ระบบแจ้งเหตุเพลิงไหม้ที่สามารถระบุตำแหน่งได้

ADDRESSABLE FIRE ALARM SYSTEM

โดย

1. นาย สถาณุ โตอดิเทพย์ 35104433
2. นาย สมพล กฤษฎี 35104444

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ สมยศ จุณณะปิยะ

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2538

ระบบแจ้งเหตุเพลิงไหม้ที่สามารถระบุตำแหน่งได้

ADDRESSABLE FIRE ALARM SYSTEM

โดย

นาย สถาณุ โตอดิเทพย์ 35104433

นาย สมพล กฤษฎี 35104444

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ สมยศ จุณณะปิยะ

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้เสนอแนวทางในการออกแบบระบบแจ้งเหตุไฟไหม้ที่สามารถระบุตำแหน่งได้ ซึ่งจะเป็นการส่งข้อมูลตอบโต้กันระหว่างไมโครโปรเซสเซอร์ตัวแม่และไมโครโปรเซสเซอร์ตัวลูก หลังจากที่ไมโครโปรเซสเซอร์ตัวแม่รับค่าสถานะของตัวลูกครบทุกๆตำแหน่ง ก็จะส่งค่าสถานะที่เก็บอยู่ทั้งหมดออกไปสู่หน้าจอมอนิเตอร์ และถ้ามีตำแหน่งใดเกิดไฟไหม้ก็จะมีสัญลักษณ์ปรากฏบนหน้าจอและมีเสียงเตือนว่ากำลังเกิดไฟไหม้อยู่ การทำงานของระบบจะถูกควบคุมโดยซอฟต์แวร์ ทำให้ระบบมีความยืดหยุ่นมากกว่าการใช้ฮาร์ดแวร์ควบคุมโดยตรง และผู้ใช้สามารถพัฒนาโปรแกรมควบคุมระบบให้ดีขึ้นหรือสามารถนำไปประยุกต์ใช้งานอื่นๆได้

Abstract

This thesis presents a method to design the addressable fire alarm system that communicates between a master and slaves. After receiving statuses from slaves, the master transmits stored statuses displaying on a PC monitor. If there have firing positions, the warning sound will alarm . This system that is software-controlled maker is more flexible than the hardware-controlled. The users can develop the control software for the other applications.

ระบบแจ้งเหตุเพลิงไหม้ที่สามารถระบุตำแหน่งได้

ADDRESSABLE FIRE ALARM SYSTEM

โดย

นาย สถาณุ โตอดิเทพย์ 35104433

นาย สมพล กฤษฎี 35104444

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ระบบแจ้งเหตุเพลิงไหม้ที่สามารถระบุตำแหน่งได้

ADDRESSABLE FIRE ALARM SYSTEM

โดย

1. นาย สถาณุ โตอดิเทพย์ 35104433
2. นาย สมพล กฤษฎี 35104444

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ สมยศ จุณณะปิยะ

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบแจ้งเหตุเพลิงไหม้ที่สามารถระบุตำแหน่งได้
ADDRESSABLE FIRE ALARM SYSTEM

ผู้จัดทำ

1. นาย สถาณุ โตอดิเทพย์ 35104433
2. นาย สมพล กฤษฎี 35104444

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ สมยศ จุณณะปิยะ)

ระบบแจ้งเหตุเพลิงไหม้ที่สามารถระบุตำแหน่งได้

ADDRESSABLE FIRE ALARM SYSTEM

โดย

นาย สถาณุ โตอดิเทพย์ 35104433

นาย สมพล กฤษฎี 35104444

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ สมยศ จุณณะปิยะ

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้เสนอแนวทางในการออกแบบระบบแจ้งเหตุไฟไหม้ที่สามารถระบุตำแหน่งได้ ซึ่งจะเป็นการส่งข้อมูลตอบโต้กันระหว่างไมโครโปรเซสเซอร์ตัวแม่และไมโครโปรเซสเซอร์ตัวลูก หลังจากที่ไมโครโปรเซสเซอร์ตัวแม่รับค่าสถานะของตัวลูกครบทุกๆตำแหน่ง ก็จะส่งค่าสถานะที่เก็บอยู่ทั้งหมดออกไปสู่หน้าจอมอนิเตอร์ และถ้ามีตำแหน่งใดเกิดไฟไหม้ก็จะมีสัญลักษณ์ปรากฏบนหน้าจอและมีเสียงเตือนว่ากำลังเกิดไฟไหม้อยู่ การทำงานของระบบจะถูกควบคุมโดยซอฟต์แวร์ ทำให้ระบบมีความยืดหยุ่นมากกว่าการใช้ฮาร์ดแวร์ควบคุมโดยตรง และผู้ใช้สามารถพัฒนาโปรแกรมควบคุมระบบให้ดีขึ้นหรือสามารถนำไปประยุกต์ใช้งานอื่นๆได้

Abstract

This thesis presents a method to design the addressable fire alarm system that communicates between a master and slaves. After receiving statuses from slaves, the master transmits stored statuses displaying on a PC monitor. If there have firing positions, the warning sound will alarm . This system that is software-controlled maker is more flexible than the hardware-controlled. The users can develop the control software for the other applications.

สารบัญ

สารบัญรูปภาพ

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

ในปัจจุบันนี้จะเห็นได้ว่าการเจริญเติบโตทางด้านเศรษฐกิจ สังคม และเทคโนโลยีได้พัฒนารุดหน้าไปอย่างรวดเร็ว ประกอบกับพื้นที่ต่อประชากรได้ลดน้อยลงไปเรื่อย ๆ จึงได้มีการก่อสร้างอาคารขนาดใหญ่มากมาย เพื่อรองรับขนาดขององค์การนั้นๆ เช่น โรงแรม ศูนย์การค้า โรงงาน หรือ แม้กระทั่งอาคารเรียนของมหาวิทยาลัย สิ่งก่อสร้างเหล่านี้ต้องใช้ทุนทรัพย์มหาศาล จึงมีความจำเป็นอย่างยิ่งที่ควร จะมีระบบป้องกันอาคารนั้นๆจากอัคคีภัย

ประกอบกับในปัจจุบันนี้เทคโนโลยีทางด้านไมโครโปรเซสเซอร์ได้พัฒนาขึ้นเป็นอย่างมาก ไมโครโปรเซสเซอร์มีบทบาทสำคัญเป็นอย่างมากในการใช้เป็นตัวควบคุมการทำงานของเครื่องมือ เครื่องจักรกล เครื่องใช้ไฟฟ้า รวมทั้งระบบอัตโนมัติต่างๆมากมาย ไมโครโปรเซสเซอร์ที่ถูกนำมาประยุกต์ใช้งานกันอย่างกว้างขวาง ได้แก่ ไมโครโปรเซสเซอร์เบอร์ต่างๆ ในตระกูล MCS-51 ที่เป็นแบบ 40-PINDIPของอินเทล ชิพตระกูลนี้ประกอบไปด้วย CPU เบอร์ต่างๆ ได้แก่ 8031, 8051, 8032, 8052, 8752 รวมทั้ง 8052 AHBASIC และ DS 5000 ของ DALLAS SEMICONDUCTOR แต่ในโครงงานนี้จะใช้ไมโครคอนโทรลเลอร์เบอร์ 80C31 โดยจะทำการติดต่อระหว่างกันเองโดยจะมีตัวหนึ่งที่ทำหน้าที่เป็นตัวแม่ (Master) และจะมีตัวที่ทำหน้าที่เป็นตัวลูก (Slave) นอกจากนั้นตัวแม่ยังทำการเชื่อมต่อกับคอมพิวเตอร์ทางพอร์ตอนุกรม เพื่อที่จะนำค่าของตัวลูกนำมาแสดงผลทางหน้าจอมอนิเตอร์ เพื่อบอกให้ทราบว่าในจุดใดบ้างที่เกิดไฟไหม้ขึ้น ในระบบนี้ส่วนใหญ่จะควบคุมโดยซอฟต์แวร์ ทำให้ระบบนี้มีความยืดหยุ่นสูงและมีการเปลี่ยนแปลงตัวระบบให้เหมาะสมตามที่ผู้ใช้ต้องการได้ง่าย

บทที่ 2

ทฤษฎีและหลักการ

2.1 การสื่อสารข้อมูลแบบอนุกรม (Serial Data Communication)

MCS-51 มีพอร์ตสำหรับสื่อสารข้อมูลแบบอนุกรมที่สามารถรับและส่งข้อมูลแบบอนุกรมได้โดยผู้ใช้ไม่จำเป็นต้องต่อชิปที่ทำหน้าที่รับส่งข้อมูลแบบอนุกรมเพิ่มเติม โดยพอร์ตสื่อสารข้อมูลแบบอนุกรมที่มีใน MCS-51 จะสามารถทำงานได้ในแบบ ฟูลดูเพล็กซ์ (full duplex) หมายถึง สามารถรับส่งข้อมูลได้พร้อมๆกัน โดยในการรับข้อมูลจะมีการบัฟเฟอร์ข้อมูลให้ด้วย จึงทำให้สามารถกำหนดการรับข้อมูลไบต์ที่สองซึ่งถูกส่งตามเข้ามาก่อนที่ไบต์แรกที่ได้รับเข้ามาจะถูกอ่านจากรีจิสเตอร์ใช้งานเฉพาะที่ใช้สำหรับรับข้อมูล (receiver register) เพื่อนำไปเก็บไว้ในหน่วยความจำต่อไป (หากไบต์แรกยังไม่ถูกอ่านเมื่อได้รับไบต์ที่สองเรียบร้อยแล้วข้อมูลจะหายไปหนึ่งไบต์)

พอร์ตสื่อสารข้อมูลแบบอนุกรมประกอบด้วยรีจิสเตอร์ขนาด 8 บิต จำนวนสองตัวแต่ละตัวมีชื่อเรียกตามหน้าที่ดังนี้คือ

- รีจิสเตอร์สำหรับรับข้อมูลใช้รับข้อมูลที่ส่งเข้ามาจากภายนอก
- รีจิสเตอร์สำหรับส่งข้อมูล (transmit register) ใช้ส่งข้อมูลออกไปภายนอก

ซึ่งรีจิสเตอร์ทั้งสองมีตำแหน่งเดียวกันในรีจิสเตอร์ใช้งานเฉพาะ คือตรงกับตำแหน่งของรีจิสเตอร์ใช้งานเฉพาะ SBUF (ตำแหน่ง 99H) ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ การเข้าถึงข้อมูลในรีจิสเตอร์แต่ละตัว MCS-51 จะทราบเองว่าผู้ใช้ต้องการติดต่อกับรีจิสเตอร์ตัวใดโดยตรวจสอบจากรหัสคำสั่ง ทั้งนี้เพราะในการเขียนข้อมูลไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ SBUF หมายถึงการโหลดข้อมูลไปที่รีจิสเตอร์สำหรับส่งข้อมูลเพื่อส่งข้อมูลออกไปภายนอก ส่วนการอ่านข้อมูลจากรีจิสเตอร์ใช้งานเฉพาะ SBUF จะหมายถึงนำค่าที่รับเข้ามาได้จากภายนอกที่เก็บไว้ในรีจิสเตอร์ สำหรับรับข้อมูลมาใช้งาน

การใช้งานที่แตกต่างกันมี 4 ประเภทนี้ มีจุดประสงค์เพื่อความคล่องตัวในการรับหรือส่งข้อมูลแบบอนุกรมแต่ละประเภทดังนี้

โหมด 0 จะทำงานเป็น shift register 8 บิต อัตราเร็วในการส่งข้อมูล 1/12 ของความถี่ออสซิเลเตอร์

โหมด 1 8 บิต UART สามารถกำหนดอัตราเร็วในการส่งหรือรับข้อมูลเองได้

โหมด 2 Multiprocessor 9 บิต UART อัตราเร็วในการส่งหรือรับข้อมูล 1/32 หรือ 1/64 ของความถี่ออสซิเลเตอร์

โหมด 3 Multiprocessor 9 บิต UART สามารถกำหนดอัตราเร็วในการส่งหรือรับข้อมูลเองได้

ในที่นี้จะกล่าวถึงโหมด 2 และ 3 เท่านั้น

2.2 การสื่อสารระหว่างไมโครโปรเซสเซอร์

การทำงานในโหมดนี้ข้อมูลจำนวน 11 บิตถูกส่งผ่านขา TXD และถูกรับเข้ามาทางขา RXD ข้อมูลทั้ง 11 บิตประกอบไปด้วยบิตเริ่มต้นของข้อมูล 1 บิต (เป็น 0 เสมอ) บิตข้อมูล 8 บิต(รับและส่งบิตต่ำสุดก่อน) ตามด้วยบิตที่ 9 ซึ่งเป็นบิตที่สามารถกำหนดค่าได้ และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล (เป็น 1 เสมอ)

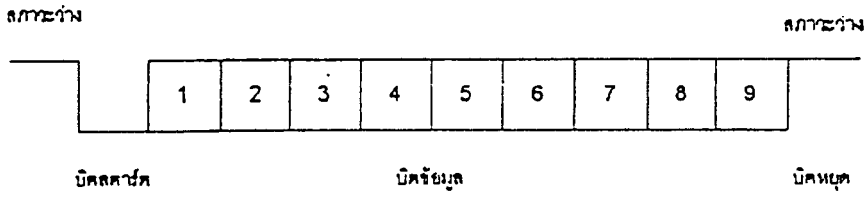
การทำงานจะเริ่มทันที เมื่อมีคำสั่งใดๆที่ใช้รีจิสเตอร์ใช้งานเฉพาะ SBUF เป็นรีจิสเตอร์ปลายทาง (destination register) เช่น MOV SBUF,A

ส่วนในการรับข้อมูลจะเริ่มต้นโดยมีเงื่อนไขดังนี้

- ในโหมด 0 เริ่มเมื่อค่าในบิต RI = 0 และบิต REN = 1
- ในโหมดอื่นๆ การรับข้อมูลเริ่มเมื่อ MCS-51 ได้รับบิตเริ่มต้นของข้อมูลเข้ามา โดยที่บิต REN ในขณะนั้นต้องมีค่าเป็น 1

โหมด 2 และ 3 นั้นต่างกันที่อัตราเร็วในการส่งข้อมูล โดยโหมด 2 จะใช้อัตราเร็วในการส่งข้อมูล 1/32 ของความถี่ออสซิเลเตอร์ ถ้า SMOD (PCON.7) เคลียร์ หรือ อัตราเร็วในการส่งข้อมูล 1/64 ของความถี่ออสซิลเลเตอร์ ถ้า SMOD ถูกเซท ตัวอย่างเช่น ใน 16 MHz จะแสดงผลในอัตราเร็วในการส่งข้อมูลที่ 500000 และ 250000 บิตต่อวินาที

อัตราความเร็วในการส่งข้อมูลสำหรับโหมด 3 สามารถโปรแกรมได้โดยใช้โอเวอร์โฟว์ของไทเมอร์ 1 โดยตรงสำหรับโหมดข้อมูลที่1 อัตราเร็วที่สูงขนาด 83333 บิตต่อวินาท จะมีความเป็นไปได้ใการใช้ 16 MHz คริสตอลอัตราเร็วนี้สามารถเข้ากันได้กับ RS 485 สายส่งเกลียวคู่



รูปที่ 2.1 การสื่อสารอนุกรมโหมดที่1 และ โหมดที่ 3

การส่งข้อมูลโดยใช้โหมด 2 และ 3 จะแสดง 11 บิตต่อคาร์แรคเตอร์ ซึ่งแสดงในรูปที่ 2.1 คาร์แรกเตอร์จะเริ่มต้นด้วยบิตเริ่มต้น (start bit) ซึ่งจะเป็นการส่งผ่านจากสูงไปต่ำ (high-to-low transition) ที่ระยะบิตสุดท้ายตามด้วย 8 ข้อมูลบิตที่สำคัญน้อยที่สุดอย่างแรก บิตที่ 10 ของคาร์แลคเตอร์เป็นบิตที่สามารถโปรแกรมได้ซึ่งจะตามด้วยบิตหยุด (stop bit) บิตหยุดยังคงอยู่ในระดับสูงสำหรับค่าอย่างน้อยของหนึ่งระยะบิต

จากรูป 2.1จะแสดงเพียงความแตกต่างระหว่างโหมด 1 และ โหมด 2 และ 3 การส่งข้อมูลคือการเพิ่มของบิต 10 ซึ่งสามารถโปรแกรมได้ในโหมด 2 และ 3

เมื่อ 8051 ส่งคาร์แรคเตอร์ในโหมด 2 และ 3 ข้อมูล 8 บิตไม่ว่าค่าใดจะถูกโหลดเข้าใน SBUF บิตที่ 10 จะมาจากค่าของบิต SCON.3 มีชื่อว่า TB8 และเมื่อส่งครบบิต TI (SCON.3)จะถูกเซต

คาร์แรคเตอร์ที่รับมาโดยใช้โหมด 2 และ 3 จะมี 8 บิตข้อมูลที่อยู่ใน SBUF และ บิตที่ 10ใน SCON.2 ที่เรียกว่า RB 8 ถ้าเงื่อนไขเป็นจริง

มี 2 เงื่อนไขที่ประยุกต์ใช้เพื่อรับคาร์แรคเตอร์ เงื่อนไขแรกคืออินเทอร์รับบิต RI (SCON . 0) ต้องถูกเคลียร์ก่อนที่บิตสุดท้ายของคาร์แรคเตอร์จะถูกได้รับ เงื่อนไขที่ 2 คือ บิต SM 2 (SCON . 5) ต้องเป็น 0 หรือบิตที่ 10 ต้องเป็น 1 ถ้าเป็นไปตามเงื่อนไขเหล่านี้แล้วต่อมา ข้อมูล 8 บิตจะถูกโหลดใน SBUF บิตที่ 10 อยู่ใน RB 8 และรีซีฟอินเทอร์รัปต์บิต RI ถูกเซต ถ้าไม่เป็นไปตามเงื่อนไขเหล่านี้ คาร์แรคเตอร์ก็จะถูกละเลยไปและวงจรจะรับรอบิตเริ่มต้น (start bit) ต่อไป

เงื่อนไขที่สำคัญคือเงื่อนไขที่ 2 ถ้า RI ถูกเซตแล้วต่อมาซอฟต์แวร์ไม่อ่านข้อมูลก่อนหน้านั้น(หรือลืมทำการรีเซต RI) และมันก็จะไม่เขียนข้อมูล การเคลียร์ SM 2 ให้เป็น 0 จะอนุญาตให้มีการรับคาร์แรคเตอร์ของมัลติโปรเซสเซอร์ คาร์แรคเตอร์ซึ่งส่งในโหมด2และ3 การเซต SM 2 เป็น 1 จะป้องกันการรับคาร์แรคเตอร์ที่มีบิต 10 เท่ากับ 0 อีกทางหนึ่ง ถ้าบิต 10 เป็น 1 การรับจะเกิดขึ้นเสมอโดยไม่สนใจค่า SM 2 ถ้าบิต 10 เป็น 0 แล้วตัวรับที่มีบิต SM 2 เซตเป็น 0 เท่านั้นที่ถูกอินเทอร์รัปต์

โหมด 2 และ 3 ได้ถูกรวมเข้าใน 8051 โดยเฉพาะเพื่อเพิ่มประโยชน์ใช้สอยของมัลติเปิล8051s ซึ่งต่อเข้ากับลูป(loop)ทั่วไปในมัลติโปรเซสเซอร์ คำว่ามัลติโปรเซสซิ่ง(multiprocessing) หมายถึงโปรเซสเซอร์หลายๆตัวทำงานร่วมกันและติดต่อถึงกันดังนั้นข้อมูลจึงสามารถแลกเปลี่ยนกันได้ เมื่อโปรเซสเซอร์ต่อเข้ากับลูปแล้วโดยทั่วไปจะมีตัวควบคุม (controlling) หรือ ทอล์คเกอร์ โปรเซสเซอร์ (talker processor) ซึ่งสั่งการส่วนอื่นในลูปนั้น (listener)

ลักษณะสำคัญอย่างหนึ่งของทอล์คเกอร์-ลิสเทนเนอร์ลูปคือลำดับการส่งข้อมูลระหว่างทอล์คเกอร์กับแต่ละลิสเทนเนอร์ข้อมูลทั้งหมดที่ถูกกระจายไปโดยทอล์คเกอร์จะถูกรับโดยลิสเทนเนอร์ทั้งหมดถึงแม้ว่าข้อมูลนั้นจะตั้งใจส่งให้แก่ลิสเทนเนอร์เพียงบางคนเวลาที่ข้อมูลถูกกระจายไปนั่นหมายถึงมันจะถูกใช้โดยลิสเทนเนอร์ทั้งหมด

มีหลายวิธีที่จะแก้ปัญหาแอดเดรสซิ่ง ระบบที่ใช้มาตราฐาน UART Technology เช่นโหมด 1 สามารถกำหนดแอดเดรสหนึ่งๆแก่ลิสเทนเนอร์ทั้งหมดได้ แต่ละข้อความจากทอล์คเกอร์สามารถเริ่มต้นด้วยแอดเดรส

ของลิสเทนเนอร์ที่เจาะจงซึ่งได้ตั้งใจส่งถึงเมื่อข้อความได้ถูกส่งลิสเทนเนอร์ทั้งหมดก็จะเข้าถึงข้อมูลและจะกระทำการต่อเมื่อแอดเดรสที่นำหน้าข้อความนั้นตรงกันกับแอดเดรสที่กำหนดไว้ ถ้าข้อความได้ถูกส่งตามลำดับลิสเทนเนอร์ก็จะเสียเวลาเข้าถึงข้อความที่ไม่ต้องการ

โหมด 2 และ 3 ลดเวลาโปรเซสโดยทำให้รีเซพชั่นคาร์แรคเตอร์สามารถอยู่ในที่ซึ่ง SM 2ในลิสเทนเนอร์และบิต 10 ในคาร์แรคเตอร์ที่ส่งลิสเทนเนอร์ทั้งหมดโดยเริ่มต้นจะมี SM 2 เซตเท่ากับ 0 หรือการรีเซตแบบธรรมดาและรับข้อความมัลติโปรเซสเซอร์ทั้งหมดแต่ละลิสเทนเนอร์จะมีแอดเดรสเป็นของตัวเอง ทอล์คเกอร์ใช้แอดเดรสของแต่ละลิสเทนเนอร์ที่ไม่สนใจและสั่งการให้เซต SM 2 ให้เป็น 1 โดยปล่อยให้ลิสเทนเนอร์ที่ต้องการติดต่อด้วย ให้เคลียร์ SM 2เป็น 0 คาร์แรคเตอร์ทั้งหมดจากทอล์คเกอร์ถึงลิสเทนเนอร์หนึ่งๆถูกส่งด้วยบิตที่ 10 เซตเป็น 0 การสื่อสารกับลิสเทนเนอร์ทั้งหมดได้ถูกทำโดยเซตบิต 10 ให้เป็น 1 ซึ่งทำให้สามารถรับคาร์แรคเตอร์โดยไม่สนใจสถานะของ SM2

อีกวิธีการหนึ่งคือการเซต SM2 ให้เป็น 1 ตลอด ข้อความแอดเดรสทั้งหมดเป็นหนึ่งในตำแหน่งของบิตที่ 10 ดังนั้นลิสเทนเนอร์ทุกตัวจะรับและจัดการข้อความแอดเดรสเพื่อดูว่าทอล์คเกอร์ต้องการให้ทำอะไร ลิสเทนเนอร์ที่ถูกเลือกจะถูกสั่งในข้อความแอดเดรสเพื่อเซต SM2 ให้เป็น 0และการสื่อสารข้อมูลจะตามมาด้วยบิตที่ 10 ถูกเคลียร์ให้เป็น 0

มัลติโปรเซสซิ่งจะทำงานได้ดีมากเมื่อมีการติดต่อกับลิสเทนเนอร์เพียงหนึ่งตัวเพราะในการส่งข้อมูลทีละน้อยบ่อยๆจะทำให้ใช้แอดเดรสหนักมากซึ่งจะทำให้เกิดการรบกวนในการโปรเซสของลิสเทนเนอร์และข้อความแอดเดรส

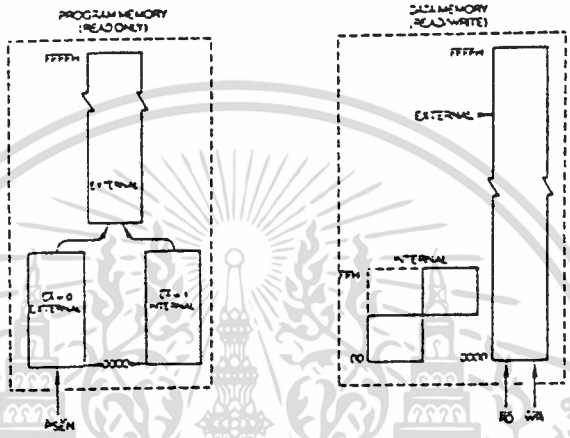
2.3 MCS-51

2.3.1 การจัดการหน่วยความจำของ 8051

หน่วยความจำของ 8051 แบ่งออกไว้เป็น 2 แบบตามลักษณะของการใช้งานคือ

2.3.1.1 หน่วยความจำโปรแกรม (Program Memory) เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปรหัสภาษาเครื่อง (Machine Language) ซึ่งต้องการให้ 8051 ทำงาน เมื่อ 8051 ทำงานก็จะอ่านข้อมูลที่เก็บในหน่วยความจำประเภทนี้เข้าไปถอดรหัสแล้ว สร้างสัญญาณควบคุมส่วนอื่น ๆ ตามการทำงานของแต่ละคำสั่งนั้น ๆ ตามการทำงานของแต่ละคำสั่งนั้น หน่วยความจำแบบนี้จะต้องเป็น รอม (ROM: Read Only Memory) และผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของหน่วยความจำเป็นรหัสภาษาเครื่องของ 8051 ตามลำดับการทำงานที่ต้องการ หน่วยความจำแบบ รอม ซึ่งเป็นแบบเมื่อปิดไฟแล้วข้อมูลก็ไม่มีการสูญหาย (Non volatite) การเขียนข้อมูลลงไปบน รอม จะต้องใช้เครื่องมือพิเศษ ในระหว่างการทำงานของ 8051 ผู้ใช้จะไม่สามารถใช้คำสั่งทำการเขียนข้อมูลลงในหน่วยความจำแบบนี้ได้ จำนวนตำแหน่งสูงสุดของหน่วยความจำแบบนี้ที่ 8051 จะใช้งานได้คือ 65536 ตำแหน่ง ค่าของตำแหน่ง (Address) จะเขียนเป็นเลขฐาน 16 ได้ตั้งแต่ 0000H ถึง FFFFH หน่วยความจำตำแหน่ง 0000H ถึง 0FFFH จำนวน 4 กิโลไบท์ นั้นผู้ใช้จะเลือกได้ว่าเป็นตำแหน่ง ของ รอม ที่อยู่ภาย

ในหรือภายนอก 8051 (ไมโครคอนโทรลเลอร์เบอร์อื่นๆเช่น 8052จะมีขนาดของรอมส่วนนี้ได้ถึง8กิโลไบท์ ตำแหน่ง 0000H ถึง 1FFFH) ถ้าต้องการให้ 8051 ทำงานตามคำสั่งที่เก็บไว้ใน รอม ภายใน 8051 ก็ให้ป้อนสัญญาณสภาวะลอจิกสูง (High :1) เข้าที่ขา EA ของ 8051 แต่ถ้าต้องการให้ทำงานในโปรแกรมที่เก็บไว้ใน รอม ภายนอก 8051 ก็ให้ต่อลอจิกต่ำ (Low :0) เข้าที่ขา EA ของ 8051 ส่วนหน่วยความจำที่ตำแหน่ง 1FFFH ถึง FFFFH จะต้องอยู่ภายนอก 8051 เสมอ ดังแสดงในแผนภูมิหน่วยความจำ (Memory Map) ดังรูปที่ 2.2



รูปที่ 2.2 แผนภูมิหน่วยความจำของ 8051

หน่วยความจำภายใน (Internal Memory) หมายถึงหน่วยความจำนั้นอยู่ภายใน 8051 ส่วน หน่วยความจำภายนอก (External Memory) หมายถึงหน่วยความจำนั้นอยู่ภายนอก 8051

- 8031 จะไม่มี รอม ขนาด 4 กิโลไบท์อยู่ภายใน ผู้ใช้จะต้องเลือกการใช้งานโปรแกรมเมมโมรี่ อยู่ภายนอกวงจรรวมทั้งหมด 64 กิโลไบท์

- 8051 จะมี รอม ขนาด 4 กิโลไบท์ ถ้าต้องการเก็บคำสั่งควบคุมการทำงานไว้ในหน่วยความจำส่วนนี้ จะต้องส่งโปรแกรมคำสั่งไปให้โรงงานผู้ผลิตทำการเขียนใส่ใน รอม ให้ตั้งแต่ในขั้นตอนของการผลิตวงจรรวม ผู้ใช้ไม่สามารถแก้ไขโปรแกรมเองได้ ถ้าจะนำมาใช้งานโดยเก็บโปรแกรมไว้ในหน่วยความจำช่วง 4 กิโลไบท์แรก อยู่ภายนอกก็สามารถทำได้ โดยการต่อ ROM ไว้ภายนอก แล้วต่อขา EA ของ 8051 ไว้กับสัญญาณที่สภาวะลอจิกเป็น 0

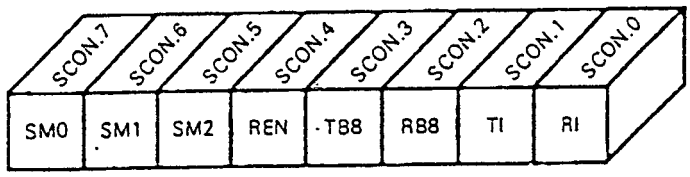
- 8751 จะมีหน่วยความจำขนาด 4 กิโลไบท์เป็นแบบ อีพรอม (EPROM :Erasable Program Read Only Memory) อยู่ภายในวงจรรวมเอาไว้ ใช้เก็บโปรแกรมคำสั่งที่จะให้ 8751 ทำงาน ผู้ใช้สามารถเขียนคำสั่งลงไป

ใน อีพรอม ได้เองโดยใช้เครื่องมือที่เรียกว่า เครื่องโปรแกรม อีพรอม (EPROM Programmer) และผู้ใช้สามารถแก้ไขโปรแกรมที่อยู่ใน อีพรอม ได้โดยการล้างข้อมูลในทุกตำแหน่งของ อีพรอม ออกด้วยการฉายแสงอุลตร้าไวโอเลต (Ultraviolet) ผ่านกระจกใสบนวงจรรวม เข้าไปยังวงจรภายใน ตามเวลาที่กำหนดในคู่มือเฉพาะ (Data Sheet) ของ 8751 จากนั้นก็ใช้เครื่องโปรแกรม อีพรอม เขียนโปรแกรมลงไปใหม่ 8751 นี้ จะสะดวกมากสำหรับการพัฒนาโปรแกรม

2.3.1.2 หน่วยความจำข้อมูล (Data Memory) เป็นหน่วยความจำที่ 8051 จะใช้สำหรับพัก, เก็บข้อมูลแล้วเรียกมาใช้ใหม่ในระหว่างการทำงานของ 8051 การอ่านหรือเขียนข้อมูลจากหน่วยความจำจะกระทำได้โดยคำสั่งที่เก็บไว้ในโปรแกรมเมมโมรี่ หน่วยความจำหลักนี้ (Random Access Memory ; RAM) ถ้ามีไฟเลี้ยงอยู่ข้อมูลที่เก็บไว้ไม่สูญหาย และถ้าปิดเครื่องหรือไม่จ่ายไฟให้แก่ แรม ข้อมูลใน แรม ก็จะสูญหายไป การสูญหายของข้อมูลไม่ได้หมายความว่าไม่มีอะไรอยู่เลย แต่เป็นการที่มีข้อมูลใหม่ที่ไม่ใช่ข้อมูลที่เก็บไว้เดิมเข้ามาอยู่แทนที่เช่นเดิม เก็บตำแหน่งข้อมูล 18 H ไว้ที่ตำแหน่ง 1900 H เมื่อปิดไฟแล้วเปิดใหม่ข้อมูลที่ตำแหน่ง 1900 H จะไม่ใช่ 18 H อาจเป็นค่าอะไรก็ได้ ซึ่งเรียกการเกิดลักษณะนี้ว่าข้อมูลสูญหายไป หน่วยความจำข้อมูล ของ 8051 จะมีอยู่ 2 ชุด ชุดหนึ่งอยู่ภายใน 8051 จำนวน128 ไบต์ ที่ตำแหน่ง 00 H ถึง 7FH (เบอร์ 8052 จะมี 256 ไบต์อยู่ที่ตำแหน่ง 00H ถึง FFH) และอีกชุดหนึ่งจะต้องอยู่ภายนอกของวงจรรวม 8051 มีได้สูงสุด 65536 ไบต์ (64 กิโลไบต์) อยู่ที่ตำแหน่ง 0000 H ถึง FFFF H ดังแสดงในรูป หน่วยความจำข้อมูล ภายใน 8051 ที่ตำแหน่ง 80 H ถึง FF H ไม่ได้มีอยู่ทุกตำแหน่ง จะมีเฉพาะในบางตำแหน่ง ซึ่งเรียกหน่วยความจำบางตำแหน่งนี้ว่า รีจิสเตอร์ที่มีหน้าที่พิเศษ (Special Function Registor : SFR) เพราะจะใช้หน่วยความจำเหล่านี้สำหรับงานพิเศษเท่านั้น แต่ละตำแหน่งของหน่วยความจำแบบรีจิสเตอร์ที่มีหน้าที่พิเศษนี้อาจเป็น แรม หรือวงจรนับ (Counter) วงจรตั้งเวลา (Timer) ก็ได้เช่น เป็น ไทม์เมอร์ 0 , ไทม์เมอร์ 1 ดังนั้นใน 8051 จึงไม่ถือว่ารีจิสเตอร์ที่มีหน้าที่พิเศษ เป็น หน่วยความจำข้อมูล ถ้าเป็น 8052 ซึ่งมีหน่วยความจำข้อมูล ขนาด 256 ไบต์ จะใช้บางตำแหน่งของหน่วยความจำช่วงตำแหน่ง 80 H ถึง FF H เป็น รีจิสเตอร์ที่มีหน้าที่พิเศษ ส่วนตำแหน่งอื่นที่เหลือก็เป็น แรม เหมือนกับหน่วยความจำช่วง 00 H ถึง FF H นั่นเอง

2.3.2 SCON รีจิสเตอร์ใช้งานเฉพาะ (Serial Port Control Register)

แต่ละบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON จะใช้สำหรับควบคุมและตรวจสอบการทำงานของพอร์ตการสื่อสารอนุกรมใน MCS-51 ดังนั้นก่อนใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรม ผู้เขียนโปรแกรมจำเป็นต้องทราบความหมายของบิตต่างๆในรีจิสเตอร์ตัวนี้ แต่ละบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON มีความหมายดังแสดงในรูปที่ 2.3



รูปที่ 2.3 แสดงรีจิสเตอร์ใช้งานเฉพาะ SCON

โดยรีจิสเตอร์ใช้งานเฉพาะ SCON เข้าถึงข้อมูลระดับบิตได้

บิต	ชื่อบิต	
SCON.7	SM0	บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดต่างๆ
SCON.6	SM1	บิตเลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดต่างๆ 00 โหมด 0 : ทำงานเป็นชิฟต์รีจิสเตอร์อัตราเร็วในการรับหรือส่งข้อมูลเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์ 01 โหมด 1 : 8 บิต UART อัตราเร็วในการรับส่งข้อมูลกำหนดเองได้ 10 โหมด 2 : 9 บิต UART อัตราเร็วในการรับส่งข้อมูลเท่ากับ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ 11 โหมด 3 : 9 บิต UART อัตราเร็วในการรับส่งข้อมูลกำหนดเองได้
SCON.5	SM2	บิตใช้เลือกการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 2 และ 3 เพื่อใช้ติดต่อระหว่างซีพียูด้วยกันเอง 1 : ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมในการติดต่อระหว่างซีพียูด้วยกันเอง เมื่อหากข้อมูลบิตที่ 9 ที่ได้ระบุมีค่าเป็น 0 (ดาต้าไบต์) บิต RI จะไม่ถูกเซต แต่หากข้อมูลบิตที่ 9 มีค่าเป็น 1 (แอดเดรสไบต์) บิต RI จะถูกเซต 0 : ใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 และโหมด 3ตามปกติ

ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 หากบิต SM2 ถูกเซต บิต RI จะไม่ถูกเซตจนกว่าบิตสิ้นสุดของข้อมูลจะถูกรับเข้ามา

ในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 บิตนี้ควรถูกเคลียร์ให้เป็น 0

SCON.4	REN	บิตควบคุมการอนุญาตให้มีการรับข้อมูล ดังนี้ 1 : อนุญาตให้มีการรับข้อมูลจากภายนอกได้ 0 : ไม่อนุญาตให้มีการรับข้อมูลจากภายนอก
SCON.3	TB8	บิตข้อมูลบิตที่ 9 ซึ่งจะถูกส่งออกไปในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 การเซตหรือเคลียร์กระทำด้วยคำสั่งในโปรแกรมเท่านั้น
SCON.2	RB8	บิตข้อมูลบิตที่ 9 ที่ได้รับเข้ามาจากภายนอกในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 2 และ 3 ส่วนในการทำงานโหมด 1 ถ้าบิต SM2 = 0 บิตนี้จะเป็นบิตสิ้นสุดของข้อมูลที่รับเข้ามาได้ และไม่ถูกกำหนดการใช้งานในโหมด 0
SCON.1	TI	บิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากส่งข้อมูล ถูกเซตโดยฮาร์ดแวร์เมื่อข้อมูลบิตที่ 8 ถูกส่งออกไปแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่นๆ จะถูกเซตโดยฮาร์ดแวร์เมื่อเริ่มส่งบิตสิ้นสุดของข้อมูลออกไป และจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น
SCON.0	RI	บิตบอกสถานะสัญญาณบิตบอกสถานะสัญญาณอินเตอร์รัปต์ที่เกิดจากรับข้อมูล ถูกเซตโดยฮาร์ดแวร์เมื่อได้รับข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 หรือจุดครึ่งทางของช่วงรับบิตสิ้นสุดของข้อมูลในการทำงานโหมดอื่น (มีข้อยกเว้นในกรณีใช้พอร์ตสื่อสารข้อมูลแบบอนุกรมติดต่อระหว่างซีพียูด้วยกันเอง) และจะต้องถูกเคลียร์โดยคำสั่งในโปรแกรมเท่านั้น

นอกจากนี้เราจะใช้บิตที่ 9 สำหรับการรับและการส่งข้อมูล (บิต TB8 และ RB8) อยู่ด้วย

2.3.3 อัตราบอด (Baud Rate)

อัตราบอด หมายถึง อัตราเร็วในการรับหรือส่งข้อมูล โดยใน MCS-51 ค่าอัตราเร็วในการรับและส่งข้อมูลจะมีค่าขึ้นอยู่กับการทำงานในโหมดต่างๆของพอร์ตสื่อสารข้อมูลแบบอนุกรม

2.3.3.1 การใช้งานไทม์เมอร์ 1 เป็นตัวกำหนดอัตราบอด

อัตราบอดที่ได้จะถูกกำหนดด้วยอัตราการเกิดโอเวอร์โฟลว์ (overflow) ของไทม์เมอร์ 1 และขึ้นอยู่กับบิต SMOD ในรีจิสเตอร์ PCON ซึ่งเขียนเป็นสมการที่ใช้คำนวณหาอัตราบอดได้ดังนี้

$$\text{อัตราบอด} = \frac{2^{SMOD}}{} * \text{อัตราการเกิดโอเวอร์โฟลว์ของไทม์เมอร์ 1}$$

เนื่องจากเมื่อเกิดโอเวอร์โฟลว์ในไทม์เมอร์ตัวใด จะทำให้เกิดสัญญาณอินเตอร์รัปต์เพื่อบอกให้ซีพียูทราบ ดังนั้นเมื่อเรานำไทม์เมอร์ 1 มาเป็นตัวกำหนดอัตราบอด จึงควรห้ามการเกิดอินเตอร์รัปต์ขึ้นในระหว่าง

การรับส่งข้อมูล และเนื่องจากตัวไทม์เมอร์ 1 เองยังสามารถถูกกำหนดให้ทำงานเป็นไทม์เมอร์หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง ซึ่งมีโหมดการทำงานย่อยลงไปอีก 4 โหมดแต่การใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมที่พบบ่อยที่สุดนั้นไทม์เมอร์ 1 จะถูกกำหนดให้ทำงานเป็นไทม์เมอร์ในโหมด 2 (Auto-Reload) ในกรณีนี้อัตราบอดจะถูกกำหนดโดยสมการดังนี้

$$\text{อัตราบอด} = \frac{2^{SMOD} * \text{ความถี่ออสซิลเลเตอร์ที่ใช้}}{32 * 12 * [\,256 - (\,TH1\,)\,]}$$

ดังนั้นค่าที่ต้องโหลดไปไว้ยังรีจิสเตอร์ TH1 เพื่อให้ได้ค่าอัตราบอดจะสามารถคำนวณได้ดังนี้

$$TH1 = 256 - \left[\, \frac{2^{SMOD} * \text{ความถี่ออสซิลเลเตอร์ที่ใช้}}{384 * \text{อัตราบอด}} \,\right]$$

เราสามารถสร้างอัตราบอดค่าต่ำๆด้วยไทม์เมอร์ 1 ได้โดยปล่อยให้ไทม์เมอร์ 1 อินเตอร์รัปต์ซีพียูได้ และกำหนดการทำงานให้เป็นไทม์เมอร์ขนาด 16 บิต (โหมด 1) และใช้ไทม์เมอร์ 1 อินเตอร์รัปต์ซีพียูเพื่อโหลดค่าใหม่เองด้วยซอฟต์แวร์ได้ (ไม่สามารถทำงานแบบ ออโต้รีโหลด)

ตารางที่ 2.1 เป็นอัตราบอดค่าต่างๆที่ใช้กันมากและบอกว่าสามารถใช้ได้จากไทม์เมอร์ 1 อย่างไร

baud rate	ความถี่ออสซิลเลเตอร์	บิต SMOD	ไทม์เมอร์ C/T	ไทม์เมอร์ โหมด	ค่าที่โหลด
Mode 0 Max: 1 MHZ	12 MHZ	X	X	X	X
Mode 2 Max: 375K	12 MHZ	1	X	X	X
Modes 1,3: 62.5K	12 MHZ	1	0	2	FFH
19.2K	11.059 MHZ	1	0	2	FDH
9.6K	11.059 MHZ	0	0	2	FDH
4.8K	11.059 MHZ	0	0	2	FAH
2.4K	11.059 MHZ	0	0	2	F4H
1.2K	11.059 MHZ	0	0	2	E8H
137.5K	11.986 MHZ	0	0	2	1DH
110K	6 MHZ	0	0	2	72H
110K	12 MHZ	0	0	1	FEEBH

จากตารางที่ 2.1 จะเห็นว่าในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 จะมีความเร็วในการส่งมากที่สุดเมื่อเปรียบเทียบกับโหมดอื่นที่ความถี่คริสตอลค่าเดียวกัน และจะเห็นหากเลือกใช้คริสตอลความถี่ 11.0592 MHz จะสามารถตั้งค่าอัตราบอดในโหมด 1 และ 3 เป็นค่ามาตรฐานที่ใช้กันทั่วไปได้ เช่น 1200, 2400, 4800, 9600, 19200 จึงเป็นเหตุผลสำคัญที่ในระบบควบคุมส่วนใหญ่เลือกใช้คริสตอล ความถี่ 11.0592 MHz มากกว่า 12 MHz

ในตารางที่ 2.1 นอกจากจะแสดงค่าอัตราบอดค่าต่างๆเปรียบเทียบให้เห็นแล้ว ตารางนี้ยังแสดงค่าที่ต้องโหลดไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ TH1 ที่ค่าอัตราบอดมาตรฐานต่างๆให้ทราบอีกด้วย ผู้เขียนโปรแกรมสามารถนำค่านี้ไปใช้ได้เลยแต่หากต้องการใช้ค่าอัตราบอดอื่นๆที่นอกเหนือไปจากนี้จะต้องคำนวณค่าที่โหลดให้รีจิสเตอร์ใช้งานเฉพาะ TH1 จากสมการที่แสดงข้างต้นนั่นเอง ดังในตัวอย่าง

ความถี่	=	11.0	MHz
บอร์ดที่ต้องการ	=	19.2	kBaud

$$TH1 = 256 - \frac{2 * (1.0 * 10^{6})}{32*12*19200}$$

$$= 253$$

$$= DH$$

2.3.4 <u>ไทม์เมอร์และเคาน์เตอร์</u> **(Timer and Counter)**

ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 **มีรีจิสเตอร์พิเศษที่สามารถเลือกใช้งานเป็นไทม์เมอร์หรือ**เคาน์เตอร์อย่างใดอย่างหนึ่ง (นับจำนวนแมชชีนไซเคิลหรือนับจำนวนพัลส์ที่เกิดขึ้นภายนอกชิป) **รีจิสเตอร์ชนิดนี้มีขนาด 16 บิต ซึ่งในโครงงานนี้จะใช้ไทม์เมอร์ 1 โหมด 2 โดยรีจิสเตอร์ที่ใช้เป็นไทม์เมอร์ 1 ประกอบขึ้นจากรีจิสเตอร์ใช้งานเฉพาะ TL1, TH1**

ไทม์เมอร์ : ค่าในรีจิสเตอร์ที่ใช้เป็นไทม์เมอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าทุกๆแมชชีนไซเคิล ดังนั้นจึงสามารถคิดว่าในขณะทำงานเป็นไทม์เมอร์หมายถึงใช้รีจิสเตอร์เป็นตัวนับจำนวนแมชชีนไซเคิลได้ และเนื่องจากใน 1 แมชชีนไซเคิลใดๆของ MCS-51 ประกอบไปด้วย 12 คาบสัญญาณออสซิลเลเตอร์ (oscillator period) ดังนั้นอัตราเร็วในการนับ (count rate) จึงมีค่าเป็น 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้

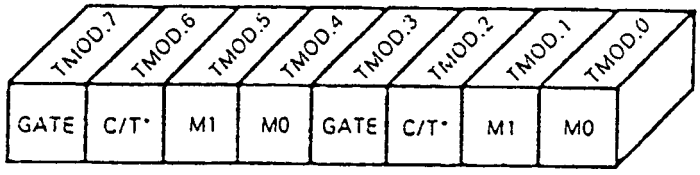
เคาน์เตอร์ : ค่าในรีจิสเตอร์ที่ใช้เป็นเคาน์เตอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าทีละหนึ่งเมื่อมีการเปลี่ยนสถานะซึ่งตรวจจับได้จากขา T0, T1 หรือ T2 (ใน 8052) ขึ้นกับรีจิสเตอร์ที่ถูกเลือกใช้งานเป็นเคาน์เตอร์ในขณะนั้น การตรวจสอบการเปลี่ยนสถานะจะตรวจเฉพาะในขณะที่สัญญาณมีการเปลี่ยนค่าจาก 1 เป็น 0 (1 to 0 trandition)

2.3.4.1 ไทม์เมอร์ 1

ผู้ใช้สามารถเลือกการทำงานให้เป็นไทม์เมอร์หรือเคาน์เตอร์อย่างใดอย่างหนึ่งโดยการกำหนดค่าบิต C/T ในรีจิสเตอร์ใช้งานเฉพาะ TMOD ดังแสดงในรูปที่ 2.4 ในไทม์เมอร์ 1 จะใช้บิต 6 โดยหากบิตนี้มีค่าเป็นหมายถึงเลือกให้รีจิสเตอร์ทำงานเป็นไทม์เมอร์ (นับจำนวนแมชชีนไซเคิล) ถ้าบิตนี้มีค่าเป็น 1 หมายถึงเลือกให้ทำงานเป็นเคาน์เตอร์ (นับจำนวนการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา T0 หรือ T1)

2.3.4.2 **รีจิสเตอร์ใช้งานเฉพาะ** TMOD (Timer/Counter Mode Control Register)

ไม่สามารถเข้าถึงข้อมูลระดับบิตได้



รูปที่ 2.4 รีจิสเตอร์ใช้งานเฉพาะ TMOD

บิตเกต เลือกการควบคุมให้รีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ทำงานโดยควบคุมจากฮาร์ดแวร์หรือซอฟต์แวร์ดังนี้

- เมื่อบิต TRx (TR0, TR1) และเกตถูกเซต ไทม์เมอร์หรือเคาน์เตอร์จะทำงานต่อเมื่อสถานะที่ขา INTx (INT0, INT1) มีค่าเป็น 1 (ควบคุมจากฮาร์ดแวร์)

- เมื่อบิตเกตถูกเคลียร์ ไทม์เมอร์หรือเคาน์เตอร์จะทำงานก็ต่อเมื่อบิต TRx ถูกเซตโดยไม่ขึ้นกับสถานะสัญญาณที่ขา INTx (ควบคุมจากซอฟต์แวร์)

C/T บิตเลือกการทำงานของรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ดังนี้

- 0 หมายถึงทำงานเป็นไทม์เมอร์ (นับจำนวนแมชชีนไซเคิล)

- 1 หมายถึงทำงานเป็นเคาน์เตอร์ (นับจำนวนพัลส์ภายนอกที่ขา Tx)

M1 บิตสำหรับเลือกโหมดการทำงานของไทม์เมอร์ 0 หรือ ไทม์เมอร์ 1

M2 บิตสำหรับเลือกโหมดการทำงานของไทม์เมอร์ 0 หรือ ไทม์เมอร์ 1

00 โหมด 0 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 13 บิต

01 โหมด 1 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 16 บิต

10 โหมด 2 : ไทม์เมอร์หรือเคาน์เตอร์ขนาด 8 บิต ที่มีการโหลดค่าเองเมื่อเกิดโอเวอร์โฟลว์

11 โหมด 3 : ไทม์เมอร์ 0

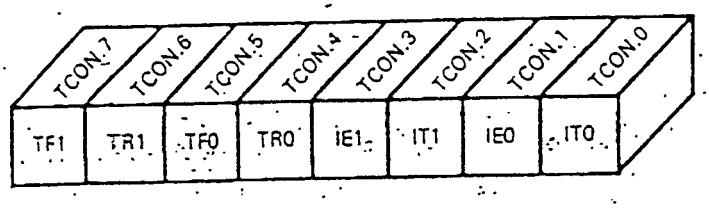
รีจิสเตอร์ TL0 ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ขนาด 8 บิต ที่ควบคุมการทำงานได้จากบิตของไทม์เมอร์ 0 เอง

รีจิสเตอร์ TH0 ใช้เป็นไทม์เมอร์ขนาด 8 บิต ทีควบคุมการทำงานได้จากบิตของไทม์เมอร์ 1

11 โหมด 3 : ไทม์เมอร์ 1 หยุดทำงาน (หยุดนับ)

2.3.4.3 รีจิสเตอร์ TCON (Timer/Counter Control Register)

เข้าถึงข้อมูลในระดับบิตได้



รูปที่ 2.5 รีจิสเตอร์ใช้งานเฉพาะ TCON

บิต

TF1 บิตแสดงการเกิดโอเวอร์โฟลว์ของไทม์เมอร์ 1 ถูกเซตเองเมื่อไทม์เมอร์ 1 เกิดโอเวอร์โฟลว์ และถูกเคลียร์เองเมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการอินเตอร์รัปต์

TR1 บิตควบคุมการนับของไทม์เมอร์ 1 ควบคุมจากโปรแกรม

1 ไทม์เมอร์ 1 เริ่มทำงานต่อ (นับต่อ)

0 ไทม์เมอร์ 1 หยุดทำงาน (หยุดนับสัญญาณนาฬิกาภายในหรือนับจำนวนพัลส์ภายนอก)

TF0 บิตแสดงการเกิดโอเวอร์โฟลว์ของไทม์เมอร์ 0 ถูกเซตเองเมื่อไทม์เมอร์ 0 เกิดโอเวอร์โฟลว์และถูกเคลียร์เองเมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการอินเตอร์รัปต์

TR0 บิตควบคุมการนับของไทม์เมอร์ 0 ควบคุมจากโปรแกรม

1 ไทม์เมอร์ 0 เริ่มทำงานต่อ (นับต่อ)

0 ไทม์เมอร์ 0 หยุดทำงาน

IE1 บิตแสดงสถานะสัญญาณอินเตอร์รัปต์ภายนอกชนิดที่ 1 จะถูกเซตเองโดยฮาร์ดแวร์เมื่อมีสัญญาณอินเตอร์รัปต์ และจะถูกเคลียร์เองโดยคำสั่ง RET1 ที่อยู่ในโปรแกรมส่วนบริการอินเตอร์รัปต์ เมื่ออินเตอร์รัปต์ที่เกิดขึ้นได้มาจากการตรวจสอบจากการเปลี่ยนสถานะของสัญญาณ

IT1 บิตเลือกประเภทการตรวจสอบสัญญาณอินเตอร์รัปต์ที่เกิดขึ้นที่ขา INT1 โดย

1 ตรวจสอบการเปลี่ยนระดับสัญญาณจาก 1 เป็น 0 ที่ขา INT1

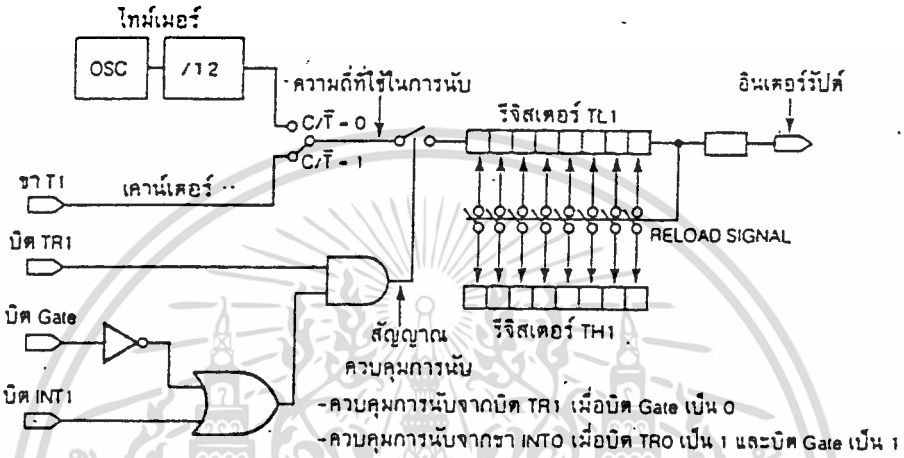
0 ตรวจสอบระดับของสัญญาณที่ขา INT1

IE0 บิตแสดงสถานะสัญญาณอินเตอร์รัปต์ภายนอกชนิดที่ 0 จะถูกเซตเองโดยฮาร์ดแวร์เมื่อมีสัญญาณอินเตอร์รัปต์ และจะถูกเคลียร์เองโดยคำสั่ง RET1 ที่อยู่ในโปรแกรมส่วนบริการอินเตอร์รัปต์ เมื่ออินเตอร์รัปต์ที่เกิดขึ้นได้มาจากการตรวจสอบจากการเปลี่ยนสถานะของสัญญาณ

IT0 บิตที่ใช้เลือกประเภทการตรวจสอบสัญญาณอินเตอร์รัปต์ที่เกิดขึ้นที่ขา INT1 เหมือนกับ IT1

2.3.4.4 การใช้งานในโหมดที่ 2

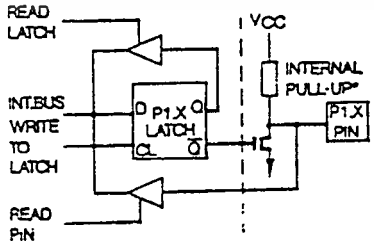
การทำงานในโหมดที่ 2 จะกำหนดให้รีจิสเตอร์ที่ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ถูกใช้ในการนับเพียง 8 บิต (จากรีจิสเตอร์ใช้งานเฉพาะ TLx) ที่มีการโหลดค่าเองด้วยค่าในรีจิสเตอร์ใช้งานเฉพาะ THx เมื่อเกิดโอเวอร์โฟลว์ในรีจิสเตอร์ TLx โดยค่าในรีจิสเตอร์ THx นี้สามารถกำหนดได้ล่วงหน้าโดยซอฟต์แวร์ และจะไม่เปลี่ยนแปลงเมื่อถูกโหลดไปไว้ในรีจิสเตอร์ TLx การทำงานในโหมด 2 มีดังแสดงในรูปที่ 2.6



รูปที่ 2.6 การทำงานของไทม์เมอร์/เคาน์เตอร์ 1 โหมด 2

2.4 พอร์ต 1 ไดร์ฟเวอร์

เป็นพอร์ตขนานขนาด 8 บิตคือขา P1.0 ถึง P1.7 (ขา 1-8) P1.0 หมายถึงบิต 0 ของพอร์ต 1 ซึ่งเป็นบิตที่สำคัญน้อยที่สุด และบิต P1.7 หมายถึงบิตที่ 7 ของพอร์ต 1 ซึ่งเป็นบิตที่สำคัญที่สุด โครงสร้างของพอร์ต 1 แต่ละบิตมีดังรูปที่ 2.7



รูปที่ 2.7 โครงสร้างของพอร์ต 1

ส่วนที่ 1 คือ พอร์ต 1 แลตช์ซึ่งจะมีการทำงานเหมือนส่วนที่ 1 ของพอร์ต 0 ส่วนที่ 2 คือ พอร์ต 1 ไดร์ฟเวอร์ ซึ่งจะมีตัวต้านทานต่ออยู่เป็น อินเทอร์นอล พูลอัพ (Internal Pull Up) พอร์ต 1 นี้จะใช้ทำหน้าที่เป็นตัวรับส่งข้อมูลเท่านั้นข้อมูลที่ส่งออกมาทางพอร์ต 1 จะถูกแลตช์ไว้แล้วส่งออกไปทางแต่ละขา ก่อนที่จะอ่านข้อมูลเข้าในทางพอร์ต 1 จะต้องเขียน 1 ไปยังทุกบิตของพอร์ต 1 เสียก่อนเพื่อให้ FET อยู่ในสภาวะปิดเสียก่อน มิฉะนั้นแล้วถ้ามีข้อมูล 0 ส่งออกมาค้างอยู่ที่ D-FF จะทำให้ FET อยู่ในสภาวะ เปิด ดังนั้นถ้าสัญญาณภายนอกส่งเข้ามาที่ขานี้ก็จะถูกลัดวงจรลงกราวนด์ โดยไม่สนใจว่าสภาวะลอจิกของสัญญาณที่เข้ามาจะเป็นอะไร ข้อมูลที่อ่านเข้าไปจึงจะเป็น 0 เสมอ

2.5 มาตรฐานของสายสัญญาณรับ-ส่งข้อมูลแบบอนุกรม

2.5.1 มาตรฐาน อาร์ เอส 232 ซี

อาร์ เอส 232 ซี คือมาตราฐานที่ถูกตีพิมพ์ใน ELECTRONIC INDUSTRIES ASSOSIATION (ELA) ในปี ค.ศ. 1969 RS มาจากคำว่า RECOMMENDED STANDARD และ 232 คือตัวเลขที่ใช้เจาะจงมาตราฐาน Cคือแสดงถึงว่ามันถูกออกแบบมาล่าสุด จุดประสงค์ของมาตราฐานนี้คือการกำหนดลักษณะทางไฟฟ้าเพื่อเชื่อมต่ออุปกรณ์ดาต้าเทอร์มินอล (DATA TERMINAL) หรือ DTE และอุปกรณ์การสื่อสารข้อมูล (DATA COMMUNICATION EQUIPMENT) หรือ DCEจากรูปที่2.8 ข้างล่างนี้จะเห็นการทำงานของเคเบิลเชื่อมต่อ อาร์ เอส 232 ซี กับคอมพิวเตอร์ (PC) ดูจากรูปนี้จะเห็นได้ว่า ตัวเชื่อม อาร์ เอส 232 ซี ได้มีบทบาทสำคัญมากในการสื่อสารของคอมพิวเตอร์ (PERSONAL-COMPUTER COMMUNICATION) มันเป็นการเชื่อมต่อที่สร้างความสะดวกให้แก่การส่งข้อมูลไปและรับมาจากคอมพิวเตอร์เครื่องอื่นโดยผ่านทางระบบโทรศัพท์

นอกจากนี้ความเร็วบิต และความยาวของเคเบิลที่ใช้ก็เกี่ยวข้องกับการทำงานของ อาร์ เอส 232 ซี กับคอมพิวเตอร์ด้วย มาตราฐานที่ใช้ในการส่งข้อมูลแบบอนุกรมคือช่วงระหว่าง 0 ถึง 20,000 บิทต่อวินาที ซึ่งเพียงพอกับกับความเร็วบิตของคอมพิวเตอร์ ซึ่งอยู่ระหว่าง50 ถึง 9600 บิทต่อวินาที มาตรฐานจำกัดความยาวของเคเบิลนั้นคือ 50 ฟุต ซึ่งควรจะครอบคลุมรูปร่างภายนอกของฮาร์ดแวร์คอมพิวเตอร์สื่อสารส่วนใหญ่ สายเคเบิลที่ยาวกว่า 50 ฟุตสามารถใช้ได้ (ส่วนใหญ่ใช้ในระบบเมนเฟรม (mainframe)) แต่การใช้งานแบบนั้นควรจะผ่านการทดสอบเพื่อทำให้แน่ใจถึงคุณภาพของสัญญาณก่อนนำออกใช้

2.5.1.1 ลักษณะของสัญญาณ อาร์ เอส 232 ซี

การที่จะให้ข้อมูลถูกส่งไปอย่างถูกต้องและอุปกรณ์ควบคุมนั้นได้ถูกใช้งานอย่างเหมาะสมเราจำเป็นต้องทำให้สัญญาณที่จะใช้เข้ากันได้เสียก่อน มาตราฐาน อาร์ เอส 232 ซีจะให้แรงดันซึ่งนับเป็นจำนวนโวลต์แก่ข้อมูลและควบคุมสัญญาณเพื่อที่จะตอบสนองความต้องการนี้ ดูได้จากตารางที่ 2.2 และ รูปที่ 2.8

อาร์ เอส 232 มีหน้าที่หลัก คือ เปลี่ยนแปลงสัญญาณไฟฟ้า ซึ่งได้แก่ สัญญาณดิจิตอลที่ได้จาก Universal Asynchronous Receiver Transmitter (UART) ไปเป็นระดับสัญญาณมาตรฐาน (EIA)

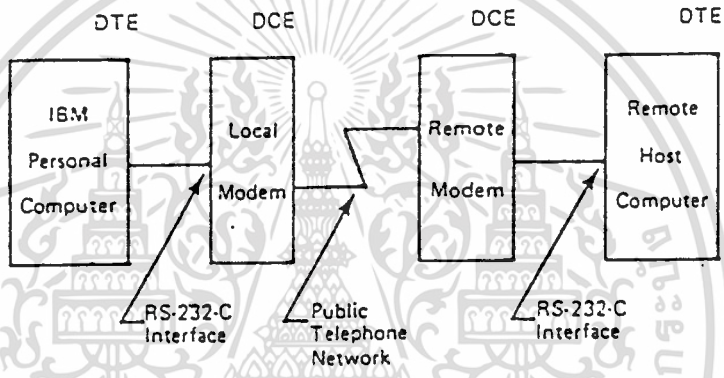
ลักษณะของสัญญาณทางไฟฟ้า (Electrical Characteristic) ที่ใช้กับ อาร์ เอส 232 จะเป็นดังนี้

2.5.1.1.1 ระดับโวลท์เตจของสัญญาณที่ใช้เป็นทั้งค่าบวก และค่าลบ โดยอยู่ระหว่าง 3โวลท์ และ 15 โวลท์

2.5.1.1.2 ระดับโวลท์เตจระหว่าง +3 โวลท์ และ -3 โวลท์ ไม่มีความหมาย

2.5.1.1.3 ค่าสัญญาณเป็นข้อมูลทางดิจิตอลสถานะหนึ่งเป็น marking state จะกำหนดระดับโวลท์เตจที่เป็นลบ และสถานะ 0 จะกำหนดด้วยโวลท์เตจที่เป็นบวก

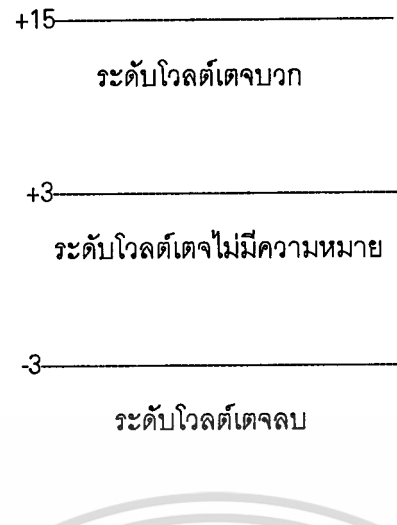
2.5.1.1.4 ถ้าสัญญาณนั้นเป็นสัญญาณควบคุมระดับโวลท์เตจที่เป็นบวกจะแสดงสภาวะเปิดของสัญญาณ และระดับโวลท์เตจที่เป็นลบจะแสดงสภาวะปิด



รูปที่ 2.8 การเชื่อมต่อโดยใช้ อาร์ เอส 232 ซี

ตารางที่ 2.2 มาตรฐานการแปลงโวลต์

การแปลงโวลต์	สถานทางโล่จิกฐาน 2	สถานะของสัญญาณ
บวก	0	space
ลบ	1	mark



รูปที่ 2.9 ระดับโวลต์เตจสัญญาณ อาร์ เอส 232 ซี

2.5.2 <u>มาตรฐาน อาร์ เอส 485 และ อาร์ เอส 422</u>

มาตรฐาน อาร์ เอส 485 นั้นมีความใกล้เคียงกับมาตรฐาน อาร์ เอส 422 แต่ต่างกันตรงที่ เอาท์พุท ไดร์เวอร์ (Out drivers) ได้ถูกออกแบบมาเพื่อเอาท์พุทแบบไตรสเตท (Tristate output) โดยลักษณะร่วมกันของมาตรฐาน อาร์ เอส 232 และ อาร์ เอส 485 มีดังนี้

มาตรฐาน อาร์ เอส 485 และ อาร์ เอส 422 จะใช้เส้นส่งสัญญาณแบบสัญญาณผลต่าง (Differential signaling) ซึ่งจะใช้คู่หนึ่งสำหรับส่งข้อมูล และใช้เส้นส่งสัญญาณอีกคู่หนึ่งรับสัญญาณจากภายนอกอีกด้วย (แต่ในโครงงานนี้จะใช้เพียงแค่อย่างละเส้น)

มาตรฐาน อาร์ เอส 422 ได้กำหนดว่าถ้าความต่างศักย์ระหว่างเส้นส่งสัญญาณคู่ใดๆ มีค่าเป็นบวก แสดงว่าเส้นส่งสัญญาณคู่นั้นมีค่าทางตรรกะเป็น 1 ซึ่งหมายความว่าสาย A จะต้องมีความต่างศักย์สูงกว่าสาย B ซึ่งจะไม่เหมือนกับในมาตรฐาน อาร์ เอส 232 ที่มีการใช้สัญญาณที่มีศักดาไฟฟ้าเป็นทั้งบวกและลบ ในมาตรฐาน อาร์ เอส 422 นี้จะใช้สัญญาณที่อยู่ในช่วง 0 ถึง +5 โวลต์ ซึ่งหมายความว่า อาร์ เอส 422 จะสามารถใช้ศักดาไฟฟ้าที่ใช้ในระบบไมโครโปรเซสเซอร์และวงจรอื่นๆ ได้

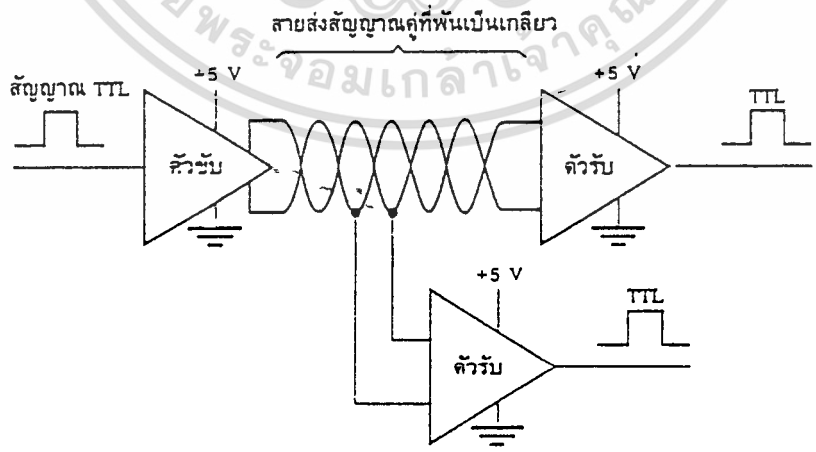
สัญญาณข้อมูลจะถูกส่งไปในรูปของผลต่างของระดับศักดาไฟฟ้าในสายส่งทั้งสองเส้นที่พันเกลียวกันไป เพราะฉะนั้นถึงแม้ว่าเราจะทราบค่าศักดาไฟฟ้าในสายส่งทั้งสองเส้นที่พันเกลียวกันไปก็ไม่มีประโยชน์อันใด ยกเว้นแต่ว่าค่าศักดาไฟฟ้าที่เทียบกับกราวด์ในสายส่งเหล่านี้จะมีค่ามากเกินกว่าที่วงจรรับสัญญาณจะทนได้ นั่นก็หมายความว่า อาร์ เอส 422 นี้สามารถทำงานได้ถึงแม้ว่าจะมีการรบกวนแบบโหมดร่วม (common mode) การรบกวนแบบโหมดร่วมคือ การเปลี่ยนแปลงค่าระดับศักดาไฟฟ้าในเส้นส่งสัญญาณทั้งสองเส้นจำนวนพอๆกัน แต่จะไม่ทำให้ค่าผลต่างของระดับศักดาไฟฟ้าในเส้นส่งสัญญาณทั้งสองนี้เปลี่ยนแปลง เพราะ

ฉะนั้นข้อมูล (ที่ถูกส่งไปในรูปของผลต่างของระดับศักดาไฟฟ้าในสายส่งสัญญาณทั้งสองนี้) จึงไม่มีการเปลี่ยนแปลง

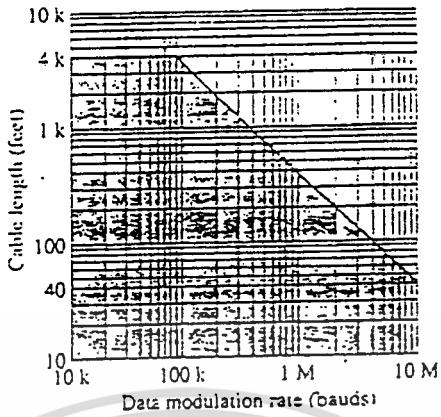
มาตรฐาน อาร์ เอส 422 ต่างจากมาตรฐาน อาร์ เอส 232 ตรงที่ว่ามาตรฐาน อาร์ เอส 422 จะใช้สายรับสัญญาณ 1 คู่ และสายส่งสัญญาณ 1 คู่เท่านั้น ไม่มีการระบุการใช้เส้นส่งสัญญาณควบคุมใดๆ และไม่ได้มีการกำหนดชนิดของตัวเชื่อมต่อที่ต้องใช้ วงจรผลต่าง (Differential circuit) ของ อาร์ เอส 422 แสดงได้ดังรูปที่ 2.10

เนื่องจากไม่มีการใช้สายส่งสัญญาณตอบรับ เราจึงต้องส่งข้อมูลควบคุมรวมไปกับข้อมูลจริงเพื่อให้การทำงานเป็นไปดังโปรโตคอล การไม่มีสายส่งสัญญาณตอบรับทำให้การส่งข้อมูลในระบบ อาร์ เอส 422 มีโอเวอร์เฮดสูงกว่าส่งข้อมูลในระบบ อาร์ เอส 232

การส่งข้อมูลในระบบ อาร์ เอส 422 ที่ใช้สัญญาณผลต่างทำให้เราสามารถส่งข้อมูลได้ด้วยอัตราที่เร็วกว่า และระยะทางไกลกว่าในระบบ อาร์ เอส 232 ระบบ อาร์ เอส 422 นี้สามารถส่งข้อมูลด้วยอัตรา 38.4 กิโลบอด ในระยะทางประมาณ 4000 ฟุต แต่ในระบบ อาร์ เอส 232 จะสามารถส่งข้อมูลด้วยอัตรา 38.4 กิโลบอด ภายในระยะทางเพียง 12.5 ฟุต โดยระยะทางสูงสุดของสายขึ้นอยู่กับอัตราข้อมูล, จำนวนสัญญาณรบกวน (noise) และความเพี้ยน (distortion) ที่สามารถทนได้ และความไม่สมดุลของสายเคเบิล ถ้าเราเพิ่มความยาวของเคเบิลจะเกิดความเพี้ยนมากขึ้น ดังนั้นสายเคเบิลจึงควรให้สั้นที่สุดเพื่อขจัดปัญหาที่อาจเกิดขึ้น ในรูปที่ 2.11 แสดงถึงความยาวของเคเบิลต่ออัตราข้อมูล



รูปที่ 2.10 ระบบที่ใช้มาตรฐาน อาร์ เอส 422 ในการส่งข้อมูล



รูปที่ 2.11 ความยาวของสายเคเบิลต่ออัตราบอด

2.6 ภาษาแอสเซมบลี

เนื่องจากคอมพิวเตอร์แบบดิจิตอล (digital computer) จะทำตามลำดับของชุดคำสั่งที่ผู้เขียนได้เขียนขึ้นที่เรียกว่า โปรแกรม ชุดของคำสั่งที่คอมพิวเตอร์จะเข้าใจและทำงานได้ตามต้องการนั้นจะต้องอยู่ในรูปของเลขฐานสอง เช่น 11011001 ชุดคำสั่งลักษณะนี้คอมพิวเตอร์เข้าใจได้โดยตรงนี้เรียกว่าภาษาเครื่อง (Machine lanuage) แต่การที่มนุษย์จะเขียนคำสั่งต่างๆด้วยเลขฐานสองทีเดียวนั้นทำได้ยากและเกิดข้อผิดพลาดได้ง่าย มนุษย์จึงเขียนโปรแกรมด้วยภาษาที่มนุษย์เข้าใจได้ง่ายแล้วมีตัวแปลภาษาจากภาษาที่มนุษย์เข้าใจไปเป็นภาษาที่เครื่องคอมพิวเตอร์เข้าใจ ภาษาที่มนุษย์เข้าใจได้ง่ายและนำมาเขียนโปรแกรมเรียกว่าเป็น ภาษาชั้นสูง (High level language) ได้แก่ ภาษาฟอร์แทน, ภาษาเบสิก หรือภาษาปาสคาล เป็นต้น คำสั่งที่จะควบคุมการทำงานของคอมพิวเตอร์ในภาษาเหล่านี้ มนุษย์จะเข้าใจได้ง่ายเพราะเป็นภาษาที่คล้ายคลึงกับภาษาอังกฤษที่ใช้กันในชีวิตประจำวัน เช่น ต้องการเอาตัวแปร A และ B บวกกันแล้วเก็บผลลัพธ์ไว้ในตัวแปร A ก็สามารถทำได้โดยใช้คำสั่ง

A := A+B ในภาษาปาสคาล

A = A+B ในภาษาเบสิกและฟอร์แทน

จะเห็นว่าลักษณะของคำสั่งในภาษาชั้นสูงแต่ละแบบก็แตกต่างกันดังนั้นถ้าผู้ใช้ต้องการเขียนโปรแกรมด้วยภาษาใด ก็จะต้องศึกษาโครงสร้างของโปรแกรมและชุดคำสั่งของภาษานั้นๆก่อน เมื่อเขียนโปรแกรมในภาษาชั้นสูงได้แล้วจะต้องป้อนเข้าไปในเครื่องคอมพิวเตอร์โดยใช้โปรแกรมประมวลคำ (Word Processor)หรืออาจใช้โปรแกรมอีดิเตอร์ สำหรับภาษาชั้นสูงๆนั้น โปรแกรมเหล่านี้จะทำการเก็บชุดคำสั่งที่ป้อนเข้าไปเก็บไว้เป็นแฟ้มข้อมูลที่เรียกว่าโปรแกรมต้นกำเนิด (Source program) โปรแกรมนี้จะถูกเปลี่ยนเป็นภาษาเครื่องโดยใช้

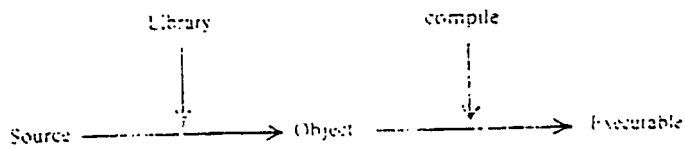
ตัวแปลโปรแกรม (Complier) ซึ่งภาษาชั้นสูงแต่ละภาษาก็จะมีตัวแปลโปรแกรมเฉพาะตัวของภาษานั้นๆการแปลภาษาขั้นสูงเป็นภาษาเครื่องมี 2 วิธี

2.6.1 การแปลแบบตัวถอดความ (Interpreter)

ตัวแปลภาษาแบบนี้จะอ่านโปรแกรมต้นกำเนิดมาทีละบรรทัดแล้วแปลเป็นภาษาเครื่องเพื่อทำงาน ตัวแปลภาษาที่ทำงานลักษณะนี้จะมีผลทำให้คอมพิวเตอร์ทำงานช้า เพราะตัวแปลโปรแกรมจะต้องแปลภาษาขั้นสูงเป็นภาษาเครื่องทุกครั้งที่จะมีการทำงาน แม้เป็นการทำงานวนรอบ ก็ยังต้องแปลเป็นภาษาเครื่องทุกครั้ง

2.6.2 ภาษาชั้นสูงทั้งหมดในโปรแกรมต้นกำเนิดจะถูกแปลเป็นรหัส สำหรับเรียกโปรแกรมย่อย (Subroutine) ที่เก็บไว้ในไลบรารีไฟล์ (Libraly file) ในช่วงการแปลนี้จะเรียกว่าการคอมไพล์ซึ่งจะทำให้ได้แฟ้มข้อมูลที่เก็บรหัสอันเกิดจากการแปลภาษาชั้นสูงเรียกว่าโปรแกรมประสงค์ (Object program) จากนั้นจะต้องมีโปรแกรมที่อ่านโปรแกรมประสงค์และแฟ้มข้อมูลในไลบรารีไฟล์เข้ามายังคอมพิวเตอร์เพื่อทำการถอดรหัสจากโปรแกรมประสงค์ โดยการใช้รหัสจากโปรแกรมประสงค์เป็นตัวชี้ไปยังโปรแกรมย่อยภาษาเครื่องที่เก็บไว้ในแฟ้มข้อมูลไลบรารี จากนั้นนำเอาโปรแกรมย่อยจากแฟ้มข้อมูลไลบรารีมาสร้างเป็นโปรแกรมภาษาเครื่องที่คอมพิวเตอร์สามารถเข้าใจได้ที่เรียกว่าโปรแกรมปฏิบัติการ (Executable program) วิธีนี้นิยมใช้กันมากในภาษาชั้นสูง ขั้นตอนการทำงานของวิธีนี้สามารถเขียนเป็นไดอะแกรมได้ดังรูปที่ 2.12

ภาษาชั้นสูงถึงแม้จะมีข้อดีที่มนุษย์เข้าใจได้ง่าย แต่มีปัญหาในเรื่องความเร็วในการทำงานของคอมพิวเตอร์ เพราะโปรแกรมย่อยที่เก็บไว้ในข้อมูลไลบรารีไม่สามารถทำให้สั้นกะทัดรัดได้เนื่องจากเขียนสำหรับงานหลายงาน ดังนั้นจึงต้องมีภาษาหนึ่งที่มนุษย์และคอมพิวเตอร์เข้าใจได้ง่าย คือภาษาแอสเซมบลีเป็นภาษาที่สูงกว่าภาษาเครื่องแต่ต่ำกว่าภาษาชั้นสูง วิธีการเขียนโปรแกรมภาษานี้ผู้ใช้จำเป็นต้องเข้าใจโครงสร้างภายในของตัวประมวลผลกลาง (Central Processing Unit) และรหัสคำสั่งช่วยจำ (Mnemonic) ของคอมพิวเตอร์นั้นๆ แต่อย่างไรก็ตามภาษาที่ใช้ในการเขียนโปรแกรมภาษาแอสเซมบลีนี้ก็ยังมีลักษณะคล้ายกับภาษาอังกฤษอย่างง่ายทำให้เขียนโปรแกรมได้สะดวก



รูปที่ 2.12 ลำดับการแปลโปรแกรมจากภาษาชั้นสูงเป็นโปรแกรมปฏิบัติการ

ในโปรแกรมภาษาแอสเซมบลีจะประกอบด้วยชุดคำสั่งที่เรียกว่ารหัสคำสั่งช่วยจำที่ป้อนเข้าไปเก็บในคอมพิวเตอร์ด้วยโปรแกรมประมวลคำเก็บไว้เป็นโปรแกรมต้นกำเนิด จากนั้นจะใช้โปรแกรมต้นกำเนิดที่เรียกว่าแอสเซมเบลอร์ทำการแปลภาษาแอสเซมบลีเป็นโปรแกรมภาษาเครื่องโดยตรงการแปลของแอสเซมเบลอร์มี 2 แบบคือ

แบบที่ 1 เรียกว่า One-pass Assembler โปรแกรมแอสเซมเบลอร์แบบนี้จะแปลรหัสคำสั่งช่วยจำเป็นภาษาเครื่องตั้งแต่บรรทัดที่ 1 จนถึงบรรทัดสุดท้าย โดยที่ในโปรแกรมต้นกำเนิดจะต้องไม่มีตัวแปรหรือสัญลักษณ์อื่นใดนอกรหัสคำสั่งช่วยจำ ทำให้ไม่ค่อยสะดวกสำหรับการเขียนโปรแกรม จึงไม่เป็นที่นิยมทั่วไป ยกเว้นในเครื่องไมโครคอมพิวเตอร์แบบบอร์ดเดี่ยว (Single Board Microcomputer)

แบบที่ 2 เรียกว่า Two-pass Assembler แอสเซมเบลอร์แบบนี้จะมีการทำงานแปลรหัสคำสั่งช่วยจำเป็นภาษาเครื่องตั้งแต่บรรทัดที่ 1 จนถึงบรรทัดสุดท้าย 2 รอบ ในรอบแรกจะตรวจสอบรูปแบบของโปรแกรมและกำหนดค่าให้กับสัญลักษณ์หรือตัวแปรในโปรแกรมต้นกำเนิด สัญลักษณ์หรือตัวแปรนี้อาจได้แก่ตำแหน่งของหน่วยความจำที่ต้องการอ้างอิงถึง ในการวนรอบที่ 2 จะทำการแปลรหัสคำสั่งช่วยจำเป็นภาษาเครื่อง และแทนค่าสัญลักษณ์หรือตัวแปลจากค่าเก็บไว้ในรอบที่1รหัสภาษาเครื่องที่ได้จะถูกเก็บเป็นโปรแกรมปฏิบัติการ

ในคอมพิวเตอร์จะมีไมโครโพรเซสเซอร์ เบอร์ต่างๆ เช่น Z-80, 8051, 8088 เป็นต้น ซึ่งไมโครโพรเซสเซอร์นี้มีรหัสคำสั่งช่วยจำ และภาษาเครื่องแตกต่างกัน ดังนั้นในการเขียนโปรแกรมภาษาแอสเซมบลีจะต้องทราบรหัสคำสั่งช่วยจำ และภาษาเครื่องของไมโครโพรเซสเซอร์นั้นเสียก่อน รหัสคำสั่งช่วยจำจะมีอยู่ในคู่มือเฉพาะ (Data Sheet) ของไมโครโพรเซสเซอร์เบอร์นั้น และในคู่มือนั้นก็จะบอกรหัสภาษาเครื่องของแต่ละรหัสคำสั่งช่วยจำด้วย โปรแกรมภาษาแอสเซมบลีที่เขียนขึ้นอาจแปลเป็นภาษาเครื่องโดยการเทียบจากตารางในคู่มือก็ได้แต่จะไม่สะดวก จึงนิยมใช้โปรแกรมแอสเซมบลีที่เขียนขึ้นสำหรับการแปลรหัสช่วยจำให้เป็นภาษาเครื่องของไมโครโพรเซสเซอร์ที่ต้องการ

2.7 สถาปัตยกรรมของไอบีเอ็มพีซี

เพื่อทำความเข้าใจวิธีการทำงานของอะแดปเตอร์อินเตอร์เฟสอนุกรมในระบบพีซี จำเป็นต้องรู้จักอุปกรณ์หลักของไอบีเอ็มพีซีและการออกแบบอุปกรณ์เหล่านี้ หัวข้อต่อไปจะกล่าวถึงชิปโปรเซสเซอร์ บัส การเข้าถึงหน่วยความจำและสายสัญญาณ IRQ

2.7.1 8088 และโปรเซสเซอร์ในตระกูล

ชิปโปรเซสเซอร์ของไอบีเอ็มพีซีรุ่นแรกคือ 8088 ซึ่งเข้ากันได้กับ 8086 ต่อมาไอบีเอ็มซีเอทีใช้ 80286 พีซีรุ่นใหม่ส่วนใหญ่ใช้ชิปในอนุกรม 80386 หรือ 80486 ชิปพวกนี้อยู่ในตระกูลเดียวกัน สร้างโดยบริษัท Intel ข้อมูลต่อไปนี้เป็นคุณสมบัติที่มีร่วมกันในชิปทุกตัวในตระกูลนี้

2.7.2 **ตัวรีจิสเตอร์**

ตัวรีจิสเตอร์ขนาด 16 บิต มีอยู่สี่ตัวคือ AX, BX, CX และ DX แต่ละตัวสามารถแบ่งออกได้เป็นรีจิสเตอร์ขนาด 8 บิต ได้เป็น AH, AL, BH, BL, CH, CL, DH และ DL การโหลดไบต์ต่ำของเลข 16 บิต ลงใน AL และไบต์สูงลงใน AH เหมือนกับการโหลดทั้ง 16 บิต ลงใน AX และการโหลด BX, CX และ DX ก็เช่นเดียวกัน

ตัวรีจิสเตอร์ที่ใหญ่ที่สุดมีขนาดเพียง 16 บิตเนื่องจากจำนวน 16 บิต สามารถอ้างหน่วยความจำได้เพียง 64K การอ้างหน่วยความจำจึงต้องใช้รีจิสเตอร์สองตัว ตัวเลขในรีจิสเตอร์ทั้งสองเรียกว่า เซกเมนต์ (segment) และออฟเซต (offset) ของแอดเดรส เซกเมนต์เป็นแอดเดรสที่มีขอบเขต 16 ไบต์ ภายในหน่วยความจำเซกเมนตรีจิสเตอร์ 16 บิต สามารถอ้างถึงเซกเมนต์ขนาด 64K แต่ละเซกเมนต์เริ่มต้นที่ทุกๆ 16 ไบต์ กล่าวอีกนัยหนึ่งคือ ภายในพื้นที่หน่วยความจำ 1 เมกะไบต์ ออฟเซตเป็นตัวบอกจำนวนไบต์นับจากจุดเริ่มต้นของเซกเมนต์

แอดเดรสใดๆ ภายในเมกะไบต์แรกของหน่วยความจำสามารถอ้างถึงได้โดยใช้เซกเมนต์รีจิสเตอร์ 16 บิต บวกกับออฟเซต 16 บิต ในเซกเมนต์นั้น เซกเมนตรีจิสเตอร์คือ CS (Code Segment), SS (Stack Segment), DS (Data Segment) และ ES (Extra Segment)

ออฟเซตสามารถใส่ไว้ในรีจิสเตอร์ AX, BX, CX และ DX หรือในรีจิสเตอร์สำหรับออฟเซตโดยเฉพาะ ได้แก่ IP (Instruction Pointer) สำหรับออฟเซตภายในโค้ดเซกเมนต์, SP (Stack Pointer) และ BP (Base Point) สำหรับออฟเซตภายในสแต็กเซกเมนต์ และ SI (Source Index) และ DI (Destination Index) สำหรับออฟเซตในดาต้าเซกเมนต์

2.7.3 **สถาปัตยกรรมบัสและการเข้าถึงหน่วยความจำหลัก**

บัสเป็นกลุ่มของวงจรที่เชื่อมต่ออุปกรณ์ภายในคอมพิวเตอร์ พีซีมีบัสสามประเภท คือ

- บัสข้อมูลมีความกว้าง 8 บิต ซึ่งข้อมูลเดินทางผ่ายในแบบขนาน
- บัสแอดเดรส มีความกว้าง 20 บิต สำหรับการส่งแอดเดรส
- บัสควบคุม ประกอบด้วยวงจรสำหรับควบคุมอุปกรณ์ โดยทั่วไปการควบคุมเหล่านี้เป็นคำสั่งบอกว่าควรทำอะไรกับข้อมูลบนบัสข้อมูลและบัสควบคุม

เมื่อโปรเซสเซอร์ต้องการอ่านหน่วยความจำ แอดเดรสจะถูกส่งออกมาตามบัสแอดเดรสและเซตสัญญาณบนสายสัญญาณอ่านของบัสควบคุมอุปกรณ์หน่วยความจำที่เกี่ยวข้องจะใส่ข้อมูลในแอดเดรสนั้นลงบนบัสข้อมูลเพื่อส่งกลับไปให้โปรเซสเซอร์

ในกระบวนการส่งข้อมูลไปยังหน่วยความจำจะกลับกันกับกระบวนการนี้สัญญาณบนสายสัญญาณเขียนบนบัสควบคุมจะถูกเซต แอดเดรสถูกใส่บนบัสแอดเดรสและข้อมูลที่ต้องการเขียนจะถูกใส่ลงบนบัสข้อมูล

2.7.4 <u>แอดเดรส I/O</u>

นอกจากหน่วยความจำหลัก 1 เมกะไบต์ ซึ่งเข้าถึงได้ในลักษณะเซกเมนต์และนอกจากออฟเซตแล้ว ยังมีหน่วยความจำ I/O หรือแอดเดรสของพอร์ตอีก 768 ตำแหน่ง สำหรับใช้กับอุปกรณ์ต่างๆ

หน่วยความจำ I/O สามารถเข้าถึงได้โดยการส่งคำสั่ง IN และ OUT ไปให้ซีพียู ตัวอย่างเช่น การส่งค่าในรีจิสเตอร์ AL ไปให้พอร์ต 3F8H ใช้คำสั่ง

```
OUT     3F8H, AL
```

การอ่านจากพอร์ตซึ่งแอดเดรสของมันอยู่ใน DX และใส่ผลลัพธ์ลงในรีจิสเตอร์ AL คำสั่งคือ

```
IN      AL, DX
```

จะเห็นว่าวิธีการทำงานคำสั่ง IN และ OUT ในภาษาต่างๆจะแตกต่างกัน

เมื่อซีพียูได้รับคำสั่งดังกล่าว มันจะส่งสัญญาณไปตามสายสัญญาณอ่าน I/O หรือสายสัญญาณเขียน I/O ตามความเหมาะสม เมื่ออุปกรณ์ I/O ตรวจพบสัญญาณบนสายสัญญาณเหล่านี้ มันต้องตรวจสอบบัสแอดเดรสว่าคำสั่งนั้นสำหรับมันหรือเปล่า เนื่องจาก I/O มีแอดเดรสที่สูงที่สุดคือ 300H การเชื่อมต่อบัสแอดเดรสกับอุปกรณ์ I/O จึงต้องการเพียง 9 บิตล่างเท่านั้น

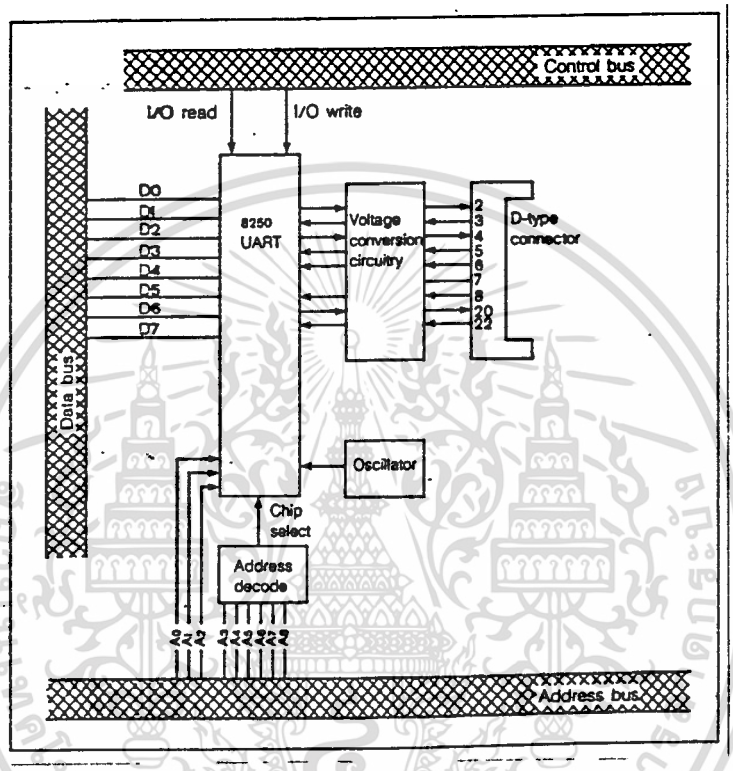
อุปกรณ์ I/O สามารถใส่ข้อมูลบนบัสข้อมูลได้เช่นเดียวกับการเข้าถึงหน่วยความจำหลัก เมื่อได้รับสัญญาณอ่าน และรับข้อมูลจากบัสข้อมูลเมื่อได้รับสัญญาณเขียน อย่างไรก็ตามอุปกรณ์ I/O อาจทำมากกว่าการใส่ข้อมูลบนบัสข้อมูล เช่น เมื่อ 8250 รับสัญญาณอ่านที่อ้างถึงบัฟเฟอร์รับข้อมูล มันไม่เพียงส่งข้อมูลในบัฟเฟอร์ออกไปเท่านั้น แต่ยังเคลียร์บัฟเฟอร์ด้วยเพื่อไม่ให้ข้อมูลเดียวกันนี้ถูกอ่านอีกในครั้งต่อไป UART อาจจะทำการรีเซตอินเตอร์รัปต์ที่มันสร้างขึ้นด้วย

2.7.5 <u>แผงวงจรอินเตอร์เฟซอนุกรม</u>

พีซีเกือบทั้งหมดใช้ UART ซึ่งส่วนประกอบอื่นที่เกี่ยวข้องกับการจัดการ UART ได้แก่

- การเชื่อมต่อทางกายภาพกับแผงวงจรหลักของคอมพิวเตอร์
- การเชื่อมต่อกับโลกภายนอกผ่านซอกเก็ตที่เหมาะสม
- การแปลงสัญญาณไฟฟ้าระหว่างระดับแรงดันไฟฟ้า ที่ใช้ในคอมพิวเตอร์กับที่ใช้ในวงจร RS-232C

สำหรับนักเขียนโปรแกรมแล้ว ไม่จำเป็นต้องสนใจส่วนประกอบเหล่านี้ในรูปที่ 2.13 แสดงวงจรหลักของอะแดปเตอร์อินเตอร์เฟซอนุกรม ซึ่งวงจรที่สมบูรณ์จะดูได้จากคู่มืออ้างอิงทางเทคนิค คู่มือเหล่านี้บรรจุข้อมูลอ้างอิงที่มีประโยชน์มากเกี่ยวกับอะแดปเตอร์อะซิงโครนัส และอะแดปเตอร์อนุกรมกับเครื่องพิมพ์ของไอบีเอ็ม ซึ่งนำไปใช้ได้กับแผงวงจรอนุกรมอื่น



รูปที่ 2.13 แผนผังลอจิกของอินเตอร์เฟซอนุกรม

2.7.6 แอดเดรส I/O ของอะแดปเตอร์อนุกรม

พอร์ต I/O สำหรับการสื่อสารแบบอนุกรมในพีซี เริ่มต้นที่แอดเดรส 3F8H สำหรับอะแดปเตอร์ตัวแรก และ 2F8H สำหรับอะแดปเตอร์ตัวที่สอง การอ้างถึงรีจิสเตอร์แต่ละตัวใน UART จะใช้การบวกออฟเซตเข้ากับเบสแอดเดรสนี้

ดังที่กล่าวมาแล้วว่า การอ้างแอดเดรส I/O นั้นใช้บัสแอดเดรสเพียง 9 บิตล่างเท่านั้น และทั้ง 3F8H และ 2F8H มีสามบิตล่างเป็นศูนย์ ดังนั้นออฟเซตที่บวกเข้าไปจึงมีได้ตั้งแต่ 0 ถึง 7 และบิตที่เปลี่ยนแปลงจะไม่เกินบิต 2 วงจรแยกแอดเดรสบนอะแดปเตอร์จึงตรวจสอบเฉพาะบิต 3 ถึง 8 บนบัสแอดเดรส ส่วนบิต 0 ถึง 2 จะถูกส่งต่อโดยตรงกับชิป UART

เมื่อวงจรแยกแอดเดรสพบแอดเดรสในช่วง 3F8H ถึง 3FFH หรือ 2F8H ถึง 2FFH ถ้าเป็น COM2 มันจะส่งสัญญาณไปที่ขา Chip Select Pin ของ UART แล้ว UART จึงดูที่บิต 0, 1 และ 2 ของบัสแอดเดรส ว่าเป็นแอดเดรสใดและดูสัญญาณการเขียนและการอ่าน I/O บนบัสควบคุมเพื่อตัดสินว่าควรทำอะไรต่อไป เบสแอดเดรสและออฟเซตของ COM1 และ COM2 แสดงไว้ในตารางที่ 2.3

ออฟเซต	อะแดปเตอร์ตัวแรก	อะแดปเตอร์ตัวที่สอง	รีจิสเตอร์
0	3F8	2F8	รีจิสเตอร์พักข้อมูลส่ง
0	3F8	2F8	รีจิสเตอร์พักข้อมูลรับ
0	3F8	2F8	แลตช์ตัวหาร LSB
1	3F9	2F9	แลตช์ตัวหาย MSB
1	3F9	2F9	รีจิสเตอร์อีนาเบิลอินเตอร์รัปต์
2	3FA	2FA	รีจิสเตอร์จำแนกอินเตอร์รัปต์
3	3FB	2FB	รีจิสเตอร์ควบคุมสายสื่อสาร
4	3FC	2FC	รีจิสเตอร์ควบคุมโมเด็ม
5	3FD	2FD	รีจิสเตอร์แสดงสถานะสายสื่อสาร
6	3FE	2FE	รีจิสเตอร์แสดงสถานะโมเด็ม

ตารางที่ 2.3 แอดเดรส I/O สำหรับ COM1 และ COM2

สังเกตว่ารีจิสเตอร์บางตัวใช้แอดเดรสร่วมกัน ความสับสนระหว่างรีจิสเตอร์พักข้อมูลส่งและรีจิสเตอร์พักข้อมูลรับไม่เกิดขึ้น เพราะคำสั่ง OUT จะเข้าถึงรีจิสเตอร์พักข้อมูลส่ง และคำสั่ง IN จะเข้าถึงรีจิสเตอร์พักข้อมูลรับ แลตช์ตัวหารก็ใช้แอดเดรสร่วมกับรีจิสเตอร์อื่นเช่นกัน การเลือกแลตช์ตัวหารใช้บิตควบคุมการเข้าถึงแลตช์ตัวหาร (DLAB) ในรีจิสเตอร์ควบคุมสายสื่อสาร เมื่อ DLAB ถูกเซตเป็น 1 จะเป็นการเข้าถึงแลตช์ตัวหาร เมื่อ DLAB เป็น 0 จะเป็นการเข้าถึงรีจิสเตอร์อื่นในแอดเดรสเดียวกัน

2.7.7 เรื่องควรพิจารณาในการโปรแกรม

หัวข้อต่อไปนี้บรรยายถึงจุดสำคัญที่ควรพิจารณาเมื่อโปรแกรมการสื่อสารอนุกรมในระดับของระบบมีดังนี้

2.7.12.1 โพลลิ่ง

การเขียนโปรแกรมโดยใช้โพลลิ่งต้องแน่ใจว่าโปรแกรมจะทำคำสั่ง IN เป็นวงรอบเพื่อตรวจสอบการรับข้อมูลหรือเหตุการณ์อื่นที่เกี่ยวข้องกันว่าเกิดขึ้นหรือไม่ โดยอ่านจากรีจิสเตอร์แสดงสถานะที่เหมาะสมของ UART ถ้ามีการรับตัวอักษรโปรแกรมควรสั่งคำสั่ง IN อีกครั้งไปยังแอดเดรสของรีจิสเตอร์พักข้อมูลรับ ถ้าตัวอักษรถูกส่งออกไปแล้ว ตัวอักษรตัวต่อไปจะถูกส่งด้วยการใช้คำสั่ง OUT ไปยังรีจิสเตอร์พักข้อมูลส่ง

2.7.12.2 ปัญหาบนพีซีความเร็วสูง

ถ้าหากพีซีทำงานเร็วกว่า 8250 มากเกินไปอาจทำให้คำสั่งที่ส่งติดต่อกันเร็วกว่าที่ 8250 สามารถประมวลผลได้ จากสาเหตุนี้จึงควรหลีกเลี่ยงการส่งคำสั่งติดต่อกันไปให้ 8250 ควรใส่คำสั่ง Short Jump ไว้ระหว่างคำสั่งเข้าถึงชิปที่ต่อเนื่องกันเพื่อลดความเร็ว วิธีนี้ใช้ได้เฉพาะกับการเขียนโปรแกรมด้วยภาษาแอสเซมบลีเท่านั้น

2.7.8 เปรียบเทียบโพลลิ่งกับอินเตอร์รัปต์

ลองนึกถึงโทรศัพท์ที่ไม่มีเสียงกริ่ง แล้วผู้ใช้ต้องคอยยกหูโทรศัพท์เพื่อฟังว่ามีใครโทรมาหาหรือไม่ตลอดเวลา ยิ่งเมื่อมีโทรศัพท์หลายเครื่องก็ยิ่งเพิ่มความยุ่งยาก โดยเฉพาะหากต้องสนทนาไปพร้อมกับการคอยยกหูโทรศัพท์เครื่องอื่น เหตุการณ์นี้ก็เช่นเดียวกับการที่โปรแกรมทำงานในลูปเพื่อคอยตรวจสอบ UART หรือเรียกฟังก์ชั่นของระบบปฏิบัติการ เพื่อดูว่ามีอะไรเกิดขึ้น คอมพิวเตอร์จะใช้เวลาส่วนใหญ่ในการตรวจสอบสถานะหรือที่เรียกว่า โพลลิ่ง (Polling)

เวลาที่ใช้ในการโพลลิ่งไม่ได้สำคัญอะไรนัก เมื่อคอมพิวเตอร์ถูกใช้ให้ทำงานเพียงอย่างเดียว แต่ในระบบหลายผู้ใช้ (Multiuser) หรือหลายงาน (Multitask) การตรวจสอบแบบนี้สามารถลดสมรรถนะการทำงานของส่วนอื่นลงได้ ยิ่งกว่านั้นข้อมูลอาจสูญหายได้ง่ายดายมาก เมื่อใช้โพลลิ่งเนื่องจากตัวอักษรอาจเข้ามาในขณะที่โปรแกรมกำลังจัดการกับตัวอักษรที่เข้ามาครั้งก่อนอยู่ และไม่ได้คอยเฝ้าดูตัวอักษรที่เข้ามาใหม่

การเขียนโปรแกรมแบบกระตุ้นด้วยอินเตอร์รัปต์เทียบได้กับโทรศัพท์ที่มีเสียงกริ่ง คอมพิวเตอร์ได้รับการแจ้งเตือนเมื่อมีเหตุการณ์สำคัญเกิดขึ้นดังนั้นโปรแกรมจึงไม่ต้องคอยตรวจสอบอีกต่อไป ด้วยเหตุผลนี้จึงขอแนะนำอย่างจริงจังว่าควรใช้อินเตอร์รัปต์เสมอเมื่อเป็นไปได้

2.8 การสื่อสารในภาษาเบสิก

เบสิกมีหลายคำสั่งที่ออกแบบมาสำหรับการสื่อสารแบบอนุกรม คำสั่งเหล่านี้มีประสิทธิภาพที่เดียวและมีประโยชน์มากกว่าฟังก์ชั่นของดอสและ รอม ไบออส (ROM BIOS) ด้วยฟังก์ชั่นทางการสื่อสารของภาษาเบสิกทำให้ไม่จำเป็นต้องเรียกฟังก์ชั่นของดอสโดยตรง ซึ่งการติดต่อระหว่างเบสิกกับระบบปฏิบัติการนั้นทำได้ยาก

นอกจากคำสั่งที่ออกแบบพิเศษสำหรับการสื่อสารแล้ว เบสิกยังมีคำสั่งที่ใช้คำสั่ง IN และ OUT ในภาษาแอสเซมบลีด้วย เพื่อใช้ในการควบคุมโดยตรง

2.8.1 การจัดการบัฟเฟอร์

เบสิกสร้างบัฟเฟอร์ข้อมูลขนาด 128 ไบต์ และบัฟเฟอร์ข้อมูลเข้าตามขนาดที่ผู้ใช้กำหนดโดยอัตโนมัติ บัฟเฟอร์ถูกเติมและถูกดึงข้อมูลในระดับของการอินเตอร์รัปต์ ถ้ามีการเปิดอินพุตสตรีม (Input stream) ตัวอักษรที่รับได้จะถูกใส่ลงในบัฟเฟอร์ขีอมูลเข้าอย่างอัตโนมัติ โดยที่โปรแกรมไม่ต้องทำอะไรทั้งสิ้น ส่วนตัวอักษรในบัฟเฟอร์ข้อมูลออกจะถูกส่งออกไปตามลำดับติดต่อกัน

ทั้งหมดที่โปรแกรมต้องทำคือ ร้องขอให้เบสิกส่งตัวอักษร และตัวอักษรจะถูกใส่ไว้ในบัฟเฟอร์ข้อมูลออกเพื่อรอการส่งผ่านทางกระบวนการอินเตอร์รัปต์ ในทำนองเดียวกันสามารถร้องขอตัวอักษรที่เข้ามาจากภาษาเบสิก และตัวอักษรนั้นจะถูกดึงออกจากบัฟเฟอร์ข้อมูลเข้า และถูกส่งต่อไปให้กับโปรแกรม

2.8.2 การทำงานกับสตรีม I/O

เนื่องจากความคล้ายคลึงกันระหว่างการเขียนไปยังอุปกรณ์และการเขียนไปยังไฟล์บนดิสก์ คำสั่งหลายๆคำสั่งที่ใช้กับการเข้าถึงไฟล์บนดิสก์ จึงถูกนำมาใช้กับการเข้าถึง I/O แบบอนุกรม คำว่า สตรีม I/O (I/O stream) ในที่นี้หมายถึงช่องทางติดต่อกับ I/O ที่มีวิธีการใช้เหมือนกับไฟล์สตรีม I/O สามารถถูกเปิด อ่าน เขียน และปิด ได้เช่นเดียวกับไฟล์บนดิสก์

2.8.3 การโปรแกรม UART ด้วยภาษาเบสิก

แม้ว่าฟังก์ชั่นการสื่อสารของเบสิกจะมีประสิทธิภาพ แต่บางครั้งจำเป็นต้องมีการควบคุมโดยตรง เช่น เมื่อต้องการส่งสัญญาณเบรก หรือเมื่อต้องการควบคุมสายสัญญาณแฮนด์เช็คกิ้งโดยตรง

UART มีรีจิสเตอร์หลายตัว แต่ละตัวบรรจุข้อมูลหนึ่งไบต์ และแต่ละบิตภายในไบต์นั้นมีความหมายแตกต่างกัน การตรวจสอบสถานะที่แน่นอนของ UART จำเป็นต้องอ่านรีจิสเตอร์และตรวจสอบแต่ละบิตภายในนั้น การเปลี่ยนสถานะของ UART ต้องเปลี่ยนแต่ละบิตภายในรีจิสเตอร์ ดังนั้นจึงต้องมีการทำฟังก์ชั่นต่อไปนี้ด้วยภาษาเบสิก

1. อ่านหนึ่งไบต์จากรีจิสเตอร์
2. ตรวจสอบบิตภายในไบต์
3. เปลี่ยนค่าบิตภายในไบต์
4. เขียนหนึ่งไบต์ไปยังรีจิสเตอร์

2.8.3.1 การอ่านหนึ่งไบต์จากพอร์ต

คำสั่ง INP ใช้ในการอ่านหนึ่งไบต์จากพอร์ต เช่นรีจิสเตอร์แสดงสถานะโมเด็มของ COM1 อยู่ที่ 3FFH ดังนั้นรีจิสเตอร์แสดงสถานะโมเด็มถูกอ่านได้โดยใช้คำสั่ง

```
MSTATUS = INP (&H3FF)
```

2.8.3.2 การเขียนหนึ่งไบต์ไปยังพอร์ต

คำสั่ง OUT ใช้ในการส่งหนึ่งไบต์ไปยังพอร์ต เช่น รีจิสเตอร์พักข้อมูลส่งสำหรับ COM1 อยู่ที่ 3F8H การส่งตัวอักษร CH ทำได้โดยใช้คำสั่ง

```
OUT (&H3F8,CH)
```

2.8.3.3 การจัดการบิต

ก่อนอื่นต้องทำความคุ้นเคยกับค่าของไบต์เมื่อมีบิตหนึ่งถูกเซต ค่าเหล่านี้เรียกว่า ค่าบิต (bit values) ค่าบิตสำหรับบิต 0 ถึง 7 มีดังนี้

บิต	ค่าบิต
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

2.8.3.4 การทดสอบบิต

การทดสอบค่าของทุกบิตทำได้โดยใช้คำสั่ง AND ซึ่งเมื่อนำจำนวนสองจำนวนมา AND กันจะทำให้ผลลัพธ์ของแต่ละบิตเป็นหนึ่งถ้าบิตที่ตรงกันเป็นหนึ่งทั้งคู่ หรือให้ผลลัพธ์เป็นศูนย์ถ้ามีบิตหนึ่งบิตใดหรือทั้งสองบิตเป็นศูนย์

การตรวจสอบบิตใดบิตหนึ่งว่าถูกเซตหรือไม่ ให้นำจำนวนมา AND กับค่าบิต เช่น ถ้า STATUS แทนสถานะของพอร์ตและต้องการรู้ว่าบิต 4 ของ STATUS ถูกเซตหรือไม่ เขียนได้ดังนี้

IF STATUS AND 16 ...

ในตัวอย่างข้างบน ถ้า STATUS เป็น 40 เบสิกจะทำการคำนวณดังนี้

STATUS	0	0	1	0	1	0	0	0
แพตเทิร์นสำหรับทดสอบ	0	0	0	1	0	0	0	0
ผลจากการ AND	0	0	0	0	0	0	0	0

ซึ่งให้ผลลัพธ์เป็นศูนย์ แสดงว่า บิต 4 ในไบต์แสดงสถานะไม่ถูกเซต
ถ้า STATUS เป็น 52 การคำนวณจะเกิดขึ้นดังนี้

STATUS	0	0	1	1	0	1	0	0
แพตเทิร์นสำหรับทดสอบ	0	0	0	1	0	0	0	0
ผลจากการ AND	0	0	0	1	0	0	0	0

ค่าผลลัพธ์ที่ไม่เป็นศูนย์ แสดงว่าบิต 4 ถูกเซต

2.8.4 <u>เงื่อนไขการเกิดข้อผิดพลาด</u>

ข่าวสารแจ้งข้อผิดพลาดที่เกี่ยวข้องกับการสื่อสารที่พบบ่อยในภาษาเบสิกมีดังต่อไปนี้

- Error 69 : Communication Buffer Overflow บอกให้ทราบว่าตัวอักษรถูกรับเข้ามา (ใส่ไว้ในบัฟเฟอร์ข้อมูลเข้าโดยระบบอินเตอร์รัปต์) เร็วกว่าที่โปรแกรมดึงออกไปทำให้บัฟเฟอร์เต็ม ตัวอักษรสูญหายไป ทางแก้ที่เป็นไปได้เช่น ใช้บัฟเฟอร์ขนาดใหญ่ขึ้น ลดอัตราบอด หรือใช้โฟลว์คอนโทรล เช่น XON/XOFF หรือฮาร์ดแวร์แฮนด์เช็คกิ้ง
- Error 25 : Device Fault บอกให้ทราบว่าไม่ได้รับสัญญาณแฮนด์เช็คกิ้งตามที่คาด ตามที่นิยามไว้ในคู่มือภาษาเบสิก มีความคล้ายคลึงกับ Error 24 มาก ดังนั้นควรจะทดสอบข้อผิดพลาดทั้งสองแบบ
- Error 57 · Device I/O Error เกิดขึ้นจากความผิดพลาดทางพาริตี้เฟรม โอเวอร์รัน หรือได้รับสัญญาณเบรก ถ้าความยาวเวิร์ดเป็นเจ็ดบิต เมื่อเกิดข้อผิดพลาดขึ้นบิตที่แปดในไบต์ที่ผิดพลาดจะถูกทำให้เป็นหนึ่ง
- Error 24 : Device Timeout ในคำสั่ง OPEN "COM.." สามารถระบุช่วงเวลาไทม์เอาต์ หรือช่วงเวลาที่ภาษาเบสิกต้องรอสัญญาณที่แน่นอน ถ้าไม่มีการระบุช่วงเวลาค่าปริยายจะถูกนำมาใช้ Error 24 บอกให้ทราบว่าเกิดสภาวะไทม์เอาต์ ตัวอย่างเช่น สมมติว่า คำสั่ง OPEN ระบุพารามิเตอร์ CS500 ถ้าไม่ได้รับ CTS ภายใน 500 มิลลิวินาที error 24 จะถูกสร้างขึ้น

- Error 68 : Device Unavailable เกิดขึ้นในกรณีอย่างเช่น การใช้คำสั่ง OPEN "COM2..." โดยที่ไม่มีอุปกรณ์ใดถูกทำให้เป็น COM2

2.8.5 การ์ดและจอภาพสำหรับการทำงานในโหมดกราฟิก

การปฏิบัติงานโปรแกรมในโหมดกราฟิก เครื่องคอมพิวเตอร์จำเป็นต้องมีความสามารถแสดงผลกราฟิกได้ ซึ่งชนิดของการ์ดควบคุมจอภาพที่สามารถทำงานในโหมดกราฟิกได้มีดังนี้

- Color Graphics Adapter (CGA)
- Enhanced Graphics Adapter (EGA)
- Multi Color Graphics Array (MCGA)
- Video Graphics Array (VGA)
- Hercules Graphics Card (HGC)

การทำงานของโปรแกรมในโหมดกราฟิก สำหรับแต่ละชนิดของการ์ดจำเป็นต้องบอก QuickBASIC ให้ทราบถึงชนิดของการ์ดที่กำลังติดต่อด้วยโดยใช้ประโยค SCREEN สำหรับกำหนดให้ QuickBASIC ทำงานในโหมดกราฟิกสำหรับจอภาพแต่ละชนิด

ประโยค SCREEN ใช้สำหรับกำหนดโหมดกราฟิกที่ใช้แสดงผลมีรูปแบบการใช้ดังนี้

```
SCREEN [mode][,[colorswitch]][,[apage]][,[vpage]]
```

mode คือนิพจน์หรือค่าคงที่ชนิดจำนวนเต็มบวก ใช้สำหรับกำหนดโหมดการทำงานสำหรับโปรแกรม สำหรับแต่ละโหมดจะสามารถใช้ได้กับชนิดของการ์ดและจอภาพเฉพาะอย่างเท่านั้น ซึ่งในโครงงานนี้ได้ใช้ mode 12 ซึ่งหมายถึง

- สำหรับโหมดกราฟิก ความละเอียดขนาด 640x480 จุด
- ขนาดของจอภาพที่สามารถกำหนดให้แสดงผลได้คือ 80x30 และ 80x60
- ขนาดของตัวอักษร 8x16 และ 8x8
- สามารถแสดงผลมากถึง 256 K สี และ 16 ระดับความเข้ม
- ชนิดของการ์ดและจอภาพที่สามารถแสดงผลได้คือ VGA

colorswitch คือนิพจน์หรือค่าคงที่ชนิดจำนวนเต็ม สามารถมีค่าได้ตั้งแต่ 0 ถึง 255 สำหรับใช้กำหนดให้สามารถแสดงกราฟิกสี สำหรับจอภาพหรือจอทีวีแบบ Composite แต่สำหรับการทำงานตั้งแต่โหมด 2 เป็นต้นไป ค่าของ colorswitch จะไม่มีผลต่อประโยค SCREEN

apage คือนิพจน์หรือค่าคงที่ชนิดจำนวนเต็ม สำหรับกำหนดหน้าของจอภาพสำหรับการเขียนรูปภาพกราฟิก ซึ่งค่าของ apage สามารถใช้ได้กับจอภาพและการ์ดที่สามารถแสดงผลได้มากกว่า 1 จอภาพ

vpage คือนิพจน์หรือค่าคงที่ชนิดจำนวนเต็ม สำหรับกำหนดหน้าของจอภาพสำหรับการแสดงผลรูปภาพกราฟิกออกทางหน้าจอ ซึ่งค่าของ vpage สามารถใช้ได้กับจอภาพและการ์ดที่สามารถแสดงผลได้มากกว่า 1 จอภาพ

สำหรับโหมดกราฟิก จอภาพโมโนโครม และการ์ดเฮอร์คิวลิส (SCREEN 3) จะต้องรันโปรแกรม MSHERC.COM ซึ่งเป็นโปรแกรมประเภทฝังตัว (Stay resident program) ซึ่งเป็นโปรแกรมไดร์เวอร์สำหรับการติดต่อระหว่างโปรแกรมกับการ์ดเมื่อทำงานในโหมดกราฟิก

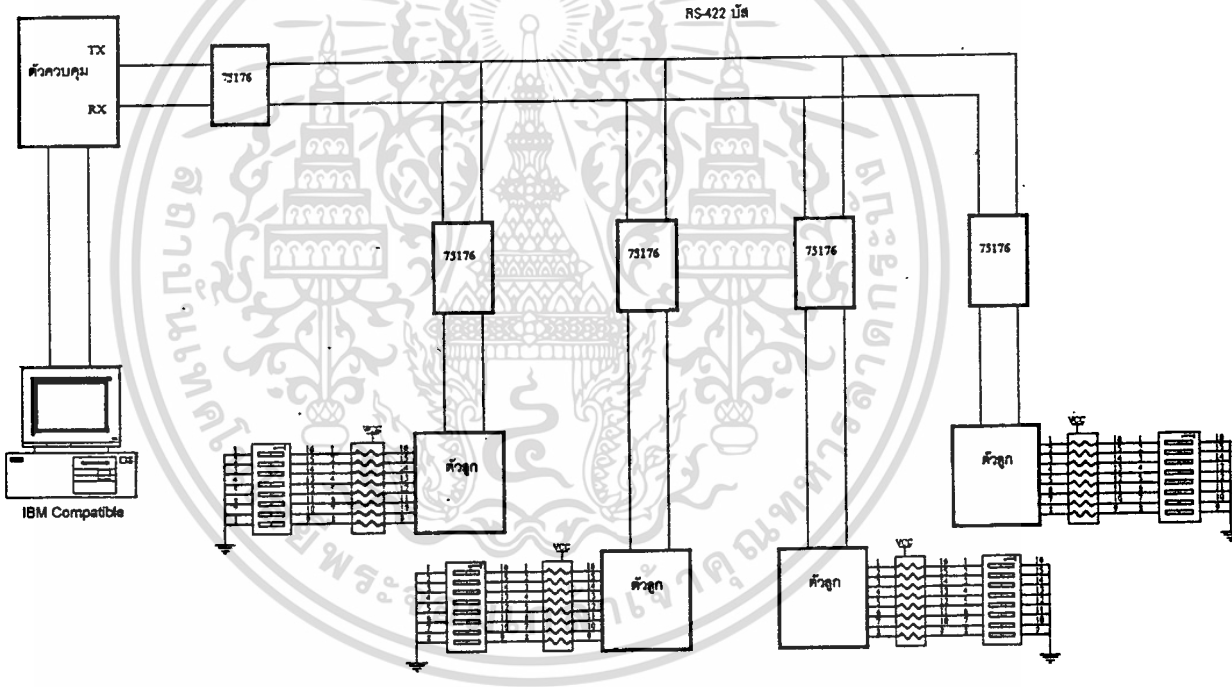
การทำงานในโหมดข้อความ QuickBASIC จะสามารถแสดงผลได้ขนาดที่เล็กที่สุดเท่ากับ 1 ตัวอักษร (cell) แต่สำหรับการทำงานในโหมดกราฟิกสามารถแสดงผลได้ละเอียดกว่าในโหมดข้อความคือ สามารถแสดงผลได้ขนาดที่เล็กที่สุดเท่ากกับ 1 จุด (pixel) ซึ่งจำนวนของจุดที่สามารถแสดงผลได้นี้ขึ้นอยู่กับชนิดของจอภาพ การ์ดที่ใช้ควบคุมจอภาพนั้นๆ และโหมดการทำงานที่ถูกกำหนดโดยประโยค SCREEN ซึ่งถ้าทำงานในโหมดที่มีจำนวนจุดมากกว่า หมายถึงว่าเราจะสามารถแสดงผลรูปภาพกราฟิกได้ละเอียดกว่านั่นเอง

บทที่ 3

แนวความคิดในการสร้างระบบ

3.1 การทำงานของระบบ

ระบบแจ้งเหตุไฟไหม้ที่สามารถระบุตำแหน่งได้จะมีรูปร่างของระบบดังแสดงในรูปที่ 3.1 ซึ่งประกอบด้วยไมโครโปรเซสเซอร์ตัวแม่และตัวลูกและคอมพิวเตอร์



รูปที่ 3.1 ระบบแจ้งเหตุไฟไหม้ที่สามารถระบุตำแหน่งได้

ในโครงงานนี้จะใช้การสื่อสารแบบอนุกรมระหว่างไมโครโปรเซสเซอร์ตัวแม่และไมโครโปรเซสเซอร์ตัวลูก โดยไมโครโปรเซสเซอร์ตัวแม่จะทำหน้าที่ควบคุมดูแลและส่งค่าแสดงผลผ่านคอมพิวเตอร์ จะมีการทำงานดังนี้

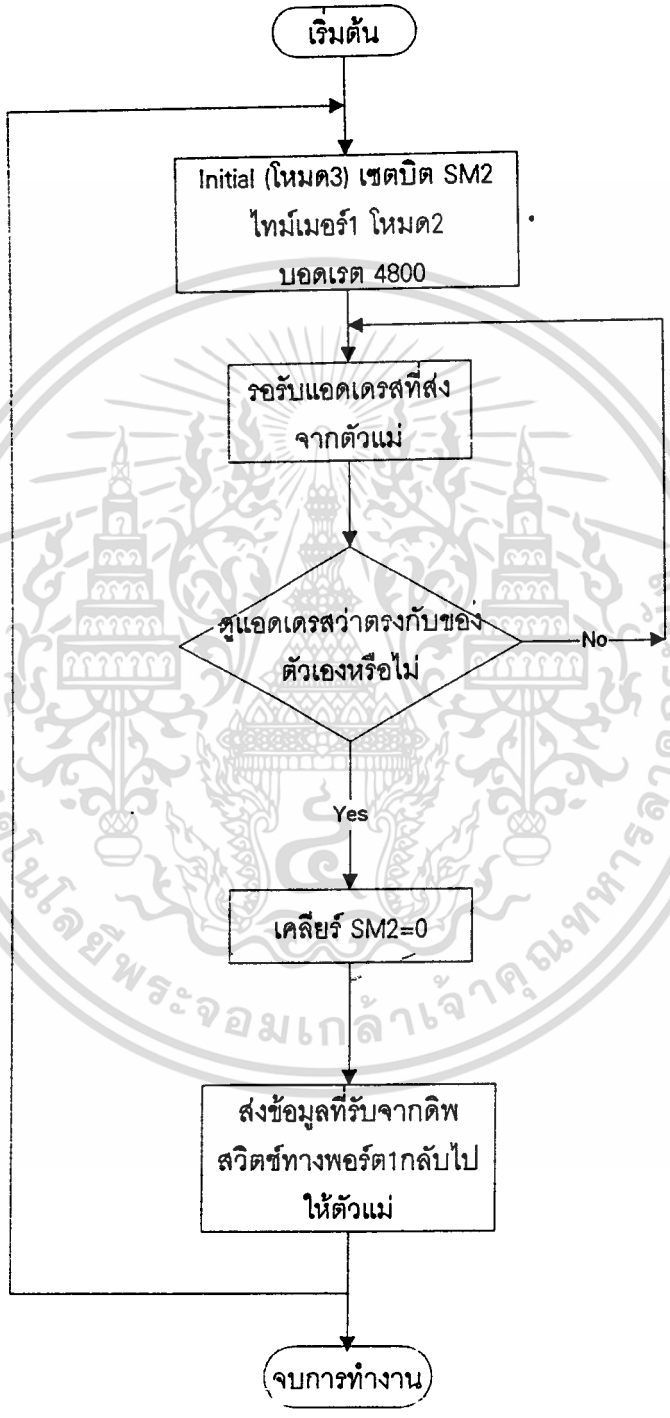
3.2 ไมโครโปรเซสเซอร์ตัวแม่

ไมโครโปรเซสเซอร์ตัวแม่จะทำการสื่อสารกับไมโครโปรเซสเซอร์ตัวลูกทั้งหมดและกับคอมพิวเตอร์ ในการสื่อสารกับไมโครโปรเซสเซอร์ตัวลูกนั้นจะใช้การสื่อสารแบบอนุกรมโหมดที่ 3 และมีรูปแบบการทำงานตามรูปที่ 3.2 คือจะเริ่มจากกำหนดการทำงานในโหมดที่ 3 และใช้ไทม์เมอร์โหมดที่ 2 (8 บิต ออโต-เมติกรีโหลด) และใช้อัตราเร็วในการส่งและรับข้อมูล 4800 บิตต่อวินาที จากนั้นมันก็จะนำการส่งข้อมูล 11 บิตไปให้ตัวลูก ข้อมูลนี้จะเป็นค่าแอดเดรสของตัวลูก (บิตที่ 10 หรือ TB8 มีค่าเป็น 1) และจะรอรับค่าข้อมูลที่ส่งกลับมาจากตัวลูกตัวนั้น แล้วนำไปเก็บไว้ในหน่วยความจำภายนอก ทำอย่างนี้ไปจนครบตัวลูกทั้งหมด แล้วก็จะทำการส่งข้อมูลจากตัวลูกที่เก็บไว้ไปให้คอมพิวเตอร์เพื่อแสดงผลต่อไป

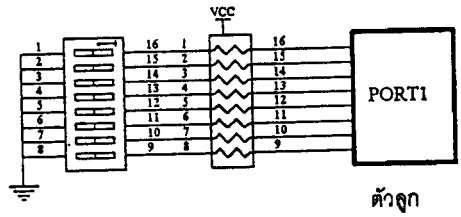
3.3 ไมโครโปรเซสเซอร์ตัวลูก

การทำงานของไมโครโปรเซสเซอร์ตัวลูกนี้จะเป็นไปตามรูปที่ 3.3 เริ่มจากการตั้งค่าการทำงานในรูปแบบการรับส่งข้อมูลอนุกรมโหมดที่ 3 ไทม์เมอร์โหมดที่ 2 (8 บิต ออโต้เมติก รีโหลด) และใช้อัตราเร็วในการรับส่งข้อมูล 4800 บิตต่อวินาที เช่นเดียวกับไมโครโปรเซสเซอร์ตัวแม่ และตั้งบิต SM2 ให้เป็น 1 จะทำให้รับข้อมูลที่มีบิตที่ 10 (TB8) เป็น 1 สามารถอินเตอร์รัปต์มันได้ทุกตัว จากนั้นตัวลูกก็จะตรวจสอบค่าแอดเดรสที่ส่งเข้ามานั้นว่าเป็นของตนหรือไม่ ถ้าไม่ใช่แอดเดรสของตัวมันเองมันก็จะไม่สนใจ แต่ถ้าเป็นค่าแอดเดรสของตัวมันเองมันก็จะส่งสถานะที่รับมาจากตัวดิพสวิตช์ผ่านทางพอร์ต 1 กลับไปให้ตัวแม่ ดังรูปที่ 3.4

รูปที่ 3.2 การทำงานของไมโครโปรเซสเซอร์ตัวแม่



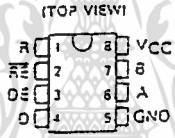
รูปที่ 3.3 การทำงานของไมโครโปรเซสเซอร์ตัวลูก



รูปที่ 3.4 การต่อไมโครโปรเซสเซอร์ตัวลูกกับดิพสวิตช์

3.4 ส่วนควบคุมการสื่อสารข้อมูลระหว่างไมโครโปรเซสเซอร์ตัวแม่และไมโครโปรเซสเซอร์ตัวลูก

ในส่วนนี้เราจะใช้การส่งข้อมูลในมาตรฐาน อาร์ เอส 485 ซึ่งใช้เส้นส่งสัญญาณเป็นแบบสัญญาณผลต่าง และค่าศักดาไฟฟ้าในช่วง 0 ถึง +5 V ซึ่งสะดวกในการใช้กับระบบไมโครโปรเซสเซอร์ ในปริญญานิพนธ์นี้จะใช้ชิพ SN 75176 A มาควบคุมโดยชิพ SN 75176 นี้จะมีลักษณะดังนี้



รูปที่ 3.5 การจัดวางขาของชิพ SN 75176

- รับส่งข้อมูลได้สองทิศทาง
- ตรงกับมาตรฐาน อาร์ เอส 422 เอ และ CCITT Recommendation V 11 และ X 27
- ออกแบบมาเพื่อการสื่อสารหลายจุดบนบัสยาวในสภาพแวดล้อมที่มีสิ่งรบกวน
- เอาท์พุตมี 3 สถานะ (3-State)
- ขาอีเนเบิ้ลของการส่งและการรับแยกจากกัน
- ช่วงศักดาอินพุต/เอาท์พุตทั้งบวกและลบมีค่ากว้าง
- เอาท์พุตสูงสุดของตัวขับ (Driver) มีค่า ± 60 mA
- มีการป้องกันการเสียหายจากความร้อน
- จำกัดค่ากระแสบวกและลบของตัวชิพ
- ใช้ไฟเลี้ยง 5 V ที่เดียว
- ต้องการกำลังต่ำ

ในการใช้งานเราจะต่อขา DE (ขา 3) และ RE\ (ขา 2) ไว้ด้วยกัน แล้วควบคุมโดยไมโครโปรเซสเซอร์ (ในไมโครโปรเซสเซอร์ตัวแม่จะใช้ขา P 1.0 ควบคุมส่วนในไมโครโปรเซสเซอร์ตัวลูกจะใช้ขา T0 ควบคุม)

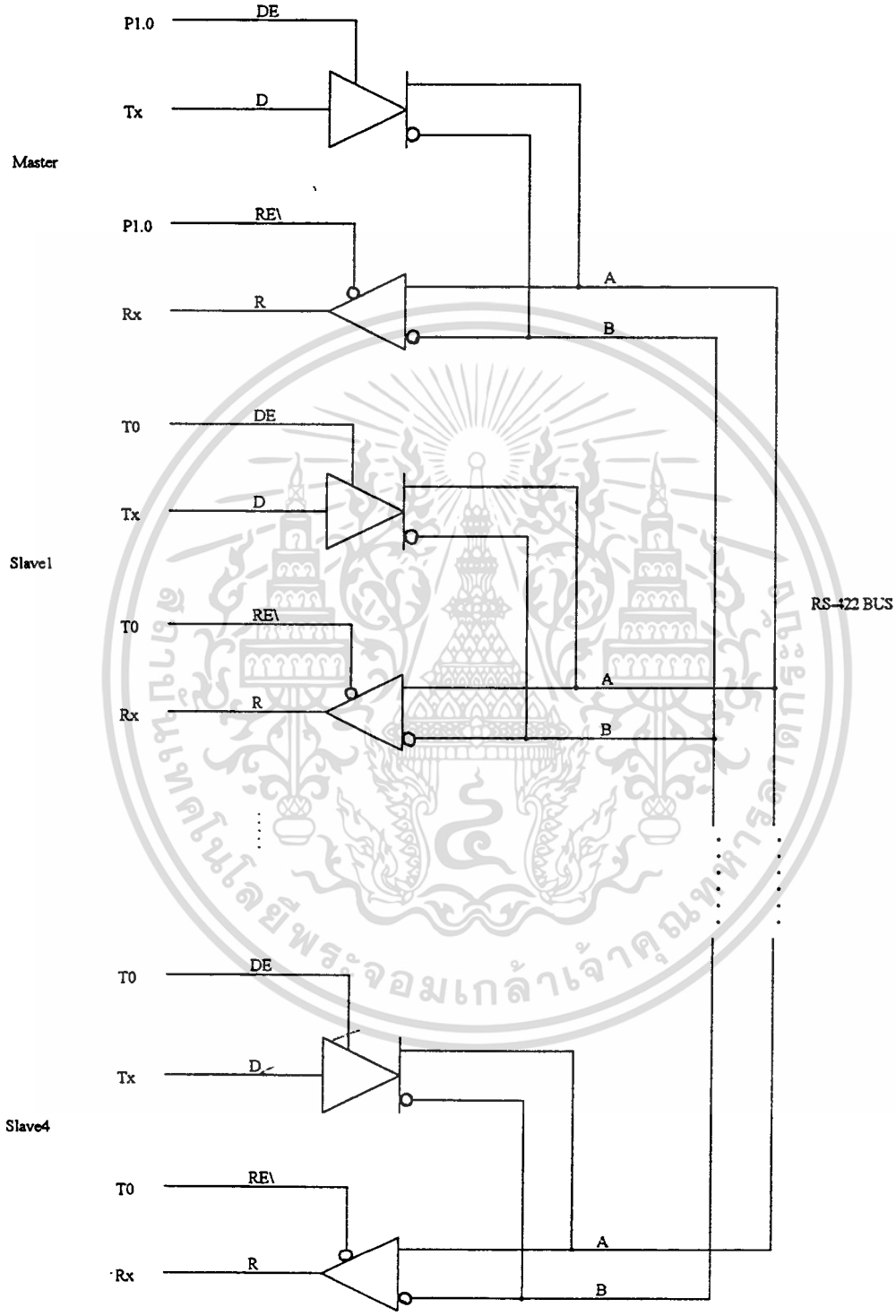
ทางด้านส่งขา TX จากไมโครโปรเซสเซอร์จะต่อเข้ากับขา D ของ 75176 ซึ่งจะแปลงอินพุตนี้ให้เป็นแบบอาร์ เอส 485 ออกที่ขา A,B เมื่อขา DE แอคทีฟและเอาท์พุตจะมีลักษณะเป็นไฮอิมพิแดนซ์ เมื่อขา DE ไม่แอคทีฟซึ่งจะทำหน้าที่เปรียบเสมือนตัวควบคุมทิศทาง

ทางด้านรับเมื่อขา RE\ แอคทีฟสัญญาณจากขา A,B จะถูกแปลงกลับเป็น TTL ออกทางขา R ของ 75176 และเอาท์พุตที่ขา R จะเป็นไฮอิมพิแดนซ์เมื่อขา RE\ ไม่แอคทีฟโดยการต่อทั้งหมดจะมีลักษณะดังรูป 3.6

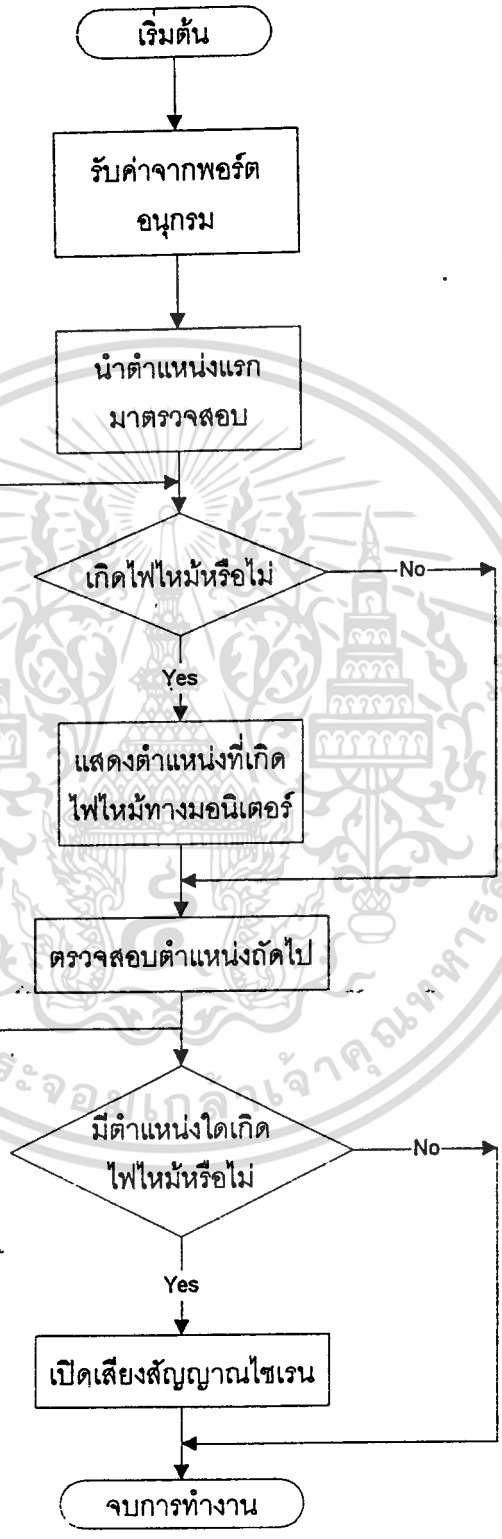
3.5 การแสดงผลออกทางหน้าจอมอนิเตอร์

เมื่อไมโครโปรเซสเซอร์ตัวแม่ส่งค่าสถานะของไมโครโปรเซสเซอร์ตัวลูกแต่ละตัวมาให้ยังเครื่องคอมพิวเตอร์เพื่อที่จะแสดงผลออกทางหน้าจอมอนิเตอร์ คอมพิวเตอร์จะรับข้อมูลเข้ามาทางพอร์ตอนุกรม ได้แก่ COM 1 (&H3F8) หรือ COM 2 (&H2F8) ไมโครโปรเซสเซอร์ตัวแม่จะส่งมาทีละไบต์ซึ่งในแต่ละไบต์นั้นจะประกอบด้วย 8 บิตในแต่ละบิตแทนเป็นแต่ละตำแหน่งที่เราสมมติให้เกิดสถานะการณ์โดยตัวโปรแกรมจะทำงานแบบวนลูฟ นำข้อมูลวนไปเรื่อย ๆ เพื่อไปแทนในตำแหน่งของแต่ละห้อง ในระหว่างนั้นจะมีการตรวจตำแหน่งนั้นด้วยว่ามีไฟไหม้เกิดขึ้นหรือไม่ ถ้าบิตใดมีค่าเป็น "1" แสดงว่าในตำแหน่งนั้นเกิดไฟไหม้ แต่ถ้าในบิตนั้นมีค่าเป็น "0" แสดงว่าตำแหน่งนั้นมีสถานะการณ์เป็นปกติ คอมพิวเตอร์จะทำการตรวจสอบทีละบิตไปจนครบทั้ง 8 บิต ถ้าบิตใดมีค่าเป็น "1" แสดงว่าเกิดไฟไหม้คอมพิวเตอร์จะนำค่าบิตนั้นไปแสดงผลทางหน้าจอมอนิเตอร์เพื่อเตือนให้รู้ว่าที่ตำแหน่งนั้นเกิดไฟไหม้ขึ้นมา ส่วนในตำแหน่งที่เกิดไฟไหม้ (ค่าบิตมีค่าเป็น "0") ก็จะไม่มีการแสดงสัญลักษณ์ที่หน้าจอมอนิเตอร์ เมื่อทำการตรวจสอบครบทั้ง 8 บิตแล้วคอมพิวเตอร์ก็จะรับค่าไบต์ใหม่เข้ามาเพื่อทำการตรวจสอบสถานะตำแหน่งที่อยู่ถัดไปและจะตรวจสอบตำแหน่งถัดไปเรื่อย ๆ จนครบ เมื่อทำการตรวจสอบได้ครบทุกค่าจำนวนของไมโครโปรเซสเซอร์ตัวลูกแล้วถ้าเกิดมีบิตใดเกิดเป็น "1" ไม่ว่าจะเกิดที่จุดก็ตามจะมีเสียงสัญญาณไซเรนดังขึ้นมาเพื่อเตือนผู้ที่ควบคุมอยู่หน้าจอมอนิเตอร์ว่าเกิดไฟไหม้ขึ้นมา และถ้าทั้งในทุกตำแหน่งนั้นไม่เกิดไฟไหม้ขึ้นเลย ก็จะไม่มีเสียงสัญญาณไซเรนเตือน เมื่อทำครบทั้งหมดทุกตำแหน่งแล้วคอมพิวเตอร์ก็จะคอยรับค่าตำแหน่งใหม่ทั้งหมดจากไมโครโปรเซสเซอร์ตัวแม่ที่จะเข้ามาทางพอร์ตอนุกรมต่อไป ซึ่งการทำงานจะวนเวียนอย่างนี้ไปเรื่อยๆ การทำงานในการแสดงผลออกทางหน้าจอมอนิเตอร์สามารถแสดงได้ตาม รูปที่ 3.7

นอกจากนี้ ในตัวโปรแกรมที่ใช้ในการแสดงผลทางจอมอนิเตอร์ เรายังสามารถที่จะใช้ในการตรวจสอบอาคารที่สูงกี่ชั้นก็ได้ และยังกำหนดความกว้างของทางเดิน, จำนวนของห้องในแต่ละชั้นได้ตามความต้องการ โดยเพียงแค่เราปรับเปลี่ยนค่าตัวเลขที่เป็นตัวแปรของการกำหนดค่าต่างๆ ภายในโปรแกรม ซึ่งจะทำให้โปรแกรมมีความยืดหยุ่นมาก และอาจเป็นประโยชน์สำหรับผู้ที่จะต้องการนำไปเป็นแนวทางในการเขียนโปรแกรมประยุกต์ในการใช้งานอื่นๆได้ตามต้องการ



รูปที่ 3.6 การติดต่อระหว่างตัวแม่และตัวลูก



รูปที่ 3.7 แสดงขั้นตอนการมาออกที่จอมอนิเตอร์

บทที่ 4

การทดลองและผลการทดลอง

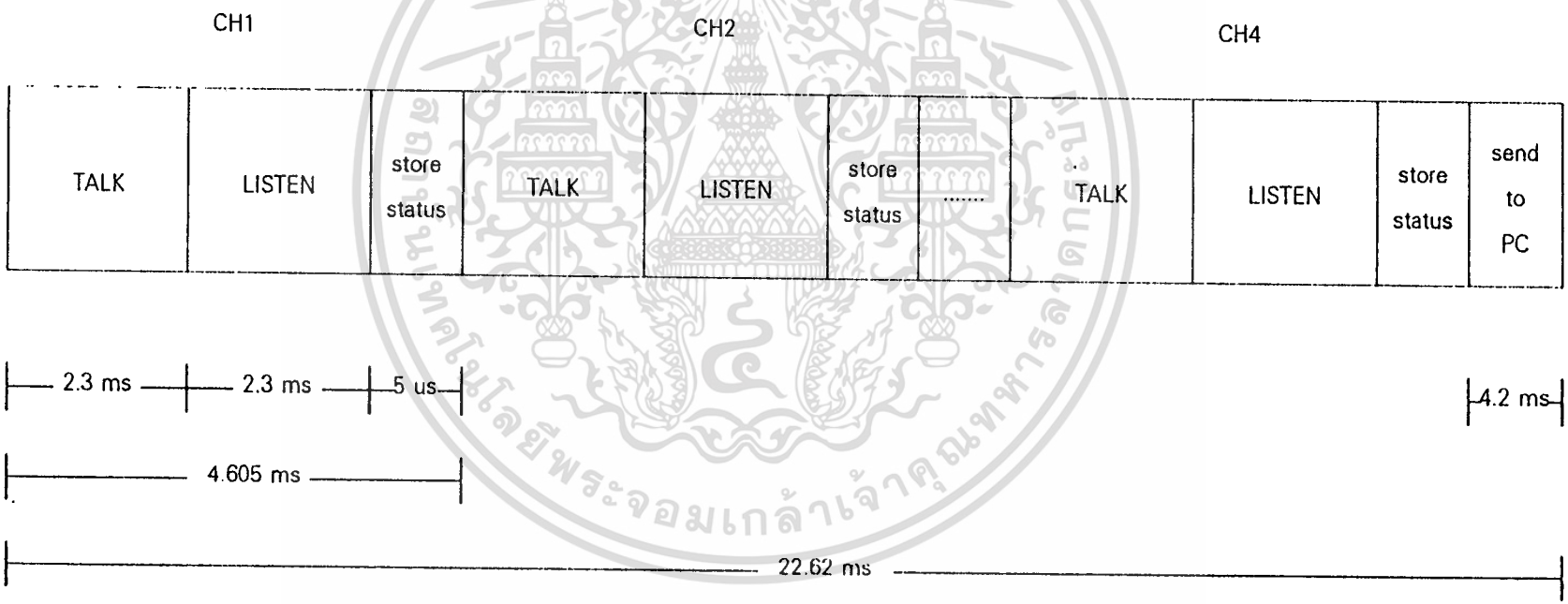
4.1 ขั้นตอนในการทดลอง

1. ทำการทดลองโดยเริ่มจากการเชื่อมต่อเครื่อง JAZZ-31 (ตัวแม่) ผ่านชิพ SN75176 A ซึ่งทำหน้าที่แปลงสัญญาณให้เป็นมาตรฐาน อาร์ เอส 485 ส่งไปให้ตัวลูก
2. เราจะใช้เครื่อง V-31 ทำหน้าที่เป็นตัวลูกซึ่งจะมีชิพ SN75176 A ทำหน้าที่แปลงสัญญาณ อาร์ เอส 485 ที่มาจากตัวแม่ให้กลับเป็นสัญญาณแบบ TTL
3. เชื่อมต่อไมโครโปรเซสเซอร์ตัวแม่ (JAZZ-31) กับคอมพิวเตอร์ผ่านพอร์ตสื่อสารอนุกรม
4. ส่วนของโปรแกรมในการทำงานนั้นมีดังนี้
   - โปรแกรมการติดต่อกันระหว่างไมโครโปรเซสเซอร์ตัวแม่และไมโครโปรเซสเซอร์ตัวลูก เราจะใช้ภาษาแอสเซมบลีของ MCS-51 ในการเขียน
   - ส่วนของการแสดงผลบนที่หน้าจอมอนิเตอร์เราจะใช้ภาษา QuickBASIC ในการเขียน

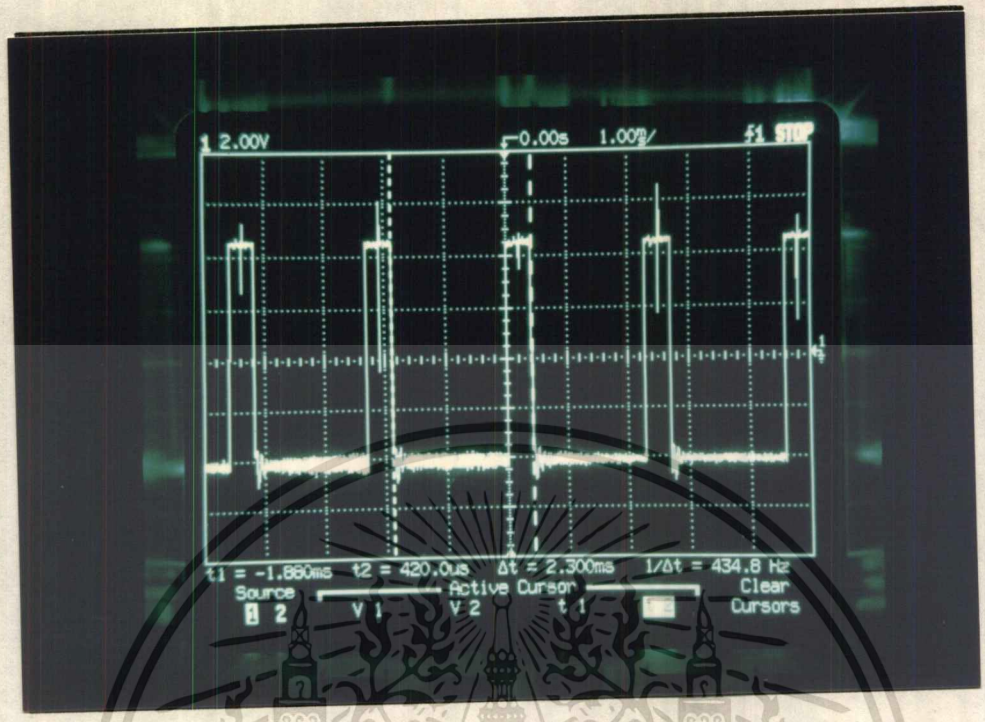
สำหรับคาบเวลาในการรับส่งข้อมูลระหว่างไมโครโปรเซสเซอร์ตัวแม่และไมโครโปรเซสเซอร์ตัวลูกจะคำนวณได้จากอัตราเร็วในการรับส่งข้อมูลที่ใช้ ซึ่งเราจะใช้ความเร็วในการรับส่งที่ 4800 บิตต่อวินาที จะใช้เวลาในการส่งต่อบิตประมาณ 2.08 ดังนั้นในการส่งข้อมูลของเรา 11 บิต จะใช้เวลาประมาณ 2.292 ms แต่ในส่วนการส่งข้อมูลจากไมโครโปรเซสเซอร์ตัวแม่ไปยังคอมพิวเตอร์ จะใช้อัตราเร็วในการส่งข้อมูลที่ 9600 บิตต่อวินาที ซึ่งจะใช้อัตราเร็วในการส่งข้อมูลต่อบิตประมาณ 0.1 ms ดังนั้นในการส่งข้อมูล 40 บิต (จากการทดลองส่งตัวลูก 4 ตัว) เท่ากับ 4.2 ms คาบเวลาที่ใช้ทั้งหมดแสดงในรูปที่ 4.1

4.2 ผลการทดลอง

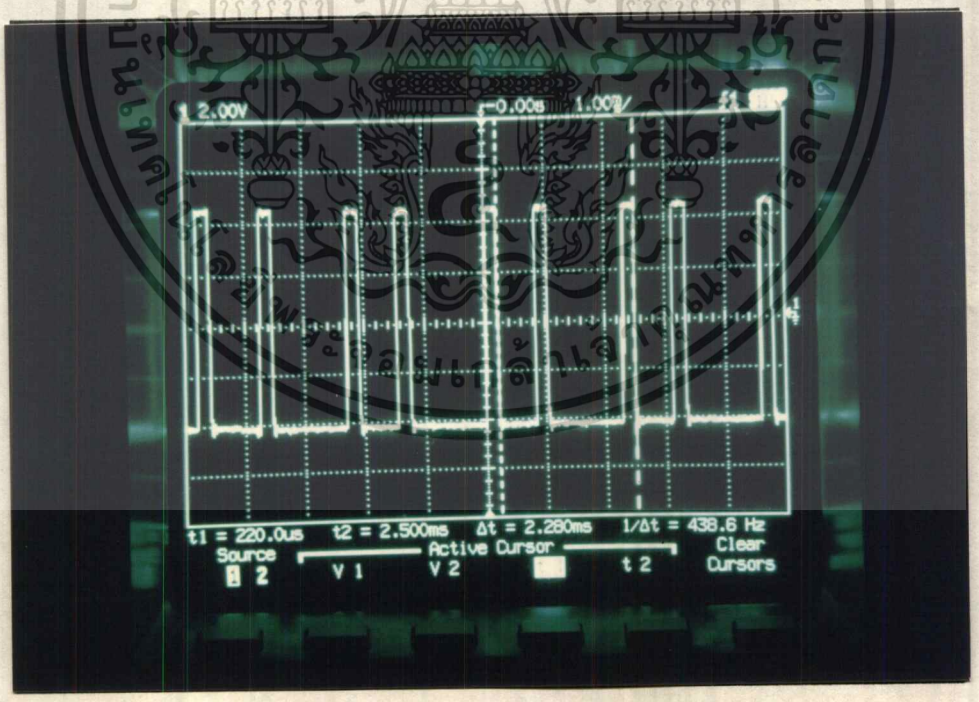
จากรูปที่ 4.2 และ รูปที่ 4.3 แสดงคาบเวลาของข้อมูลที่ส่งออกมาจากไมโครโปรเซสเซอร์ตัวแม่หลังจากที่สัญญาณผ่านชิพ SN75176 A มาแล้ว (วัดที่ขา A เทียบกับขา B)



รูปที่ 4.1 ตามแบบในมาตรฐาน IEEE



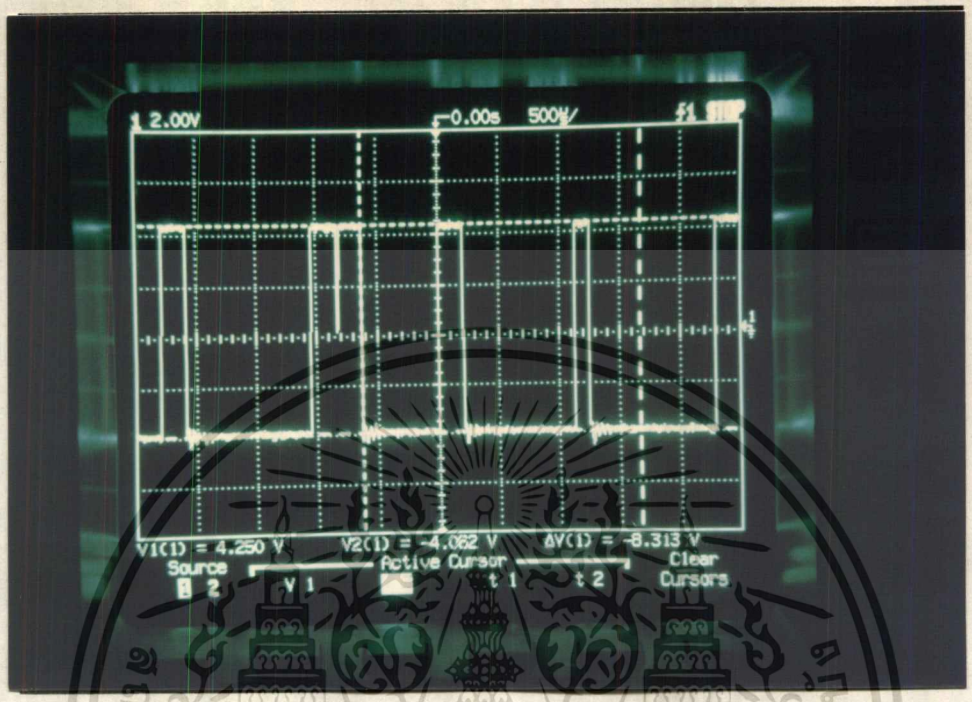
รูปที่ 4.2 คาบเวลาของแอดเดรสที่ส่งจากตัวแม่ที่มีแอดเดรสเท่ากับ 00H



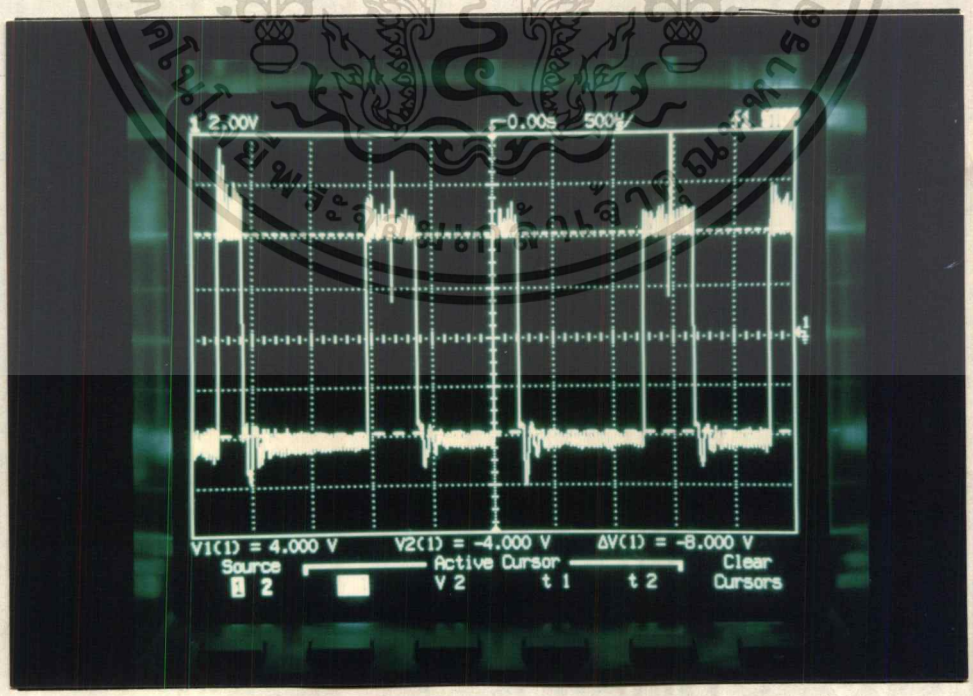
รูปที่ 4.3 คาบเวลาของแอดเดรสที่ส่งจากตัวแม่ที่มีแอดเดรสเท่ากับ 04H

จะเห็นได้ว่าข้อมูลประกอบด้วย บิตเริ่มต้น 1 บิต (มีค่า 0), บิตข้อมูล 8 บิต, บิตที่9 ซึ่งเป็น 1 (แสดงว่าเป็นแอดเดรส) และ บิตสุดท้าย 1 บิต (มีค่า 1)

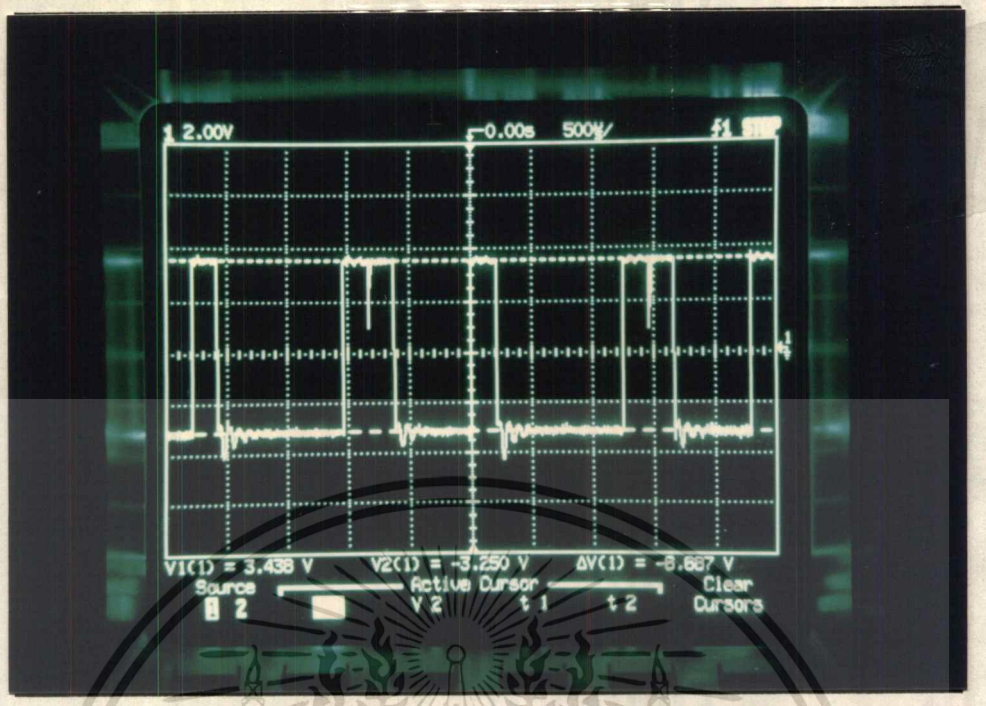
รูปที่ 4.4 ถึง รูปที่ 4.8 แสดงระดับโวลเตจของสัญญาณจากตัวแม่ไปตัวลูก โดยมีระยะทางจากตัวแม่ถึงตัวลูกต่างๆกัน



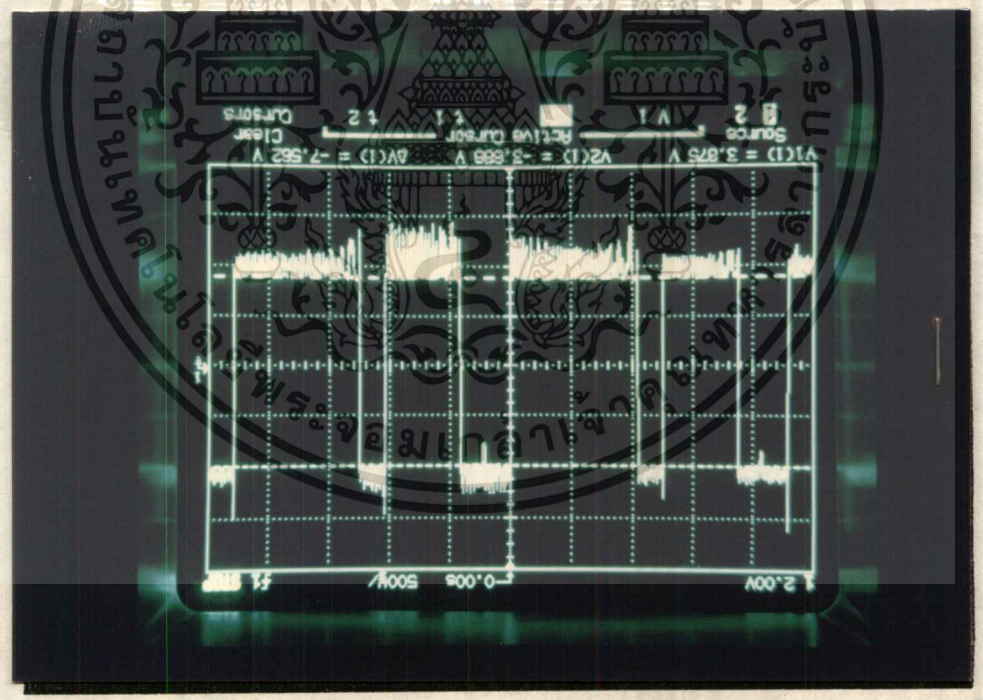
รูปที่ 4.4 ระดับโวลเตจจากตัวแม่ไปยังตัวลูก 1 ตัวที่ระยะทาง 1 ฟุต



รูปที่ 4.5 ระดับโวลเตจจากตัวแม่ไปยังตัวลูก 1 ตัวที่ระยะทาง 200 เมตร



รูปที่ 4.6 ระดับโวลเตจจากตัวแม่ไปยังตัวลูก 4 ตัวที่ระยะทาง 1 ฟุต

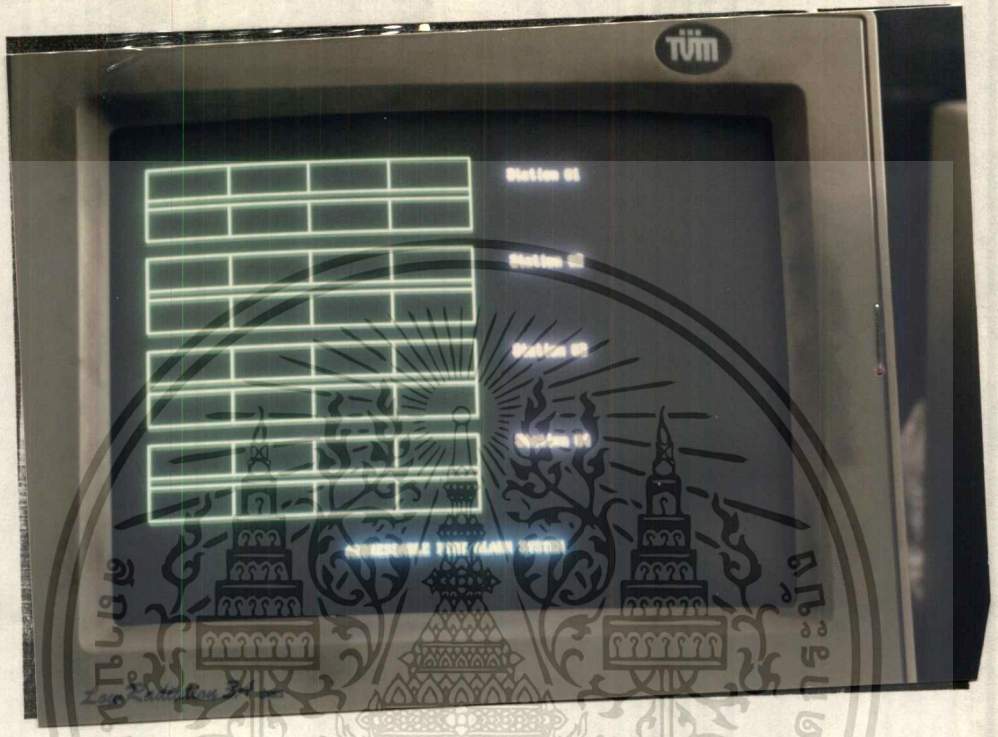


รูปที่ 4.7 ระดับโวลเตจจากตัวแม่ไปยังตัวลูก 2 ตัวที่ระยะทาง 100 เมตร

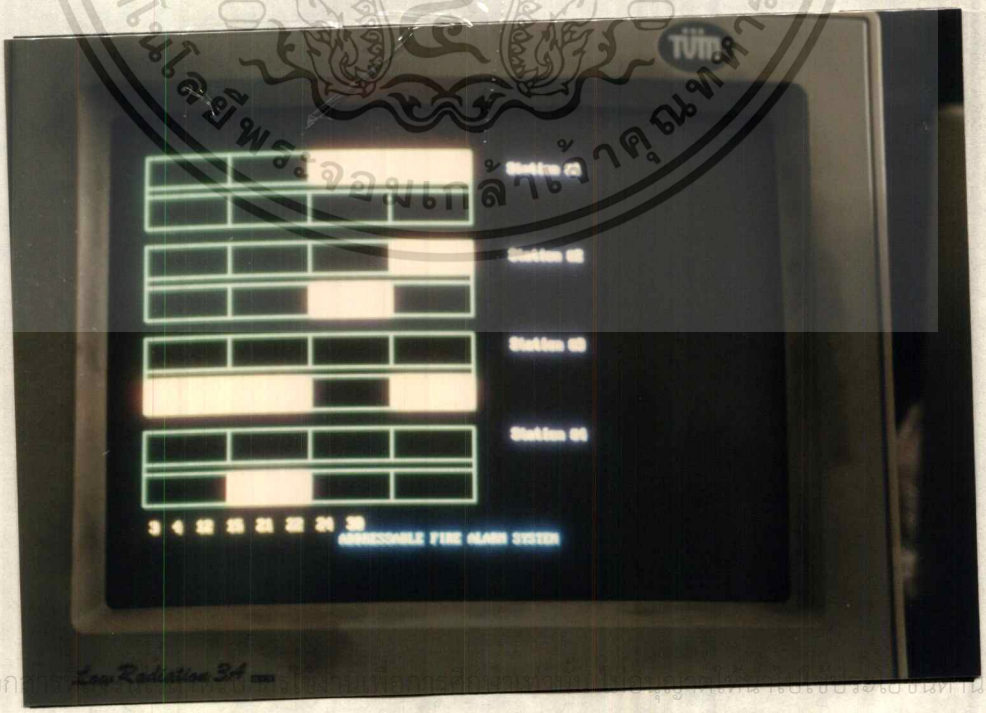
เมื่อเปรียบเทียบรูปที่ 4.4 กับ รูปที่ 4.5 จะเห็นว่าเมื่อสายส่งมีระยะทางยาวขึ้น (ตั้งแต่ 1 ฟุต ถึง 200 เมตร) ระดับโวลเตจจะตกลง และจะเกิดสัญญาณรบกวน (noise) ขึ้นเนื่องจากอาจเป็นเพราะว่าในการทดลองไม่ได้ใช้สายส่งเกลียวคู่ แต่ใช้สายโทรศัพท์ธรรมดาแทน จึงอาจเกิดค่าคาปาซิเตอร์ระหว่างสายส่ง) แต่เมื่อ

เปรียบเทียบกับรูปที่ 4.4 และรูปที่ 4.6 จะได้ว่าเมื่อมีตัวลูกมากขึ้นระดับโวลเตจจะตกลงมากกว่ากรณีแรก แต่ไม่มีสัญญาณรบกวนเกิดขึ้นเพราะว่าสายส่งมีระยะทางสั้น

ถ้ารวมทั้งสองกรณีจะเป็นดังรูปที่ 4.7 คือมีตัวลูกสองตัว แต่ละตัวมีระยะทาง 100 เมตร



รูปที่ 4.8 จอมอนิเตอร์ในสภาวะปกติ



รูปที่ 4.9 จอมอนิเตอร์เมื่อเกิดไฟไหม้

จากรูปที่ 4.8 จะแสดงให้เห็นหน้าจอมอนิเตอร์ในสภาวะปกติ ในรูปจะเห็นการแบ่งเป็นในแต่ละตัวลูก การทดลองนี้จะแบ่งเป็น 4 ตัวลูก ซึ่งตัวลูกแต่ละตัวจะแบ่งเป็น 8 จุดถ้าในจุดใดเกิดไฟไหม้ก็จะแสดงผลเตือนออกมาที่หน้าจอ และจะมีเสียงไซเรนเตือนซึ่งจะไม่สามารถแสดงได้ในที่นี้ สำหรับในสภาวะเมื่อเกิดไฟไหม้จะแสดงในรูปที่ 4.9 จากรูปเราสมมติให้เกิดไฟไหม้ขึ้นที่จุดต่างๆ และจะแสดงจุดที่เกิดเพลิงไหม้ขึ้นที่หน้าจอมอนิเตอร์ด้วย ในตัวลูกตัวแรกจะส่งข้อมูลเข้ามาเป็น 0CH จึงไหม้ในห้องที่ 3 และห้องที่ 4 ในตัวลูกตัวที่สองจะส่งข้อมูลเข้ามาเป็น 48H นั่นคือไฟไหม้ในห้องที่ 12 และ 15 ในตัวลูกตัวที่สามจะส่งข้อมูลเข้ามาเป็น B0H ไฟไหม้ในห้องที่ 21, 22 และ 24 ในตัวลูกตัวที่สี่จะส่งข้อมูลเข้ามาเป็น 20H ไฟไหม้ในห้องที่ 30

บทที่ 5

บทวิจารณ์และบทสรุป

ในปริญญานิพนธ์ฉบับนี้เป็นการนำเอาไมโครโปรเซสเซอร์ MCS-51 มาประยุกต์ใช้งาน โดยจะมีไมโครโปรเซสเซอร์ตัวหนึ่งทำหน้าที่เป็นตัวแม่คอยสื่อสารกับตัวลูกโดยการใช้สื่อสารแบบอนุกรม และรวบรวมข้อมูลสถานะจากตัวลูกส่งออกไปแสดงผลทางคอมพิวเตอร์

ข้อดีของระบบนี้คือใช้การควบคุมการทำงานโดยโปรแกรมควบคุม ซึ่งถ้าผู้ใช้ต้องการเปลี่ยนแปลงการทำงานใดๆก็จะสามารถทำการแก้ไขได้ที่โปรแกรมโดยตรงทำให้ระบบมีความยืดหยุ่นสูง

สำหรับสัญญาณเสียงไซเรนจะต้องนำมาไว้ที่ท้ายโปรแกรมหลังจากที่ทำการตรวจสอบตำแหน่งทั้งหมดทุกตำแหน่งแล้ว เนื่องจากถ้านำไปไว้ในลูประหว่างที่ทำการตรวจสอบแต่ละตำแหน่ง เมื่อมีไฟไหม้แล้วนอกจากจะมีสัญญาณเตือนบนหน้าจอมอนิเตอร์และมีสัญญาณเสียงไซเรนเตือนด้วย จะทำให้เวลาที่จะรันโปรแกรมจะกินเวลามาก อาจทำให้การตรวจสอบในตำแหน่งหลังจากตำแหน่งที่เกิดไฟไหม้นี้อาจเกิดการพลาดได้ เนื่องจากตัวโปรแกรมจะไม่มีเวลาในการตรวจสอบเพราะจะต้องใช้เวลาในการเอาไปทำลูปที่ทำให้เกิดสัญญาณไซเรนซึ่งต้องใช้เวลาพอสมควร

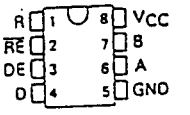
ภาคผนวก

SN75176A
DIFFERENTIAL BUS TRANSCEIVER

D2619, JUNE 1984—REVISED AUGUST 1989

- Bidirectional Transceiver
- Meets EIA Standards RS-422A and CCITT Recommendations V.11 and X.27
- Designed for Multipoint Transmission on Long Bus Lines in Noisy Environments
- 3-State Driver and Receiver Outputs
- Individual Driver and Receiver Enables
- Wide Positive and Negative Input/Output Bus Voltage Ranges
- Driver Output Capability. . . ±60 mA Max
- Thermal Shutdown Protection
- Driver Positive and Negative Current Limiting
- Receiver Input Impedance . . . 12 kΩ Min
- Receiver Input Sensitivity . . . ±200 mV
- Receiver Input Hysteresis . . . 50 mV Typ
- Operates from Single 5-V Supply
- Low Power Requirements

D OR P
DUAL-IN-LINE PACKAGE
(TOP VIEW)



FUNCTION TABLE (DRIVER)

INPUT	ENABLE	OUTPUTS	
D	DE	A	B
H	H	H	L
L	H	L	H
X	L	Z	Z

FUNCTION TABLE (RECEIVER)

DIFFERENTIAL INPUTS	ENABLE	OUTPUT
A − B	RE	R
$V_{ID} > 0.2$ V	L	H
-0.2 V $< V_{ID} < 0.2$ V	L	?
$V_{ID} < -0.2$ V	L	L
X	H	Z

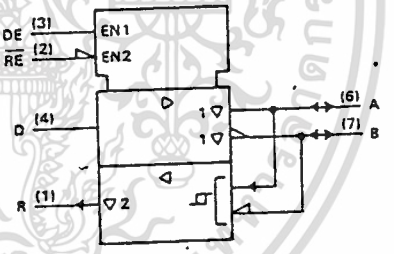
H = high level, L = low level, ? = indeterminate,
X = irrelevant, Z = high impedance (off)

description

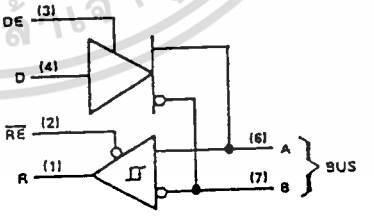
The SN75176A differential bus transceiver is a monolithic integrated circuit designed for bidirectional data communication on multipoint bus transmission lines. It is designed for balanced transmission lines and meets EIA Standard RS-422A and CCITT Recommendations V.11 and X.27.

The SN75176A combines a 3-state differential line driver and a differential-input line receiver both of which operate from a single 5-V power supply. The driver and receiver have active-high and active-low enables, respectively, that can be externally connected together to function as direction control. The driver differential outputs and the receiver differential inputs are connected internally to form differential input/output (I/O bus ports that are designed to offer minimum loading to the bus whenever the driver is disabled or V_{CC} = 0. These ports feature wide positive and negative common-mode voltage ranges making the device suitable for party-line applications.

logic symbol



logic diagram (positive logic)



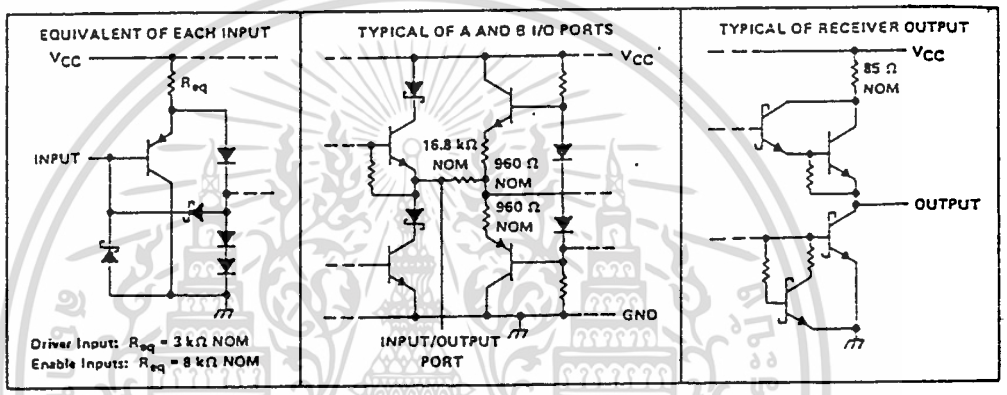
TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

description (continued)

The driver is designed to handle loads up to 60 mA of sink or source current. The driver features positive- and negative-current limiting and thermal shutdown for protection from line fault conditions. Thermal shutdown is designed to occur at a junction temperature of approximately 150°C. The receiver features a minimum input impedance of 12 kΩ, input sensitivity of ±200 mV, and a typical input hysteresis of 50 mV.

The SN75176A can be used in transmission line applications employing the SN75172 and SN75174 quadruple differential line drivers and the SN75173 and SN75175 quadruple differential line receivers.

schematics of inputs and outputs



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1)	7 V
Voltage at any bus terminal	−10 V to 15 V
Enable input voltage	5.5 V
Continuous total dissipation at (or below) 25°C free-air temperature (see Note 2):	
D package	725 mW
P package	1000 mW
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	260°C

NOTES: 1. All voltage values, except differential input/output bus voltage, are with respect to network ground terminal.
2. For operation above 25°C free-air temperature, derate the D package to 464 mW at 70°C at the rate of 5.8 mW/°C and derate the P package to 640 mW at 70°C at the rate of 8.0 mW/°C.

recommended operating conditions

		MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		4.75	5	5.25	V
Voltage at any bus terminal (separately or common-mode), V_I or V_{IC}		−7		12	V
High-level input voltage, V_{IH}	D, DE, and $\overline{RE}$	2			V
Low-level input voltage, V_{IL}	D, DE, and $\overline{RE}$			0.8	V
Differential input voltage, V_{ID} (see Note 3)				±12	V
High-level output current, I_{OH}	Driver			−60	mA
	Receiver			−400	μA
Low-level output current, I_{OL}	Driver			60	mA
	Receiver			8	
Operating free-air temperature, T_A		0		70	°C

NOTE 3: Differential-input/output bus voltage is measured at the noninverting terminal A with respect to the inverting terminal B.

DRIVER SECTION

driver electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted).

PARAMETER		TEST CONDITIONS			MIN	TYP†	MAX	UNIT
V_{IK}	Input clamp voltage	I_I = −18 mA					−1.5	V
V_{OH}	High-level output voltage	V_{IH} = 2 V, I_{OH} = −33 mA	V_{IL} = 0.8 V,			3.7		V
V_{OL}	Low-level output voltage	V_{IH} = 2 V, I_{OL} = 33 mA	V_{IL} = 0.8 V,			1.1		V
$\|V_{OD1}\|$	Differential output voltage	I_O = 0					2 V_{OD2}	V
$\|V_{OD2}\|$	Differential output voltage	R_L = 100 Ω,	See Figure 1		2	2.7		V
		R_L = 54 Ω,	See Figure 1		1.5	2.4		
$\Delta\|V_{OD}\|$	Change in magnitude of differential output voltage‡						±0.2	V
V_{OC}	Common-mode output voltage§	R_L = 54 Ω or 100 Ω,	See Figure 1				3	V
$\Delta\|V_{OC}\|$	Change in magnitude of common-mode output voltage‡						±0.2	V
I_O	Output current	Output disabled, See Note 4		V_O = 12 V			1	mA
				V_O = −7 V			−0.8	
I_{IH}	High-level input current	V_I = 2.4 V					20	μA
I_{IL}	Low-level input current	V_I = 0.4 V					−400	μA
I_{OS}	Short-circuit output current	V_O = −7 V					−250	mA
		V_O = V_{CC}					250	
		V_O = 12 V					500	
I_{CC}	Supply current (total package)	No load		Outputs enabled		35	50	mA
				Outputs disabled		25	40	

†All typical values are at V_{CC} = 5 V and T_A = 25°C.
‡$\Delta|V_{OD}|$ and $\Delta|V_{OC}|$ are the changes in magnitude of V_{OD} and V_{OC} respectively, that occur when the input is changed from a high level to a low level.
§In EIA Standard RS-422A, V_{OC}, which is the average of the two output voltages with respect to ground, is called output offset voltage, V_{OS}.
NOTE 4: This applies for both power on and power off. Refer to EIA Standard RS-422A for exact conditions.

driver switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
t_{DD}	Differential-output delay time	R_L = 60 Ω,	See Figure 3		40	60	ns
t_{TD}	Differential-output transition time				65	95	ns
t_{PZH}	Output enable time to high level	R_L = 110 Ω,	See Figure 4		55	90	ns
t_{PZL}	Output enable time to low level	R_L = 110 Ω,	See Figure 5		30	50	ns
t_{PHZ}	Output disable time from high level	R_L = 110 Ω,	See Figure 4		55	130	ns
t_{PLZ}	Output disable time from low level	R_L = 110 Ω,	See Figure 5		20	40	ns

RECEIVER SECTION

receiver electrical characteristics over recommended ranges of common-mode input voltage, supply voltage, and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS			MIN	TYP†	MAX	UNIT
V_{TH}	Differential-input high-threshold voltage	V_O = 2.7 V,	I_O = −0.4 mA				0.2	V
V_{TL}	Differential-input low-threshold voltage	V_O = 0.5 V,	I_O = 8 mA		−0.2‡			V
$V_{T+} - V_{T-}$	Hysteresis§					50		mV
V_{IK}	Enable-input clamp voltage	I_I = −18 mA					−1.5	V
V_{OH}	High-level output voltage	V_{ID} = −200 mV, See Figure 2	I_{OH} = −400 µA,		2.7			V
V_{OL}	Low-level output voltage	V_{ID} = −200 mV, See Figure 2	I_{OL} = 8 mA,				0.45	V
I_{OZ}	High-impedance-state output current	V_O = 0.4 V to 2.4 V					±20	µA
I_I	Line input current	Other input = 0 V, See Note 4	V_I = 12 V				1	mA
			V_I = −7 V				−0.8	
I_{IH}	High-level enable-input current	V_{IH} = 2.7 V					20	µA
I_{IL}	Low-level enable-input current	V_{IL} = 0.4 V					−100	µA
r_I	Input resistance				12			kΩ
I_{OS}	Short-circuit output current				−15		−85	mA
I_{CC}	Supply current (total package)	No load		Outputs enabled		35	50	mA
				Outputs disabled		26	40	

†All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡The algebraic convention, where the less-positive (more-negative) limit is designated minimum, is used in this data sheet for common-mode input voltage and threshold voltage levels only.
§Hysteresis is the difference between the positive-going input threshold voltage, V_{T+}, and the negative-going input threshold voltage, V_{T-}. See Figure 4.
NOTE 4: This applies for both power on and power off. Refer to EIA Standard RS-422A for exact conditions.

receiver switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output	V_{ID} = −1.5 V to 1.5 V, C_L = 15 pF,	See Figure 6		21	35	ns
t_{PHL}	Propagation delay time, high-to-low-level output				23	35	ns
t_{PZH}	Output enable time to high level	C_L = 15 pF,	See Figure 7		10	30	ns
t_{PZL}	Output enable time to low level				12	30	ns
t_{PHZ}	Output disable time from high level	C_L = 15 pF,	See Figure 7		20	35	ns
t_{PLZ}	Output disable time from low level				17	35	ns

PARAMETER MEASUREMENT INFORMATION

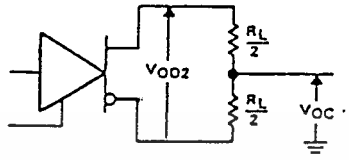


FIGURE 1. DRIVER V_{OD} AND V_{OC}

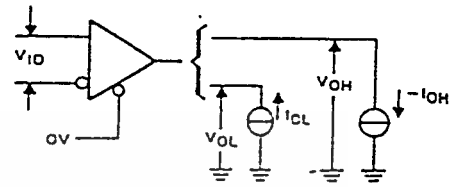


FIGURE 2. RECEIVER V_{OH} AND V_{OL}

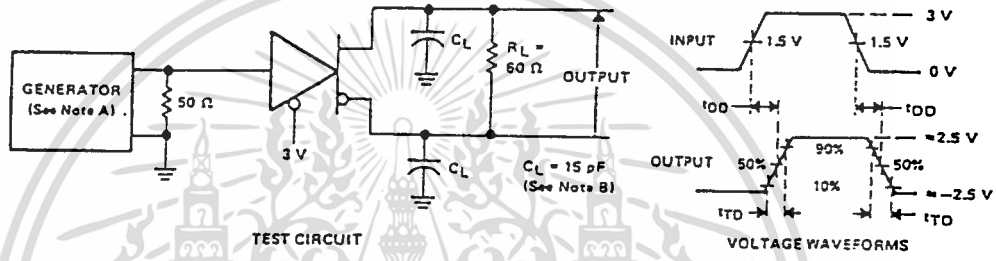


FIGURE 3. DRIVER DIFFERENTIAL-OUTPUT DELAY AND TRANSITION TIMES

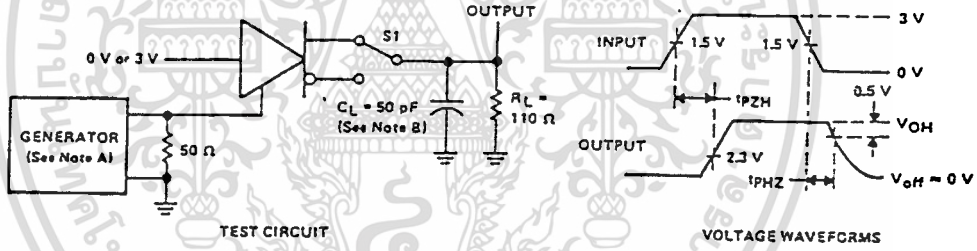


FIGURE 4. DRIVER ENABLE AND DISABLE TIMES

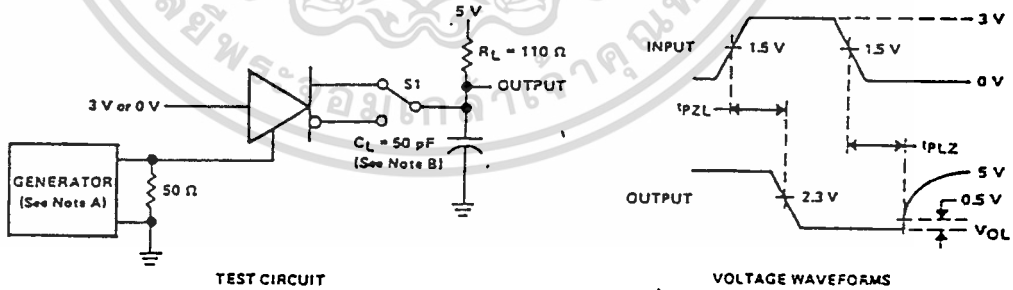


FIGURE 5. DRIVER ENABLE AND DISABLE TIMES

NOTES: A. The input pulse is supplied by a generator having the following characteristics: PRR = 1 MHz, 50% duty cycle, t_r ≤6 ns, t_f ≤6 ns, Z_{out} = 50 Ω.
B. C_L includes probe and jig capacitance.

PARAMETER MEASUREMENT INFORMATION

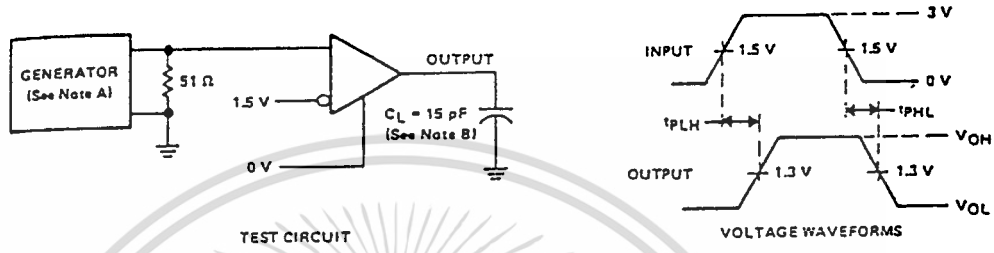


FIGURE 6. RECEIVER PROPAGATION DELAY TIMES

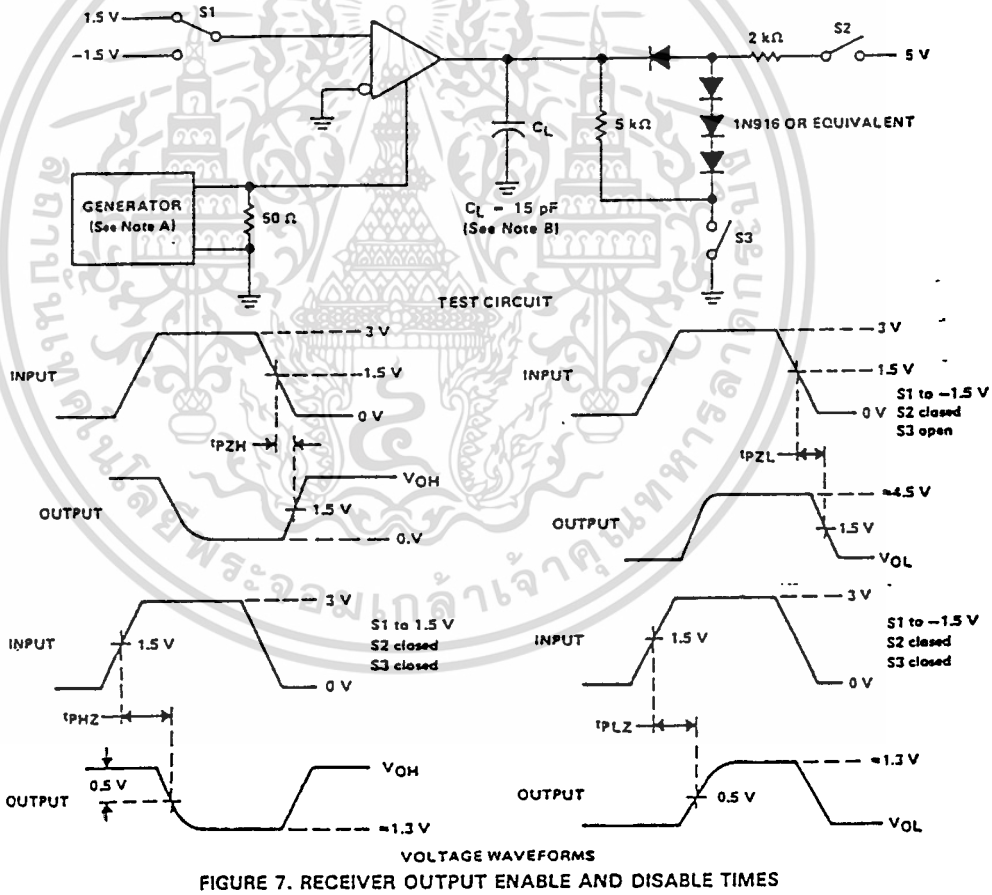


FIGURE 7. RECEIVER OUTPUT ENABLE AND DISABLE TIMES

NOTES: A. The input pulse is supplied by a generator having the following characteristics: PRR = 1 MHz, 50% duty cycle, t_r ≤6 ns, t_f ≤6 ns, Z_{out} = 50 Ω.
B. C_L includes probe and jig capacitance.

TYPICAL CHARACTERISTICS

DRIVER HIGH-LEVEL OUTPUT VOLTAGE
vs
DRIVER HIGH-LEVEL OUTPUT CURRENT

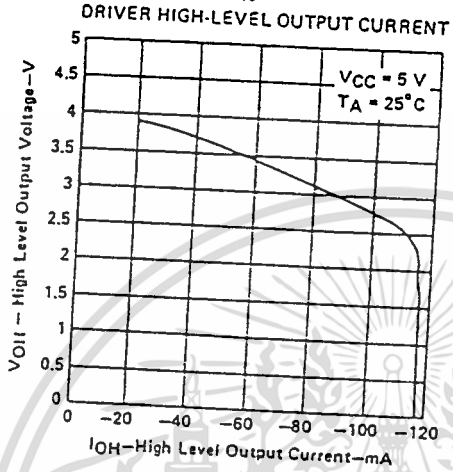


FIGURE 8

DRIVER LOW-LEVEL OUTPUT VOLTAGE
vs
DRIVER LOW-LEVEL OUTPUT CURRENT

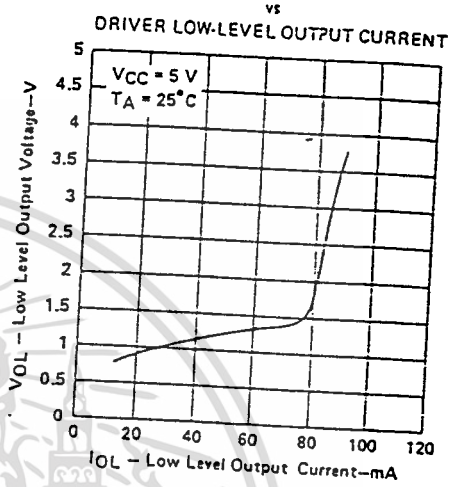


FIGURE 9

DRIVER DIFFERENTIAL OUTPUT VOLTAGE
vs
DRIVER OUTPUT CURRENT

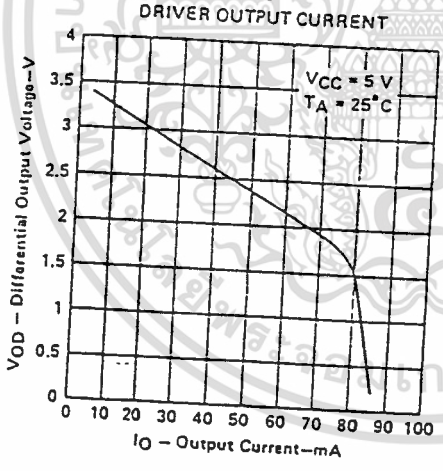


FIGURE 10

RECEIVER LOW-LEVEL OUTPUT VOLTAGE
vs
RECEIVER LOW-LEVEL OUTPUT CURRENT

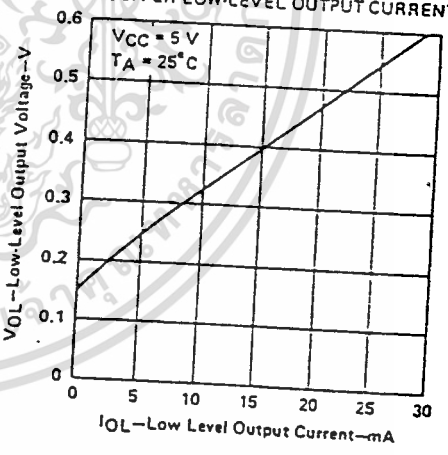


FIGURE 11

TYPICAL CHARACTERISTICS

RECEIVER LOW-LEVEL OUTPUT VOLTAGE
vs
FREE-AIR TEMPERATURE

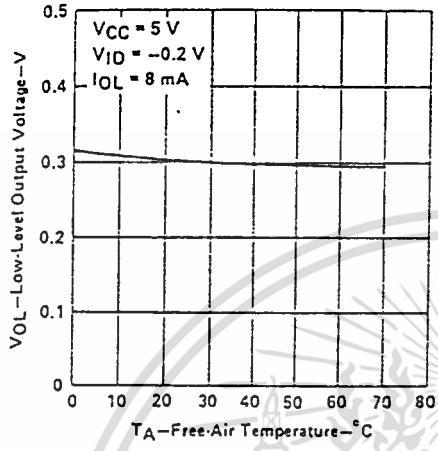


FIGURE 12

RECEIVER OUTPUT VOLTAGE
vs
ENABLE VOLTAGE

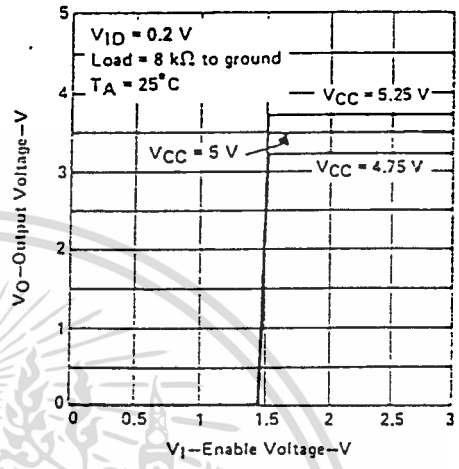


FIGURE 13

RECEIVER OUTPUT VOLTAGE
vs
ENABLE VOLTAGE

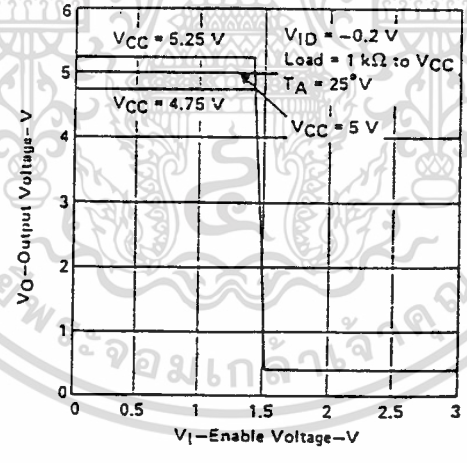
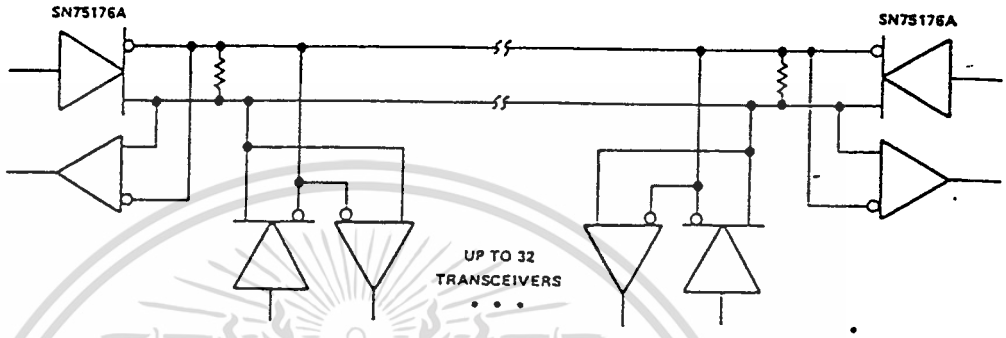


FIGURE 14

TYPICAL APPLICATION



NOTE: The line should be terminated at both ends in its characteristic impedance. Stub lengths off the main line should be kept as short as possible.

```
DECLARE SUB firecase (x!, y!)
DECLARE SUB notfire (x!, y!)
DECLARE SUB alarm ()
DECLARE FUNCTION roomposx! (number)
DECLARE FUNCTION roomposy! (number)
CLS
DIM SHARED floory AS INTEGER, floorx AS INTEGER, floorsp AS INTEGER,
floornum AS INTEGER
DIM SHARED path AS INTEGER, roomnum AS INTEGER, floor AS INTEGER
DIM SHARED onfire AS INTEGER, roomcolor AS INTEGER
floory = 100
floorx = 350
floorsp = 20
floornum = 4
path = 30
roomnum = 4
roomcolor = 10

SCREEN 12

FOR fl = 1 TO floornum

        LINE  (0, ((fl - 1) * floory) + floorsp)-(floorx, (fl *
              floory)), roomcolor, B
        LINE  (0, ((fl - 1) * floory) + floorsp + ((floory - path) /
              2))- (floorx, ((fl - 1) * floory) + floorsp + ((floory
              - path) / 2)), roomcolor
        LINE  (0, (fl * floory) - ((floory - path) / 2))-(floorx, (fl
              * floory) - ((floory - path) / 2)), roomcolor

        FOR room = 1 TO roomnum
              LINE (room * (floorx / roomnum), ((fl - 1) * floory)
                   + floorsp)-(room * (floorx / roomnum), ((fl - 1)
                   * floory) + floorsp + ((floory - path) / 2)),
                   roomcolor
               LINE (room * (floorx / roomnum), (fl * floory) -
                   ((floory - path) / 2))-(room * (floorx /
                   roomnum), fl * floory), roomcolor
        NEXT room

NEXT fl

LOCATE 3, 50
PRINT "Station #1"
LOCATE 9, 50
PRINT "Station #2"
LOCATE 15, 50
PRINT "Station #3"
LOCATE 21, 50
PRINT "Station #4"

'Note on Using Room Position Function...
'There only 1 parameter of 2 function.
'It's Integer...the room number#
' eg.   x = roomposx(room%)
'       y = roomposy(room%)

WHILE INKEY$ = ""
onfire = 0
LINE (0, 410)-(640, 460), 0, BF
LOCATE 27
FOR i% = 1 TO 4
```

```
floor = 8 * (i% - 1)
OUT (&H2F8), &H60                    'Send 60H to COM port
fire = INP(&H2F8)                    'Get value from COM port

a1 = fire AND 1                      'Check bit 1 whether on fire
        x = roomposx(1 + floor)      'Define position x1
        y = roomposy(1 + floor)      'Define position y1
        IF a1 > 0 THEN
                PRINT (1 + floor);   'Type firing position1
                firecase x, y        'Go to sub-program
                  ELSE
                notfire x, y         'Go to sub-program
        END.IF

a2 = fire AND 2                      'Check bit 2 whether on fire
        x = roomposx(2 + floor)      'Define position x2
        y = roomposy(2 + floor)      'Define position y2
        IF a2 > 0 THEN
                PRINT (2 + floor);   'Type firing position2
                firecase x, y        'Go to sub-program
                  ELSE
                notfire x, y         'Go to sub-program
        END IF

a3 = fire AND 4                      'Check bit 3 whether on fire
        x = roomposx(3 + floor)      'Define position x3
        y = roomposy(3 + floor)      'Define position y3
        IF a3 > 0 THEN
                PRINT (3 + floor);   'Type firing position3
                firecase x, y        'Go to sub-program
                  ELSE
                notfire x, y         'Go to sub-program
        END IF

a4 = fire AND 8                      'Check bit 4 whether on fire
        x = roomposx(4 + floor)      'Define position x4
        y = roomposy(4 + floor)      'Define position y4
        IF a4 > 0 THEN
                PRINT (4 + floor);   'Type firing position4
                firecase x, y        'Go to sub-program
                  ELSE
                notfire x, y         'Go to sub-program
        END IF

a5 = fire AND 16                     'Check bit 5 whether on fire
        x = roomposx(5 + floor)      'Define position x5
        y = roomposy(5 + floor)      'Define position y5
        AF a5 > 0 THEN
                PRINT (5 + floor);   'Type firing position5
                firecase x, y        'Go to sub-program
                  ELSE
                notfire x, y         'Go to sub-program
        END IF

a6 = fire AND 32                     'Check bit 6 whether on fire
        x = roomposx(6 + floor)      'Define position x6
        y = roomposy(6 + floor)      'Define position y6
        IF a6 > 0 THEN
                PRINT (6 + floor);   'Type firing position6
                firecase x, y        'Go to sub-program
                  ELSE
                notfire x, y         'Go to sub-program
        END IF
```

```
        a7 = fire AND 64                        'Check bit 7 whether on fire
                x = roomposx(7 + floor)         'Define position x7
                y = roomposy(7 + floor)         'Define position y7
                IF a7 > 0 THEN
                        PRINT (7 + floor);      'Type firing position7
                        firecase x, y           'Go to sub-program
                          ELSE
                        notfire x, y            'Go to sub-program
                END IF

        a8 = fire AND 128                       'Check bit 8 whether on fire
                x = roomposx(8 + floor)         'Define position x8
                y = roomposy(8 + floor)         'Define position y8
                IF a8 > 0 THEN
                        PRINT (8 + floor);      'Type firing position8
                        firecase x, y           'Go to sub-program
                          ELSE
                        notfire x, y            'Go to sub-program
                END IF

NEXT i%

IF onfire = 1 THEN                              'Have firing position(s)
        alarm                                   'Make warning sound
        onfire = 0
END IF
WEND
END

SUB alarm
FOR i = 500 TO 1000 STEP 20
        SOUND i, i / 1000
NEXT
FOR i = 1000 TO 500 STEP -20
        SOUND i, i / 1000
NEXT
END SUB

SUB firecase (x, y)
LINE (x - 42, y - 15)-(x + 42, y + 15), 12, BF
onfire = 1
END SUB

SUB notfire (x, y)
LINE (x - 42, y - 15)-(x + 42, y + 15), 0, BF
END SUB

FUNCTION roomposx (number)
temp = number MOD roomnum
IF temp = 0 THEN temp = roomnum
roomposx = ((floorx / roomnum) * temp) - ((floorx / roomnum) / 2)
END FUNCTION

FUNCTION roomposy (number)
temp = number / roomnum
IF temp > INT(temp) THEN temp = INT(temp) + 1
temp = temp / 2
temp = temp + .5
IF temp = INT(temp) THEN
        'case Upper Row
        roomposy = ((INT(temp) - 1) * floory) + floorsp + ((floory -
                path) / 4)
ELSE
        'case Lower Row
```

```
        roomposy = (INT(temp) * floory) - ((floory - path) / 4)
END IF
END FUNCTION
```

โปรแกรมแสดงผลทางมอนิเตอร์

```
            ORG   8000H

            IND   EQU   21H

START:      MOV     DPTR,#9000H     ;INITIALIZE DATA POINTER TO
                                     POINT TO
                                    ;ADDRESS 9000H OF EXTERNAL RAM
            MOV     R1,#00H         ;INITIALIZE REGISTER R1 THE FIRST
                                    ;ADDRESS VALUE
BEGIN:      MOV     SCON,#0C8H      ;MODE3,SETBIT TB8,RESETBIT REN,SM2
            MOV     TMOD,#020H      ;TIMER1 MODE2 AUTORELOAD
            MOV     TH1,#0FAH       ;BAUD RATE 4.8KHz
            SETB    TR1             ;START TIMER
            MOV     A,R1            ;MOVE ADDRESS TO ACCUMULATOR
            INC     R1
            ACALL   TRANS
            SJMP    RECV

TRANS:      CLR     TI              ;BEGIN TO TRANSMIT ADDRESS
            SETB    P1.0            ;WRITE RS 485
            MOV     SBUF,A
            JNB     TI,$
            CLR     P1.0
            CLR     TI
            RET

RECV:       CLR     RI              ;BEGIN TO RECEIVE STATUS PROCESS
            SETB    REN
            CLR     P1.0            ;READ RS 485
LOOP1:      JBC     RI,RECV1        ;WAIT FOR STATUS FROM SLAVE
            SJMP    LOOP1

RECV1:      MOV     A,SBUF
            CLR     RI
            MOVX    @DPTR,A         ;MOVE STATUS TO EXTERNAL RAM
                                     POINTED
                                    ;BY DATA POINTER
            INC     DPTR
            CJNE    R1,#04H,BEGIN   ;IF TRANSMIT PROCESS HAS NOT
                                     FINISHED
                                    ;YET,TRANSMIT NEXT ADDRESS

SENDPC:     MOV     SCON,#040H      ;MODE1
            MOV     TMOD,#020H      ;TIMER1 MODE2 AUTORELOAD
            MOV     TH1,#0FAH       ;BAUD RATE 4.8KHz
            SETB    TR1             ;START TIMER
            MOV     DPTR,#8FFFH

WAITPC:     CLR     RI
            SETB    REN
LOOP3:      JBC     RI,CHECK        ;WAIT TRIGGER FROM PC
            SJMP    LOOP3
CHECK:      MOV     A,SBUF
            CLR     RI
            XRL     A,#060H         ;CHECK NUMBER FROM PC
            JNZ     LOOP3          ;IF RECEIVE WRONG NUMBER, RECEIVE
                                    AGAIN
LOOP4:      INC     DPTR
            MOVX    A,@DPTR         ;MOVE STATUS FROM EXTERNAL RAM TO
                                    ;ACCUMULATOR
            CLR     TI
            MOV     SBUF,A          ;TRANSMIT STORED STATUS TO PC
```

```
            JNB     TI,$
            CLR     TI
            MOV     A,DPL           ;MOVE THE VALUE OF LOW BYTE DATA
                                    ;POINTER (DPL) TO ACCUMULATOR
            CJNE    A,#04H,WAITPC

            MOV     IND,#00H        ;BEGIN TO SHOW 7-SEGMENT 'FINISH'
                                     TASK
            MOV     R4,#04H
      X:    MOV     R3,#00H
 FINISH:    MOV     R7,#0F8H         ;REGISTER R7 STORES THE DIGIT OF
                                      7-SEGMENT
            MOV     IND,#00H
SEGMENT:    MOV     A,IND
            INC     IND
            MOV     DPTR,#TABLE
            MOVC    A,@A+DPTR        ;MOVE DATA FROM TABLE TO ACC
            MOV     DPTR,#0F800H     ;DATA POINTER POINT TO PORT A
            MOVX    @DPTR,A
            INC     DPTR             ;DATA POINTER POINT TO PORT B
            MOV     A,R7
            INC     R7
            MOVX    @DPTR,A
            ACALL   DELAY
            CJNE    R7,#0FEH,SEGMENT ;RUN UNTIL THE LAST VALUE IN THE
                                      TABLE
            DJNZ    R3,FINISH
            DJNZ    R4,X
            MOV     A,#00H
            MOV     DPTR,#0F800H
            MOVX    @DPTR,A
            ACALL   DELAY2
            LJMP    START

  DELAY:    MOV     R5,#03H
   DEL2:    MOV     R6,#00H
   DEL1:    DJNZ    R6,$
            DJNZ    R5,DEL2
            RET

 DELAY2:    MOV     R5,#00H
   DEL3:    MOV     R6,#00H
   DEL4:    DJNZ    R6,$
            DJNZ    R5,DEL3
            RET

  TABLE:    DB   071H,06H,037H,06H,06DH,076H

            END
```

โปรแกรมการสื่อสารของตัวแม่

```
        ORG     0000H

START:  ACALL   DELAY       ;POWER ON DELAY
        MOV     R0,#00H     ;REG R0 STORE SLAVE'S ADDRESS
        MOV     SCON,#0F0H  ;MODE 3, SETBIT REN
        MOV     TMOD,#020H  ;TIMER1 MODE2
        MOV     TH1,#0FAH   ;BAUD RATE 4.8KHz
        SETB    TR1

        CLR     P3.4        ;READ RS-422
        CLR     RI
LOOP:   JBC     RI,RECV     ;WAIT FOR ADDRESS FROM MASTER
        SJMP    LOOP

RECV:   MOV     A,SBUF      ;RECEIVE ADDRESS
        CLR     RI
        CLR     SM2
        XRL     A,R0        ;ADDRESS IS MATCHED OR NOT
        JNZ     LOOP        ;IF NOT MATCH RETURN

PORT:   MOV     P1,#0FFH    ;TURN OFF 'FET' IN PORT1
        MOV     A,P1        ;RECEIVE STATUS FROM SENSOR
        SETB    P3.4        ;WRITE RS-422
        MOV     SBUF,A      ;ECHO TO MASTER
        JNB     TI,$
        CLR     P3.4
        CLR     TI
        LJMP    START

DELAY:  MOV     R3,#00H     ;POWER ON DELAY
        DJNZ    R3,$
        RET

        END
```

โปรแกรมการสื่อสารของตัวลูก

หนังสืออ้างอิง

1. Jordan & Churchill ; "Communications and networking for the IBM PC and compatibles", Brady , 1990.

2. Kenneth J. Ayala ; "The 8051 Microcontroller Architecture, programming, and applications " , West Publishing company, 1991.

3. Peter W. Gofton ; "Mastering Serisl Communication", Sybex Inc., 1994

4. สุเจตน์ จันทรังษ์ ," Single Chip Microcontroller 8051 " , กรุงเทพ : สำนักพิมพ์วิทยาลัยมหานคร , 2535.

5. ปรเมษฐ์ ประนยานันท์ และ ปิยะพงศ์ เผ่าวณิช , " คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51 " , กรุงเทพ : บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2536.

6. พิพัฒน์ เลาหสงคราม , " ภาษาแอสเซมบลีสำหรับไมโครคอนโทรลเลอร์ MCS-51 ", กรุงเทพ : สำนักพิมพ์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2537.

7. จิรศักดิ์ เหลืองอุไร, "การสื่อสารอนุกรมบน PC", กรุงเทพ : บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2538

8. สุทธิศักดิ์ พงศ์ธนาพาณิช, "การใช้งานโปรแกรม QuickBASIC", กรุงเทพ : บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2537

9. พิพัฒน์ เลาหสงคราม, "ไมโครคอนโทรลเลอร์", กรุงเทพ : สำนักพิมพ์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2537