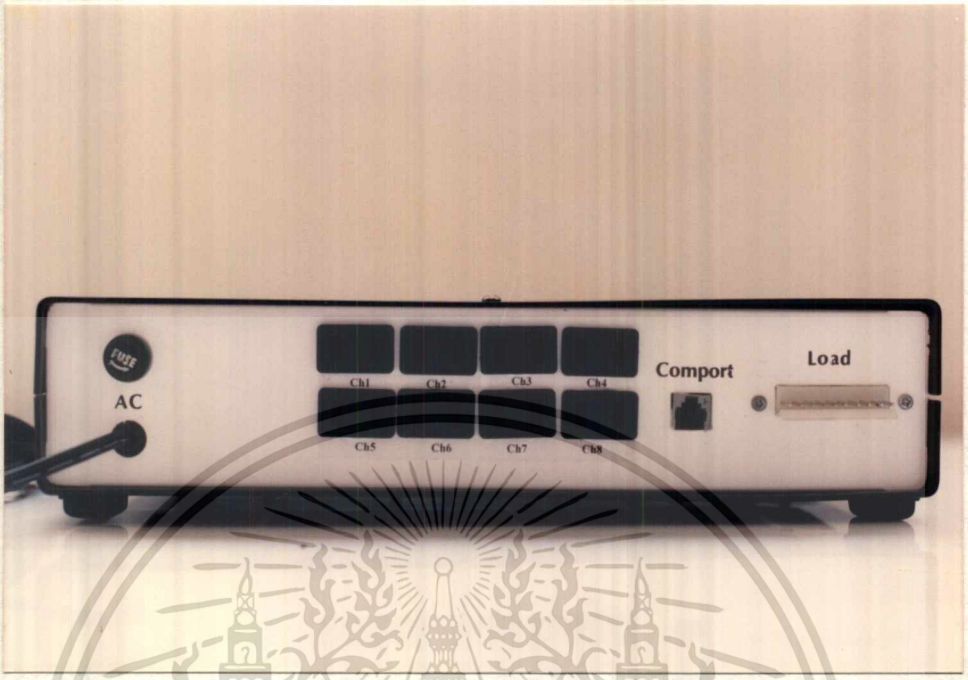
เครื่องต้นแบบ



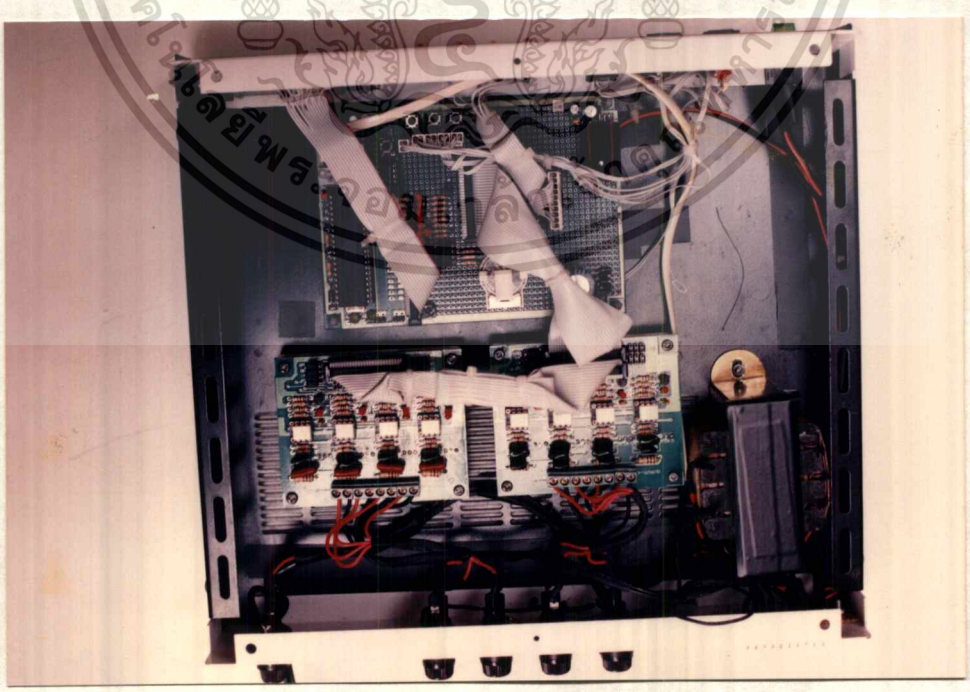
รูปที่ ก.1 เครื่องต้นแบบ



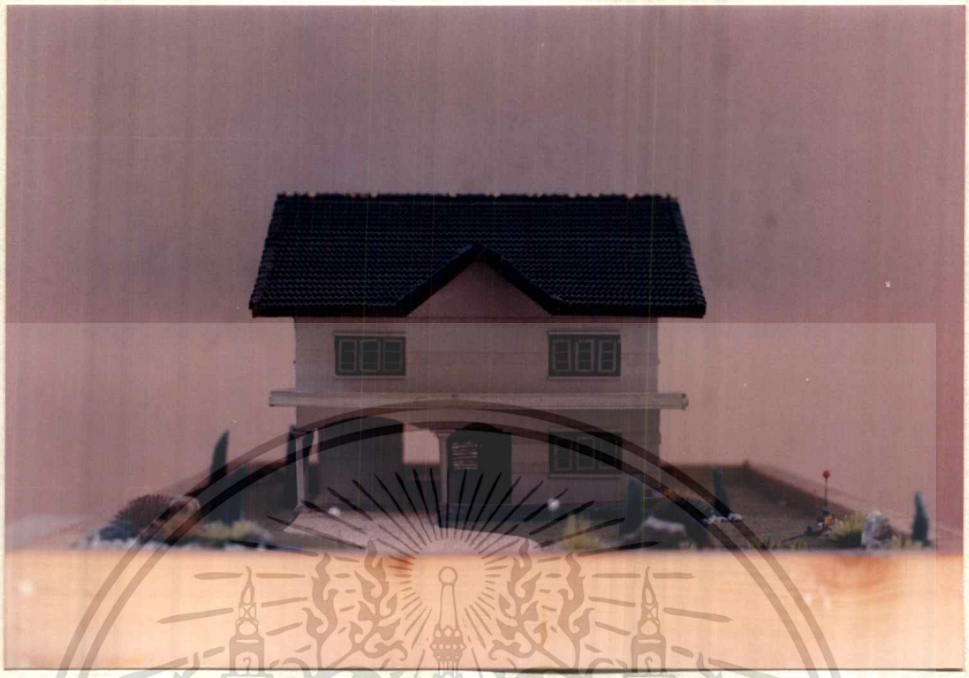
รูปที่ ก.2 ด้านหน้าเครื่องต้นแบบ



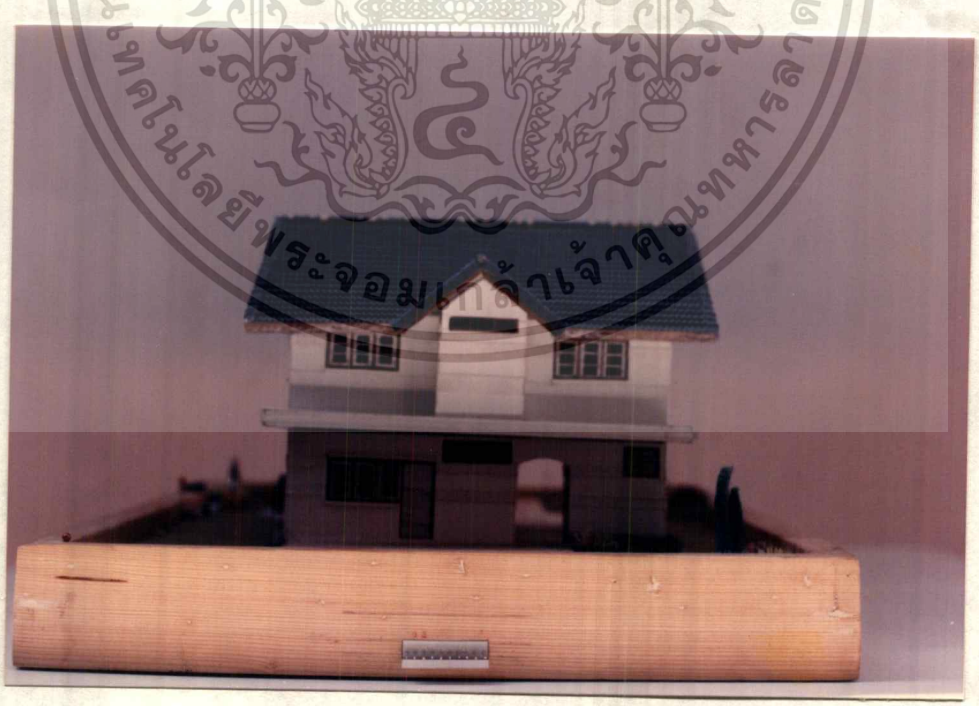
รูปที่ ก.3 ด้านหลังเครื่องต้นแบบ



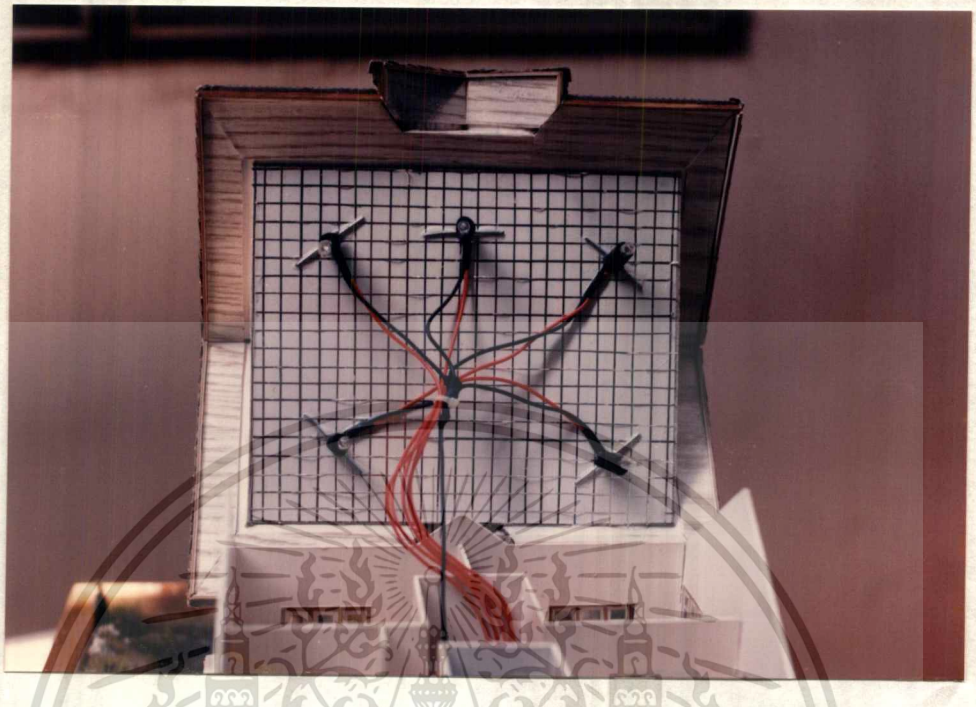
รูปที่ ก.4 การเดินสายภายในเครื่องต้นแบบ



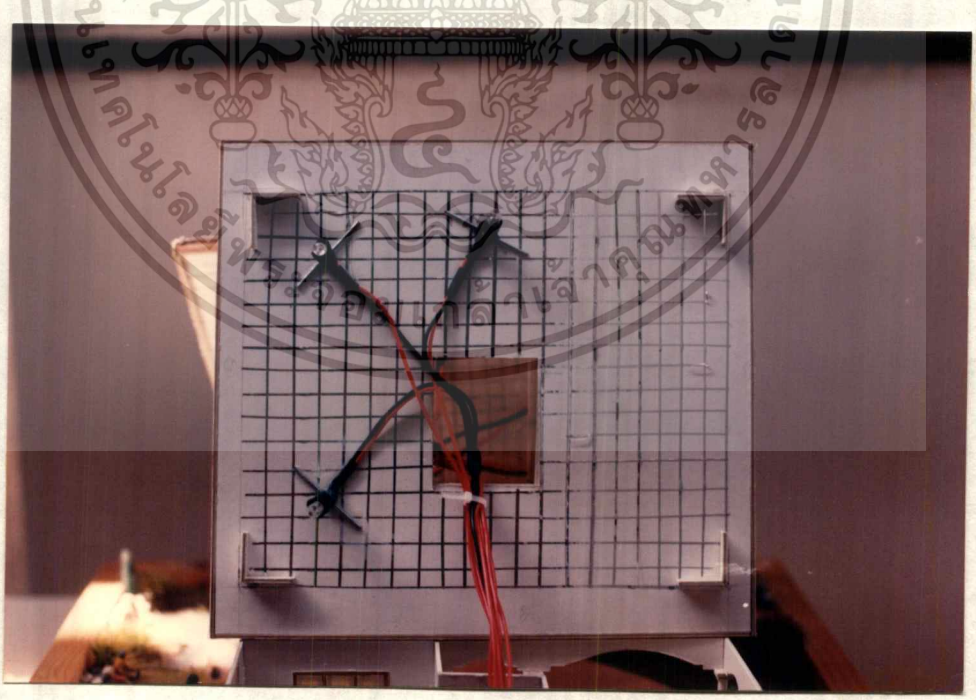
รูปที่ ก.5 แบบบ้านจำลองด้านหน้า



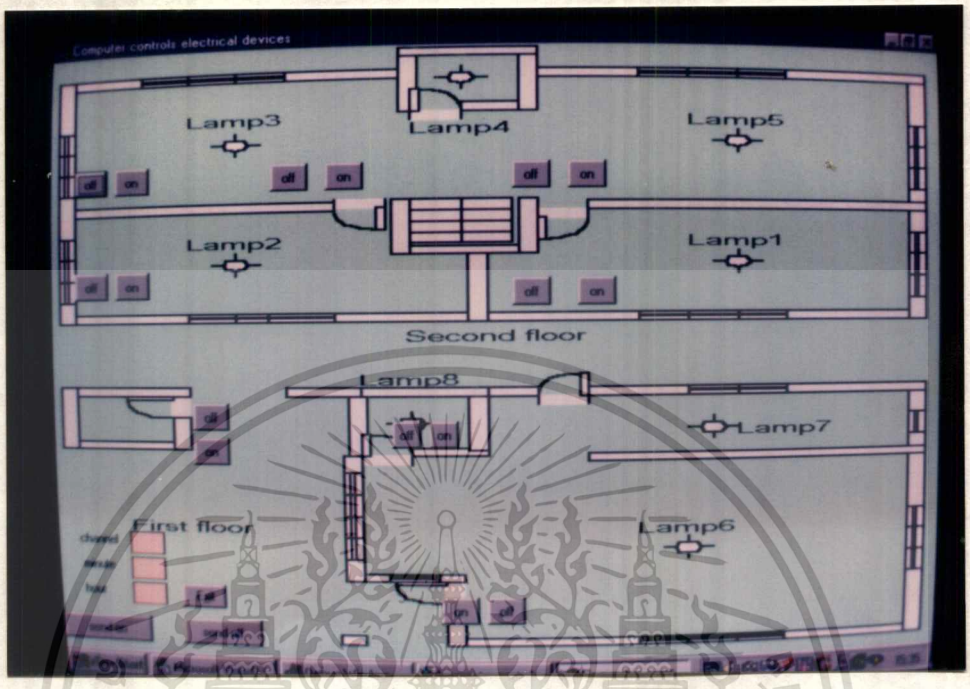
รูปที่ ก.6 แบบบ้านจำลองด้านหลังพร้อมจุดต่อแรงดันไฟฟ้ากระแสสลับ 220 โวลต์



รูปที่ ก.7 การเดินสายชั้นบนภายในบ้านจำลอง



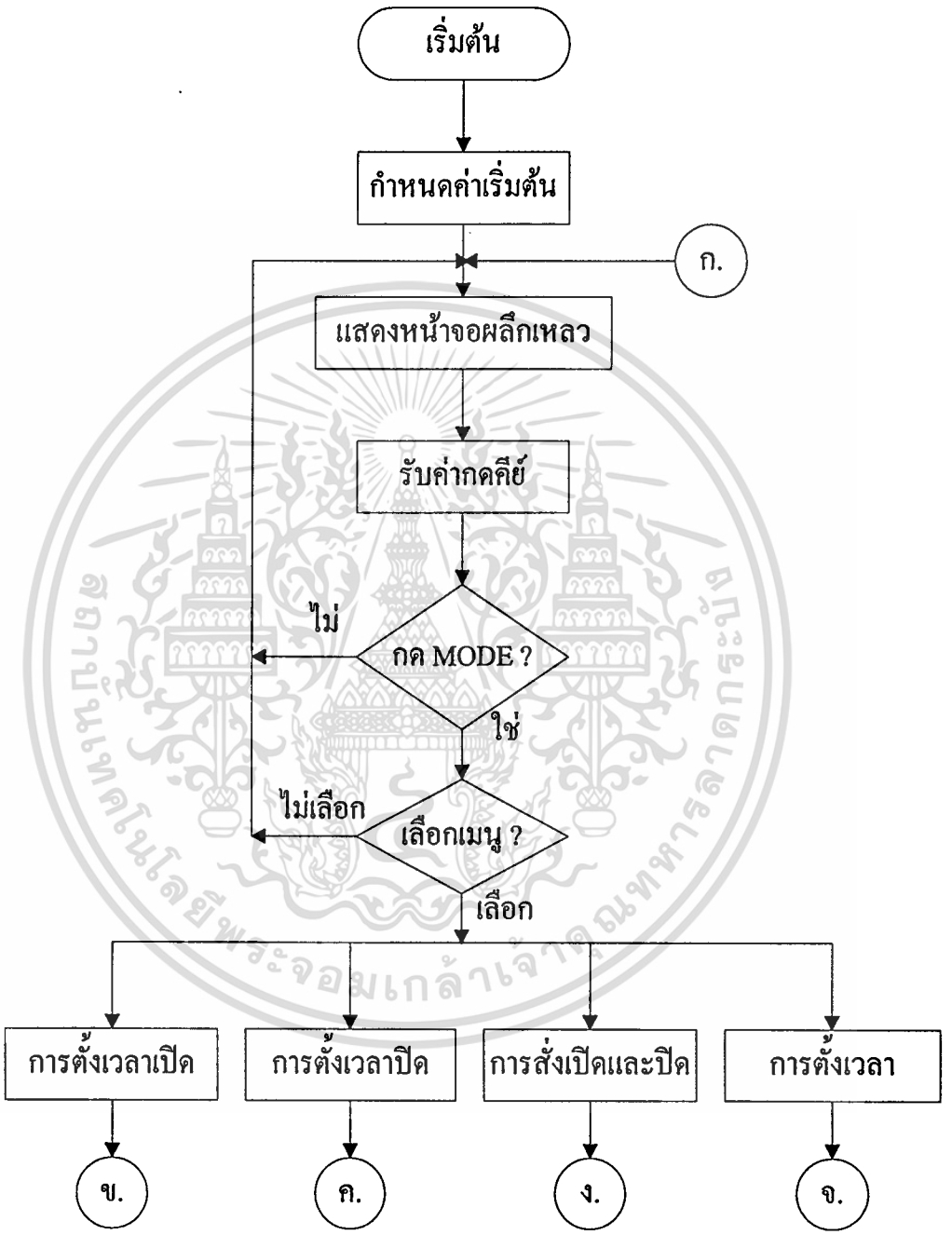
รูปที่ ก.8 การเดินสายชั้นล่างภายในบ้านจำลอง



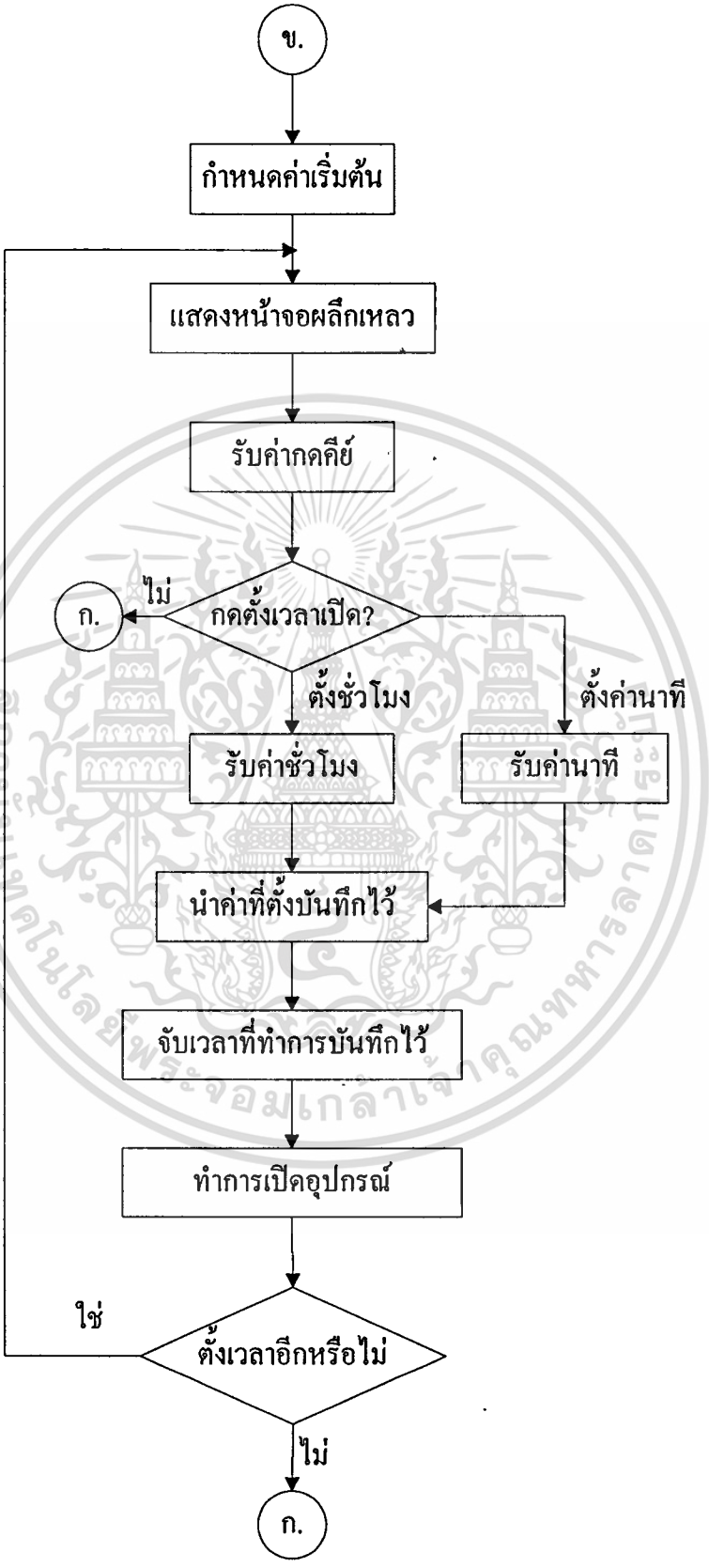
รูปที่ ก.9 ผังบ้านในโปรแกรมวิชวลเบสิค

ภาคผนวก ข

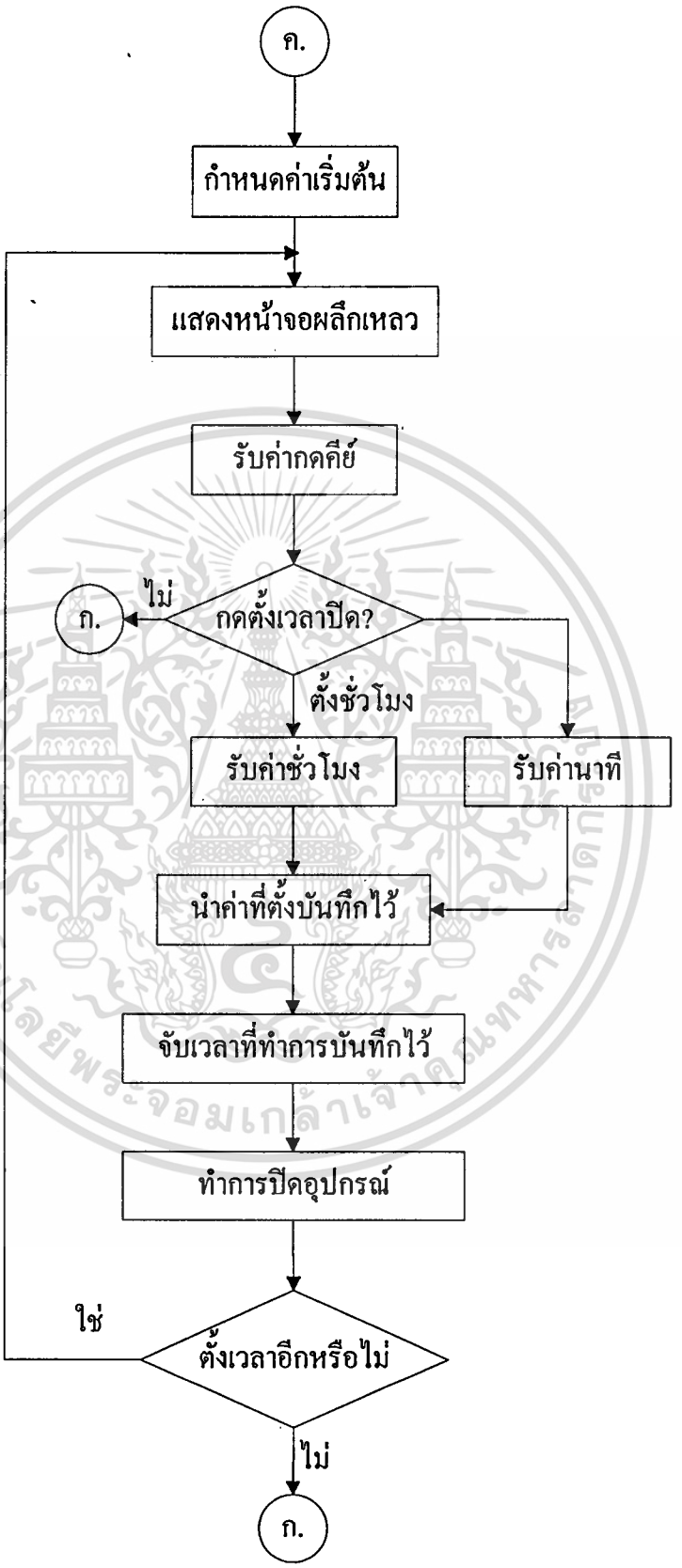
แผนผังการทำงาน



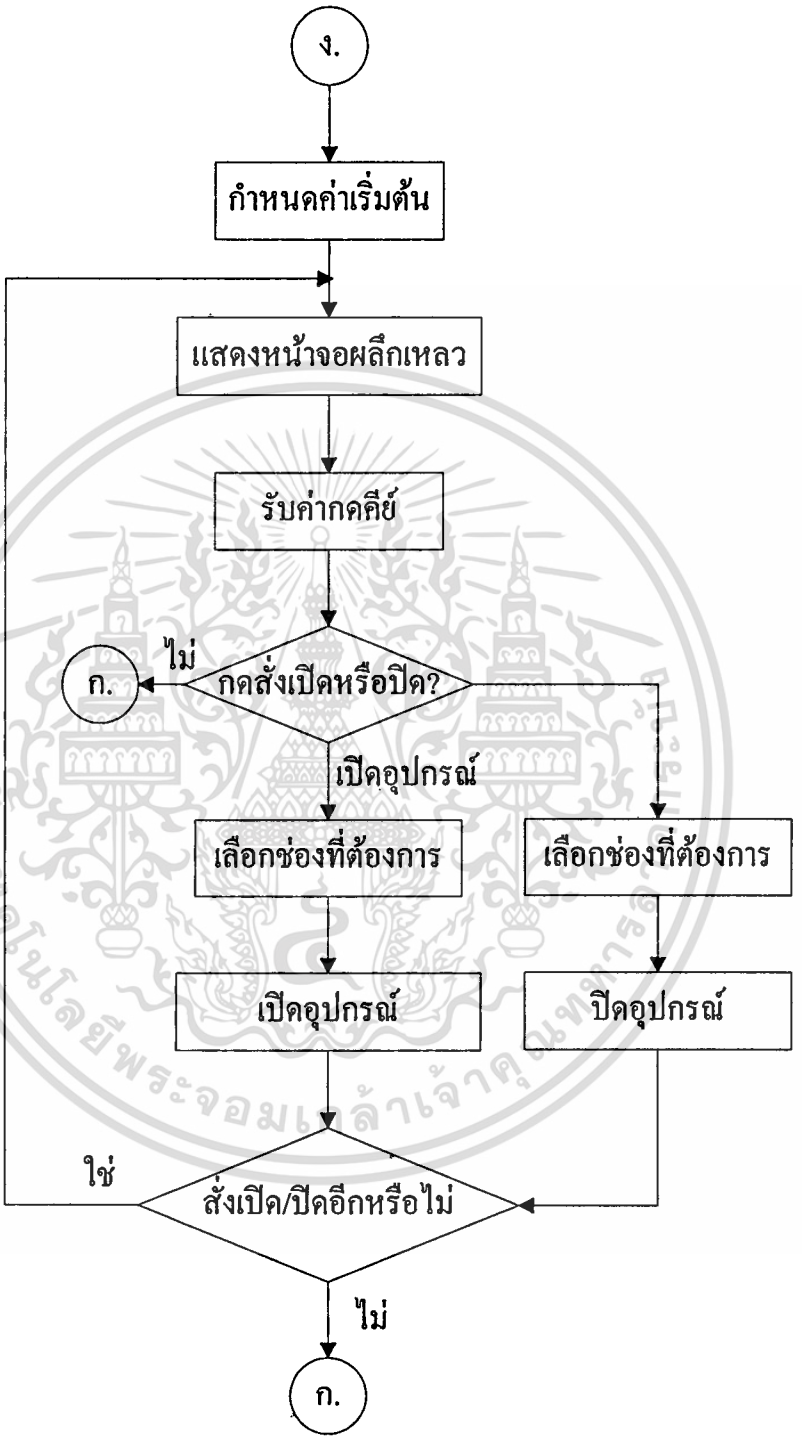
รูปที่ ข.1 ผังการทำงานของโปรแกรม



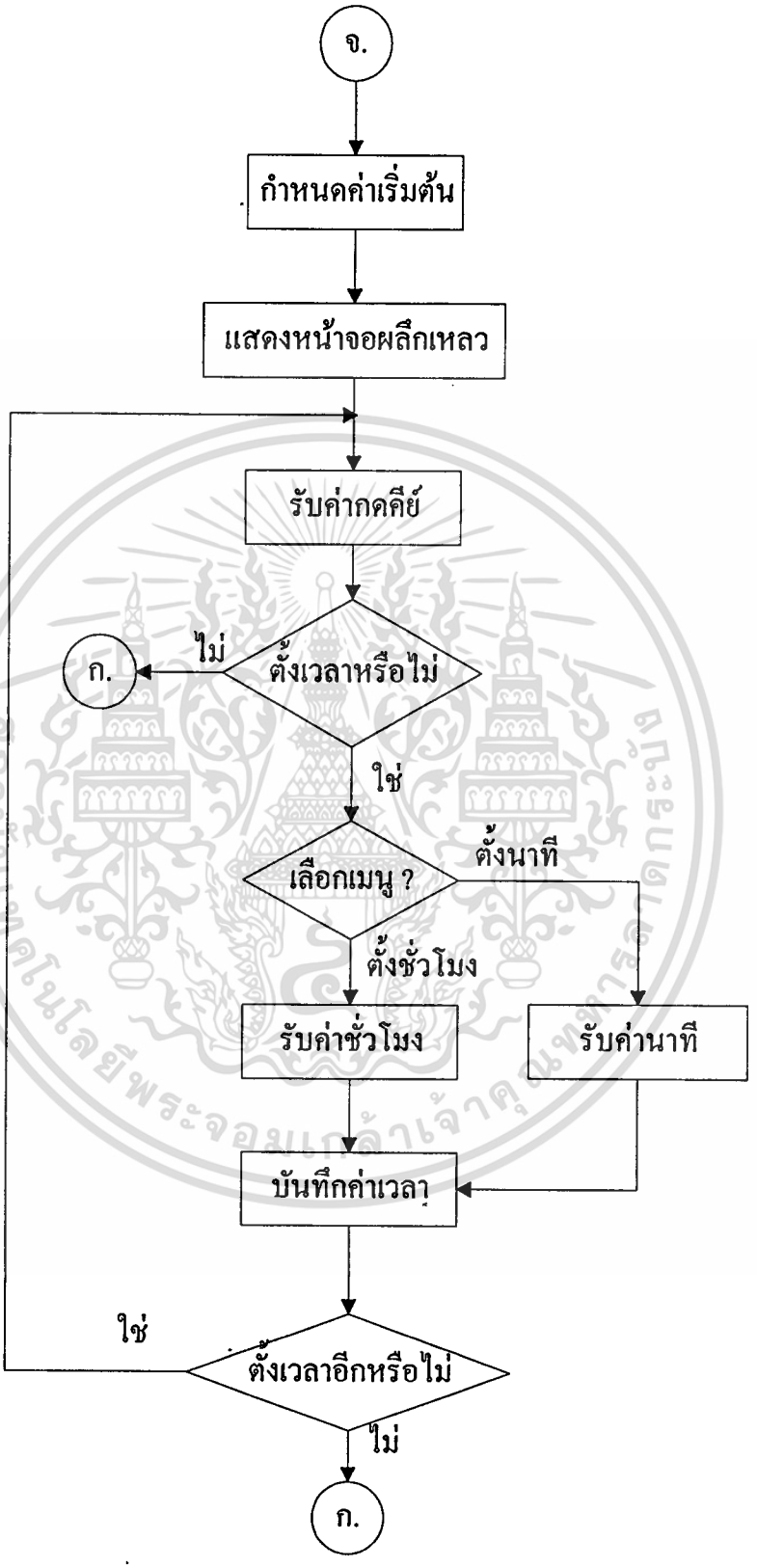
รูปที่ ข.2 ผังการทำงานของโปรแกรมการตั้งเวลาเปิด



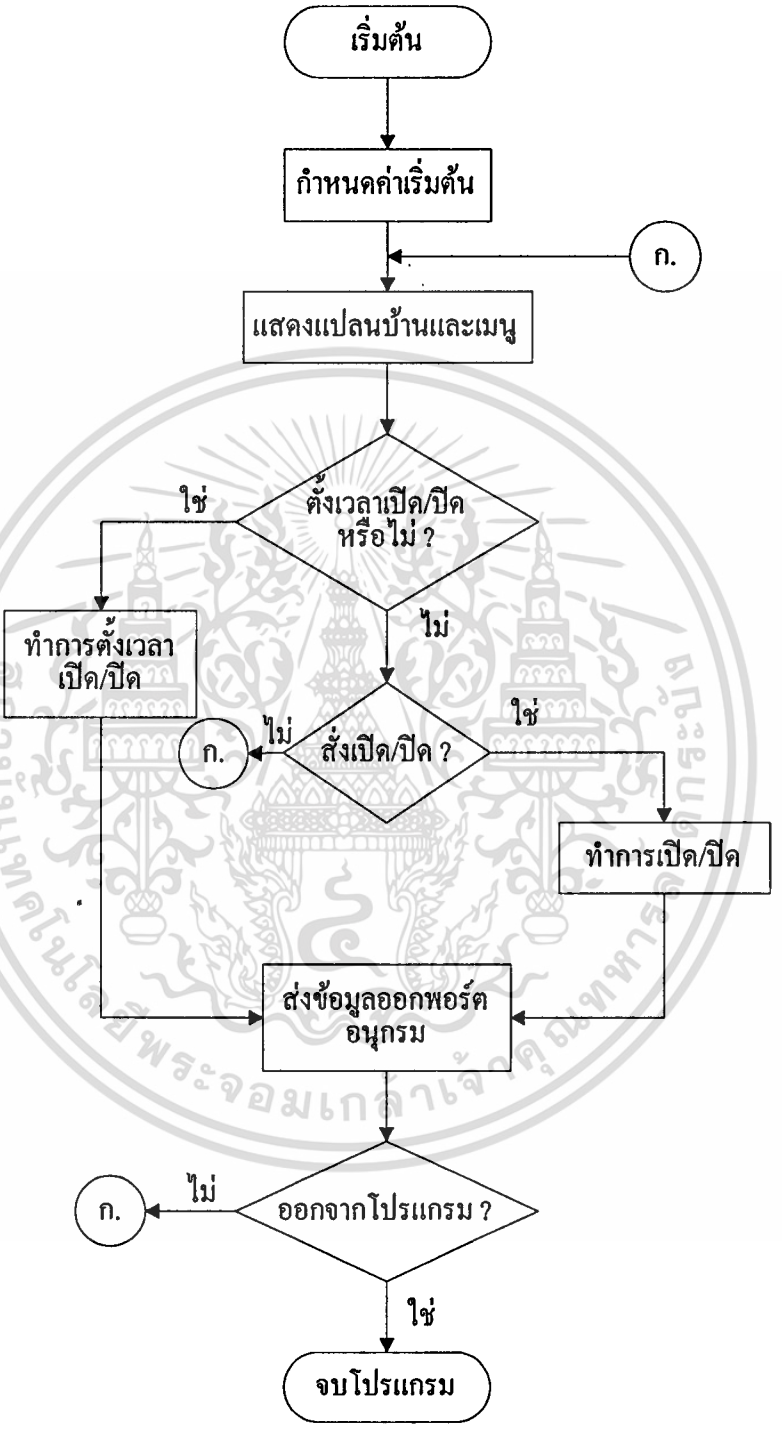
รูปที่ ข.3 ผังการทำงานของโปรแกรมการตั้งเวลาปิด



รูปที่ ข.4 ผังการทำงานของโปรแกรมการสั่งเปิด และปิด



รูปที่ ข.5 ผังการทำงานของโปรแกรมการตั้งเวลา



รูปที่ ข.6 ผังการทำงานของ Visual Basic

ภาคผนวก ค

โปรแกรม

โปรแกรม Visual Basic

```
VERSION 4.00
Begin VB.Form commdat
   BackColor       = &H00000000&
   Caption         = "ตัวอย่างการ ติดต่อกับ RS 232"
   ClientHeight    = 4470
   ClientLeft      = 1560
   ClientTop       = 1500
   ClientWidth     = 7830
   BeginProperty Font
      name         = "MS Sans Serif"
      charset      = 1
      weight       = 700
      size         = 8.25
      underline    = 0   'False
      italic       = 0   'False
      strikethrough = 0   'False
   EndProperty
   Height          = 4875
   Left            = 1500
   LinkTopic       = "Form1"
   ScaleHeight     = 4470
   ScaleWidth      = 7830
   Top             = 1155
   Width           = 7950
   WindowState     = 2  'Maximized
   Begin VB.Timer Timer1
      Enabled      = 0   'False
      Interval     = 100
      Left         = 5160
      Top          = 2880
   End
   Begin MSCommLib.MSComm Comm1
```

```
    Left        = 1080
    Top         = 1680
    _version    = 65536
    _extentx    = 847
    _extenty    = 847
    _stockprops = 0
    cdtimeout   = 0
    commport    = 1
    ctstimeout  = 0
    dsrtimeout  = 0
    dtrenable   = -1 'True
    handshaking = 0
    inbuffersize = 1024
    inputlen    = 0
    interval    = 1000
    nulldiscard = 0 'False
    outbuffersize = 512
    parityreplace = "?"
    rthreshold  = 0
    rtsenable   = 0 'False
    settings    = "9600,N,8,1"
    sthreshold  = 0
  End
  Begin VB.Label Label7
    BorderStyle = 1 'Fixed Single
    Height      = 255
    Left        = 4560
    TabIndex    = 6
    Top         = 960
    Width       = 1935
  End
  Begin VB.Label Label6
    BorderStyle = 1 'Fixed Single
    Height      = 255
```

```
  Left        = 4560
  TabIndex    = 5
  Top         = 480
  Width       = 1935
End
Begin VB.Label Label5
  BackStyle   = 0 'Transparent
  Caption     = "Code out"
  ForeColor   = &H000000FF&
  Height      = 255
  Left        = 3240
  TabIndex    = 4
  Top         = 960
  Width       = 1335
End
Begin VB.Label Label4
  BackStyle   = 0 'Transparent
  Caption     = "Sand  Ascii out"
  ForeColor   = &H000000FF&
  Height      = 255
  Left        = 3240
  TabIndex    = 3
  Top         = 480
  Width       = 1335
End
Begin VB.Label Label3
  BackStyle   = 0 'Transparent
  Caption     = "Code input"
  ForeColor   = &H000000FF&
  Height      = 375
  Left        = 4320
  TabIndex    = 2
  Top         = 2400
  Width       = 975
```

```
End
Begin VB.Image Image5
  Height        =  330
  Left          =  2520
  Picture       =  "COMMDAT.frx":0000
  Top           =  480
  Width         =  360
End
Begin VB.Image Image4
  Height        =  330
  Left          =  1800
  Picture       =  "COMMDAT.frx":018A
  Top           =  480
  Width         =  360
End
Begin VB.Image Image3
  Height        =  330
  Left          =  1080
  Picture       =  "COMMDAT.frx":0314
  Top           =  480
  Width         =  360
End
Begin VB.Label Label2
  BorderStyle   =  1  'Fixed Single
  Height        =  255
  Left          =  5280
  TabIndex      =  1
  Top           =  2400
  Width         =  2415
End
Begin VB.Label Label1
  BackColor     =  &H00000000&
  Caption       =  "Loading Now!"
  BeginProperty Font
```

```
    name        = "MS Sans Serif"
    charset     = 1
    weight      = 700
    size        = 12
    underline   = 0  'False
    italic      = 0  'False
    strikethrough = 0  'False
  EndProperty
  ForeColor     = &H00FF0000&
  Height        = 375
  Left          = 1800
  TabIndex      = 0
  Top           = 2400
  Width         = 2175
End
Begin VB.Image imgLine_G
  Height        = 315
  Left          = 1680
  Picture       = "COMMDAT.frx":049E
  Top           = 3120
  Visible       = 0  'False
  Width         = 2040
End
Begin VB.Image imgLine_R
  Height        = 315
  Left          = 1680
  Picture       = "COMMDAT.frx":0AB4
  Top           = 3120
  Visible       = 0  'False
  Width         = 2040
End
Begin VB.Image ImgLine_main
  Height        = 315
  Left          = 1680
```

```
    Picture      = "COMMDAT.frx":10CA
    Top          = 3120
    Width        = 2040
  End
  Begin VB.Image Image2
    Height       = 480
    Left         = 3720
    Picture      = "COMMDAT.frx":16E0
    Top          = 3000
    Width        = 480
  End
  Begin VB.Image Image1
    Height       = 480
    Left         = 1200
    Picture      = "COMMDAT.frx":19EA
    Top          = 3000
    Width        = 480
  End
End
Attribute VB_Name = "commdat"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
DefInt A-Z
  'Port Communication RS 232
  Const Com1 = 1
  Const Com2 = 2

Dim Datastart As String
Dim Datain1 As Integer



Private Sub command3d2_click()
```

```
Unload Me

End Sub

Private Sub Comm1_Click()

End Sub

Private Sub Form_Load()

  'Set Communication port
  Comm1.CommPort = 1   'Com2
  Comm1.Settings = "9600,n,8,1"
  Comm1.InputLen = 1
  Comm1.PortOpen = True


End Sub

Private Sub Image3_Click()
   commdat.Comm1.output = 1
   commdat.Comm1.output = 1
   label6.Caption = Chr$(65)
   Label7.Caption = "65"
   commdat.Timer1.Enabled = True
End Sub

Private Sub Image4_Click()
   commdat.Comm1.output = Chr$(66)
   label6.Caption = Chr$(66)
   Label7.Caption = "66"
   commdat.Timer1.Enabled = True

End Sub
```

```
Private Sub Image5_Click()
    commdat.Comm1.output = Chr$(67)
    label6.Caption = Chr$(67)
    Label7.Caption = "67"
    commdat.Timer1.Enabled = True

End Sub

Private Sub MSComm1_OnComm()

End Sub

Private Sub imgLine_G_Click()

End Sub

Private Sub Timer1_Timer()
 'Wait Start Valume
  DA$ = Comm1.Input
    If DA$ = "" Then
     Static PickBmp As Integer
     If PickBmp Then
       imgLine_main.Picture = imgLine_R.Picture 'RED Line
     Else
       imgLine_main.Picture = imgLine_G.Picture 'Green Line
     End If
   PickBmp = Not PickBmp
  Else
    commdat.Timer1.Enabled = False
   'Index 2 As Plot From Data In Rs232
    commdat.Label2.Caption = DA$
  End If
End Sub
```

โปรแกรม MCS-51

```
;*******************************************
;                    main  program
; r0 use lcd ( r4 use command  r5 use data of ds1202 )
;*******************************************

    rst   equ     p0.5
    sclk  equ     p0.3
    io    equ     p0.4

      ORG   00H
      ajmp    start

      ORG     23H
      JMP     IN_SER

     ORG   40H
start:  clr     ri
        setb    ea
        setb    es
        mov     tmod,#20h
        mov     th1,#0fdh
        mov     tl1,#0fdh
        mov     scon,#50h
        setb    tr1
;*************************************************************
        mov     p1,#00h
        clr     p0.7
        lcall   clearlcd        ; clear lcd and set start up
        mov     r4,#8eh         ; write protect command
        mov     r5,#00h         ; with zero bits
        lcall   bytewr          ; write to ds1202
        mov     r4,#80h         ; write seconds command
```

```
        lcall   wa_dog
        mov     r5,#00h         ; with value of 00 sec
        lcall   bytewr          ; write to ds1202
        lcall   delaykey
        lcall   name            ; PC OPERATING
next_a: lcall   date            ; date and time
        call    check
        call    wa_dog
        jnb     p0.0,menu
        mov     a,52h
        setb    p1.6
        sjmp    next_a
        menu:   call menu0
exit_0: call    delaykey
        call    name
        jmp     next_a
        ret

        $include "set_on.asm"
        $include "clock.asm"
        $include "date.asm"
        $include "name.asm"
        $include "out.asm"
        $include "lcd.asm"
        $include "delay.asm"
        $include "ds1202.asm"
        $include "count.asm"
        $include "count1.asm"
        $include "wa_dog.asm"
        $include "check.asm"
        $include "direct.asm"
        $include "in_ser.asm"
        $include "menu.asm"
        $include "menu1.asm"
```

```
        $include "menu2.asm"
        $include "menu3.asm"
        $include "menu4.asm"
        end
;***************** set_on.asm **********************
set_on:         call    delaykey
                call    set_on1
                mov     r0,#0cbh
                call    writeinst
                mov     r0,#'1'
                call    writechar
                call    minute_1
                mov     5ah,#60h
                call    coo1
                mov     30h,r3
                call    hour_1
                mov     5ah,#24h
                call    coo1
                mov     31h,r3
                mov     r0,#0cbh
                call    writeinst
                mov     r0,#'2'
                call    writechar
              call      minute_1
                mov     5ah,#60h
                call    coo1
                mov     32h,r3
                call    hour_1
                mov     5ah,#24h
                call    coo1
                mov     33h,r3
                mov     r0,#0cbh
                call    writeinst
                mov     r0,#'3'
```

```
call    writechar
call    minute_1
mov     5ah,#60h
call    coo1
mov     34h,r3
call    hour_1
mov     5ah,#24h
call    coo1
mov     35h,r3
mov     r0,#0cbh
call    writeinst
mov     r0,#'4'
call    writechar
call    minute_1
mov     5ah,#60h
call    coo1
mov     36h,r3
call    hour_1
mov     5ah,#24h
call    coo1
mov     37h,r3
mov     r0,#0cbh
call    writeinst
mov     r0,#'5'
call    writechar
call    minute_1
mov     5ah,#60h
call    coo1
mov     38h,r3
call    hour_1
mov     5ah,#24h
call    coo1
mov     39h,r3
mov     r0,#0cbh
```

```
        call    writeinst
        mov     r0,#'6'
        call    writechar
        call    minute_1
        mov     5ah,#60h
        call    coo1
        mov     3ah,r3
        call    hour_1
        mov     5ah,#24h
        call    coo1
        mov     3bh,r3
        mov     r0,#0cbh
        call    writeinst
        mov     r0,#'7'
        call    writechar
        call    minute_1
        mov     5ah,#60h
        call    coo1
        mov     3ch,r3
        call    hour_1
        mov     5ah,#24h
        call    coo1
        mov     3dh,r3
mov  r0,#0cbh
call writeinst
mov  r0,#'8'
call writechar
call minute_1
mov  5ah,#60h
call coo1
mov  3eh,r3
call hour_1
mov  5ah,#24h
call coo1
```

```
        mov  3fh,r3
        ret

set_off1:    call set_off
        mov  r0,#0cbh
        call writeinst
        mov  r0,#'1'
        call writechar
        call minute_1
        mov  5ah,#60h
        call coo1
        mov  40h,r3
        call hour_1
        mov  5ah,#24h
        call coo1
        mov  41h,r3
        mov  r0,#0cbh
        call writeinst
        mov  r0,#'2'
        call writechar
        call minute_1
        mov  5ah,#60h
        call coo1
        mov  42h,r3
        call hour_1
        mov  5ah,#24h
        call coo1
        mov  43h,r3
        mov  r0,#0cbh
        call writeinst
        mov  r0,#'3'
        call writechar
        call minute_1
        mov  5ah,#60h
```

```
call coo1
mov  44h,r3
call hour_1
mov  5ah,#24h
call coo1
mov  45h,r3
mov  r0,#0cbh
call writeinst
mov  r0,#'4'
call writechar
call minute_1
mov  5ah,#60h
call coo1
mov  46h,r3
call hour_1
mov  5ah,#24h
call coo1
mov  47h,r3
mov  r0,#0cbh
call writeinst
mov  r0,#'5'
call writechar
call minute_1
mov  5ah,#60h
call coo1
mov  48h,r3
call hour_1
mov  5ah,#24h
call coo1
mov  49h,r3
mov  r0,#0cbh
call writeinst
mov  r0,#'6'
call writechar
```

```
call minute_1
mov  5ah,#60h
call coo1
mov  4ah,r3
call hour_1
mov  5ah,#24h
call coo1
mov  4bh,r3
mov  r0,#0cbh
call writeinst
mov  r0,#'7'
call writechar
call minute_1
mov  5ah,#60h
call coo1
mov  4ch,r3
call hour_1
mov  5ah,#24h
call coo1
mov  4dh,r3
mov  r0,#0cbh
call writeinst
mov  r0,#'8'
call writechar
call minute_1
mov  5ah,#60h
call coo1
mov  4eh,r3
call hour_1
mov  5ah,#24h
call coo1
mov  4fh,r3
lcall  name
jmp next_a
```

```
        ret


;************ clock.asm******************
 set_on1: call wa_dog
        mov  r0,#01h
        call writeinst
        mov  r0,#82h
        call writeinst
        mov  r0,#'s'  ;set time on
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#'t'
        call writechar
        mov  r0,#' '
        call writechar
        mov  r0,#'t'
        call writechar
        mov  r0,#'i'
        call writechar
        mov  r0,#'m'
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#' '
        call writechar
        mov  r0,#'o'
        call writechar
        mov  r0,#'n'
        call writechar
        mov  r0,#0c2h
        call writeinst
        mov  r0,#'c'      ; chanal
```

```
        call  writechar
        mov   r0,#'h'
        call  writechar
        mov   r0,#'a'
        call  writechar
        mov   r0,#'n'
        call  writechar
        mov   r0,#'n'
        call  writechar
        mov   r0,#'e'
        call  writechar
        mov   r0,#'l'
        call  writechar
        ret
set_off: mov   r0,#01h
        call  writeinst
        mov   r0,#82h
        call  writeinst
        mov   r0,#'s'   ;set time on
        call  writechar
        mov   r0,#'e'
        call  writechar
        mov   r0,#'t'
        call  writechar
        mov   r0,#' '
        call  writechar
        mov   r0,#'t'
        call  writechar
        mov   r0,#'i'
        call  writechar
        mov   r0,#'m'
        call  writechar
        mov   r0,#'e'
        call  writechar
```

```
        mov  r0,#' '
        call writechar
        mov  r0,#'o'
        call writechar
        mov  r0,#'f'
        call writechar
        mov  r0,#'f'
        call writechar
        mov  r0,#0c2h
        call writeinst
        mov  r0,#'c'      ; channel
        call writechar
        mov  r0,#'h'
        call writechar
        mov  r0,#'a'
        call writechar
        mov  r0,#'n'
        call writechar
        mov  r0,#'n'
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#'l'
        call writechar
        ret
minute_1: mov  r0,#93h
        call writeinst
        mov  r0,#'m'
        call writechar
        mov  r0,#'i'
        call writechar
        mov  r0,#'n'
        call writechar
        mov  r0,#'u'
```

```
        call  writechar
        mov   r0,#'t'
        call  writechar
        mov   r0,#'e'
        call  writechar
        ret
 hour_1: mov   r0,#93h
        call  writeinst
        mov   r0,#'h'
        call  writechar
        mov   r0,#'o'
        call  writechar
        mov   r0,#'u'
        call  writechar
       ret
□

;*************** date.asm *****************
clock1: mov   r0,#01h
      lcall  writeinst
      mov   r0,#81h
      lcall  writeinst
      lcall  wa_dog
      mov   r0,#'s    ;set sec
      lcall  writechar
      mov   r0,#'e'
      lcall  writechar
      mov   r0,#'t'
      lcall  writechar
      mov   r0,#' '
      lcall  writechar
      mov   r0,#'s'
      lcall  writechar
      mov   r0,#'e'
```

```
lcall  writechar
mov   r0,#'c'
lcall  writechar
lcall  wa_dog
mov    5ah,#60h
lcall  count
mov   a,r3
mov   r5,a
mov   r4,#80h
lcall  bytewr
lcall  delaykey
mov   r0,#01h
lcall  writeinst
mov   r0,#81h
lcall  writeinst
lcall  wa_dog
mov   r0,#'s    ;set minute
lcall  writechar
mov   r0,#'e'
lcall  writechar
mov   r0,#'t'
lcall  writechar
mov   r0,#' '
lcall  writechar
mov   r0,#'m'
lcall  writechar
mov   r0,#'i'
lcall  writechar
mov   r0,#'n'
lcall  writechar
mov   r0,#'u'
lcall  writechar
mov   r0,#'t'
lcall  writechar
```

```
mov  r0,#'e'
lcall writechar
lcall wa_dog
mov   5ah,#60h
lcall count
mov  a,r3
mov  r5,a
mov  r4,#82h
lcall bytewr
lcall delaykey
mov  r0,#01h   ;clear   lcd
lcall writeinst
mov  r0,#81h
lcall writeinst
mov  r0,#'s    ;set hour
lcall writechar
mov  r0,#'e'
lcall writechar
mov  r0,#'t'
lcall writechar
mov  r0,#' '
lcall writechar
mov  r0,#'h'
lcall writechar
lcall wa_dog
mov  r0,#'o'
lcall writechar
mov  r0,#'u'
lcall writechar
mov  r0,#'r'
lcall writechar
mov   5ah,#24h
lcall count
mov  a,r3
```

```
mov  r5,a
mov  r4,#84h
lcall bytewr
lcall delaykey
mov  r0,#01h   ;clear   lcd
lcall writeinst
mov  r0,#81h
lcall writeinst
mov  r0,#'s    ;set day
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'d'
call writechar
mov  r0,#'a'
call writechar
call wa_dog
mov  r0,#'y'
call writechar
mov   5ah,#32h
call count
mov  a,r3
mov  r5,a
mov  r4,#86h
call bytewr
call delaykey
mov  r0,#01h   ;clear   lcd
call writeinst
mov  r0,#81h
call writeinst
```

```
 mov  r0,#'s    ;set month
 call writechar
 mov  r0,#'e'
 call writechar
 mov  r0,#'t'
 call writechar
 mov  r0,#' '
 call writechar
 mov  r0,#'m'
 call writechar
 mov  r0,#'o'
 call writechar
 mov  r0,#'n'
 call writechar
 mov  r0,#'t'
 call writechar
 call wa_dog
 mov  r0,#'h'
 call writechar
 mov  5ah,#13h
 call count
 mov  a,r3
 mov  r5,a
 mov  r4,#88h
 call bytewr
 call delaykey
 mov  r0,#01h   ;clear   lcd
 call writeinst
 mov  r0,#81h
 call writeinst
 mov  r0,#'s    ;set year
 call writechar
 mov  r0,#'e'
 call writechar
```

```
        mov  r0,#'t'
        call writechar
        mov  r0,#' '
        call writechar
        mov  r0,#'y'
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#'a'
        call writechar
        call wa_dog
        mov  r0,#'r'
        call writechar
        mov  5ah,#00h
        call count
        mov  a,r3
        mov  r5,a
        mov  r4,#8ch
        call bytewr
        ret
□
;************* name.asm ****************
name:   mov     r0,#01h
        call    writeinst
        mov     r0,#80h
        call    writeinst
    mov  r0,#'C'
    call writechar
    mov  r0,#'o'
    call writechar
    mov  r0,#'m'
    call writechar
    mov  r0,#'p'
    call writechar
```

```
mov  r0,#'u'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'r'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'c'
call writechar
mov  r0,#'o'
call writechar
mov  r0,#'n'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#'r'
call writechar
mov  r0,#'o'
call writechar
mov  r0,#'l'
call writechar
mov  r0,#0c0h
call writeinst
mov  r0,#'e'
call writechar
mov  r0,#'l'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'c'
call writechar
```

```
        mov  r0,#'t'
        call writechar
        mov  r0,#'r'
        call writechar
        mov  r0,#'i'
        call writechar
        mov  r0,#'c'
        call writechar
        mov  r0,#' '
        call writechar
        mov  r0,#'d'
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#'v'
        call writechar
        mov  r0,#'i'
        call writechar
        mov  r0,#'c'
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#'s'
        call writechar
        ret

;*************** out.asm ***************
 outlcd:   mov b,#00h
           mov a,r5
           swap a
           anl a,#0fh
           call bcd
           mov r0,a
           call writechar
```

```
        mov  a,r5
        anl  a,#0fh
        call bcd
        mov  r0,a
        call writechar
        ret
;*********** lcd.asm **************
clearlcd:  clr  p3.3
        clr  p3.4
        clr  p3.5
     acall  delay
     mov    p2,#38h
     setb   p3.5
     clr    p3.5
     acall  delay
     mov    p2,#38h
     setb   p3.5
     clr    p3.5
     acall  delay
     mov    p2,#38h
     setb   p3.5
     clr    p3.5
     acall  delay

display:  mov    r0,#38h
     acall  writeinst
     mov    r0,#0ch
     acall  writeinst
     mov    r0,#01h
     acall  writeinst
     mov    r0,#06h
     acall  writeinst
     ret
```

```
;**************** delay.asm*************
delay: mov    dptr,#0h
loop:  inc    dptr
       mov    r1,dph
       cjne   r1,#0bh,loop
       ret

;*************** ds1202.asm***************
writechar: acall  busywait
       clr   p3.4
       setb  p3.3
       mov   p2,r0
       setb  p3.5
       clr   p3.5
       ret
writeinst: acall  busywait
       clr   p3.4
       clr   p3.3
       mov   p2,r0
       setb  p3.5
       clr   p3.5
       ret
busywait: mov  p2,#0ffh
       clr  p3.3
       setb p3.4
       setb p3.5
busy:    jb   p2.7,busy
       clr  p3.5
       ret
byterd:   clr  rst
       clr  sclk
       setb rst
       lcall delayt ;delay timer
```

```
        mov  b,#8
        clr  c
byterd1:  mov  a,r4
        rrc  a
        mov  r4,a
        mov  io,c
        lcall sclkw
        djnz  b,byterd1
        mov  b,#8
        mov  r5,#0
byterd2:  lcall sclkr
        mov  a,r5
        mov  c,io
        rrc  a
        mov  r5,a
        djnz  b,byterd2
        clr  rst
        lcall delayt
        ret
bytewr:  clr  rst
        clr  sclk
        setb  rst
        lcall delayt
        mov  b,#8
        clr  c
bytewr1:  mov  a,r4
        rrc  a
        mov  r4,a
        mov  io,c
        lcall sclkw
        djnz  b,bytewr1
        mov  b,#8
        clr  c
bytewr2:  mov  a,r5
```

```
        rrc  a
        mov  r5,a
        mov  io,c
        lcall sclkw
        djnz b,bytewr2
        clr  rst
        lcall delayt
        ret
sclkw:  clr  sclk
        lcall delayt
        setb sclk
        lcall delayt
        ret
sclkr:  setb  sclk
        lcall delayt
        clr   sclk
        ret
delayt: mov  r1,#5
        djnz r1,$
        ret
;*************** count.asm **********************
; r3 data of count
count:      mov   a,#00h
            mov   b,a
            call  wa_dog
            mov   r3,a
            call  output
vv:         mov   a,r3
            jb    p0.2,el11
            inc   a
            lcall delaykey
            add   a,b
            da    a
            mov   r3,a
```

```
        lcall   output
el11:     mov    a,r3
        jb     p0.1,el1
        add    a,b
        da     a
        dec    a
        lcall   delaykey
        add    a,b
        da     a
        mov    r3,a
        lcall   output
el1:      jnb    p0.6,next3
        call   wa_dog
        mov    a,r3
        cjne   a,5ah,bbb
        sjmp   count
bbb:      sjmp   vv
next:     lcall   delaykey
        ret
output:     mov    r0,#8dh
        call   writeinst
        mov    a,r3
        swap   a
        anl    a,#0fh
        lcall   bcd
        call   wa_dog
        mov    r0,a
        call   writechar
        mov    a,r3
        anl    a,#0fh
        lcall   bcd
        mov    r0,a
        call   writechar
        ret
```

```
    bcd:    inc  a
            movc a,@a+pc
            ret
            db  '0123456789'
□
;************* count1.asm ********************
; r3 data of count
coo1:       mov   a,#00h
            mov   b,a
            call  wa_dog
            mov   r3,a
            call  output1
 vv1:       mov   a,r3
            jb    p0.2,eld11
            inc   a
            lcall delaykey
            add   a,b
            da    a
            mov   r3,a
            lcall output1
eld11:      mov   a,r3
            jb    p0.1,eld1
            add   a,b
            da    a
            dec   a
            lcall delaykey
            add   a,b
            da    a
            mov   r3,a
            lcall output1
eld1:       jnb   p0.6,next3
            call  wa_dog
            mov   a,r3
            cjne  a,5ah,ddd
```

```
        sjmp   coo1
ddd:      sjmp   vv1
next3:      lcall  delaykey
        mov    r0,#9ah
        call   writeinst
        mov    r0,#' '
        call   writechar
        mov    r0,#' '
        call   writechar
        ret

;*************** wa_dog.asm **************
output1:     mov    r0,#9ah
        call   writeinst
        mov    a,r3
        swap   a
        anl    a,#0fh
        lcall  bcd1
        call   wa_dog
        mov    r0,a
        call   writechar
        mov    a,r3
        anl    a,#0fh
        lcall  bcd1
        mov    r0,a
        call   writechar
        ret
   bcd1:    inc  a
        movc a,@a+pc
        ret
        db  '0123456789'
```

```
;************** check.asm ****************
 check:  mov  r2,51h
        mov  r1,50h      ;check on ch1
        mov  a,r2
        cjne a,31h,ch_2  ;r2= hor
        mov  a,r1        ;31= hor
        cjne a,30h,ch_2  ;r1= min
        setb p1.0        ;30= min
 ch_2:  mov  a,r2              ;check on ch2
        cjne a,33h,ch_3
        mov  a,r1
        cjne a,32h,ch_3
        setb p1.1
 ch_3:  mov  a,r2              ;check on ch3
        cjne a,35h,ch_4
        mov  a,r1
        cjne a,34h,ch_4
        setb p1.2
 ch_4:  mov  a,r2              ;check on ch4
        cjne a,37h,ch_5
        mov  a,r1
        cjne a,36h,ch_5
        setb p1.3
 ch_5:  mov  a,r2              ;check on ch5
        cjne a,39h,ch_6
        mov  a,r1
        cjne a,38h,ch_6
        setb p1.4
 ch_6:  mov  a,r2              ;check on ch6
        cjne a,3bh,ch_7
        mov  a,r1
        cjne a,3ah,ch_7
        setb p1.5
 ch_7:  mov  a,r2              ;check on ch7
```

```
        cjne a,3dh,ch_8
        mov  a,r1
        cjne a,3ch,ch_8
        setb p1.6
ch_8:   mov  a,r2          ;check on ch8
        cjne a,3fh,cha_1
        mov  a,r1
        cjne a,3eh,cha_1
        setb p0.7
cha_1:  mov  a,r2          ;check off ch1
        cjne a,41h,cha_2
        mov  a,r1
        cjne a,40h,cha_2
        clr  p1.0
cha_2:  mov  a,r2          ;check off ch2
        cjne a,43h,cha_3
        mov  a,r1
        cjne a,42h,cha_3
        clr  p1.1
cha_3:  mov  a,r2          ;check off ch3
        cjne a,45h,cha_4
        mov  a,r1
        cjne a,44h,cha_4
        clr  p1.2
cha_4:  mov  a,r2          ;check off ch4
        cjne a,47h,cha_5
        mov  a,r1
        cjne a,46h,ch_5
        clr  p1.3
cha_5:  mov  a,r2          ;check off ch5
        cjne a,49h,cha_6
        mov  a,r1
        cjne a,48h,cha_6
        clr  p1.4
```

```
cha_6:  mov  a,r2          ;check off ch6
        cjne a,4bh,cha_7
        mov  a,r1
        cjne a,4ah,cha_7
        clr  p1.5
cha_7:  mov  a,r2          ;check off ch7
        cjne a,4dh,cha_8
        mov  a,r1
        cjne a,4ch,cha_8
        clr  p1.6
cha_8:  mov  a,r2          ;check off ch8
        cjne a,4fh,cha_9
        mov  a,r1
        cjne a,4eh,cha_9
        clr  p0.7
cha_9:  ret

□

;********** direct.asm ***************
direct: call delaykey
        call wa_dog
        mov  r0,#01h
        call writeinst
        call chanal
        mov  r0,#8ch
        call writeinst
        mov  r0,#'1'
        call writechar
rtt0:   jnb  p0.2,rtt1
        call seton
        setb p1.0
        jb   p0.1,rtt0
        call setoff
```

```
        call delaykey
        call delaykey
        clr  p1.0
rtt1:   call delaykey
        mov  r0,#8ch
        call writeinst
        mov  r0,#'2'
        call writechar
rtt_1:  jnb  p0.2,rtt2
        call seton
        setb p1.1
        jb   p0.1,rtt_1
        call setoff
        clr  p1.1
        call delaykey
        call delaykey
rtt2:   call delaykey
        mov  r0,#8ch
        call writeinst
        mov  r0,#'3'
        call writechar
rtt_2:  jnb  p0.2,rtt3
        call seton
        setb p1.2
        jb   p0.1,rtt_2
        call setoff
        clr  p1.2
        call delaykey
        call delaykey
rtt3:   call delaykey
        mov  r0,#8ch
        call writeinst
        mov  r0,#'4'
        call writechar
```

```
rtt_3:  jnb  p0.2,rtt4
        call seton
        setb p1.3
        jb   p0.1,rtt_3
        call setoff
        clr  p1.3
        call delaykey
        call delaykey
rtt4:   call delaykey
        mov  r0,#8ch
        call writeinst
        mov  r0,#'5'
        call writechar
rtt_4:  jnb  p0.2,rtt5
        call seton
        setb p1.4
        jb   p0.1,rtt_4
        call setoff
        clr  p1.4
        call delaykey
        call delaykey
rtt5:   call delaykey
        mov  r0,#8ch
        call writeinst
        mov  r0,#'6'
        call writechar
rtt_5:  jnb  p0.2,rtt6
        call seton
        setb p1.5
        jb   p0.1,rtt_5
        call setoff
        clr  p1.5
        call delaykey
        call delaykey
```

```
rtt6:  call  delaykey
       mov  r0,#8ch
       call  writeinst
       mov  r0,#'7'
       call  writechar
rtt_6:  jnb  p0.2,rtt7
       call  seton
       setb  p1.6
       jb   p0.1,rtt_6
       call  setoff
       clr  p1.6
       call  delaykey
       call  delaykey
rtt7:  call  delaykey
       mov  r0,#8ch
       call  writeinst
       mov  r0,#'8'
       call  writechar
rtt_7:  jnb  p0.2,rtt8
       call  seton
       setb  p0.7
       jb   p0.1,rtt_7
       call  setoff
       clr  p0.7
       call  delaykey
       call  delaykey
rtt8:   ret

chanal:  mov r0,#83h
       call writeinst
       mov r0,#'c'
       call writechar
       mov r0,#'h'
       call writechar
```

```
        call wa_dog
        mov  r0,#'a'
        call writechar
        mov  r0,#'n'
        call writechar
        mov  r0,#'n'
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#'l'
        call writechar
seton:  call wa_dog
        mov  r0,#0c6h
        call writeinst
        mov  r0,#'O'
        call writechar
        mov  r0,#'N'
        call writechar
        mov  r0,#' '
        call writechar
        ret
setoff: mov  r0,#0c6h
        call writeinst
        mov  r0,#'O'
        call writechar
        mov  r0,#'F'
        call writechar
        ret
;************ in_ser.asm *****************
in_ser: call  wa_dog
        jnb  ri,in_ser
        mov  a,sbuf
        mov  52h,a   ;address
        clr  ri
```

```
dfg:    call wa_dog
        jnb ri,dfg
        mov a,sbuf
        mov 53h,a   ;hor
        clr ri
ghj:    call wa_dog
        jnb ri,ghj
        mov a,sbuf
        mov 54h,a   ;min
        clr ri
        call wa_dog
        mov a,52h
        cjne a,#01h,zz_1 ;on
        mov 31h,53h ;hor
        mov 30h,54h ;mim
        ajmp zzz_end
zz_1:   mov a,52h
        cjne a,#02,zz_2 ;on
        mov 33h,53h ;hor
        mov 32h,54h ;min
        ajmp zzz_end
zz_2:   mov a,52h
        cjne a,#03,zz_3 ;on
        mov 35h,53h ;hor
        mov 34h,54h ;min
        ajmp zzz_end
zz_3:   mov a,52h
        cjne a,#04,zz_4 ;on
        mov 37h,53h ;hor
        mov 36h,54h ;min
        ajmp zzz_end
zz_4:   mov a,52h
        cjne a,#05,zz_5 ;on
        mov 39h,53h ;hor
```

```
        mov 38h,54h  ;min
        ajmp zzz_end
zz_5:   mov a,52h
        cjne a,#06,zz_6 ;on
        mov 3bh,53h  ;hor
        mov 3ah,54h  ;min
        ajmp zzz_end
zz_6:   mov a,52h
        cjne a,#07,zz_7 ;on
        mov 3dh,53h  ;hor
        mov 3ch,54h  ;min
        ajmp zzz_end
zz_7:   mov a,52h
        cjne a,#08,zzz_1 ;on
        mov 3fh,53h  ;hor
        mov 3eh,54h  ;min
        ajmp zzz_end
zzz_1:  mov a,52h
        cjne a,#09,zzz_2 ;off
        mov 41h,53h  ;hor
        mov 40h,54h  ;min
        ajmp zzz_end
zzz_2:  mov a,52h
        cjne a,#10,zzz_3 ;off
        mov 43h,53h  ;hor
        mov 42h,54h  ;min
        ajmp zzz_end
zzz_3:  mov a,52h
        cjne a,#11,zzz_4 ;off
        mov 45h,53h  ;hor
        mov 44h,54h  ;min
        ajmp zzz_end
zzz_4:  mov a,52h
        cjne a,#12,zzz_5 ;off
```

```
        mov 47h,53h  ;hor
        mov 46h,54h  ;min
        ajmp zzz_end
zzz_5:  mov a,52h
        cjne a,#13h,zzz_6 ;off
        mov 49h,53h  ;hor
        mov 48h,54h  ;min
        ajmp zzz_end
zzz_6:  mov a,52h
        cjne a,#14,zzz_7 ;off
        mov 4bh,53h  ;hor
        mov 4ah,54h  ;min
        ajmp zzz_end
zzz_7:  mov a,52h
        cjne a,#15h,zzz_8 ;off
        mov 4dh,53h  ;hor
        mov 4ch,54h  ;min
        ajmp zzz_end
zzz_8:  mov a,52h
        cjne a,#16,zzz_9 ;off
        mov 4fh,53h  ;hor
        mov 4eh,54h  ;min
        ajmp zzz_end
zzz_9:  mov a,52h
        cjne a,#11h,zzz_10 ;on direct ch1
        setb p1.0
        ajmp zzz_end
zzz_10: mov a,52h
        cjne a,#12h,zzz_11 ;off direct ch1
        clr p1.0
        ajmp zzz_end
zzz_11: mov a,52h
        cjne a,#13h,zzz_12 ;on direct ch2
        setb p1.1
```

```
        ajmp  zzz_end
zzz_12:   mov  a,52h
        cjne a,#14h,zzz_13 ;off direct ch2
        clr  p1.1
        ajmp  zzz_end
zzz_13:   mov  a,52h
        cjne a,#15h,zzz_14 ;on direct ch3
        setb  p1.2
        ajmp  zzz_end
zzz_14:   mov  a,52h
        cjne a,#16h,zzz_15 ;off direct ch3
        clr  p1.2
        ajmp  zzz_end
zzz_15:   mov  a,52h
        cjne a,#17h,zzz_16 ;on direct ch4
        setb p1.3
        ajmp  zzz_end
zzz_16:   mov  a,52h
        cjne a,#18h,zzz_17 ;off direct ch4
        clr  p1.3
        ajmp  zzz_end
zzz_17:   mov  a,52h
        cjne a,#19h,zzz_18 ;on direct ch5
        setb p1.4
        ajmp  zzz_end
zzz_18:   mov  a,52h
        cjne a,#1ah,zzz_19 ;off direct ch5
        clr  p1.4
        ajmp  zzz_end
zzz_19:   mov  a,52h
        cjne a,#1bh,zzz_20 ;on direct ch6
        setb p1.5
        ajmp  zzz_end
zzz_20:   mov  a,52h
```

```
        cjne a,#1ch,zzz_21 ;off direct ch6
        clr  p1.5
        ajmp  zzz_end
zzz_21:   mov  a,52h
        cjne a,#1dh,zzz_22 ;on direct ch7
        setb p1.6
        ajmp  zzz_end
zzz_22:   mov  a,52h
        cjne a,#1eh,zzz_23 ;off direct ch7
        clr  p1.6
        ajmp  zzz_end
zzz_23:   mov  a,52h
        cjne a,#1eh,zzz_24 ;on direct ch8
        setb p0.7
        ajmp  zzz_end
zzz_24:   mov  a,52h
        cjne  a,#1fh,zzz_25 ;off direct ch8
        clr p0.7
        ajmp  zzz_end
zzz_25:   mov  a,52h
        cjne  a,20h,zzz_end
        mov  r4,#82h
        mov  r5,53h
        mov  r4,#84h
        mov  r5,54h
 zzz_end: reti

;********** menu.asm *****************
 menu0:   call  menu1
  sw1:   call  delaykey  ;set clock
        jnb  p0.1,sw2
        jnb  p0.2,sw4
        call  wa_dog
        call  delaykey
```

```
        jnb  p0.0,sw6
        jb   p0.6,sw1
        call clock1
sw2:    call menu2
sw_2:   call delaykey
        jnb  p0.1,sw3  ;set direct
        jnb  p0.2,menu0
        call delaykey
        call wa_dog
        jnb  p0.0,sw6
        jb   p0.6,sw_2
        call direct
sw3:    call menu3
sw_3:   call delaykey
        jnb  p0.1,sw4
        jnb  p0.2,sw2
        call wa_dog
        call delaykey
        jnb  p0.0,sw6
        jb   p0.6,sw_3
        call set_on
sw4:    call menu4
        call delaykey
sw_4:   jnb  p0.1,sw5
        jnb  p0.2,sw3
        call delaykey
        call wa_dog
        jnb  p0.0,sw6
        jb   p0.6,sw_4
        call set_off1
 sw5:   sjmp  menu0
 sw6:   call  exit_0
        ret
```

□

```
;*************** menu1.asm ******************
menu1: mov  r0,#01h
       call writeinst
       mov  r0,#84h
       call writeinst
       mov  r0,#7eh
       call writechar
       mov  r0,#'s'
       call writechar
       mov  r0,#'e'
       call writechar
       mov  r0,#'t'
       call writechar
       mov  r0,#' '
       call writechar
       mov  r0,#'c'
       call writechar
       call wa_dog
       mov  r0,#'l'
       call writechar
       mov  r0,#'o'
       call writechar
       mov  r0,#'c'
       call writechar
       mov  r0,#'k'
       call writechar
       mov  r0,#0c5h
       call writeinst
       mov  r0,#'s'
       call writechar
       mov  r0,#'e'
       call writechar
       mov  r0,#'t'
       call writechar
```

```
mov  r0,#' '
call writechar
mov  r0,#'d'
call writechar
mov  r0,#'i'
call writechar
mov  r0,#'r'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'c'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#95h
call writeinst
mov  r0,#'s'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'o'
call writechar
mov  r0,#'n'
call writechar
mov  r0,#0d5h
call writeinst
mov  r0,#'s'
call writechar
mov  r0,#'e'
call writechar
```

```
        mov  r0,#'t'
        call writechar
        mov  r0,#' '
        call writechar
        mov  r0,#'o'
        call writechar
        mov  r0,#'f'
        call writechar
        mov  r0,#'f'
        call writechar
        ret
□
;*************** menu2.asm ******************
menu2: mov  r0,#01h
       call writeinst
       mov  r0,#85h
       call writeinst
       mov  r0,#'s'
       call writechar
       mov  r0,#'e'
       call writechar
       mov  r0,#'t'
       call writechar
       mov  r0,#' '
       call writechar
       mov  r0,#'c'
       call writechar
       call wa_dog
       mov  r0,#'l'
       call writechar
       mov  r0,#'o'
       call writechar
       mov  r0,#'c'
       call writechar
```

```
mov  r0,#'k'
call writechar
mov  r0,#0c4h
call writeinst
mov  r0,#7eh
call writechar
mov  r0,#'s'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'d'
call writechar
mov  r0,#'i'
call writechar
mov  r0,#'r'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'c'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#95h
call writeinst
mov  r0,#'s'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
```

```
        mov  r0,#' '
        call writechar
        mov  r0,#'o'
        call writechar
        mov  r0,#'n'
        call writechar
        mov  r0,#0d5h
        call writeinst
        mov  r0,#'s'
        call writechar
        mov  r0,#'e'
        call writechar
        mov  r0,#'t'
        call writechar
        mov  r0,#' '
        call writechar
        mov  r0,#'o'
        call writechar
        mov  r0,#'f'
        call writechar
        mov  r0,#'f'
        call writechar
        ret
□
;*************** menu3.asm ******************
menu3: mov  r0,#01h
       call writeinst
       mov  r0,#85h
       call writeinst
       mov  r0,#'s'
       call writechar
       mov  r0,#'e'
       call writechar
       mov  r0,#'t'
```

```
call writechar
mov r0,#' '
call writechar
mov r0,#'c'
call writechar
mov r0,#'l'
call writechar
mov r0,#'o'
call writechar
mov r0,#'c'
call writechar
call wa_dog
mov r0,#'k'
call writechar
mov r0,#0c5h
call writeinst
mov r0,#'s'
call writechar
mov r0,#'e'
call writechar
mov r0,#'t'
call writechar
mov r0,#' '
call writechar
mov r0,#'d'
call writechar
mov r0,#'i'
call writechar
mov r0,#'r'
call writechar
mov r0,#'e'
call writechar
mov r0,#'c'
call writechar
```

```
mov  r0,#'t'
call writechar
mov  r0,#94h
call writeinst
 mov  r0,#7eh
call writechar
mov  r0,#'s'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'o'
call writechar
mov  r0,#'n'
call writechar
mov  r0,#0d5h
call writeinst
mov  r0,#'s'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'o'
call writechar
mov  r0,#'f'
call writechar
mov  r0,#'f'
call writechar
```

```
        ret
□
;************** menu4.asm ********************
menu4: mov  r0,#01h
        call  writeinst
        mov  r0,#85h
        call  writeinst
        mov  r0,#'s'
        call  writechar
        mov  r0,#'e'
        call  writechar
        mov  r0,#'t'
        call  writechar
        mov  r0,#' '
        call  writechar
        mov  r0,#'c'
        call  writechar
        mov  r0,#'l'
        call  writechar
        mov  r0,#'o'
        call  writechar
        mov  r0,#'c'
        call  writechar
        call  wa_dog
        mov  r0,#'k'
        call  writechar
        mov  r0,#0c5h
        call  writeinst
        mov  r0,#'s'
        call  writechar
        mov  r0,#'e'
        call  writechar
        mov  r0,#'t'
        call  writechar
```

```
mov  r0,#' '
call writechar
mov  r0,#'d'
call writechar
mov  r0,#'i'
call writechar
mov  r0,#'r'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'c'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#95h
call writeinst
mov  r0,#'s'
call writechar
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'o'
call writechar
mov  r0,#'n'
call writechar
mov  r0,#0d4h
call writeinst
mov  r0,#7eh
call writechar
mov  r0,#'s'
call writechar
```

```
mov  r0,#'e'
call writechar
mov  r0,#'t'
call writechar
mov  r0,#' '
call writechar
mov  r0,#'o'
call writechar
mov  r0,#'f'
call writechar
mov  r0,#'f'
call writechar
ret
```

ภาคผนวก ง

คู่มือการใช้งาน

คู่มือการใช้งาน

(User Manual)

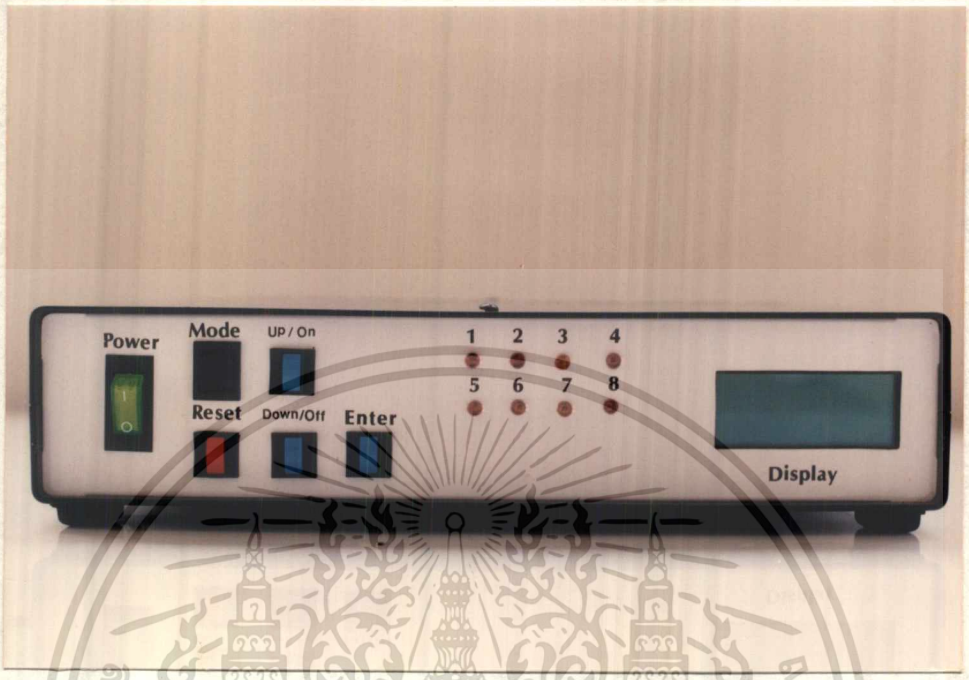
คู่มือการใช้งานนี้จะกล่าวถึงการติดตั้งระบบต่างๆ เพื่อใช้งาน การควบคุมการทำงานผ่านทางไมโครคอนโทรลเลอร์ และการควบคุมการทำงานผ่านทางคอมพิวเตอร์

ในการควบคุมการทำงานผ่านทางไมโครคอนโทรลเลอร์จะอธิบายวิธีการตั้งเวลา การตั้งเปิด-ปิดโดยตรง การตั้งเวลาเปิด และ การตั้งเวลาปิด

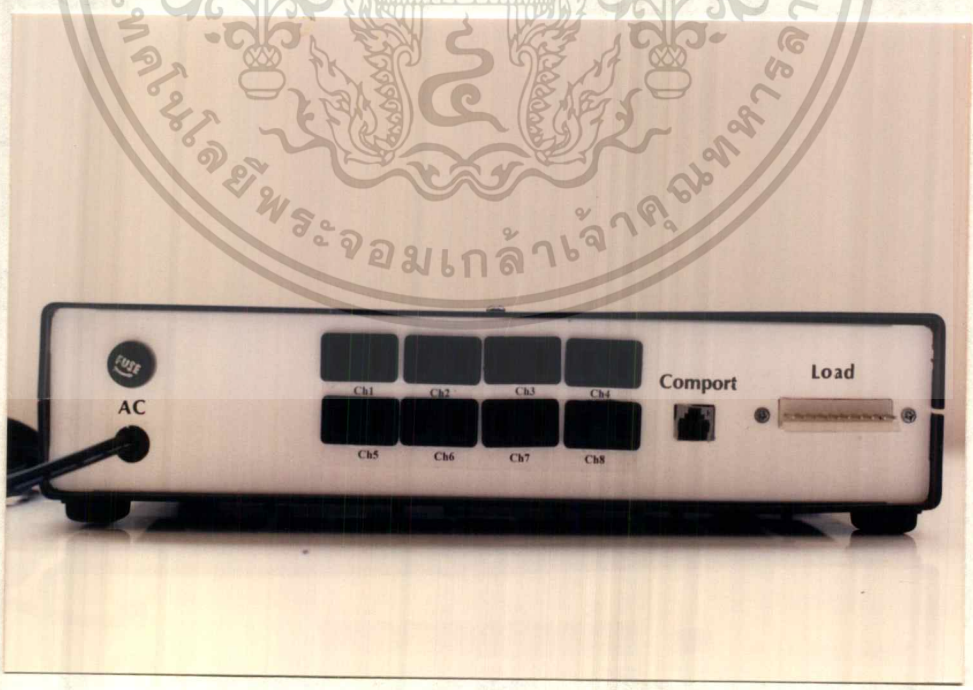
ในการควบคุมการทำงานผ่านทางคอมพิวเตอร์จะอธิบายการใช้งานโปรแกรมสำหรับวินโดวส์ 95 เท่านั้นโดยจะกล่าวตั้งแต่ความต้องการของระบบ การติดตั้งโปรแกรมบนวินโดวส์ 95 การตั้งเปิด-ปิดโดยตรง การตั้งเวลาเปิด การตั้งเวลาปิด จนกระทั่งการออกจากโปรแกรม

1. การติดตั้งระบบ

รูปที่ ง.1 ตัวเครื่องไมโครคอนโทรลเลอร์



รูปที่ ง.2 หน้าปัทม์เครื่องไมโครคอนโทรลเลอร์



รูปที่ ง.3ด้านหลังเครื่องไมโครคอนโทรลเลอร์

1.1 ขั้นตอนการติดตั้ง

1.เชื่อมต่อสาย RS-232C ระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ โดยต่อคอนเนคเตอร์ที่ COM1 หรือ COM2 และเสียบแจ็คที่คอมพอร์ต (Comport) ซึ่งอยู่ด้านหลังของเครื่องไมโครคอนโทรลเลอร์ ดูรูปที่ ง.3 ประกอบ

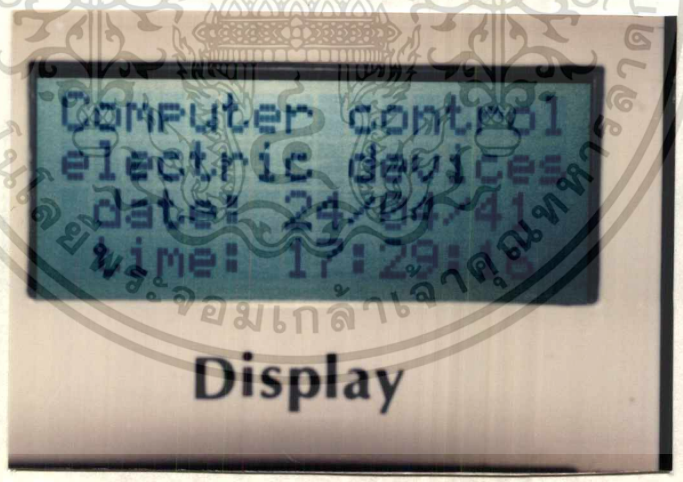
2.ต่อโหลดเข้าในช่องสำหรับต่อโหลดที่อยู่ด้านหลังของเครื่องไมโครคอนโทรล-เลอร์ ดูรูปที่ ง.3 ประกอบ

3.เสียบปลั๊กเครื่องไมโครคอนโทรลเลอร์ (เมื่อเสียบปลั๊กแล้ว ให้ระวังแรงดัน 220 โวลต์ ที่ช่องสำหรับต่อตัวภาระด้วย)

2. การควบคุมการทำงานผ่านทางไมโครคอนโทรลเลอร์

1. ติดตั้งระบบ

2. เปิดสวิตช์ POWER จอแอลซีดี จะแสดงวันที่ และเวลา ดังรูปที่ ง.4 ซึ่งถือว่าเป็นหน้าจอหลักแสดงว่าเครื่องพร้อมที่จะทำงาน



รูปที่ ง.4 การแสดงผลของแอลซีดี เมื่อเริ่มเปิดเครื่อง

3. กดสวิตช์ MODE เข้าสู่เมนูหลัก

รูปที่ ง.5 เมนูหลัก

4. เลื่อนลูกศรไปยังฟังก์ชั่นที่ต้องการ โดยการกดสวิตช์ UP หรือ DOWN

5. กดสวิตช์ ENTER เข้าสู่ฟังก์ชั่นที่เลือกไว้ และการตั้งการทำงานของฟังก์ชั่นนั้นๆ

2.1 การตั้งเวลา วัน เดือน ปี (Set clock)

1. เลื่อนลูกศรในเมนูหลักมายังตำแหน่ง Set clock

2. กดสวิตช์ ENTER เข้าสู่การทำงานของการตั้งเวลา วัน เดือน ปี โดยจะเรียงจากการตั้งวินาที นาที ชั่วโมง วัน เดือน และปี ตามลำดับ

3. กดสวิตช์ UP เพื่อตั้งค่าตามที่ต้องการ

4. กดสวิตช์ ENTER เมื่อตั้งค่าเรียบร้อยแล้ว และโปรแกรมจะเลื่อนไปทำงานในส่วนอื่นต่อไป โดยอัตโนมัติ โดยจะเรียงจากการตั้งวินาที นาที ชั่วโมง วัน เดือน และปี ตามลำดับเช่นกัน

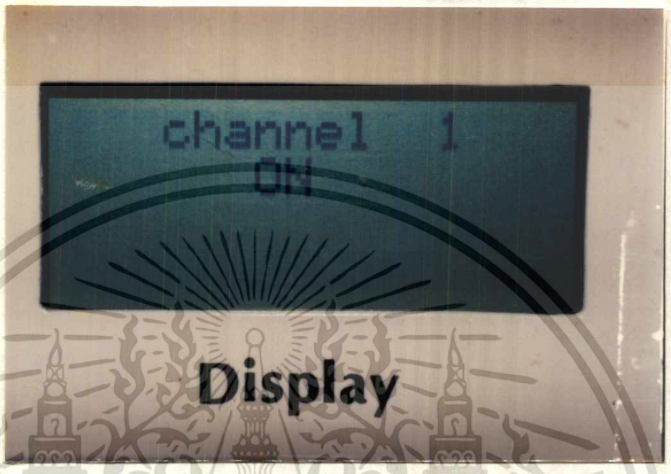
5. ทำซ้ำข้อ 3 และข้อ 4

6. เมื่อทำการตั้งค่าปีเรียบร้อยแล้ว และเป็นขั้นตอนสุดท้าย ในการตั้งเวลา วัน เดือน ปี กดสวิตช์ ENTER จะกลับเข้าสู่เมนูหลัก

2.2 การเปิด-ปิดโดยตรง (Set direct)

1. เลื่อนลูกศรในเมนูหลักมายังตำแหน่ง Set direct

2. กดสวิตช์ ENTER เข้าสู่การทำงานของการตั้งเปิด-ปิดโดยตรงโดยจะเรียงจากช่องที่ 1 จนถึงช่องที่ 8 ตามลำดับ



รูปที่ ง.6 การแสดงผลของแอลซีดี เมื่อเข้าสู่การตั้งเปิด-ปิดโดยตรง

3. กดสวิตช์ UP เมื่อต้องการเปิด หรือกดสวิตช์ DOWN เมื่อต้องการปิด จากนั้นโปรแกรมจะเลื่อนไปช่องต่อไป โดยอัตโนมัติ

4. ทำซ้ำข้อ 2 และข้อ 3

5. ในช่องที่ 8 หลังจากตั้งการเปิดหรือปิดเรียบร้อยแล้ว โปรแกรมจะกลับเข้าสู่เมนูหลัก

6. หากต้องการเลือกฟังก์ชั่นอื่นๆ ให้เลื่อนลูกศรไปยังฟังก์ชั่นที่ต้องการ แล้วกดสวิตช์ ENTER หากต้องการกลับสู่หน้าจอหลัก ให้กดสวิตช์ MODE

2.3 การตั้งเวลาเปิด (Set on)

1. เลื่อนลูกศรในเมนูหลักมายังตำแหน่ง Set on

2. กดสวิตช์ ENTER เข้าสู่การทำงานของการตั้งเวลาเปิดโดยจะเรียงจากช่องที่ 1 จนถึงช่องที่ 8 ตามลำดับ

รูปที่ ง.7 การแสดงผลของแอลซีดี เมื่อเข้าสู่การตั้งเวลาเปิด

3. ตั้งเวลาที่ต้องการเปิด โดยการกดสวิตช์ UP เพื่อตั้งนาที

4. กดสวิตช์ ENTER เมื่อตั้งนาทีเรียบร้อยแล้ว โปรแกรมจะเข้าสู่การตั้งชั่วโมง

5. กดสวิตช์ UP เพื่อตั้งชั่วโมง

6. กดสวิตช์ ENTER เมื่อตั้งชั่วโมงเรียบร้อยแล้ว โปรแกรมจะเปลี่ยนไปช่องถัดไป

7. ทำซ้ำข้อ 3 , 4 , 5 และ 6 ในช่องถัดไป

8. ในช่องที่ 8 หลังจากตั้งชั่วโมง และกดสวิตช์ ENTER โปรแกรมจะกลับข้าสู่เมนูหลัก

9. หากต้องการเลือกฟังก์ชั่นอื่นๆ ให้เลื่อนลูกศรไปยังฟังก์ชั่นที่ต้องการ แล้วกด สวิตช์ ENTER หากต้องการกลับสู่หน้าจอหลัก ให้กดสวิตช์ MODE

2.4 การตั้งเวลาปิด (Set off)

1. เลื่อนลูกศรในเมนูหลักมายังตำแหน่ง Set off

2. กดสวิตช์ ENTER เข้าสู่การทำงานของการตั้งเวลาเปิดโดยจะเรียงจากช่องที่ 1 จนถึงช่องที่ 8 ตามลำดับ

รูปที่ ง.8 การแสดงผลของแอลซีดี เมื่อเข้าสู่การตั้งเวลาปิด

3. ตั้งเวลาที่ต้องการปิด โดยการกดสวิตช์ UP เพื่อตั้งนาที

4. กดสวิตช์ ENTER เมื่อตั้งนาทีเรียบร้อยแล้ว โปรแกรมจะเข้าสู่การตั้งชั่วโมง

5. กดสวิตช์ UP เพื่อตั้งชั่วโมง

6. กดสวิตช์ ENTER เมื่อตั้งชั่วโมงเรียบร้อยแล้ว โปรแกรมจะเปลี่ยนไปช่องถัดไป

7. ทำซ้ำข้อ 3 , 4 , 5 และ 6 ในช่องถัดไป

8. ในช่องที่ 8 หลังจากตั้งชั่วโมง และกดสวิตช์ ENTER โปรแกรมจะกลับข้าสู่หน้าจอหลัก

9. หากต้องการเลือกฟังก์ชั่นอื่นๆ ให้กดสวิตช์ MODE เพื่อเข้าเมนูหลัก

3. การควบคุมการทำงานผ่านทางคอมพิวเตอร์

สำหรับโปรแกรมที่ใช้ในการควบคุมการทำงานผ่านทางคอมพิวเตอร์เป็นโปรแกรมวิชวลเบสิคเวอร์ชั่น 4.0 สำหรับวินโดวส์ และก่อนการติดตั้งโปรแกรมควรตรวจสอบระบบว่าสามารถรองรับโปรแกรมได้หรือไม่

3.1 ความต้องการของระบบ

1. เครื่องคอมพิวเตอร์ที่ใช้ CPU 386 หรือสูงกว่า

2. RAM 8 MB หรือสูงกว่า

3. เนื้อที่ฮาร์ดดิสก์ 2MB

4. ดิสก์ไดร์ฟขนาด 3.5 นิ้ว 1 ตัว

5. Windows 3.1 (สนับสนุน Windows95)

6. เม้าส์ หรืออุปกรณ์ชี้อื่นๆ ที่เข้ากันได้

3.2 วิธีเข้าสู่โปรแกรม

1. ติดตั้งระบบ

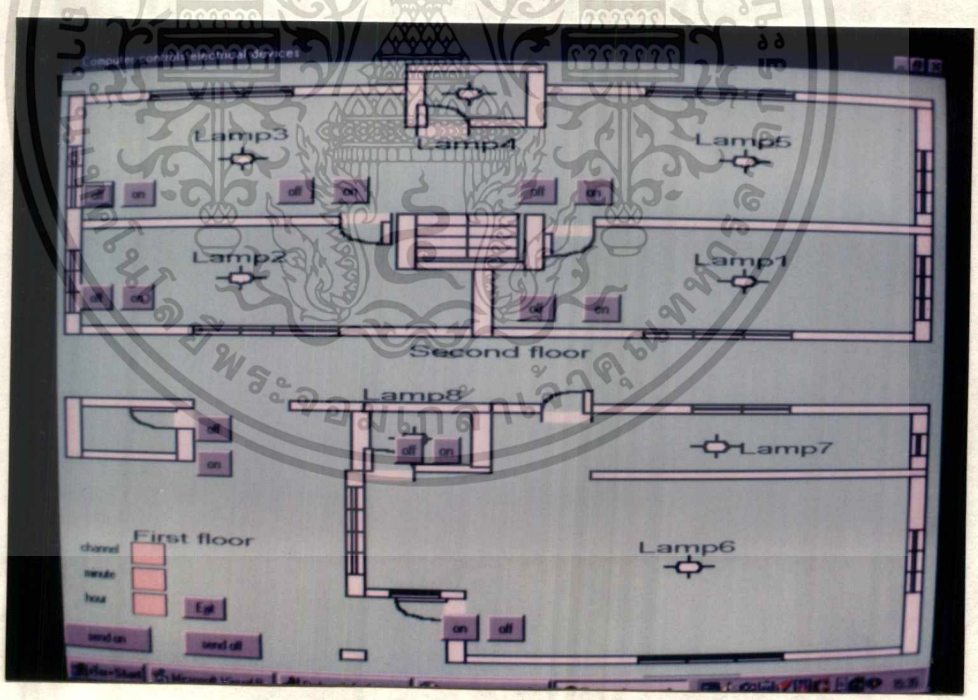
2. เปิดเครื่อง และคอมพิวเตอร์

3. เข้าสู่โปรแกรมวินโดวส์

4. คลิกเม้าส์ที่ Start Menu ชี้ที่ Programs แล้วเลือกที่ Visual Basic 4.0

5. คลิกเม้าส์ที่ Microsoft Visual Basic 4.0-32-bit และเปิดแฟ้มข้อมูลชื่อ Project1.Mak

6. กด F5 เพื่อเริ่มการทำงานโปรแกรม และจะปรากฏผังของแบบจำลอง สวิตช์ควบคุมการเปิด-ปิด และตัวตั้งเวลาเพื่อเปิด-ปิด

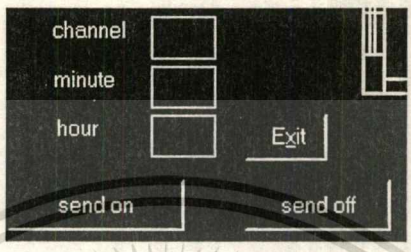


รูปที่ ง.9 ผังของแบบจำลอง และตำแหน่งของสวิตช์ควบคุม

3.3 การเปิด-ปิดโดยตรง

คลิกเม้าส์ที่ปุ่ม ON เมื่อต้องการเปิด หรือคลิกเม้าส์ที่ปุ่ม OFF เมื่อต้องการปิด

รูปที่ ง.10 ลักษณะสวิตช์เปิด-ปิดในโปรแกรมวิชวลเบสิค



รูปที่ ง.11 ลักษณะของส่วนที่ใช้ตั้งเวลาเปิด และปิดในโปรแกรมวิชวลเบสิค

3.4 การตั้งเวลาเปิด

1.คลิ๊กเม้าส์ในช่องของ Channel เพื่อเลือกช่องที่ต้องการเปิด และพิมพ์เลขประจำช่องลงไป

2. คลิ๊กเม้าส์ในช่องของ Minute เพื่อตั้งนาที

3. คลิ๊กเม้าส์ในช่องของ Hour เพื่อตั้งชั่วโมง

4. คลิ๊กเม้าส์ที่ปุ่ม Send on

3.5 การตั้งเวลาปิด

1. คลิ๊กเม้าส์ในช่องของ Channel เพื่อเลือกช่องที่ต้องการเปิด และพิมพ์เลขประจำช่องลงไป

2. คลิ๊กเม้าส์ในช่องของ Minute เพื่อตั้งนาที

3. คลิ๊กเม้าส์ในช่องของ Hour เพื่อตั้งชั่วโมง

4. คลิ๊กเม้าส์ที่ปุ่ม Send off

3.6 การออกจากโปรแกรม

เมื่อต้องการออกจากโปรแกรม คลิ๊กเม้าส์ที่ปุ่ม Exit

ภาคผนวก จ

รายการแสดงคุณสมบัติของอุปกรณ์

รายการแสดงคุณสมบัติของอุปกรณ์

DALLAS SEMICONDUCTOR

DS1202, DS1202S
Serial Timekeeping Chip

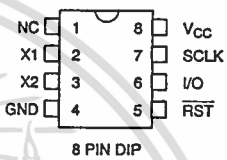
FEATURES

- Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation
- 24 x 8 RAM for scratchpad data storage
- Serial I/O for minimum pin count
- 2.0-5.5 volt full operation
- Uses less than 300 nA at 2 volts
- Single-byte or multiple-byte (burst mode) data transfer for read or write of clock or RAM data
- 8-pin DIP or optional 16-pin SOIC for surface mount
- Simple 3-wire interface
- TTL-compatible (V_{CC} = 5V)
- Optional industrial temperature range -40°C to +85°C

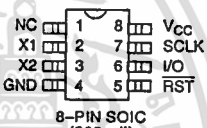
ORDERING INFORMATION

DS1202	8–pin DIP
DS1202S	16–pin SOIC
DS1202S8	8–pin SOIC

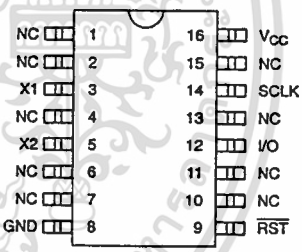
PIN ASSIGNMENT



8 PIN DIP



8–PIN SOIC (208 mil)



16–PIN SOIC

PIN DESCRIPTION

NC	–	No Connection
X1, X2	–	32.768 KHz Crystal Input
GND	–	Ground
$\overline{RST}$	–	Reset
I/O	–	Data Input/Output
SCLK	–	Serial Clock
V_{CC}	–	Power Supply Pin

DESCRIPTION

The DS1202 Serial Timekeeping Chip contains a real time clock/calendar and 24 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real time clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Interfacing the DS1202 with a microprocessor is simplified by using synchronous serial communication. Only three wires are required to communicate with the clock/RAM: (1) $\overline{RST}$ (Reset), (2) I/O (Data line), and (3) SCLK (Serial clock). Data can be transferred to and from the clock/

RAM one byte at a time or in a burst of up to 24 bytes. The DS1202 is designed to operate on very low power and retain data and clock information on less than 1 microwatt.

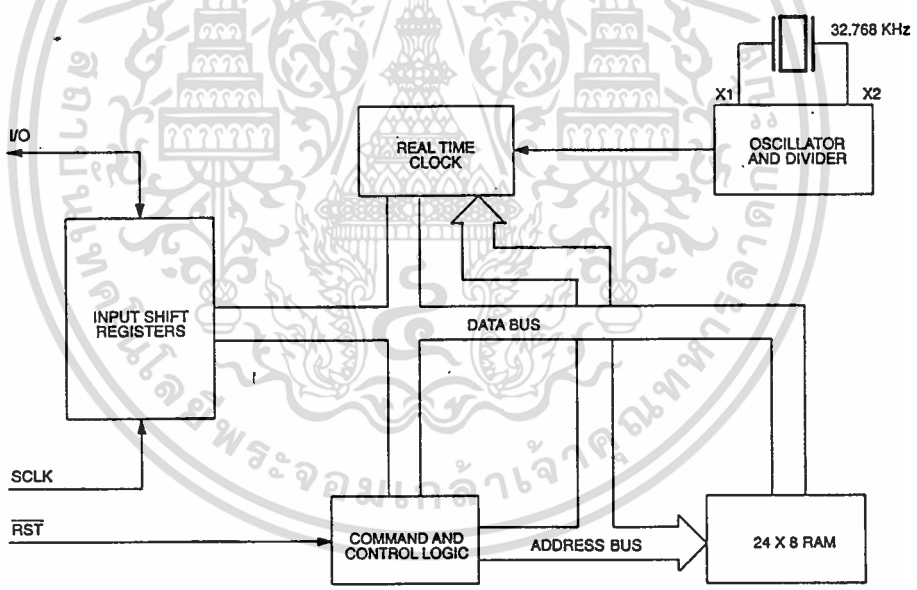
OPERATION

The main elements of the Serial Timekeeper are shown in Figure 1: shift register, control logic, oscillator, real time clock, and RAM. To initiate any transfer of data, $\overline{RST}$ is taken high and eight bits are loaded into the shift register providing both address and command information. Data is serially input on the rising edge of the SCLK. The first eight bits specify which of 32 bytes will be accessed, whether a read or write cycle will take place, and whether a byte or burst mode transfer is to occur. After the first eight clock cycles have occurred which load the command word into the shift register, additional clocks will output data for a read or input data for a write. The number of clock pulses equals eight plus eight for byte mode or eight plus up to 192 for burst mode.

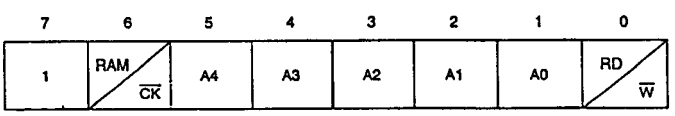
COMMAND BYTE

The command byte is shown in Figure 2. Each data transfer is initiated by a command byte. The MSB (Bit 7) must be a logic 1. If it is zero, further action will be terminated. Bit 6 specifies clock/calendar data if logic 0 or RAM data if logic 1. Bits one through five specify the designated registers to be input or output, and the LSB (Bit 0) specifies a write operation (input) if logic 0 or read operation (output) if logic 1. The command byte is always input starting with the LSB (bit 0).

DS1202 BLOCK DIAGRAM Figure 1



ADDRESS/COMMAND BYTE Figure 2



RESET AND CLOCK CONTROL

All data transfers are initiated by driving the $\overline{RST}$ input high. The $\overline{RST}$ input serves two functions. First, $\overline{RST}$ turns on the control logic which allows access to the shift register for the address/command sequence. Second, the $\overline{RST}$ signal provides a method of terminating either single byte or multiple byte data transfer. A clock cycle is a sequence of a falling edge followed by a rising edge. For data inputs, data must be valid during the rising edge of the clock and data bits are output on t he falling edge of clock. All data transfer terminates if the $\overline{RST}$ input is low and the I/O pin goes to a high impedance state. Data transfer is illustrated in Figure 3.

DATA INPUT

Following the eight SCLK cycles that input a write command byte, a data byte is input on the rising edge of the next eight SCLK cycles. Additional SCLK cycles are ignored should they inadvertently occur. Data is input starting with bit 0.

DATA OUTPUT

Following the eight SCLK cycles that input a read command byte, a data byte is output on the falling edge of the next eight SCLK cycles. Note that the first data bit to be transmitted occurs on the first falling edge after the last bit of the command byte is written. Additional SCLK cycles retransmit the data bytes should they inadvertently occur so long as $\overline{RST}$ remains high. This operation permits continuous burst mode read capability. Data is output starting with bit 0.

BURST MODE

Burst mode may be specified for either the clock/calendar or the RAM registers by addressing location 31 decimal (address/command bits one through five = logical one). As before, bit six specified clock or RAM and bit 0 specifies read or write. There is no data storage capacity at locations 8 through 31 in the Clock/Calendar Registers or locations 24 through 31 in the RAM registers. When writing to the clock registers in the burst mode, the first eight registers must be written in order for the data to be transferred.

However, when writing to RAM in burst mode it is not necessary to write all 24 bytes for the data to transfer. Each byte that is written to will be transferred to RAM regardless of whether all 24 bytes are written or not.

CLOCK/CALENDAR

The clock/calendar is contained in eight write/read registers as shown in Figure 4. Data contained in the clock/calendar registers is in binary coded decimal format (BCD).

CLOCK HALT FLAG

Bit 7 of the seconds register is defined as the clock halt flag. When this bit is set to logic 1, the clock oscillator is stopped and the DS1202 is placed into a low-power standby mode with a current drain of not more than 100 nanoamps. When this bit is written to logic 0, the clock will start.

AM-PM/12-24 MODE

Bit 7 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

WRITE PROTECT REGISTER

Bit 7 of write protect register is the write protect bit. The first seven bits (bits 0-6) are forced to zero and will always read a zero when read. Before any write operation to the clock or RAM, bit 7 must be zero. When high, the write protect bit prevents a write operation to any other register.

CLOCK/CALENDAR BURST MODE

The clock/calendar command byte specifies burst mode operation. In this mode the eight clock/calendar registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

RAM

The static RAM is 24 x 8 bytes addressed consecutively in the RAM address space.

RAM BURST MODE

The RAM command byte specifies burst mode operation. In this mode, the 24 RAM registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

REGISTER SUMMARY

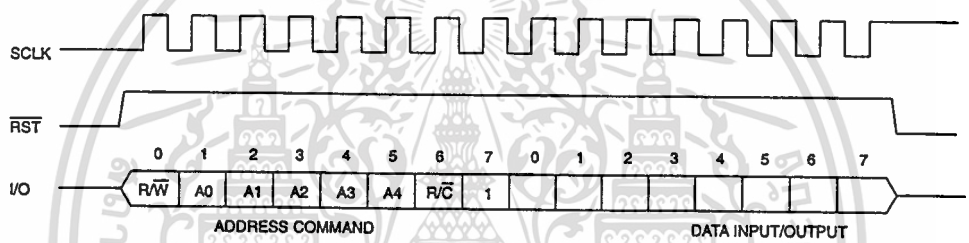
A register data format summary is shown in Figure 4.

CRYSTAL SELECTION

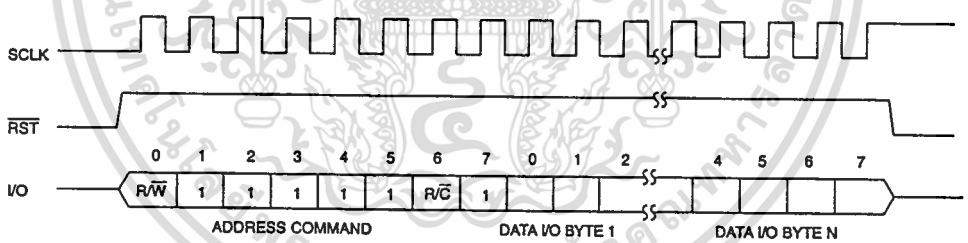
A 32.768 KHz crystal, Daiwa Part No. DT26S, Seiko Part No. DS-VT-200 or equivalent, can be directly connected to the DS1202 via pins 2 and 3 (X1, X2). The crystal selected for use should have a specified load capacitance (CL) of 6 pF. The crystal is connected directly to the X1 and X2 pins. There is no need for external capacitors or resistors. Note: X1 and X2 are very high impedance nodes. It is recommended that they and the crystal be guard–ringed with ground and that high frequency signals be kept away from the crystal area. For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real Time Clocks".

DATA TRANSFER SUMMARY Figure 3

SINGLE BYTE TRANSFER

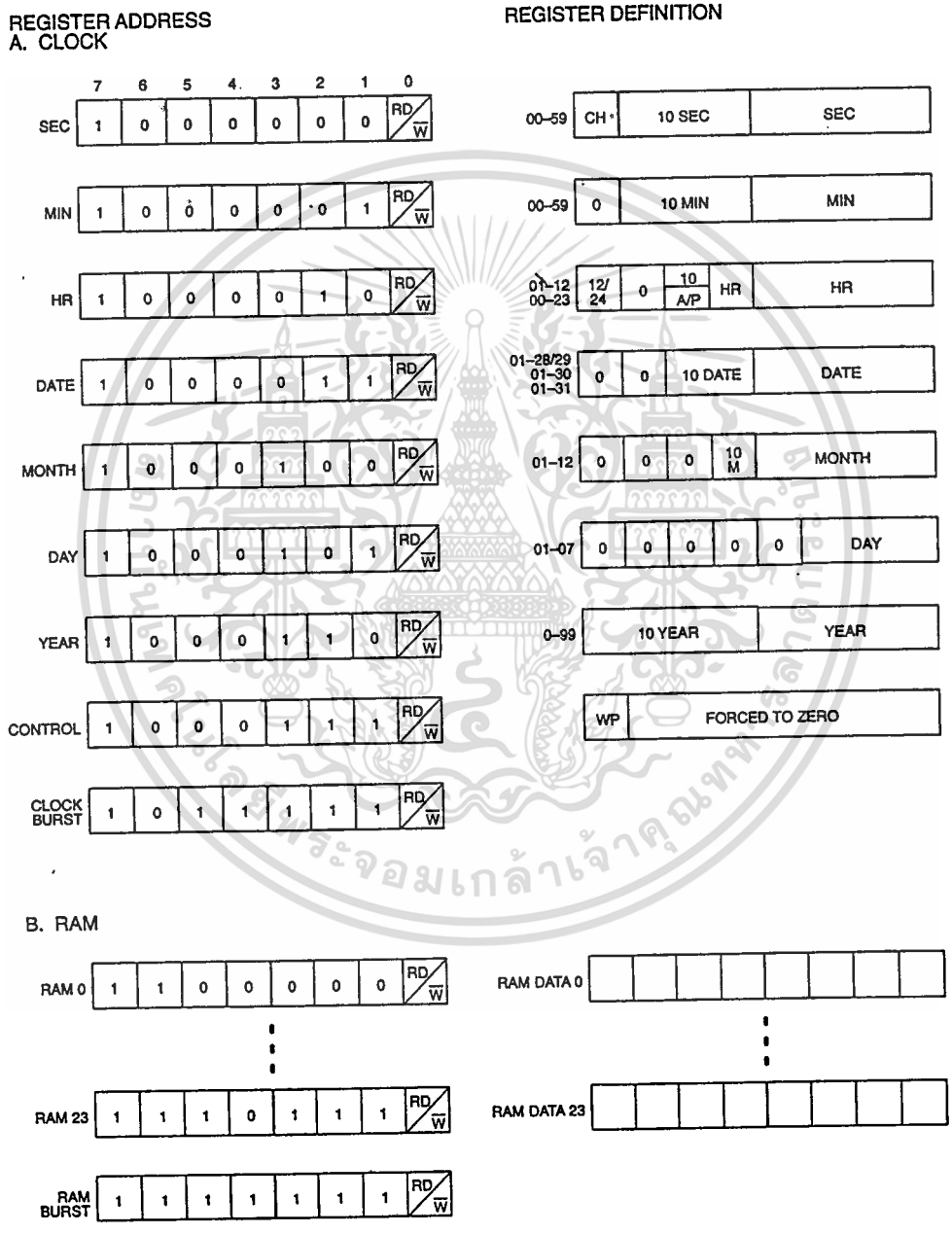


BURST MODE TRANSFER



FUNCTION	BYTE N	SCLK n
CLOCK	8	72
RAM	24	200

REGISTER ADDRESS/DEFINITION Figure 4



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.3V to +7.0V
Operating Temperature	0°C to 70°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

(0°C to 70°C)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}		2.0		5.5	V	1
Logic 1 Input	V_{IH}		2.0		V_{CC}+0.3	V	1
Logic 0 Input	V_{IL}	V_{CC}=2.0V	–0.3		+0.3	V	1
		V_{CC}=5V	–0.3		+0.8		

DC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC} = 2.0 to 5.5V*)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Input Leakage	I_{LI}				+500	µA	6
I/O Leakage	I_{LO}				+500	µA	6
Logic 1 Output	V_{OH}	V_{CC}=2V	1.6			V	2
		V_{CC}=5V	2.4				
Logic 0 Output	V_{OL}	V_{CC}=2V			0.4	V	3
		V_{CC}=5V			0.4		
Active Supply Current	I_{CC}	V_{CC}=2V			.4	mA	5
		V_{CC}=5V			1.2		
Timekeeping Current	I_{CC1}	V_{CC}=2V			0.3	µA	4
		V_{CC}=5V			1		
Leakage Current	I_{CC2}	V_{CC}=2V			100	nA	10
		V_{CC}=5V			100		

*Unless otherwise noted.

CAPACITANCE

(t_A = 25°C)

PARAMETER	SYMBOL	CONDITION	TYP	MAX	UNITS	NOTES
Input Capacitance	C_I		5		pF	
I/O Capacitance	$C_{I/O}$		10		pF	
Crystal Capacitance	C_X		6		pF	

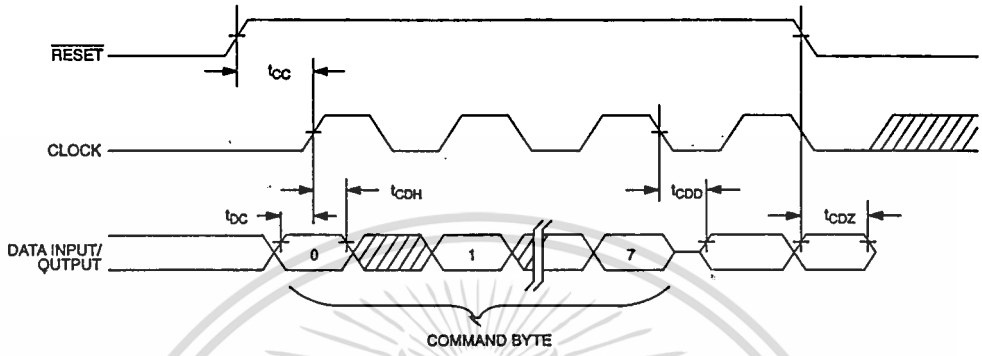
AC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC} = 2.0 to 5.5V*)

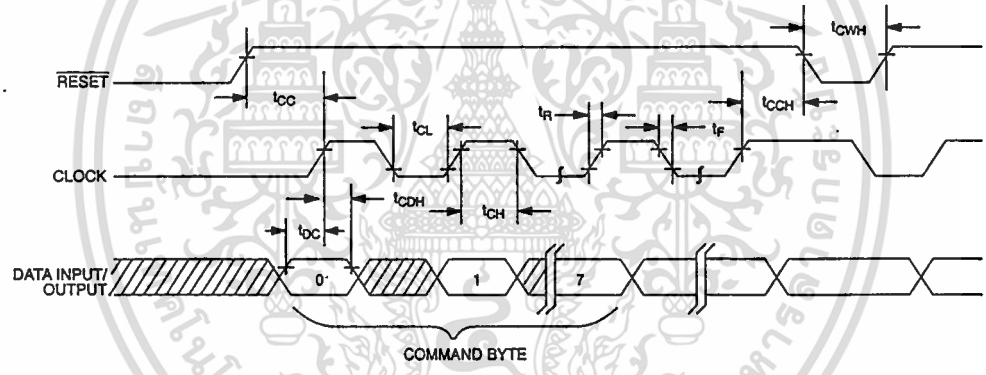
PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Data to CLK Setup	t_{DC}	V_{CC}=2V	200			ns	7
		V_{CC}=5V	50				
CLK to Data Hold	t_{CDH}	V_{CC}=2V	280			ns	7
		V_{CC}=5V	70				
CLK to Data Delay	t_{CDD}	V_{CC}=2V			800	ns	7, 8, 9
		V_{CC}=5V			200		
CLK Low Time	t_{CL}	V_{CC}=2V	1000			ns	7
		V_{CC}=5V	250				
CLK High Time	t_{CH}	V_{CC}=2V	1000			ns	7, 12
		V_{CC}=5V	250				
CLK Frequency	f_{CLK}	V_{CC}=2V			0.5	MHz	7, 12
		V_{CC}=5V	DC		2.0		
CLK Rise and Fall	t_R, t_F	V_{CC}=2V			2000	ns	
		V_{CC}=5V			500		
$\overline{RST}$ to CLK Setup	t_{CC}	V_{CC}=2V	4			µs	7
		V_{CC}=5V	1				
CLK to $\overline{RST}$ Hold	t_{CCH}	V_{CC}=2V	1000			ns	7
		V_{CC}=5V	250				
$\overline{RST}$ Inactive Time	t_{CWH}	V_{CC}=2V	4			µs	7
		V_{CC}=5V	1				
$\overline{RST}$ to I/O High Z	t_{CDZ}	V_{CC}=2V			280	ns	7
		V_{CC}=5V			70		

*Unless otherwise noted.

TIMING DIAGRAM: READ DATA TRANSFER Figure 5



TIMING DIAGRAM: WRITE DATA TRANSFER Figure 6



NOTES:

1. All voltages are referenced to ground.
2. Logic one voltages are specified at a source current of 1 mA at V_{CC}=5V and .4 mA at V_{CC}=2V, V_{OH}=V_{CC} for capacitive loads.
3. Logic zero voltages are specified at a sink current of 4 mA at V_{CC}=5V and 1.5 mA at V_{CC}=2V.
4. I_{CC1} is specified with I/O open, $\overline{RST}$ set to a logic 0, and clock halt flag=0 (oscillator enabled).
5. I_{CC} is specified with the I/O pin open, $\overline{RST}$ high, SCLK=2 MHz at V_{CC}=5V; SCLK=500 KHz, V_{CC}=2V and clock halt flag=0 (oscillator enabled).
6. $\overline{RST}$, SCLK, and I/O all have 40KΩ pulldown resistors to ground.
7. Measured at V_{IH}=2.0V or V_{IL}=0.8V and 10 ms maximum rise and fall time.
8. Measured at V_{OH}=2.4V or V_{OL}=0.4V.
9. Load capacitance = 50 pF.

19-4323; Rev 6; 3/96

MAXIM

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

General Description

The MAX220-MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, and in particular, for those applications where ±12V is not available.

These parts are particularly useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245-MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers
Low-Power Modems
Interface Translation
Battery-Powered RS-232 Systems
Multi-Drop RS-232 Networks

Features

Superior to Bipolar

- ♦ **Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)**
- ♦ **Low-Power Receive Mode in Shutdown (MAX223/MAX242)**
- ♦ **Meet All EIA/TIA-232E and V.28 Specifications**
- ♦ **Multiple Drivers and Receivers**
- ♦ **3-State Driver and Receiver Outputs**
- ♦ **Open-Line Detection (MAX243)**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

*Contact factory for dice specifications.

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	4.7/10	No		120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes		200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 + receivers active in shutdown
MAX225	+5	5/5	0	–	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes		120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No		120	Standard +5/+12V or battery supplies; same functions as MAX232.
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No		120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No		200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	–	No		120	No external caps
MAX233A	+5	2/2	0	–	No		200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No		120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	–	Yes		120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes		120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No		120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No		120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No		120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes		120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes		120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No		200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No		120	High slew rate
MAX245	+5	8/10	0	–	Yes	✔	120	High slew rate, int. caps, two shutdown modes.
MAX246	+5	8/10	0	–	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	–	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} - 0.3V)
R_{IN} ..... ±30V
T_{OUT} (Note 1) ..... ±15V
Output Voltages
T_{OUT} ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) ... 842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ... 889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) ... 440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ... 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) ... 762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C) ... 762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) ... 800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) ... 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) ... 800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C) ... 842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ..... 0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ..... -40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1-C4 = 0.1µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	Normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate	Except MAX220, normal operation			200	116	kbits/ sec
	MAX220			22	20	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	Except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	Except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	Except MAX243, V_{CC} = 5V, no hyst. in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1-C4 = 0.1µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

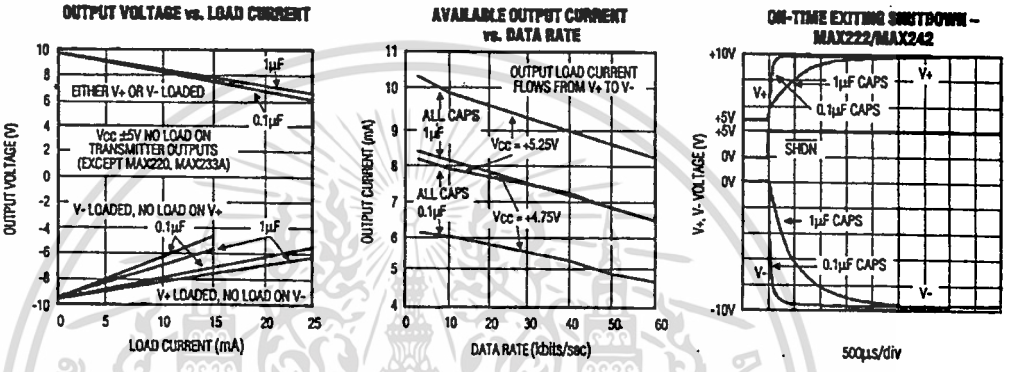
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
POWER SUPPLY						
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 9, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0° to +70°C		2	50	
		T_A = -40° to +85°C		2	50	
		T_A = -55° to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
AC CHARACTERISTICS						
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 2: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243



+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX223/MAX230-MAX241

V_{CC} ..... -0.3V to +6V
V+ ..... (V_{CC} - 0.3V) to +14V
V- ..... +0.3V to -14V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} ..... ±30V
Output Voltages
T_{OUT} ..... (V+ + 0.3V) to (V- - 0.3V)
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Short-Circuit Duration, T_{OUT} ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C) 800mW
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
20-Pin Plastic DIP (derate 11.11 mW/°C above +70°C) 889mW
24-Pin Narrow Plastic DIP
(derate 13.33mW/°C above +70°C) 1.07W
24-Pin Plastic DIP (derate 9.09mW/°C above +70°C) 500mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) 800mW
24-Pin Wide SO (derate 11.76mW/°C above +70°C) 941mW
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
44-Pin Plastic FP (derate 11.11 mW/°C above +70°C) 889mW
14-Pin CERDIP (derate 9.09mW/°C above +70°C) 727mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
20-Pin CERDIP (derate 11.11mW/°C above +70°C) 889mW
24-Pin Narrow CERDIP
(derate 12.50mW/°C above +70°C) 1W
24-Pin Sidebraze (derate 20.0mW/°C above +70°C) 1.6W
28-Pin SSOP (derate 9.52mW/°C above +70°C) 762mW
Operating Temperature Ranges
MAX2__C__ ..... 0°C to +70°C
MAX2__E__ ..... -40°C to +85°C
MAX2__M__ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX223/MAX230-MAX241

(MAX223/230/232/234/236/237/238/240/241 V_{CC} = +5V ±10%, MAX233/MAX235 V_{CC} = 5V ±5%, C1-C4 = 1.0µF MAX231/MAX239 V_{CC} = 5V ±10%, V+ = 7.5V to 13.2V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to ground		±5.0	±7.3		V
V_{CC} Power-Supply Current	No load, T_A = +25°C	MAX232/233		5	10	mA
		MAX223/230/234-238/240/241		7	15	
		MAX231 /239		.4	1	
V+ Power-Supply Current		MAX231		1.8	5	mA
		MAX239		5	15	
Shutdown Supply Current	T_A = +25°C	MAX223		15	50	µA
		MAX230/235/236/240/241		1	10	
Input Logic Threshold Low	T_{IN}; EN, $\overline{SHDN}$ (MAX223), $\overline{EN}$, SHDN (MAX230/235-241)				0.8	V
Input Logic Threshold High	T_{IN}		2.0			V
	EN, $\overline{SHDN}$ (MAX223), $\overline{EN}$, SHDN (MAX230/235/236/240/241)		2.4			
Logic Pull-Up Current	T_{IN} = 0V			1.5	200	µA
Receiver Input Voltage Operating Range			-30		30	V

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

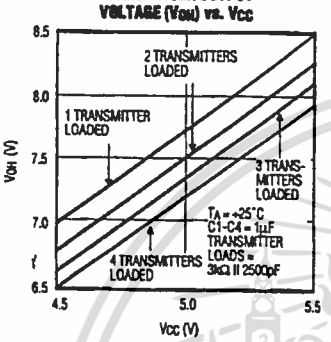
ELECTRICAL CHARACTERISTICS—MAX223/MAX230-MAX241 (continued)

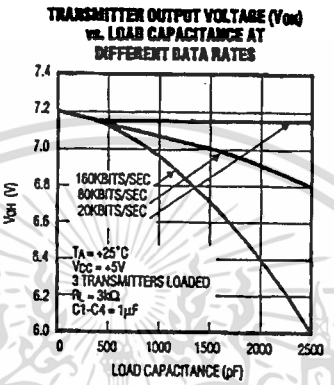
(MAX223/230/232/234/236/237/238/240/241 V_{CC} = +5V ±10%, MAX233/MAX235 V_{CC} = 5V ±5%; C1-C4 = 1.0µF MAX231/MAX239 V_{CC} = 5V ±10%, V+ = 7.5V to 13.2V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

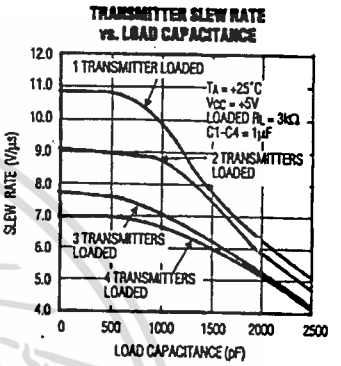
PARAMETER	CONDITIONS			MIN	TYP	MAX	UNITS
RS-232 Input Threshold Low	T_A = +25°C, V_{CC} = 5V	Normal operation $\overline{SHDN}$ = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)		0.8	1.2		V
		Shutdown (MAX223) $\overline{SHDN}$ = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)		0.6	1.5		
RS-232 Input Threshold High	T_A = +25°C, V_{CC} = 5V	Normal operation $\overline{SHDN}$ = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)			1.7	2.4	V
		Shutdown (MAX223) $\overline{SHDN}$ = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)			1.5	2.4	
RS-232 Input Hysteresis	V_{CC} = 5V; no hysteresis in shutdown			0.2	0.5	1.0	V
RS-232 Input Resistance	T_A = +25°C, V_{CC} = 5V			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 1.6mA (MAX231-233 I_{OUT} = 3.2mA)					0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1mA			3.5	V_{CC} - 0.4		V
TTL/CMOS Output Leakage Current	0V ≤ R_{OUT} ≤ V_{CC}; EN = 0V (MAX223); $\overline{EN}$ = V_{CC} (MAX235-241)				0.05	±10	µA
Receiver Output Enable Time	Normal operation	MAX223			600		ns
		MAX235/236/239/240/241			400		
Receiver Output Disable Time	Normal operation	MAX223			900		ns
		MAX235/236/239/240/241			250		
Propagation Delay	RS-232 IN to TTL/CMOS OUT, C_L = 150pF	Normal operation			0.5	10	µs
		$\overline{SHDN}$ = 0V (MAX223)	t_{PHLS}		4	40	
			t_{PLHS}		6	40	
Transition Region Slew Rate	MAX223/MAX230/MAX234-241 T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V			3	5.1	30	V/µs
	MAX231/MAX232/MAX233 T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V				4	30	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V			300			Ω
Transmitter Out Short-Circuit Current					±10		mA

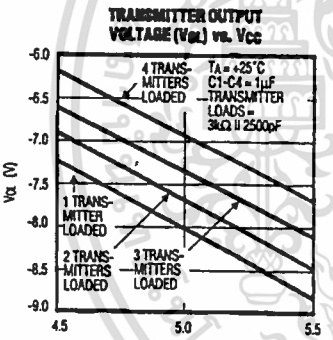
Typical Operating Characteristics

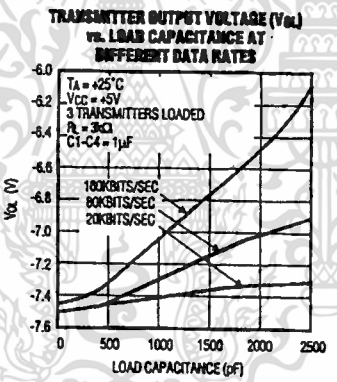
MAX223/MAX230-MAX241

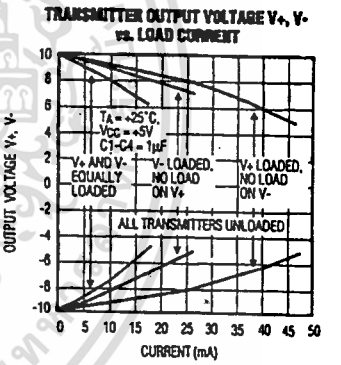


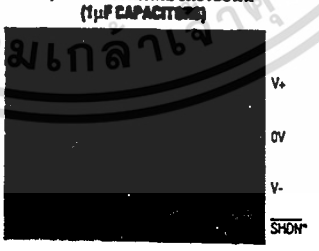












*SHUTDOWN POLARITY IS REVERSED FOR THE MAX241

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX225/MAX244-MAX249

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN}, $\overline{ENA}$, $\overline{ENB}$, $\overline{ENR}$, $\overline{ENT}$, $\overline{ENRA}$, $\overline{ENRB}$, $\overline{ENTA}$, $\overline{ENTB}$ ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} ..... ±25V
T_{OUT} (Note 3) ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Short Circuit (one output at a time)
T_{OUT} to GND ..... Continuous
R_{OUT} to GND ..... Continuous

Continuous Power Dissipation (T_A = +70°C)
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
40-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ... 611mW
44-Pin PLCC (derate 13.33mW/°C above +70°C) 1.07W
Operating Temperature Ranges
MAX225C_ _, MAX24 C_ _ ..... 0°C to +70°C
MAX225E_ _, MAX24_E_ _ ..... -40°C to +85°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Note 3: Input voltage measured with transmitter output in a high-impedance state, shutdown, or V_{CC} = 0V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX225/MAX244-MAX249

(MAX225 V_{CC} = 5.0V ±5%; MAX244-MAX249 V_{CC} = +5.0V ±10%, external capacitors C1-C4 = 1µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTER						
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	Tables 1a-1d	Normal operation		10	50	µA
		Shutdown		±0.01	±1	
Data Rate	Tables 1a-1d, normal operation			120	64	kbits/sec
Output Voltage Swing	All transmitter outputs loaded with 3Ωk to GND		±5	±7.5		V
Output Leakage Current (shutdown)	Tables 1a-1d	$\overline{ENA}$, $\overline{ENB}$, $\overline{ENT}$, $\overline{ENTA}$, $\overline{ENTB}$ = V_{CC}, V_{OUT} = ±15V		±0.01	±25	µA
		V_{CC} = 0V, V_{OUT} = ±15V		±0.01	±25	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V (Note 4)		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±30		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±25	V
RS-232 Input Threshold Low	V_{CC} = 5V		0.8	1.3		V
RS-232 Input Threshold High	V_{CC} = 5V			1.8	2.4	V
RS-232 Input Hysteresis	V_{CC} = 5V		0.2	0.5	1.0	V
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		
TTL/CMOS Output Leakage Current	Normal operation, outputs disabled, Tables 1A-1D, 0V ≤ V_{OUT} ≤ V_{CC}, $\overline{ENR}$_ = V_{CC}			±0.05	±0.10	µA

ELECTRICAL CHARACTERISTICS—MAX225/MAX244-MAX249 (continued)

(MAX225 V_{CC} = 5.0V ±5%; MAX244-MAX249 V_{CC} = +5.0V ±10%, external capacitors C1-C4 = 1µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

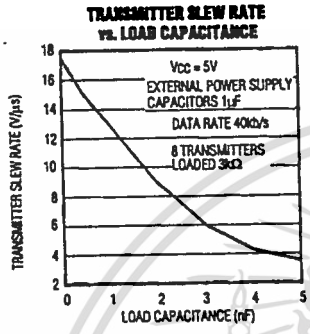
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
POWER SUPPLY AND CONTROL LOGIC						
Operating Supply Voltage		MAX225	4.75		5.25	V
		MAX244-MAX249	4.5		5.5	
V_{CC} Supply Current (normal operation)	No load	MAX225		10	20	mA
		MAX244-MAX249		11	30	
	3kΩ loads on all outputs	MAX225		40		
		MAX244-MAX249		57		
Shutdown Supply Current	T_A = +25°C			8	25	µA
	T_A = T_{MIN} to T_{MAX}				50	
Control Input	Leakage current				±1	µA
	Threshold low			1.4	0.8	V
	Threshold high		2.4	1.4		
AC CHARACTERISTICS						
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V		5	10	30	V/µs
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}			1.3	3.5	µs
	t_{PLHT}			1.5	3.5	
Receiver Propagation Delay TLL to RS-232 (normal operation), Figure 2	t_{PHLR}			0.6	1.5	µs
	t_{PLHR}			0.6	1.5	
Receiver Propagation Delay TLL to RS-232 (low-power mode), Figure 2	t_{PHLS}			0.6	10	µs
	t_{PLHS}			3.0	10	
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}			350		ns
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}			350		ns
Receiver-Output Enable Time, Figure 3	t_{ER}			100	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}			100	500	ns
Transmitter Enable Time	t_{ET}	MAX246-MAX249 (excludes charge-pump start-up)		5		µs
		MAX225/MAX245-MAX249 (includes charge-pump start-up)		10		ms
Transmitter Disable Time, Figure 4	t_{DT}			100		ns

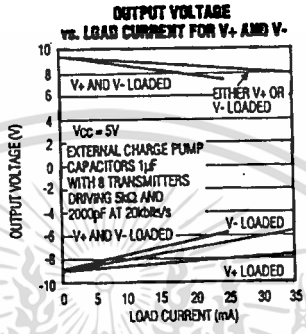
Note 4: The 300Ω minimum specification complies with EIA/TIA-232E, but the actual resistance when in shutdown mode or V_{CC} = 0 is 10MΩ as is implied by the leakage specification.

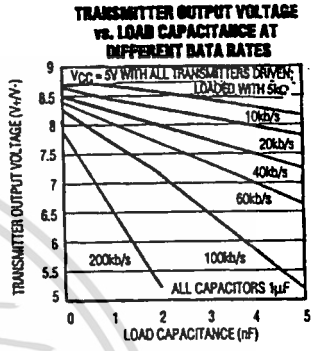
+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX225/MAX244-MAX249







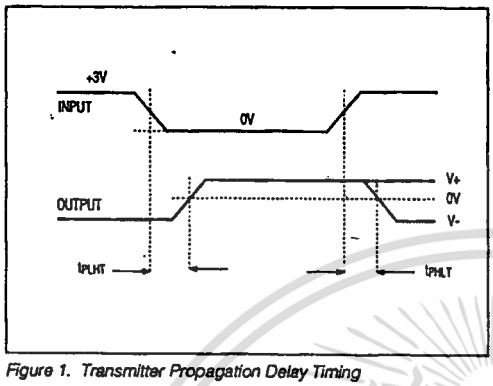


Figure 1. Transmitter Propagation Delay Timing

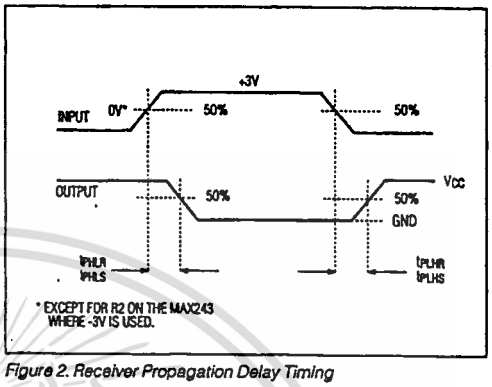


Figure 2. Receiver Propagation Delay Timing

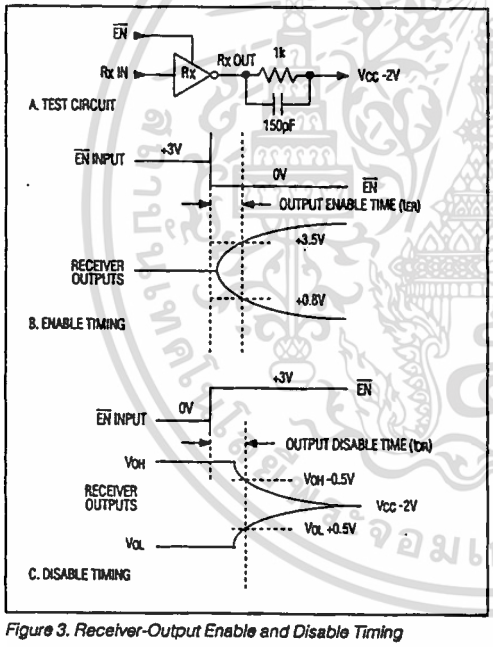


Figure 3. Receiver-Output Enable and Disable Timing

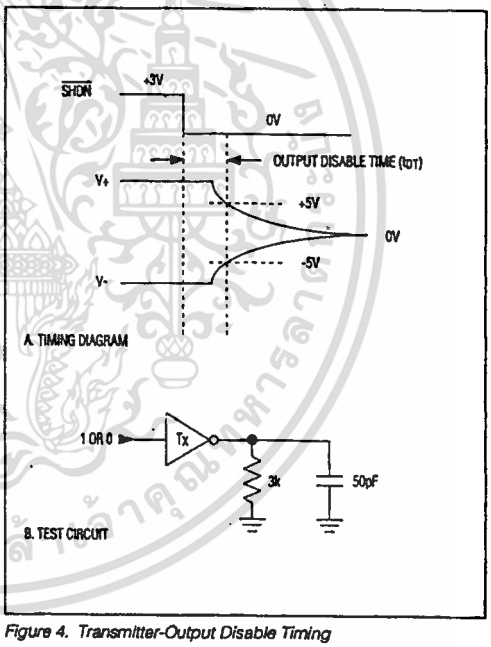


Figure 4. Transmitter-Output Disable Timing

Table 1a. MAX225 Control Pin Configurations

$\overline{ENT}$	$\overline{ENR}$	OPERATION STATUS	TRANSMITTERS	RECEIVERS
0	0	Normal Operation	All Active	All Active
0	1	Normal Operation	All Active	All 3-State
1	0	Shutdown	All 3-State	All Low-Power Receive Mode
1	1	Shutdown	All 3-State	All 3-State

Table 1b. MAX245 Control Pin Configurations

$\overline{ENT}$	$\overline{ENR}$	OPERATION STATUS	TRANSMITTERS		RECEIVERS	
			TA1-TA4	TB1-TB4	RA1-RA5	RB1-RB5
0	0	Normal Operation	All Active	All Active	All Active	All Active
0	1	Normal Operation	All Active	All Active	RA1-RA4 3-State RA5 Active	RB1-RB4 3-State RB5 Active
1	0	Shutdown	All 3-State	All 3-State	All Low Power Receiver Mode	All Low Power Receiver Mode
1	1	Shutdown	All 3-State	All 3-State	RA1-RA4 3-State RA5 Low-Power Receiver Mode	RB1-RB4 3-State RA5 Low-Power Receiver Mode

Table 1c. MAX246 Control Pin Configurations

$\overline{ENA}$	$\overline{ENB}$	OPERATION STATUS	TRANSMITTERS		RECEIVERS	
			TA1-TA4	TB1-TB4	RA1-RA5	RB1-RB5
0	0	Normal Operation	All Active	All Active	All Active	All Active
0	1	Normal Operation	All Active	All 3-State	All Active	RB1-RB4 3-State RB5 Active
1	0	Shutdown	All 3-State	All Active	RA1-RA4 3-State RA5 Active	All Active
1	1	Shutdown	All 3-State	All 3-State	RA1-RA4 3-State RA5 Low-Power Receiver Mode	RB1-RB4 3-State RA5 Low-Power Receiver Mode

Table 1d. MAX247/248/249 Control Pin Configurations

ENTA	ENTB	ENRA	ENRB	OPERATION STATUS		TRANSMITTERS		RECEIVERS	
					MAX247	TA1–TA4	TB1–TB4	RA1–RA4	RB1–RB5
					MAX248	TA1–TA4	TB1–TB4	RA1–RA4	RB1–RB4
					MAX249	TA1–TA3	TB1–TB3	RA1–RA5	RB1–RB5
0	0	0	0	Normal Operation		All Active	All Active	All Active	All Active
0	0	0	1	Normal Operation		All Active	All Active	All Active	All 3-State, except RB5 stays active on MAX247
0	0	1	0	Normal Operation		All Active	All Active	All 3-State	All Active
0	0	1	1	Normal Operation		All Active	All Active	All 3-State	All 3-State, except RB5 stays active on MAX247
0	1	0	0	Normal Operation		All Active	All 3-State	All Active	All Active
0	1	0	1	Normal Operation		All Active	All 3-State	All Active	All 3-State, except RB5 stays active on MAX247
0	1	1	0	Normal Operation		All Active	All 3-State	All 3-State	All Active
0	1	1	1	Normal Operation		All Active	All 3-State	All 3-State	All 3-State, except RB5 stays active on MAX247
1	0	0	0	Normal Operation		All 3-State	All Active	All Active	All Active
1	0	0	1	Normal Operation		All 3-State	All Active	All Active	All 3-State, except RB5 stays active on MAX247
1	0	1	0	Normal Operation		All 3-State	All Active	All 3-State	All Active
1	0	1	1	Normal Operation		All 3-State	All Active	All 3-State	All 3-State, except RB5 stays active on MAX247
1	1	0	0	Shutdown		All 3-State	All 3-State	Low-Power Receive Mode	Low-Power Receive Mode
1	1	0	1	Shutdown		All 3-State	All 3-State	Low-Power Receive Mode	All 3-State, except RB5 stays active on MAX247
1	1	1	0	Shutdown		All 3-State	All 3-State	All 3-State	Low-Power Receive Mode
1	1	1	1	Shutdown		All 3-State	All 3-State	All 3-State	All 3-State, except RB5 stays active on MAX247

+5V-Powered, Multi-Channel RS-232 Drivers/Receivers

Detailed Description

The MAX220-MAX249 contain four sections: dual charge-pump DC-DC voltage converters, RS-232 drivers, RS-232 receivers, and receiver and transmitter enable control inputs.

Dual Charge-Pump Voltage Converter

The MAX220-MAX249 have two internal charge-pumps that convert +5V to ±10V (unloaded) for RS-232 driver operation. The first converter uses capacitor C1 to double the +5V input to +10V on C3 at the V+ output. The second converter uses capacitor C2 to invert +10V to -10V on C4 at the.V- output.

A small amount of power may be drawn from the +10V (V+) and -10V (V-) outputs to power external circuitry (see Typical Operating Characteristics), except on the MAX225 and MAX245-MAX247, where these pins are not available. V+ and V- are not regulated, so the output voltage drops with increasing load current. Do not load V+ and V- to a point that violates the minimum ±5V EIA/TIA-232E driver output voltage when sourcing current from V+ and V- to external circuitry.

When using the shutdown feature in the MAX222, MAX225, MAX230, MAX235, MAX236, MAX240, MAX241, and MAX245-MAX249 avoid using V+ and V- to power external circuitry. When these parts are shut down, V- falls to 0V, and V+ falls to +5V. For applications where a +10V external supply is applied to the V+ pin (instead of using the internal charge pump to generate +10V), the C1 capacitor must not be installed and the $\overline{SHDN}$ pin must be tied to V_{CC}. This is because V+ is internally connected to V_{CC} in shutdown mode.

RS-232 Drivers

The typical driver output voltage swing is ±8V when loaded with a nominal 5kΩ RS-232 receiver and V_{CC} = +5V. Output swing is guaranteed to meet the EIA/TIA-232E and V.28 specification, which calls for ±5V minimum driver output levels under worst-case conditions. These include a minimum 3kΩ load, V_{CC} = +4.5V, and maximum operating temperature. Unloaded driver output voltage ranges from (V+ -1.3V) to (V- +0.5V).

Input thresholds are both TTL and CMOS compatible. The inputs of unused drivers can be left unconnected since 400kΩ input pull-up resistors to V_{CC} are built-in. The pull-up resistors force the outputs of unused drivers low because all drivers invert. The internal input pull-up resistors typically source 12µA, except in shutdown mode where the pull-ups are disabled. Driver outputs turn off and enter a high-impedance state—where leakage current is typically microamperes (maximum 25µA)—when in shutdown mode, in three-state mode, or when device power is removed. Outputs can be driven to ±15V. The power-supply current typically drops to 8µA in shutdown mode.

The MAX239 has a receiver 3-state control line, and the MAX223, MAX225, MAX235, MAX236, MAX240, and MAX241 have both a receiver 3-state control line and a low-power shutdown control. The receiver TTL/CMOS outputs are in a high-impedance, 3-state mode whenever the 3-state $\overline{EN}$able line is high, and are also high-impedance whenever the shutdown control line is high.

When in low-power shutdown mode, the driver outputs are turned off and their leakage current is less than 1µA with the driver output pulled to ground. The driver output leakage remains less than 1µA, even if the transmitter output is backdriven between 0V and (V_{CC} + 6V). Below -0.5V, the transmitter is diode clamped to ground with 1kΩ series impedance. The transmitter is also zener clamped to approximately V_{CC} + 6V, with a series impedance of 1kΩ.

The driver output slew rate is limited to less than 30V/µs as required by the EIA/TIA-232E and V.28 specifications. Typical slew rates are 24V/µs unloaded and 10V/µs loaded with 3Ω and 2500pF.

RS-232 Receivers

EIA/TIA-232E and V.28 specifications define a voltage level greater than 3V as a logic 0, so all receivers invert. Input thresholds are set at 0.8V and 2.4V, so receivers respond to TTL level inputs as well as EIA/TIA-232E and V.28 levels.

The receiver inputs withstand an input overvoltage up to ±25V and provide input terminating resistors with nominal 5kΩ values. The receivers implement Type 1 interpretation of the fault conditions of V.28 and EIA/TIA-232E.

The receiver input hysteresis is typically 0.5V with a guaranteed minimum of 0.2V. This produces clear output transitions with slow-moving input signals, even with moderate amounts of noise and ringing. The receiver propagation delay is typically 600ns and is independent of input swing direction.

Low-Power Receive Mode

The low-power receive-mode feature of the MAX223, MAX242, and MAX245-MAX249 puts the IC into shutdown mode, but still allows it to receive information. This is important for applications where systems are periodically awakened to look for activity. Using low-power receive mode, the system can still receive a signal that will activate it on command and prepare it for communication at faster data rates. This operation conserves system power.

Negative Threshold—MAX243

The MAX243 is pin compatible with the MAX232A, differing only in that RS-232 cable fault protection is removed on one of the two receiver inputs. This means that control lines such as CTS and RTS can either be driven or left floating without interrupting communication. Different cables are not needed to interface with different pieces of equipment.

The input threshold of the receiver without cable fault protection is -0.8V rather than +1.4V. Its output goes positive only if the input is connected to a control line that is actively driven negative. If not driven, it defaults to the 0 or "OK to send" state. Normally, the MAX243's other receiver (+1.4V threshold) is used for the data line (TD or RD), while the negative threshold receiver is connected to the control line (DTR, DTS, CTS, RTS, etc.).

Other members of the RS-232 family implement the optional cable fault protection as specified by EIA/TIA-232E specifications. This means a receiver output goes high whenever its input is driven negative, left floating, or shorted to ground. The high output tells the serial communications IC to stop sending data. To avoid this, the control lines must either be driven or connected with jumpers to an appropriate positive voltage level.

Shutdown—MAX222-MAX242

On the MAX222, MAX235, MAX236, MAX240, and MAX241, all receivers are disabled during shutdown. On the MAX223 and MAX242, two receivers continue to operate in a reduced power mode when the chip is in shutdown. Under these conditions, the propagation delay increases to about 2.5µs for a high-to-low input transition. When in shutdown the receiver acts as a CMOS inverter with no hysteresis. The MAX223 and MAX242 also have a receiver output enable input ($\overline{EN}$) that allows receiver output control independent of $\overline{SHDN}$. With all other devices, $\overline{SHDN}$ also disables the receiver outputs.

The MAX225 provides five transmitters and five receivers, while the MAX245 provides ten receivers and eight transmitters. Both devices have separate receiver and transmitter-enable controls. The charge pumps turn off and the devices shut down when a logic high is applied to the ENT input. In this state, the supply current drops to less than 25µA and the receivers continue to operate in a low-power receive mode. Driver outputs enter a high-impedance state (three-state mode). On the MAX225, all five receivers are controlled by the $\overline{ENR}$ input. On the MAX245, eight of the receiver outputs are controlled by the $\overline{ENR}$ input, while the remaining two receivers (RA5 and RB5) are always active. RA1-RA4 and RB1-RB4 are put in a three-state mode when $\overline{ENR}$ is a logic high.

Receiver and Transmitter Enable Control Inputs

The MAX225 and MAX245-MAX249 feature transmitter and receiver enable controls.

The receivers have three modes of operation: full-speed receive (normal active), three-state (disabled), and low-power receive (enabled receivers continue to function at lower data rates). The receiver enable inputs control the full-speed receive and three-state modes. The transmitters have two modes of operation: full-speed transmit (normal active) and three-state (disabled). The transmitter enable inputs also control the shutdown mode. The device enters shutdown mode when all transmitters are disabled. Enabled receivers function in the low-power receive mode when in shutdown.

Tables 1a-1d define the control states. The MAX244 has no control pins and is not included in these tables.

The MAX246 has ten receivers and eight drivers with two control pins, each controlling one side of the device. A logic high at the A-side control input ($\overline{ENA}$) causes the four A-side receivers and drivers to go into a three-state mode. Similarly, the B-side control input ($\overline{ENB}$) causes the four B-side drivers and receivers to go into a three-state mode. As in the MAX245, one A-side and one B-side receiver (RA5 and RB5) remain active at all times. The entire device is put into shutdown mode when both the A and B sides are disabled ($\overline{ENA}$ = $\overline{ENB}$ = +5V).

The MAX247 provides nine receivers and eight drivers with four control pins. The $\overline{ENRA}$ and $\overline{ENRB}$ receiver enable inputs each control four receiver outputs. The $\overline{ENTA}$ and $\overline{ENTB}$ transmitter enable inputs each control four drivers. The ninth receiver (RB5) is always active. The device enters shutdown mode with a logic high on both $\overline{ENTA}$ and $\overline{ENTB}$.

The MAX248 provides eight receivers and eight drivers with four control pins. The $\overline{ENRA}$ and $\overline{ENRB}$ receiver enable inputs each control four receiver outputs. The $\overline{ENTA}$ and $\overline{ENTB}$ transmitter enable inputs control four drivers each. This part does not have an always-active receiver. The device enters shutdown mode and transmitters go into a three-state mode with a logic high on both $\overline{ENTA}$ and $\overline{ENTB}$.

The MAX249 provides ten receivers and six drivers with four control pins. The $\overline{ENRA}$ and $\overline{ENRB}$ receiver enable inputs each control five receiver outputs. The $\overline{ENTA}$ and $\overline{ENTB}$ transmitter enable inputs control three drivers each. There is no always-active receiver. The device enters shutdown mode and transmitters go into a three-state mode with a logic high on both $\overline{ENTA}$ and $\overline{ENTB}$. In shutdown mode, active receivers operate in a low-power receive mode at data rates up to 20kbits/s.

Applications Information

Figures 5 through 25 show pin configurations and typical operating circuits. In applications that are sensitive to power-supply noise, V_{CC} should be decoupled to ground with a capacitor of the same value as C1 and C2 connected as close as possible to the device.

บรรณานุกรม

คณะผู้จัดทำ บริษัท สกายบุ๊กส์ จำกัด, การสร้างแอพพลิเคชันบนวินโดวส์ด้วย Visual Basic For Windows 3.0, บริษัท สกายบุ๊กส์ จำกัด, 2538.

ฐิติกรณ์ ผลผูล, ระบบควบคุมอุปกรณ์ไฟฟ้า และป้องกันขโมยภายในบ้าน, ปริญญานิพนธ์, 2538.

ปรเมษฐ์ ประนยานันทน์, คู่มือประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51, บริษัท ซีเอ็ดยูเคชั่น จำกัก (มหาชน) , 2521.

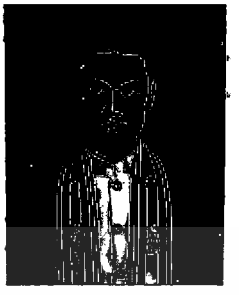
สุเจตน์ จันทรังษ์, ไมโครคอนโทรลเลอร์ชิพเดี่ยว 8051, โครงการตำราวิชาการวิทยาลัยมหานคร, 2535.

Ayla, K.J. , "The 8051 Microcontroller Architecture , Programming , and Applications," West Publishing Company , 1991.

AZZ-31 Version 2.0 Usar's Manual , บริษัท ศิลารีเสิร์ช จำกัด.

J31 LAB 2 , บริษัท ศิลารีเสิร์ช จำกัด.

ประวัติผู้แต่ง

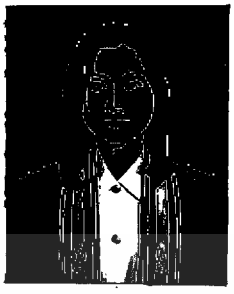


ชื่อผู้ทำปริญญานิพนธ์	นายนริศ บุญศักดิ์เฉลิม
วันเดือนปีเกิด	15 ตุลาคม 2517
สถานที่เกิด	จังหวัดฉะเชิงเทรา
ภูมิลำเนาเดิม	จังหวัดฉะเชิงเทรา
ที่อยู่ปัจจุบัน	15/1 หมู่ 6 ต. จุกเฌอ อ. เมือง จ. ฉะเชิงเทรา 24000
โทรศัพท์	-

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนบ้านบางไผ่
มัธยมศึกษาตอนต้น	โรงเรียนพุทธโสธร
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคฉะเชิงเทรา
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคฉะเชิงเทรา
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับรางวัล	-
ทุนการศึกษา	-
คติพจน์	หลังก้อนเมฆยังมีพระอาทิตย์ส่องแสงอยู่

ประวัติผู้แต่ง



ชื่อผู้ทำปริญญานิพนธ์	นางสาวนันทินี แป้นแจ้ง
วันเดือนปีเกิด	14 กรกฎาคม 2518
สถานที่เกิด	จังหวัดสุพรรณบุรี
ภูมิลำเนาเดิม	จังหวัดสุพรรณบุรี
ที่อยู่ปัจจุบัน	42/2 หมู่ 3 ต. พิหารแดง อ. เมือง จ. สุพรรณบุรี 72000
โทรศัพท์	035 - 523551
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนอนุบาลสุพรรณบุรี
มัธยมศึกษาตอนต้น	โรงเรียนสงวนหญิง
ประกาศนียบัตรวิชาชีพ (ปวช.)	
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส. 4 ปี)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตนนทบุรี
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับรางวัล	-
ทุนการศึกษา	-
คติพจน์	ความอดทนเป็นสิ่งขมขื่น แต่ผลของมันหวานชื่นเสมอ

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายอานุภาพ เสนีย์พัลย์
วันเดือนปีเกิด	10 ธันวาคม 2517
สถานที่เกิด	จังหวัดมุดาหาร
ภูมิลำเนาเดิม	จังหวัดมุกดาหาร
ที่อยู่ปัจจุบัน	25 หมู่ 21 ซอย จันทรเสนีย์ ถ. เมืองใหม่ อ.เมือง จ.มุกดาหาร 49000
โทรศัพท์	042 - 612968
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนเมืองใหม่
มัธยมศึกษาตอนต้น	โรงเรียนนวมินทราชูทิศอีสาน
ประกาศนียบัตรวิชาชีพ (ปวช.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตขอนแก่น
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตขอนแก่น
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับรางวัล	-
ทุนการศึกษา	-
คติพจน์	ขยัน ซื่อสัตย์ ประหยัด อดทน

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายพันธ์ศักดิ์ ภูวัน
วันเดือนปีเกิด	2 กุมภาพันธ์ 2516
สถานที่เกิด	จังหวัดลำปาง
ภูมิลำเนาเดิม	จังหวัดลำปาง
ที่อยู่ปัจจุบัน	45 หมู่ 4 ต.เกาะคา อ.เกาะคา จ.ลำปาง 52130
โทรศัพท์	054-281695

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนบ้านหนองหล่าย
มัธยมศึกษาตอนต้น	โรงเรียนเกาะคาวิทยาคม
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคลำปาง
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตพายัพ
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับรางวัล	-
ทุนการศึกษา	-
คติพจน์	ทำวันนี้ให้ดีที่สุด