ระบบรักษาความปลอดภัยสำหรับอาคาร

THE ELECTRONICS SECURITY SYSTEM IN BUILDING

โดย

นาย กัลป์	คติแสงศุภร	321011
นาย สุรศักดิ์	อุไรรัตน์	321006
นาย เอกรัฐ	ธรรมทอง	321434

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเลคทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ระบบรักษาความปลอดภัยสำหรับอาคาร

นาย กัลป์ คติแลงศุภร 32.1011
นาย ลุรคักดิ์ อุไรรัตน์ 32.1066
นาย เอกรัฐ ธรรมทอง 32.1434

อาจารย์ที่ปรึกษา รศ.ดร.มนัส สังวรศิลป์

บทคัดย่อ

โครงงานชิ้นนี้ เป็นการออกแบบและพัฒนาระบบรักษาความปลอดภัยในอาคารซึ่งมีพื้นที่เป็นบริเวณกว้าง โดยการระบบแบบเครือข่าย แบ่งออกเป็นโซน มีศูนย์กลางการควบคุมโดยใช้ IBM PC จัดการรับคำสั่งควบคุม แสดงผล และบันทึกข้อมูล โดยการส่งข้อมูลแบบอนุกรมตามมาตรฐาน RS422 ไปยังตัวประมวลผลสื่อสาร(Communication Processor) เพื่อจัดการในการติดต่อกับ ตัวประมวลผลปฏิบัติการ(Operating Processor) ตัวประมวลผลทั้ง 2 แบบ ใช้ CPU เบอร์ 8031 ในส่วนปฏิบัติการนี้ จะกระจายไปตามพื้นที่ต่าง ๆ เมื่อมีการตรวจจับเหตุการณ์ต่าง ๆ ตามลักษณะของตัวตรวจจับแต่ละชนิด เช่น ตรวจจับความร้อน(ไฟไหม้),ผู้บุกรุก เป็นต้น

การพัฒนาซอฟท์แวร์ ทำให้ระบบมีความสามารถสูง มีความยืดหยุ่นได้มาก โดยผู้ควบคุมสามารถตั้งค่าต่าง ๆ ให้เหมาะสมกับลักษณะการการปฏิบัติงานของบุคคลในสถานที่นั้น

THE ELECTRONICS SECURITY SYSTEM IN BUILDINGS

Mr. Kalp Sasisangsuporn

Mr. Surasak Urairat

Mr. Akarat Thammathong

Assoc.Prof.Dr. Manas Sungworasilp

Abtract

This project is designing and development security system in buildings having a large area. The project is a network system. The network system is divided in zone. And the center is IBM PC that manages about display ,recieving intruction.data recording. The center (IBM PC) communicates with COMMUNICATION PROCESSOR by serial communication (RS422 standard). And the COMMUNICATION PROCESSOR communicates with several OPERATING PROCESSOR.

Both COMMUNICATION and OPERATING PROCESSOR use 8031 CPU. The OPERATING PROCESSOR distributed in many area for sense the unusual events following by the kinds of sensors such as heat sensor(fire), infra-red sensor(invader) etc.

The development of software made the system having high capability and easy to apply. User can set parameters to appropriate performance for each person and each place.

สารบัญ

บทที่ 1
บทนำ
ระบบรักษาความปลอดภัยในอาคาร

การรักษาความปลอดภัยในขอบเขตบริเวณกว้างเป็นการยากที่จะดูแลได้ทั่วถึง ในปัจจุบันจึงมีการพัฒนาระบบทางอิเล็คทรอนิกส์มาช่วยในการรักษาความปลอดภัย ดังเช่นปริญญานิพนธ์นี้ ได้ออกแบบให้ระบบเป็นแบบระบบการควบคุมจากศูนย์กลางโดยมีสถานีลูกข่ายระยะไกล ได้มีการนำเอาไมโครคอนโทรลเลอร์มาใช้ในระบบ ทำให้ระบบมีความคล่องตัวในการปฏิบัติงาน และมีความยืดหยุ่นในการติดตั้งระบบ

1.1 ลักษณะการทำงาน

การทำงานของระบบ จะมีศูนย์กลางการควบคุม ที่จุดนี้ผู้ปฏิบัติงานสามารถควบคุม, เฝ้าดูการแสดงผล, ทำรายงานเหตุการณ์ที่เกิดขึ้น ซึ่งการสั่งการ, การประมวลผลข้อมูลที่เกิดขึ้น ได้มาจากการสื่อสารข้อมูลกับสถานีลูกข่ายระยะไกล ระบบประกอบกันขึ้นเป็นโครงข่ายแบบต้นไม้ (Tree Network) กล่าวคือสถานีลูกข่ายจะกระจายออกจากศูนย์กลาง

สถานีลูกข่ายจะมี 2 ระดับคือ

1) หน่วยประมวลผลการสื่อสารข้อมูล (Datacommunication Processing Unit : DPU) ที่หน่วยนี้จะจัดการการสื่อสารข้อมูลระหว่างศูนย์กลางกับหน่วยปฏิบัติการซึ่งเป็นสถานีระดับล่างสุด

2) หน่วยประมวลผลปฏิบัติการ (Operating Processing Unit : OPU) เป็นการทำงานด้านการตรวจจับ (Sensing) เหตุการณ์ต่าง เช่นการบุกรุก, ไฟไหม้ เป็นต้น แล้วรายงานผลให้ศูนย์กลางทราบ รวมทั้งการรับคำสั่งในการควบคุมอุปกรณ์ป้องกันที่ติดตั้งไว้

1.2 การติดตั้งระบบ

ที่ศูนย์กลางใช้เครื่องคอมพิวเตอร์ IBM PC หรือเครื่อง Compatible ทำงานสถานีลูกข่ายจะติดตั้งให้เหมาะสมตามแบบแปลนของอาคาร โดยคำนึงถึงความประหยัด และประสิทธิภาพ

สถานีลูกข่ายใช้บอร์ด Microcontroller 8031 (ตระกูล MCS51) ทั้ง 2 ระดับ แต่จะมีวงจร Interface ที่แตกต่างกัน

1.3 การใช้งาน

ซอฟแวร์ของระบบได้รับการออกแบบให้ใช้งานได้ง่าย โดยมีการแสดงผลแบบกราฟฟิก มีฟังก์ชั่นการใช้งานต่างๆดังนี้

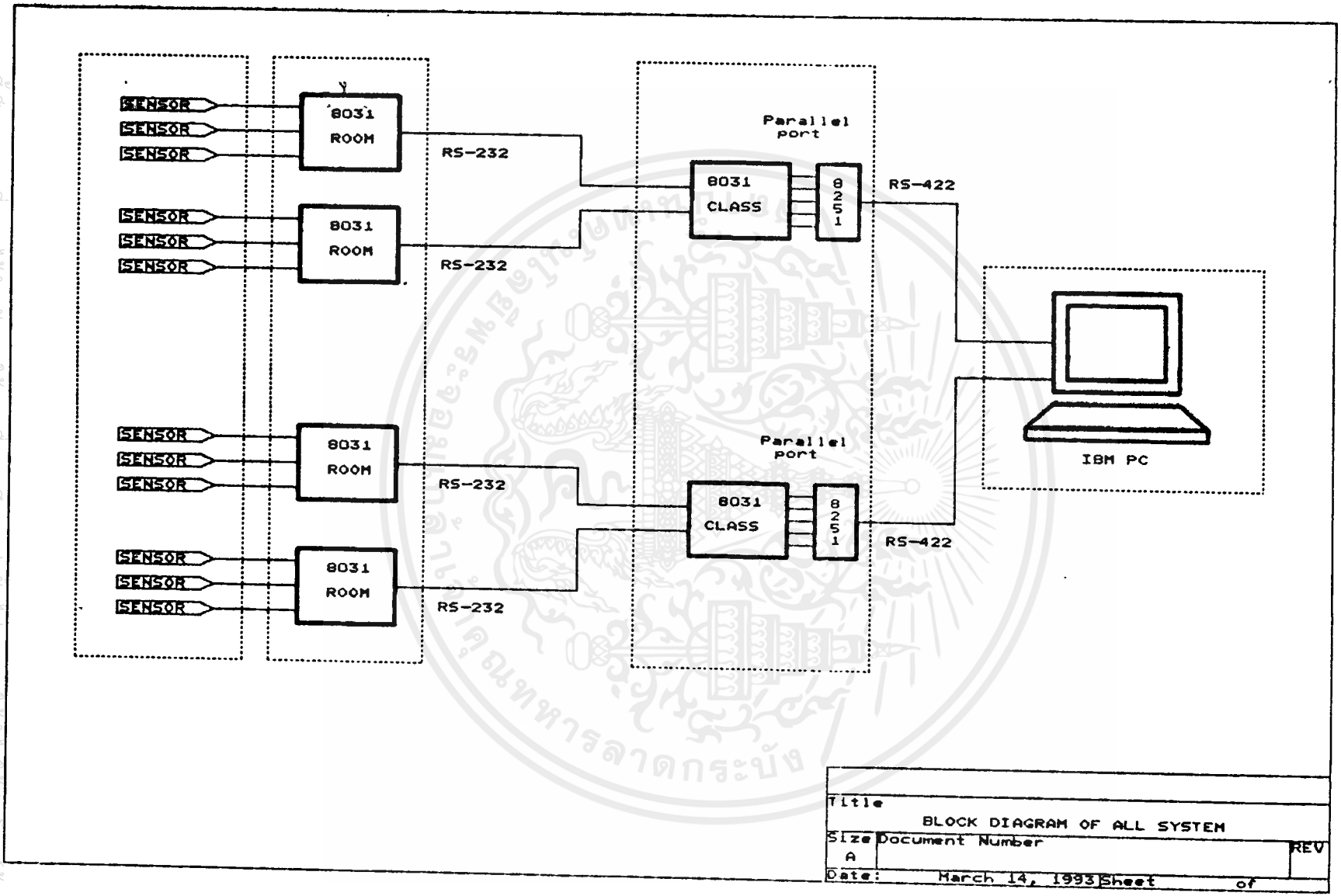
ก) การแสดงผลกราฟฟิก การแสดงผลทางจอมอนิเตอร์เป็นแบบกราฟฟิก คือมีรูปแผนผังอาคาร และข้อมูลต่างๆ รวมทั้งแสดงเหตการณ์ที่เกิดขึ้นด้วย

ข) การบันทึกข้อมูล ระบบจะบันทึกการตรวจจับที่เกิดขึ้น ทั้งที่ปกติและผิดปกติ ผู้ควบคุมสามารถขอดูได้

ค) การควบคุมอุปกรณ์ป้องกัน ผู้ควบคุมสามารถควบคุมด้วยตัวเอง หรือตั้งให้ระบบควบคุมอัตโนมัติก็ได้

ง) ฟังก์ชั่นพิเศษอื่นๆ เพื่อช่วยอำนวยความสะดวกต่างๆ

นอกจากนี้ ระบบยังสามารถทำงานพื้นฐานได้ แม้ว่าศูนย์กลางจะขัดข้อง และถ้าส่วนใดส่วนหนึ่งของสถานีลูกข่ายเสียระบบสามารถตรวจสอบได้



รูปที่ 1.1 แสดง Block diagram ของระบบทั้งหมด

บทที่ 2
ทฤษฎีและหลักการ

2.1 การทำงานของไมโครคอนโทรลเลอร์ MCS-51

การจัดหน่วยความจำ

ในระบบของ MCS-51 จะมีการแบ่งหน่วยความจำเหมือนกับ ซีพียูทั่วๆไปคือจะแบ่งเป็น 2 ลักษณะตามชนิดของข้อมูลที่จัดเก็บดังนี้

-หน่วยความจะข้อมูล(data memory)

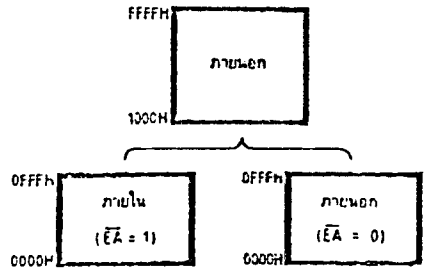
-หน่วยความจำตามโปรแกรม(programe memory)

หน่วยความจำข้อมูลหมายถึง หน่วยความจำในส่วนที่เป็น RAM ซึ่งเราสามารถอ่านหรือเขียนข้อมูลเปลี่ยนแปลงได้ตลอดเวลา แต่ไม่สามารถรันโปรแกรมในส่วนนี้ได้

หน่วยความจำโปรแกรมจะหมายถึงหน่วยความจำที่สามารถอ่านข้อมูลได้อย่างเดียว ซึ่งเป็นส่วนบรรจุโปรแกรมให้ MCS-51 ทำงาน โดยหน่วยความจำทั้งสองลักษณะนี้จะถูกแยกออกจากกันด้วยคำสั่งทางซอฟต์แวร์และลักษณะการติดต่อกันทางฮาร์ดแวร์ด้วย กล่าวคือจะมีคำสั่งเฉพาะการติดต่อกับหน่วยความจำชนิดใดชนิดหนึ่งและจัดสัญญาณสโตรบในการติดต่อกับหน่วยความจำแต่ละชนิด

หน่วยความจำโปรแกรม(programe memory)

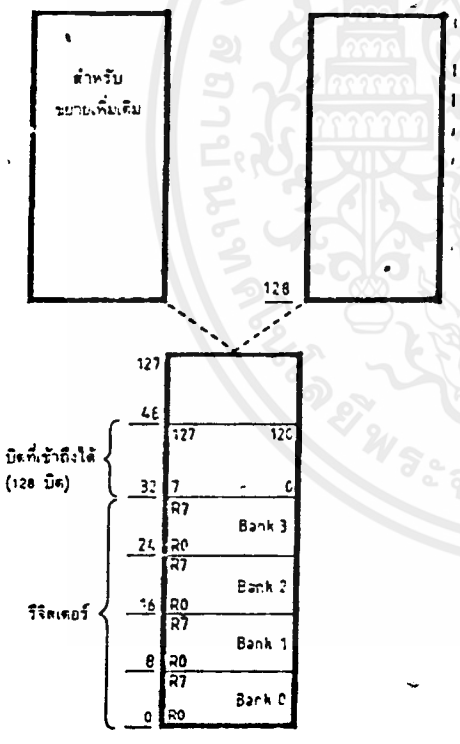
ใน MCS-51 จะแบ่งหน่วยความจำโปรแกรมออกเป็นสองส่วน หน่วยความจำโปร--แกรมภายในและหน่วยความจำโปรแกรมภายนอก หน่วยความจำโปรแกรมภายในได้แก่ หน่วยความจำประเภท ROM หรือ EPROM ที่อยู่ภายในตัว MCS-51 เบอร์ 8051 หรือ 8751 ซึ่งมีขนาด 4 กิโลไบต์ ส่วนหน่วยความจำโปรแกรมภายนอกหมายถึง หน่วยความจำที่ต่ออยู่ภายนอก MCS-51 ซึ่งอาจเป็นการต่อหน่วยความจำโปรแกรมภายนอกทั้งหมดก็เป็นได้ ในกรณีของ 8031 เพราะภายในตัวของ 8031 เองไม่มีหน่วยความจำโปรแกรมเหมือนเบอร์ 8051 หรือ 8751 โดยที่ MCS-51 สามารถเลือกให้รัน ในหน่วยความจำโปรแกรมภายในหรือภายนอกก็ได้ โดยควบคุมที่ขา EA แต่ใน กรณีของ 8031 จะต้องต่อขา EA ลงกราวด์เสมอ การจะแบ่งเนื้อที่หน่วยความจำโปรแกรมภายในและภายนอก แสดงในรูปที่ 2.1.1



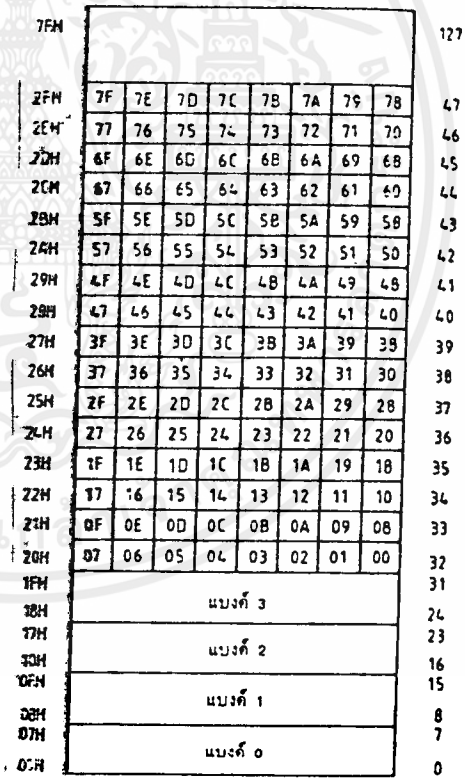
รูปที่ 2.1.1 แสดงการจัดพื้นที่หน่วยความจำโปรแกรมภายในและภายนอก

หน่วยความจำข้อมูล

หน่วยความจำข้อมูลของ MCS-51 ก็สามารถแบ่งออกเป็น 2 ส่วนเช่นกัน คือ หน่วยความจำข้อมูลภายใน และ หน่วยความจำข้อมูลภายนอก โดยหน่วยความจำข้อมูลภายนอกจะสามารถเข้าถึงได้ด้วยคำสั่ง MOVX เท่านั้น สำหรับหน่วยความจำข้อมูลภายในของ 8031, 8051 และ 8751 จะมีทั้งหมด 256 ไบต์ โดยแบ่งเป็น 128 ไบต์ส่วนบน ซึ่งใช้เป็นที่เก็บของรีจิสเตอร์พังก์ชั่นพิเศษ และอีก 128 ไบต์ส่วนล่าง ซึ่งจะถูกใช้งานทั่วไป ดังรูปที่ 2.1.2 และ 2.1.3



รูปที่ 2.1.2 การจัดพื้นที่หน่วยความจำข้อมูล



รูปที่ 2.1.3 แสดงพื้นที่การกำหนดตำแหน่งบิตของหน่วยความจำข้อมูลภายในรูปที่ 2.1.2

จากรูปที่ 2.3 สามารถบอกได้ว่าพื้นที่หน่วยความจำข้อมูลภายในส่วน 128 ไบต์ล่าง (ตำแหน่ง00H-7FH) จะถูกแบ่งเป็น 3 ส่วนคือ ส่วนของรีจิสเตอร์แบงค์ (00H-1FH)ส่วนพิเศษที่ สามารถเข้าถึงตำแหน่งบิตได้โดยตรง (20H-2FH) และส่วนที่ใช้งานทั่วไป (30H -7FH) ส่วนของรีจิสเตอร์ทั้งหมดมี 4 แบงค์ แต่สามารถใช้งานได้ทีละ 1 แบงค์เท่านั้น

การเลือกใช้แบงค์ใดสามารถกำหนดค่าในรีจิสเตอร์ PSW

คำสั่งที่ใช้ในการติดต่อหน่วยความจำข้อมูล ซึ่งสามารถแบ่งลักษณะการกำหนดเลขที่อยู่ของหน่วยความจำข้อมูลได้ 4 โหมดด้วยกันคือ

1) โหมดการกำหนดเลขที่อยู่รีจิสเตอร์ เป็นการติดต่อกับข้อมูลที่อยู่ในรีจิสเตอร์นั้นโดยตรง

2) โหมดการกำหนดเลขที่อยู่โดยตรง จะเป็นการติดต่อกับข้อมูลที่ตำแหน่งแอดเดรสที่กำหนดโดยตรง ซึ่งจะต้องอยู่บริเวณตำแหน่งของ RAM ภายใน 256 ไบต์

3) โหมดการกำหนดเลขที่อยู่ข้อมูลโดยทันที จะเป็นการติดต่อกับข้อมูลคงที่ ตัวอย่าง

เช่น MOV A,40H ย้ายข้อมูล 40H ไปไว้ที่ ACC

ADD A,41H บวกข้อมูลใน ACC กับข้อมูลที่ตำแหน่ง 41H

4) โหมดการกำหนดเลขที่อยู่รีจิสเตอร์โดยอ้อม จะเป็นการติดต่อกับข้อมูลที่ใช้ค่าในตัวรีจิสเตอร์ R0 หรือ R1 เป็นตัวชี้ตำแหน่ง ตัวอย่างเช่น

MOV A,#22H นำค่า 22H ไปไว้ใน ACC

ADD A,#41H นำค่า 41H บวกเข้ากับค่าใน ACC

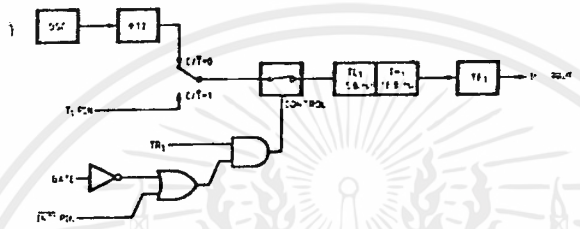
ตัวจับเวลาและตัวนับ (Timer & Counter)

ในไอซีเบอร์ 8031 8051 และ 8751 จะมีตัวจับเวลาและตัวนับขนาด 16 บิตจำ-นวน 2 ตัวคือ Timer/counter 0 และ Timer/counter 1 ส่วนเบอร์ 8032/8052 จะมีเพิ่มอีก 1 ตัว โดยแต่ละตัวสามารถที่จะกำหนดให้ทำงานเป็นตัวจับเวลาหรือตัวนับได้ โดยการเซตหรือเคลียร์บิต C/T ที่ตัวรีจิสเตอร์ควบคุม TMOD ซึ่งอยู่ในกลุ่มรีจิสเตอร์ฟังก์ชั่นพิเศษในการกำหนดให้ทำงานเป็นตัวจับเวลา ตัวรีจิสเตอร์ TH1 และ TL1 ซึ่งทำหน้าที่เป็นตัวเก็บค่าจำนวนพัลส์ที่เข้ามาจะเพิ่มค่าขึ้นทุกๆ แมชีนไซเกิลจะประกอบด้วย 12 คาบออสซิล-เลเตอร์ ดังนั้นอัตราการนับในแต่ละครั้งจะใช้เวลาเท่ากับ 1/12 ของความถี่ ออสซิลเล-เตอร์ ซึ่งส่วนใหญ่ใช้ในงานอินเตอร์รัพต์ RTC(Real Time Clock)และถ้าให้ทำงานเป็นตัวนับ รีจิสเตอร์ตัวนับจะเพิ่มค่าทุกครั้งที่มีการเปลี่ยนแปลง สถานะจาก "1" เป็น "0" ที่ขา T0 หรือ T1 โดยอัตราความถี่สูงสุดที่สามารถนับได้ต้องไม่เกิน 1/24 ของความถี่ออส-ซิลเลเตอร์ตัวจับเวลาและตัวนับสามารถโปรแกรมให้ทำงานได้ต่างกันถึง 4 โหมดโดยการตั้งค่าในรีจิสเตอร์ TMOD ซึ่งมีการทำงานในแต่ละโหมดดังนี้

<u>โหมด0</u> รีจิสเตอร์ตัวนับจะถูกกำหนดให้มี 13 บิต ประกอบด้วยรีจิสเตอร์ TH1 จำนวน 8 บิต และ TL1 อีก 5 บิตอันดับต่ำซึ่งสามารถกำหนดให้เป็นตัวจับเวลาหรือตัวนับ

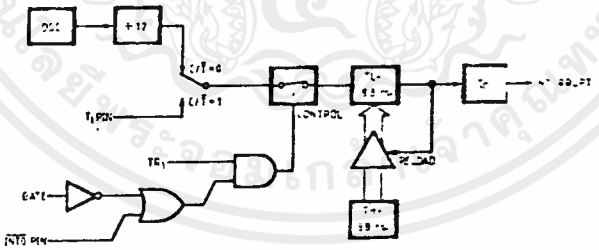
ได้โดยการเซตหรือเคลียร์ที่บิต C/T ในตัวรีจิสเตอร์ TMOD ...ทำงานของรีจิสเตอร์ ตัวนับจับขึ้นครั้งละ 1 เมื่อมีสัญญาณเข้ามา 1 ลูกและเมื่อนับจนเป็น ... ทุกบิตก็จะกลับมาเป็น 0 อีกครั้งหมดทุกบิตใหม่ ซึ่งจะเป็นการเกิดโอเวอร์โฟลว์ไปทดแทนอินเตอร์รัพต์ TF1 ให้เป็น 1 ลักษณะวงจรแสดงดังรูปที่ 3.4

โหมด 1 การทำงานจะเหมือนกับโหมด 0 ทุกอย่างยกเว้นรีจิสเตอร์ ตัวนับจะเป็นขนาด 16 บิต



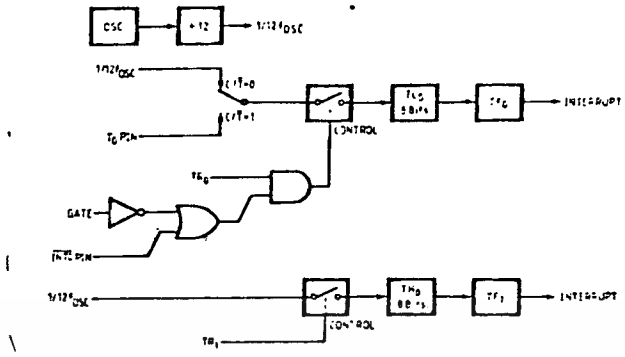
รูปที่ 2.1.4 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 0 และโหมด 1

โหมด2 จะใช้รีจิสเตอร์ TL1 เป็นตัวนับเพียงตัวเดียวและเมื่อ TL1 นับจนเป็น 1 หมดทุกบิต จะมีการโหลดค่าจากรีจิสเตอร์ TH1 เข้ามาไว้ใน TL1 โดยอัตโนมัติ และทำการทดแฟลกอินเตอร์รัพต์ TF1 ให้เป็น 1 ค่าใน TH1 นี้ เราสามารถตั้งค่าได้ด้วย Software ลักษณะวงจรแสดง ดังรูปที่ 2.5



รูปที่ 2.1.5 แสดงการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 2

โหมด 3 เป็นการเพิ่มตัวจับเวลาขึ้นอีก 1 ตัว และมีขนาด 8 บิตทั้งคู่ซึ่งลักษณะการทำงานอื่นๆ เหมือนกันกับโหมด 0 การทำงานแสดงดังรูปที่ 2.6



รูปที่ 2.1.6 แสดงวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 0 ในโหมด 3

สำหรับรายละเอียดของบิตควบคุมในรีจิสเตอร์ TMOD และรีจิสเตอร์ TCON แสดงในตารางตามลำดับ

ตารางที่ 2.1.1 แสดงบิตควบคุมรีจิสเตอร์ TMOD

TIMER/COUNTER1				TIMER/COUNTER2			
GATE	C/T	M1	M0	GATE	C/T	M1	M0

GATE : เซตเป็น 1 จะเป็นการอินาเบิลตัวจับเวลา/ตัวนับให้ถูกควบคุมด้วยขา INT ต้องมีสถานะสูงและบิต TRX ใน TCON ต้องเซตเป็น 1 จึงจะเริ่มทำงานแต่ถ้า GATE เป็น 0 ตัวจับเวลา/ตัวนับจะถูกควบคุมให้ทำงานด้วยบิต TRXเท่านั้น

C/T : เป็นบิตควบคุมในการเลือกการทำงานว่าทำงานเป็นตัวจับเวลาหรือเป็นตัวนับ ถ้าบิตมีค่าเป็น 0 จะทำงานเป็นตัวจับเวลา ถ้าเป็น 1 จะทำงานเป็นตัวนับ

M1,M0 : เป็นตัวเลือกโหมดการทำงานดังนี้

M1	M2	โหมดการทำงาน
0	0	โหมด 0
0	1	โหมด 1
1	0	โหมด 2
1	1	โหมด 3

ตารางที่ 2.1.2 แสดงบิตควบคุมรีจิสเตอร์ TCON

TF_1	TR_1	TF_0	TR_0	IE_1	IT_1	IE_0	IT_0

TF_x : เป็นแฟลกอินเตอร์รัพต์จะถูกเซตด้วยฮาร์ดแวร์เมื่อเกิดโอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับ และจะเคลียร์ตัวเองโดยอัตโนมัติเมื่อทำการอินเตอร์รัพต์ นั้นเรียบร้อยแล้ว

TR_x : เป็นบิตควบคุมให้ตัวจับเวลา/ตัวนับเริ่มทำงานโดยเซตให้เป็น 1 และให้หยุดทำงานโดยการเคลียร์ให้เป็น 0 โดยซอฟต์แวร์

IE_x : เป็นแฟลกอินเตอร์รัพต์จากสัญญาณภายนอก เซตด้วยฮาร์ดแวร์เมื่อมีสัญญาณอินเตอร์รัพต์ ปรากฏที่ขา INT_x และจะเคลียร์ตัวเองโดยอัตโนมัติ เมื่อกระโดดไปทำงานบริการอินเตอร์รัพต์ที่ขอมาเรียบร้อยแล้ว

IT_x : เป็นบิตควบคุมรูปแบบสัญญาณอินเตอร์รัพต์ภายนอกจะเซต/เคลียร์ด้วยซอฟต์แวร์ โดยถ้าเซตเป็น 1 จะถูกอินเตอร์รัพต์ด้วยสัญญาณขอบขาลงและถ้าเคลียร์เป็น 0 จะถูกอินเตอร์รัพต์ด้วยสัญญาณแรงดันต่ำ

พอร์ตอนุกรม

ได้พร้อมกันเพราะมีบัฟเฟอร์ 2 ตัว ใช้ในการรับตัวหนึ่งและส่งอีกตัวหนึ่ง โดยโครงสร้างของรีจิสเตอ บัฟเฟอร์ทั้ง 2 ตัวจะแยกออกจากกัน แต่การติดต่อจะใช้ชื่อเดียวกันคือ SBUF พอร์ตอนุกรมของ MCS-51 สามารถที่จะโปรแกรมให้ทำงานได้แตกต่างกัน 4 โหมดดังนี้

โหมด 0 ข้อมูลจะเข้าและออกทางขา RXD โดยการเลื่อนสัญญาณนาฬิกาออกที่ขา TXD ข้อมูลจะเป็น 8 บิตโดยจะส่งที่มีนัยสำคัญต่ำก่อน โดยที่อัตราบ็อตจะคงที่ที่ 1/12 ของความถี่ออสซิลเลเตอร์

โหมด 1 เป็นการรับส่งข้อมูลขนาด 10 บิตโดยการส่งออกทางขา TXD และรับเข้าทางขา RXD รูปแบบบิตประกอบด้วย 1 บิตสตาร์เป็น "0", 8 บิตข้อมูล และ 1 สต็อปบิตเป็น "1" อัตราบ็อด(baud rate) แปรผันได้ตามการตั้งตัวจับเวลาตัวที่ 1โดยมีสูตรดังนี้

$$\text{อัตราบ็อด} = \frac{2^{SMOD}}{32} * \frac{\text{ความถี่ออสซิลเลเตอร์}}{12 * (256-TH1)}$$

โหมด 2 เป็นการรับส่งข้อมูลขนาด 11 บิต เข้าทางขา RXD และส่งออกทางขา TXD ประกอบด้วย 1 บิตสตาร์ตมีค่า "0",9 บิตข้อมูลและ 1 บิตสต็อปโดยการรับข้อมูลบิตที่ 9 จะถูกนำมาเก็บที่บิต RB8 ใน SCON ส่วนการส่งจะต้องใส่บิตที่ 9 ไว้ใน TB8ของ SCON ก่อนอัตราบ็อด สามารถเลือกได้ 2 อัตราคือ 1/32 หรือ 1/64 ของความถี่ออสซิล--เลเตอร์ขึ้นอยู่กับการเซตบิต SMOD ในรีจิสเตอร์ PCON ซึ่งรายละเอียดของรีจิสเตอร์ SCON แสดงในตารางดังนี้

ตารางที่ 2.1.3 แสดงบิตควบคุมที่อยู่ในรีจิสเตอร์ SCON

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0,SM1 : เป็นตัวกำหนดโหมดการใช้งานของพอร์ตอนุกรมดังนี้

SM0	SM1	โหมด
0	0	0
0	1	1
1	0	2
1	1	3

SM2 : ควบคุมอีนาเบิลการใช้โปรเซสเซอร์หลายตัวในการสื่อสารซึ่งกันและกัน ในโหมด 2 และ 3

REN : ตัวอีนาเบิลอนุกรมการรับเมื่อเซตเป็น 1 และถ้าเป็น 0 เป็นการดิสเอเบิล

TB8 : เป็นตัวเก็บข้อมูลบิตที่ 9 ที่จะส่งในโหมด 2 และ 3

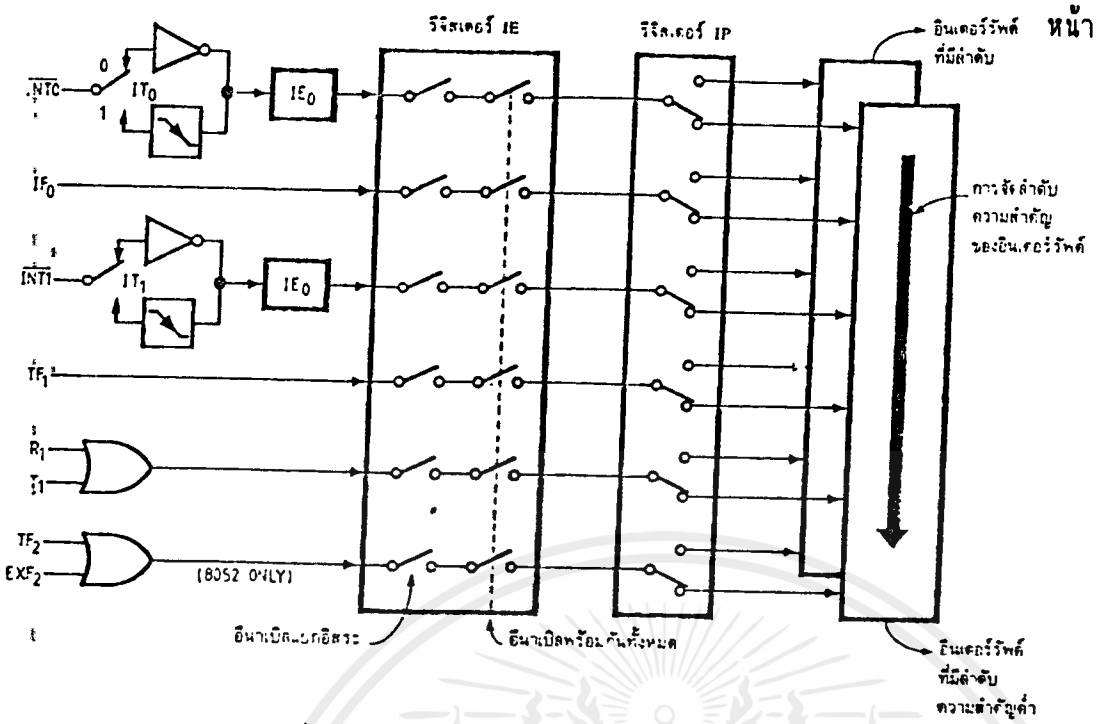
RB8 : เป็นตัวรับข้อมูลบิตที่ 9 ที่จะส่งในโหมด 2 และ 3 ส่วนในโหมด 1 จะทำหน้าที่เป็นสต็อปบิต

TI : เป็นแฟลกอินเตอร์รัพต์การเซตด้วยฮาร์ดแวร์ที่ปลายช่วงของบิตที่ 8 ในโหมด 0 หรือที่จุดเริ่มต้นของบิตสต็อปในโหมดอื่น ในการส่งแบบอนุกรมของทุกโหมดจะต้องเคลียร์บิตนี้ด้วยโปรแกรมหลังการส่ง

RI : เป็นแฟลกอินเตอร์รัพต์การรับ เซตด้วยฮาร์ดแวร์ที่ปลายช่วงของบิตที่ 8 ในโหมด 0 หรือจุดครึ่งของของช่วงบิตสต็อปในโหมดอื่น ในการรับแบบอนุกรมจะต้องเคลียร์บิตนี้ด้วย โปรแกรมหลังการรับทุกครั้ง

การอินเตอร์รัพต์

ในระบบของ MCS-51 จะอินเตอร์รัพต์ได้จาก 5 แหล่งโดยสามารถตั้งระดับความสำคัญได้ 2 ระดับคือ ระดับสูงและระดับต่ำ ซึ่งจะมีแผนภูมิทางฮาร์ดแวร์ของการอินเตอร์รัพต์แสดงไว้ในรูปที่ 2.1.7



รูปที่ 2.1.7 แสดงแผนภูมิทางฮาร์ดแวร์ของแหล่งอินเตอร์รัพต์ชนิดต่างๆ

การอินเตอร์รัพต์ความสำคัญต่ำ สามารถที่จะถูกอินเตอร์รัพต์จากอินเตอร์รัพต์ตัวอื่นที่ระดับความสำคัญสูงได้แต่ไม่สามารถถูก อินเตอร์รัพต์จากอินเตอร์รัพต์ตัวอื่นที่มีระดับเดียวกันได้ แต่ถ้าเป็นการอินเตอร์รัพต์จากตัวที่มีความสำคัญสูงจะไม่สามารถถูกอินเตอร์รัพต์จากตัวอื่นได้เลย โดยเราสามารถตั้งระดับความสำคัญของแต่ละแหล่งอินเตอร์รัพต์ได้ที่รีจิสเตอร์ IP และถ้าในเหตุการณ์ที่มีการร้องขออินเตอร์รัพต์ในระดับความสำคัญเดียวกันเข้ามาพร้อมกัน ก็จะต้องมีการจัดระดับความสำคัญของแต่ละแหล่ง ดังตารางที่ 2.4

ตารางที่ 2.1.4 แสดงลำดับความสำคัญของแหล่งการอินเตอร์รัพต์ต่างๆ

แหล่งการอินเตอร์รัพต์	ลำดับความสำคัญ
อินเตอร์รัพต์ 0 จากภายนอก	1
อินเตอร์รัพต์โอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับที่ 0	2
อินเตอร์รัพต์ 1 จากภายนอก	3
อินเตอร์รัพต์โอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับที่ 1	4
อินเตอร์รัพต์ของพอร์ตอนุกรม	5
อินเตอร์รัพต์โอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับที่ 2	6

การถูกขัดจังหวะให้มาทำงานบริการอินเตอร์รัพต์แต่ละตำแหน่งก็จะต้องมีตำแหน่งที่จะกระโดดไปทำงานที่แน่นอนซึ่งใน MCS-51 จำหนดแอดเดรสที่อยู่ของตำแหน่งที่จะกระโดด

ไปทำงานเมื่อมีการอินเตอร์รัพต์ ดังในตารางที่ 5 ส่วนรายละเอียดบิตควบคุมของรีจิสเตอร์ IE แสดงในรูปที่ 2.8

ตารางที่ 2.1.5 แสดงตำแหน่งที่อยู่ของการกระโดดแต่ละแหล่งอินเตอร์รัพต์

แหล่งการอินเตอร์รัพต์	ตำแหน่งที่อยู่ใน การกระโดดทำงาน
External Interrupt 0	0003H
Timer 0 Overflow	000BH
External Interrupt 1	0013H
Timer 1 Overflow	001BH
Serial Port Interrupt	0023H

EA	–	ET2	ES	ET1	EX1	ET0	EX0

รูปที่ 2.1.8 แสดงบิตควบคุมการอินเตอร์รัพต์ในรีจิสเตอร์ IE

EA : ถ้า EA = 0 จะดิสเอเบิ้ลการอินเตอร์รัพต์ทั้งหมด ถ้า EA = 1 จะสามารถอินเตอร์รัพต์ได้ โดยแต่ละแหล่งอินเตอร์รัพต์จะมีอิสระในการเซตและเคลียร์ให้อินาเบิลหรือดิสเอเบิ้ลได้

ET_x : จะเป็นบิตควบคุมการตอบรับอินเตอร์รัพต์โอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับ x โดยถ้าเซตเป็น 1 จะยอมรับแต่ถ้าเป็น 0 จะไม่ยอมรับ

ES : จะเป็นบิตควบคุมการยอมรับอินเตอร์รัพต์พอร์ตอนุกรม โดยที่ ES="1" จะยอมรับแต่ถ้า ES = "0" จะไม่ยอมรับ

EX_x : เป็นบิตควบคุมการยอมรับอินเตอร์รัพต์ภายนอกจากขา INT_x ถ้าเป็น "1" จะเป็นการยอมรับ แต่ถ้าเป็น "0" จะไม่ยอมรับ

ชุดคำสั่งของ MCS-51

ในส่วนนี้จะกล่าวถึงชุดคำสั่งของ MCS-51 ซึ่งถูกแบ่งเป็นลักษณะการทำงานตามฟังก์ชั่นได้ 5 กลุ่มคือ

- กลุ่มการถ่ายเทข้อมูล
- กลุ่มคณิตศาสตร์
- กลุ่มตรรกศาสตร์
- กลุ่มของบูลลีน
- กลุ่มของการกระโดด

<u>กลุ่มการถ่ายเท่ข้อมูล</u>

แบ่งออกเป็น 2 ชุดคำสั่งคือชุดคำสั่งถ่ายเทข้อมูลใน RAM ภายในและชุดคำสั่งถ่ายเทข้อมูลใน RAM ภายนอก และหน่วยความจำโปรแกรม โดยมีรายละเอียดดังนี้

การถ่ายเทข้อมูลใน RAM ภายใน

ชุดคำสั่งของการถ่ายเท RAM ภายในนั้น แสดงดังตารางที่ 6 ซึ่งเวลาที่ใช้ในคำสั่งจะเป็นเวลาเมื่อขณะใช้ความถี่ในการทำงานของ CPU ที่ความถี่ 12 MHz และรายละเอียดของแต่ละคำสั่งมีดังนี้

MOV จะทำงานในลักษณะการถ่ายเทข้อมูลเป็นขนาดไบต์หรือบิตก็ได้ จากแหล่งกำเนิดเข้าสู่การรับข้อมูลในนิลด์โอเปอร์แรนด์

PUSH จะทำงานโดยเพิ่มค่าให้รีจิสเตอร์ SP ก่อน แล้วจึงถ่ายข้อมูล 1ไบต์จากแหล่งกำเนิดที่นิลด์โอเปอร์แรนด์ที่กำหนดไว้ไปยังบริเวณสแต็กตามตำแหน่งที่ รีจิสเตอร์ SP กำหนด

POP การถ่ายเทข้อมูลขนาด 1ไบต์จากบริเวณสแต็กตามตำแหน่งที่รีจิสเตอร์ SP กำหนดไปยังรีจิสเตอร์ ที่โอเปอร์แรนด์กำหนด และหลังจากนั้นรีจิสเตอร์ SP จะลดค่าลง1

XCH คำสั่งแลกเปลี่ยนไบต์ระหว่างแหล่งกำเนิด โอเปอร์แรนด์กับรีจิสเตอร์ A

XCHD คำสั่งแลกเปลี่ยนขนาดนิบเบิลทางอันดับต่ำของแหล่งกำเนิดโอเปอร์แรนด์กับนิบเบิลอันดับต่ำของแอกคูมูเลเตอร์

ตาราง 2.1.6 แสดงชุดคำสั่งของการถ่ายเทข้อมูลใน RAM ภายใน

มีโมนิก	การกระทำ	โหมดการแอดเดรส				เวลาการเอ็กซีคิวต์ (µs)
		Dir	Ind	Reg	Imm	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>,<src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant.				X	2
PUSH <src>	INC SP : MOV "@SP",<src>	X				2
POP <dest>	MOV <dest>, "@SP" : DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

การถ่ายเทข้อมูลใน RAM ภายนอก และหน่วยความจำโปรแกรม

คำสั่งในการเคลื่อนย้ายข้อมูลใน RAM ภายนอกนั้นสามารถดูคำสั่งได้ในตารางที่ 7 ซึ่งจะมีตัวชี้อยู่ 2 แบบคือ ใช้รีจิสเตอร์ R0 และ R1 เป็นตัวชี้และอีกแบบหนึ่งคือ ใช้รีจิส-เตอร์ DPTR เป็นตัวชี้ ซึ่งเราใช้ R1 (R0 หรือ R1) เป็นตัวชี้นั้นจะชี้ได้เพียง 256 ไบต์ แต่ถ้าใช้ DPTR จะสามารถชี้ได้ถึง 64 กิโลไบต์ ส่วนชุดคำสั่งในการถ่ายเทข้อมูลในหน่วยความจำโปรแกรมนั้นจะแสดงอยู่ในตารางที่ 7

ตาราง 2.1.7 แสดงชุดคำสั่งในการถ่ายเทข้อมูลในหน่วยความจำโปรแกรม

มีโมนิก	การกระทำ	เวลาการเอ็กซีคิวต์ (µs)
MOVC A,@A+DPTR	Read Pgm Memory at (A+DPTR)	2
MOVC A,@A+PC	Read Pgm Memory at (A+PC)	2

<u>กลุ่มคำสั่งคณิตศาสตร์</u>

กลุ่มคำสั่งนี้แสดงในตารางที่ 8 ช่วงเวลาการทำงานแต่ละคำสั่งนั้นกำหนดที่ความถี่สัญญาณนาฬิกา 12 MHz คำสั่งทางคณิตศาสตร์ส่วนใหญ่ใช้เวลา 1 µs ยกเว้นคำสั่ง INC DPTR ซึ่งใช้เวลา 2 µs คำสั่งคูณและหารใช้เวลา 4 µs

ตารางที่ 2.1.8 แสดงชุดคำสั่งทางคณิตศาสตร์

นีโมนิก	การกระทำ	โหมดการแอดเดรส				เวลาการเอ็กซีคิวต์ (μs)
		Dir	Ind	Reg	Imm	
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A − <byte> − C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A − 1	Accumulator only				1
DEC <byte>	<byte> = <byte> − 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

รายละเอียดการทำงานของแต่ละคำสั่งมีดังนี้

INC เป็นการบวกหนึ่งเข้ากับแหล่งกำเนิดโอเปอร์แรนด์และใส่คำใหม่กลับเข้าสู่ตัวโอเปอร์แรนด์

DEC เป็นการลบหนึ่งออกจากตัวเลขที่อยู่ในแหล่งกำเนิดโอเปอร์แรนด์และนำผลลัพธ์กลับมาเก็บที่ตัวโอเปอร์แรนด์นั้น

ADD เป็นการบวกค่าในแอคคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์

ADC เป็นการบวกค่าในแอคคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์และบวกเข้ากับบิตตัวทดด้วย

SUBB เป็นการนำตัวเลขที่แหล่งกำเนินของโอเปอร์แรนด์ลบออกจากตัวเลขใน A และนำค่าบิตตัวที่ทดนำมาลบออกอีกและได้ผลลัพธ์ใส่ลงในแอกคิวมูเลเตอร์ A

MUL เป็นการคูณแบบไม่คิดเครื่องหมายของตัวเลขที่อยู่ในแอคคิวมูเลเตอร์กับเลขในรีจิสเตอร์ B แล้วได้ผลลัพธ์ 2 ไบต์ นำมาเก็บไว้ที่ AB โดย A จะรับอันดับต่ำส่วน B จะรับอันดับสูง

DIV เป็นคำสั่งการหารโดยไม่คิดเครื่องหมายที่อยู่ในแอคคิวมูเลเตอร์หารด้วยตัวเลขในรีจิสเตอร์ B แล้วนำผลลัพธ์ไปเก็บไว้ในแอคคิวมูเลเตอร์และเศษเหลือจะอยู่ในรีจิสเตอร์ B

DA ใช้สำหรับการบวกกันทางระบบตัวเลข BCDเป็นการปรับค่ารวมซึ่งเป็นผลลัพธ์จากการบวกกันทางไบนารี่ของระบบตัวเลข BCD ขนาด 2 หลัก สองจำนวนการปรับค่าตัวเลขผลรวมด้วยการใช้คำสั่ง DA จะได้ผลลัพธ์กลับมาที่แอคคิวมูเลเตอร์

กลุ่มคำสั่งทางตรรกศาสตร์

กลุ่มคำสั่งทางตรรกศาสตร์สามารถแสดงได้ดังในตารางที่ 9 ซึ่งคำสั่งเหล่านี้สามารถที่จะทำงานในลักษณะของบูลลีนได้ทั้งขนาดไบต์หรือบิต

ตารางที่ 2.1.9 แสดงชุดคำสั่งของกลุ่มตรรกศาสตร์

มีโมนิก	การกระทำ	โหมดการแอดเดรส				เวลาการเอ็กซีคิวต์ (μs)
		Dir	Ind	Reg	Imm	
ANL A,<byte>	A = A .AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND. A	X				1
ANL <byte>,#data	<byte> = <byte> .AND. #data	X				2
ORL A,<byte>	A = A .OR. <byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR. A	X				1
ORL <byte>,#data	<byte> = <byte> .OR. #data	X				2
XRL A,<byte>	A = A .XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR. A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR. #data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT. A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

รายละเอียดของแต่ละคำสั่งมีดังนี้

CPL เป็นคำสั่งการสั่งกลับค่าหรือทำคอมพลิเมนต์(Complement)ข้อมูลในแอคคิวมูเลเตอร์ โดยไม่ผลใดๆต่อค่าแฟลกใน PSW หรือการให้ตำแหน่งแอดเดรสตามบิตนั้นๆ

RL,RLC,RR,RRC,SWAP ทั้ง 5 คำสั่งนี้เป็นการสั่งทำงานตามการวนบิต บนตัวแอคคิวมูเลเตอร์ซึ่ง RL เป็นการวนซ้าย, RR เป็นวนขวา, RLC เป็นการวนซ้ายผ่านบิตทด ,RRC เป็นการวนขวาผ่านบิตทด และ SWAP เป็นการวนซ้ายสี่ครั้ง

ANL เป็นการ AND กันทางตรรก ระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ ซึ่งทำให้ทำงานตรรกข้อมูลขนาดเป็นไบต์หรือบิตก็ได้

ORL เป็นการ OR ทางตรรกระหว่างแหล่งกำเนินสองโอเปอร์แรนด์ ซึ่งจะสั่งให้ทำงานตรรกข้อมูลขนาดเป็นไบต์หรือบิตก็ได้

XRL เป็นการ XOR กันทางตรรก ระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ ซึ่งทำให้ทำงานตรรกข้อมูลขนาดเป็นไบต์หรือบิตก็ได้

กลุ่มคำสั่งแบบบูลีน

กลุ่มชุดคำสั่งเกี่ยวกับบูลลีนสามารถแสดงได้ในตารางที่ 10 การเข้าถึงข้อมูลโดยตรงในระดับบิต โดยมีการใช้บิตแอดเดรสได้ตั้งแต่ 00-7FH ในพื้นที่ 128 บิตหน่วยความจำข้อมูลภายในและบิตแอดเดรส 80-FFH ในบริเวณกลุ่มรีจิสเตอร์ ฟังก์ชั่นพิเศษ ขอให้ดูรูปที่ 3.3 และ 3.4 ประกอบ

ตารางที่ 2.10 แสดงชุดคำสั่งของบูลลีน

นีโมนิก		การกระทำ	เวลาการเอ็กซีคิวต์ (μs)
ANL	C,bit	C = C .AND. bit	2
ANL	C,/bit	C = C .AND. .NOT bit	2
ORL	C,bit	C = C .OR. bit	2
ORL	C,/bit	C = C .OR. NOT bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT. C	1
CPL	bit	bit = .NOT. bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

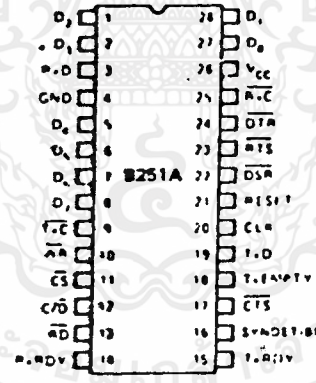
2.2 ลักษณะและวิธีการใช้งานไอซี 8251 USART

ไอซี 8251 USART (UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER /TRANSMITTER) เป็นพอร์ตที่ใช้ในการรับส่งข้อมูลแบบอนุกรม

สำหรับส่วนแรกที่จะกล่าวถึงก็คือ DATA BUS BUFFER ซึ่งส่วนนี้ 8251 จะใช้ในการเชื่อมต่อระหว่าง 8251 กับ พอร์ต1 ของ 8031 ส่วนต่อไปก็คือ READ/WRITE CONTROL LOGIC ซึ่งจะทำหน้าที่ในการควบคุมการรับส่งข้อมูลภายในของ 8251 ให้เป็นไปอย่างถูกต้อง

ส่วนที่จะกล่าวต่อไปก็คือ TRANSMIT BUFFER (P-S) และ TRANSMIT CONTROL (P-S;PARALLEL TO SERIAL CONVERSION) ซึ่งใช้ในการส่งและควบคุมการส่งข้อมูลไปตามสายส่ง

สำหรับส่วนสุดท้ายที่จะกล่าวถึงก็คือ RECEIVE BUFFER (S-P;SERIAL TO PARALLEL CONVERSION) ซึ่งทำหน้าที่ในการรับและควบคุมการรับข้อมูลของ 8251



รูปที่ 2.2.1 การจัดเรียงขาบน 8251

การจัดเรียงขา และหน้าที่

8251 เป็นไอซีขนาด 28 ขา ซึ่งแสดงไว้ในรูปที่ และเราสามารถจะแบ่งขาของ 8251 ออกเป็นกลุ่มๆ ได้ดังนี้คือ

1.กลุ่มที่ใช้ในการติดต่อกับ CPU

1.1) D0-D7: ใช้ในการติดต่อกับบัสข้อมูลของ CPU โดยตรง ซึ่งจะทำหน้าที่รับส่งข้อมูลและคำสั่งต่างๆ ระหว่าง 8251 กับ CPU

1.2) RESET : 8251 จะถูกรีเซ็ต เมื่อขานี้ได้รับลอจิก "1" ซึ่งเราอาจจะต่อมาจากขา RESET ของ 8031 โดยผ่าน INVERTER ก่อนก็ได้

1.3) CLK (CLOCK): ใช้ในการควบคุมช่วงเวลาการทำงานภายในของ 8251 สำหรับการใช้งานนั้นจะต่อเข้าโดยตรงกับระบบ อย่างไรก็ตามสัญญาณที่ขา CLK นี้ไม่เกี่ยวข้องกับอัตราการรับส่งข้อมูลหรือ BAUD RATE แต่อย่างใด

1.4) RD : เมื่อขานี้ได้รับลอจิก "0" 8251 จะทำการส่งข้อมูลแบบขนานออกมาที่ DATA BUS เพื่อส่งให้กับ CPU

1.5) WR : เมื่อขานี้ได้รับลอจิก "0" 8251 จะทำการรับข้อมูลแบบขนานจาก DATA BUS ของระบบ

1.6) C/D (CONTROL/DATA): ขา C/D นี้จะใช้ในการทำให้ 8251 ทราบว่า CPU ต้องการที่จะติดต่อกับ CONTROL REGISTER หรือ DATA REGISTER โดยที่ถ้าขานี้ได้รับลอจิก "1"ก็แสดงว่า CPU ต้องการที่จะติดต่อ CONTROL REGISTER แต่ถ้าได้รับลอจิก "0" ก็แสดงว่า CPU ต้องการที่จะติดต่อกับ DATA REGISTER

1.7) CS (CHIP SELECT) : ในกรณีที่ขานี้ได้รับลอจิก"0"ก็จะเป็นการ ENABLE 8251 โดยทั่วไปแล้วสัญญาณที่ขานี้จะได้มาจากการถอดรหัสพอร์ทแอดเดรส ดังที่ใช้กับ CHIP SUPPORT อื่นๆ

2.กลุ่มที่ใช้ในการส่งข้อมูล

2.1) TxD (TRANSMIT DATA OUTPUT): เป็นขาที่ใช้ในการส่งข้อมูลไปตามสายส่ง

2.2) TxC (TRANSMIT BAUD RATE CLOCK): ขานี้เป็นขาที่ใช้ในการส่งสัญญาณคล็อกที่ใช้ในการส่งข้อมูล ซึ่งก็คือความถี่ที่ใช้ในการกำหนด BAUD RATE นั่นเอง โดยปกติแล้วจะต้องช้ากว่าสัญญาณคล็อกของระบบไม่น้อยกว่า 30 เท่า

2.3) TxRDY: ขานี้จะใช้ในการทำให้ CPU ทราบว่า 8251 พร้อมที่จะรับข้อมูลจาก CPU เพื่อที่จะทำการส่งต่อไปแล้วหรือยัง และขานี้อาจจะนำไปใช้ในการขออินเทอร์รัมพ์ก็ได้

2.4) TxEMPTY : ขานี้จะใช้ในการแสดงว่าข้อมูลที่ CPU ส่งให้กับ 8251 นั้นได้ถูกส่งออกไปให้อุปกรณ์อื่น ๆ หมดแล้ว โดยที่ 8251 จะทำให้ขานี้เป็น "1" และเมื่อ CPU ทำการส่งข้อมูลชุดต่อไปให้กับ 8251 ขา TxEMPTY ก็จะเป็น "0" จนกว่า 8251 จะทำการส่งข้อมูลนี้ออกไปหมด 8251 ก็จะทำให้ขานี้กลับเป็น "0" อีกครั้ง

3.กลุ่มที่ใช้ในการรับส่งข้อมูล

3.1) RxD : ใช้ในการรับข้อมูลแบบอนุกรมจากสายส่ง

3.2) RxC : เป็นขาที่ใช้ในการรับสัญญาณคล็อกที่ใช้ในการรับข้อมูลโดยปกติแล้วจะทำการต่อเข้ากับ TxC โดยตรง

3.3) RxRDY : จะใช้ในการแสดงว่า 8251 พร้อมที่จะส่งข้อมูลให้กับ CPU และขานี้อาจจะใช้ในการขออินเทอร์รัพท์ได้เช่นเดียวกับขา TxRDY

3.4) SYNDET : ขานี้จะใช้ในการรับข้อมูลแบบ SYNCHRONOUS เท่านั้น (8251 สามารถที่จะทำการรับส่งข้อมูลได้ทั้งแบบ SYNCHRONOUS และแบบ ASYNCHRONOUS ซึ่งได้กล่าวถึงต่อไป) โดยที่เราสามารถที่จะโปรแกรมให้ขานี้เป็นอินพุทหรือเอาท์พุทก็ได้ โดยที่เมื่อขา SYNDET นี้ถูกโปรแกรมเป็นเอาท์พุทนั้นขา SYNDET จะให้ลอจิก "1" เมื่อ 8251 สามารถที่จะตรวจจับ SYNC CHARACTER ได้และจะให้ลอจิก "0" เมื่อ CPU ทำการอ่านรีจิสเตอร์สถานะ สำหรับขา SYNDET นี้จะให้ลอจิก "1" ในอีกกรณีหนึ่งคือ เมื่อ 8251 ได้รับข้อมูลจากสายส่งเป็น "0" หมดตั้งแต่ START BIT จนถึง STOP BIT

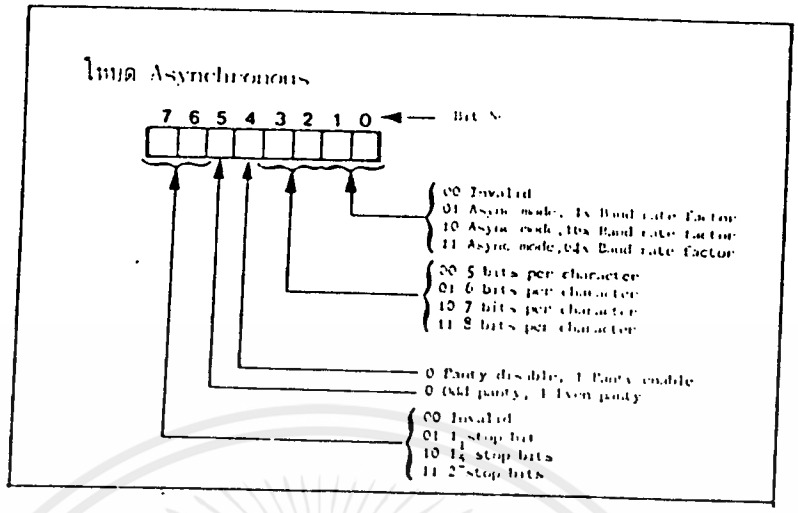
ในกรณีที่ขา SYNDET ถูกโปรแกรมให้เป็นอินพุทนั้น ถ้าขานี้ได้รับสัญญาณขอบขาขึ้น (สัญญาณเปลี่ยนจากลอจิก "0" เป็น "1") 8251 ก็จะถือว่าข้อมูลที่ขา RxD เป็นข้อมูลทันที และเราสามารถที่จะทำให้ลอจิกที่ขานี้กลับเป็น "0" ได้ในสัญญาณ RxC ลูกต่อไป

4.กลุ่มไฟเลี้ยงของ 8251

8251 ใช้ไฟเลี้ยงเพียงชุดเดียว คือ +5V กับ GND เท่านั้น ดังนั้นขาไฟเลี้ยงของ 8251 จึงมีเพียง 2 ขา คือ Vcc กับ GND

<u>การโปรแกรม 8251</u>

การรับส่งข้อมูลบน 8251 นั้น สามารถที่จะทำได้ 2 แบบ คือ SYNCHRONOUS และแบบ ASYNCHRONOUS แต่ในโครงงานนี้เราใช้ 8251 ในการติดต่อกับ IBM-PC แบบ ASYNCHRONOUS รูปที่ 2.2.2



รูป 2.2.2 การจัดเรียงบิทใน MODE WORD

จากรูป แสดงการจัดเรียงบิทใน CONTROL WORD (MODE WORD) ทั้งใน ASYNCHRONOUS และ SYNCHRONOUS MODE ซึ่ง MODE WORD นี้จะถูกส่งให้กับรีจิสเตอร์ควบคุมโดยการอ้างถึงพอร์ต-C ของ 8255 พร้อมกับการให้สัญญาณ WR กับ 8251 และ 8251จะถือว่าข้อมูลที่จะส่งให้กับรีจิสเตอร์ควบคุมเป็น MODE WORD ในกรณีที่ข้อมูลนี้เป็นข้อมูลไบท์แรกที่ถูกส่งให้กับ 8251 หลังจากที่ถูกรีเซ็ท และจะทำการโปรแกรมให้ทำงานใน ASYNCHRONOUS MODE (บิท"0"และ"1"จะกำหนดว่าให้ 8251ทำงานในโหมดใด ในกรณีที่บิททั้ง 2 นี้เป็นศูนย์ทั้งคู่เท่านั้น จึงจะเป็นการเลือกให้ 8251 ทำงานใน SYNCHRONOUS MODE) ดังนั้นจึงต้องทำการอ้างถึงพอร์ต-C ของ 8255 และกำหนดการทำงานของ 8251 ดังนี้

1. การกำหนดให้ใช้ค่า BAUD RATE = 4800 ในกรณีที่เราใช้ความถี่ของคล็อกที่ขา TxC และ RxC เท่ากับ BAUD RATE พอดี ก็เลือกโปรแกรมให้บิท 0 และ 1 เป็น "1" และ "0" ตามลำดับ แต่ในกรณีที่ความถี่ของคล็อกที่ใช้มีค่ามากกว่า BAUD RATE ที่ใช้ 16 หรือ 64 เท่า ก็จะต้องทำการโปรแกรมให้ BAUD RATE FACTOR เป็น 16* หรือ 64* ตามลำดับ

ในโครงงานนี้เราใช้ CRYSTAL 18 MHz นำมาหาร 6 เพราะฉะนั้นจึงต้องใช้ 64* ให้ได้ BAUD RATE 4800

2. กำหนดให้ข้อมูลที่จะส่งออกไปมีจำนวน 8 บิท (บิท 2 และ 1 เป็น"1") ซึ่งเราอาจจะโปรแกรมให้เป็น 5 , 6 หรือ 7 บิทก็ได้ ในกรณีที่ข้อมูลที่เราต้องการที่จะส่งออกไปมีน้อยกว่า 8 บิท 8251 จะทำการตัดบิทสูงทิ้งไป เช่น ถ้าเลือกให้ส่งเพียง 5 บิท 8251 ก็จะทำการตัดบิท D7-D5 ทิ้งไป

3. กำหนดให้ PARITY BIT เป็น PARITY คู่ (บิท 4 และ 5 เป็น "1")

4. เลือกใช้ STOP BIT จำนวน 1 บิท(บิท 6 เป็น "0" และ บิท 7 เป็น "1")

ในโหมด ASYNCHRONOUS นั้น หลังจากที่ได้ส่ง MODE WORD ให้กับ 8251 แล้ว ข้อมูลไบท์ต่อไปที่จะถูกส่งให้กับรีจิสเตอร์ควบคุมจะถือว่าเป็น COMMAND WORD ทั้งสิ้น จนกว่า 8251 จะได้รับการรีเซ็ทอีก จึงจะถือว่าข้อมูลที่ส่งมาให้กับรีจิสเตอร์ควบคุมนั้นเป็น MODE WORD

2.3 ระบบอินเตอร์เฟส RS-232 และ RS-422

การส่งข้อมูลแบบอนุกรมมาตรฐาน RS-232 เป็นที่นิยมกันมาก และเป็นแบบแรกที่ใช้สื่อสารของคอมพิวเตอร์ส่วนบุคคล จากคุณสมบัติของ RS-232 ที่ง่ายต่อการเข้าใจและกฎเกณฑ์ไม่มาก จึงทำให้มีการใช้อย่างแพร่หลาย แต่กระนั้น RS-232 ก็มีขีดจำกัดอยู่หลายอย่างในการสื่อสาร ซึ่งยังถือว่ายังเป็นระบบที่ยังไม่สมบูรณ์

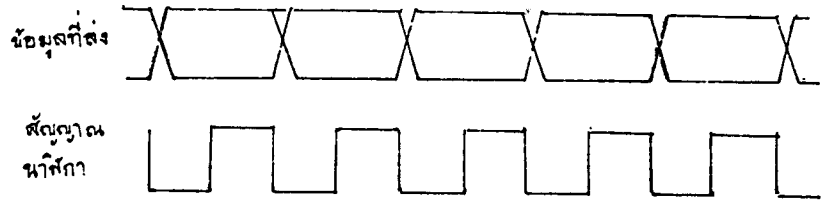
ยังมีมาตรฐานอีกมาตรฐานหนึ่ง ตามคุณสมบัติแล้วมีความสามารถเหนือกว่า RS-232 นั่นคือ มาตรฐาน RS-422 ซึ่งยังสามารถใช้งานในที่ที่มีสภาวะแวดล้อมที่เลวร้ายได้ เช่น ตามโรงงานอุตสาหกรรมต่าง ๆ แตกต่างจาก RS-232 ซึ่งจะอยู่ในที่แบบนี้ไม่ได้เลย และยังมีข้อดีตรงที่สามารถส่งข้อมูลได้ไกล และความเร็วในการส่งสูงกว่า RS-232

พื้นฐานการส่งข้อมูลแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรม สามารถแบ่งออกได้เป็น 2 อย่าง คือ แบบซิงโครนัส และแบบอะซิงโครนัส

การติดต่อสื่อสารแบบซิงโครนัส

ลักษณะการติดต่อแบบซิงโครนัส สัญญาณนาฬิกาจะเป็นตัวกำหนดของแต่ละข้อมูลที่จะส่งมา โดยตัวส่งจะเปลี่ยนแปลงข้อมูลที่ตำแหน่งขอบขาลงของสัญญาณนาฬิกา ส่วนตัวรับจะอ่านข้อมูลที่ตำแหน่งของขอบขาขึ้นของสัญญาณนาฬิกา ซึ่งข้อมูลจะคงสภาวะอยู่ไม่เปลี่ยนแปลง ลักษณะการรับส่งจะเป็นดังรูปที่ 1



รูป 2.3.1 แสดงถึงข้อมูลที่ส่งและข้อมูลที่รับซึ่งสัมพันธ์กับสัญญาณนาฬิกา

การติดต่อสื่อสารแบบซิงโครนัส จะได้เปรียบหรือมีข้อดีก็ตรงที่สามารถส่งข้อมูลได้เร็ว และมีความเที่ยงตรงสูงกว่าแบบอะซิงโครนัส อย่างไรก็ตาม การติดต่อสื่อสารแบบนี้ก็มีความยุ่งยาก และละเอียดอ่อนกว่าแบบอะซิงโครนัส การติดต่อสื่อสารแบบซิงโครนัสนี้ ส่วนใหญ่จะใช้สำหรับมินิคอมพิวเตอร์ และคอมพิวเตอร์ระบบเมนเฟรม เพราะระบบต่าง ๆ ต้องใช้ความเร็วในการติดต่อสื่อสารและต้องการความถูกต้องสูง

การติดต่อสื่อสารแบบอะซิงโครนัส

การติดต่อสื่อสารแบบนี้ เป็นการติดต่อที่ง่าย และมีความซับซ้อนน้อย นิยมใช้กับเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป การติดต่อแบบอะซิงโครนัสไม่จำเป็นต้องใช้สัญญาณนาฬิกาเข้ามาเกี่ยวข้องด้วยเลย

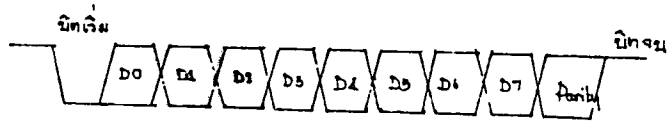
การติดต่อสื่อสารแบบอะซิงโครนัสนี้ ตัวรับและตัวส่งจะต้องกำหนดความเร็วในการรับและส่ง หรือ บ็อดเรต (baud rate = จำนวนบิตข้อมูลที่ส่งใน 1 วินาที) ให้เท่ากัน ข้อมูลจะถูกส่งทีละชุด คือจะต้องมีบิตเริ่ม(startbit)และบิตจบ (stop bit) เป็นตัวบ่งบอก ซึ่งบิตเริ่มจะมีสภาวะลอจิกเป็น"ต่ำ" (LOW) ตามด้วยบิตข้อมูล อาจเป็น 7 บิต หรือ 8 บิต ก็แล้วแต่ ต่อจากนั้นจะตามด้วยพาริตี้ บิต และปิดท้ายด้วยบิตจบ ซึ่งมีสภาวะลอจิกเป็น "สูง" (HIGH) มีความกว้างเท่ากับ 1, 1.5 หรือ 2 เท่าของบิตข้อมูล

จุดมุ่งหมายของการมีบิตจบ คือ

1. ใช้ตรวจสอบข้อมูลที่ได้รับว่าถูกต้องหรือไม่ ถ้าไม่มีการพบบิตจบหลังจากสิ้นสุดการรับข้อมูล แฟลกจะแสดงข้อผิดพลาดให้ทราบ

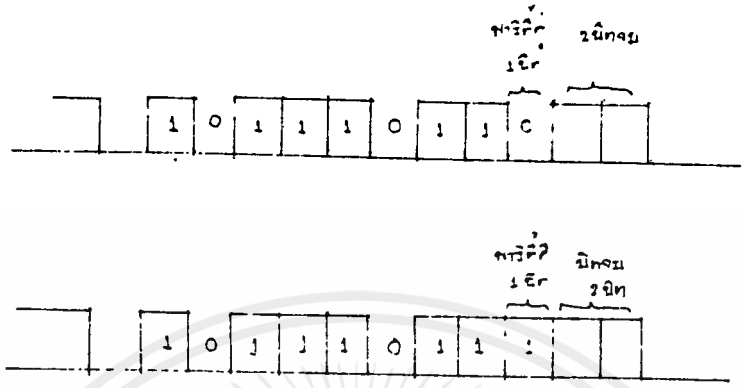
2. เพื่อให้ทราบได้ว่าข้อมูลชุดก่อนได้สิ้นสุด และพร้อมที่จะส่งข้อมูลชุดใหม่ได้

รูปที่ 2.3.2 เป็นรูปแสดงการส่งข้อมูลแบบอะซิงโครนัส จะเห็นว่า มีบิตเริ่ม บิตข้อมูล พาริตี้ บิต และบิตจบ



รูปที่ 2.3.2 ลักษณะของสัญญาณของการรับส่งข้อมูลแบบอะซิงโครนัส

ยกตัวอย่างการส่งข้อมูล 8 บิต มีข้อมูลเป็น "11011101" การส่งข้อมูลจะเป็นดังรูปที่ 2.3.3



รูปที่ 2.3.3 ตัวอย่างการส่งข้อมูล 8 บิต "11011101" แบบพาริตี้คู่ (ก) และพาริตี้คี่ (ข)

จากรูป จะเป็นการส่งข้อมูลแบบมีพาริตี้คู่ และแบบมีพาริตี้คี่ ซึ่งความสำคัญของการส่งข้อมูลแบบมีพาริตี้ คือ เพื่อใช้ตรวจสอบว่าข้อมูลที่ได้รับนั้นถูกต้องตามที่ส่งมาหรือไม่

การติดต่อสื่อสารแบบนี้มักใช้ชิพไอซีที่ทำหน้าที่เป็นตัวรับข้อมูลแบบอะซิงโครนัส โดยมีชื่อว่า UART (Universal Asynchronous Receiver/Transmitter) ซึ่งใช้ความถี่ของสัญญาณนาฬิกาเป็นตัวกำหนดอัตราการรับข้อมูล โดยมีค่าเท่ากับ 16 x ค่าบ็อดเรต เช่น มีค่าเป็น 153,600 Hz สำหรับบ็อดเรต 9600 บิตต่อวินาที โดยปกติ ถ้ายังไม่มีการส่งข้อมูลใด ๆ ในสายส่งสัญญาณจะเป็นลอจิก "สูง"อยู่

เมื่อมีการเริ่มส่งข้อมูล สภาวะของสายส่งสัญญาณจะเปลี่ยนเป็น"ต่ำ" แสดงถึงบิตเริ่ม ตัวรับ UART จะหน่วงเวลาโดยการนับสัญญาณนาฬิกาไป 8 ลูกคลื่น เพื่อเช็คดูอีกครั้งให้แน่ใจ (ที่จุดนี้จะเป็นจุดกึ่งกลางของบิต เพราะ 1 บิต จะกว้าง 16 ลูกคลื่นของสัญญาณนาฬิกา) ถ้าพบว่ามีสภาวะลอจิกเป็น "สูง" แสดงว่า เมื่อสักครู่เป็นสัญญาณรบกวนก็จะไม่สนใจ ถ้าพบว่าลอจิกยังคงเป็น "ต่ำ" มันก็จะเริ่มต้นรับข้อมูล โดยการนับสัญญาณนาฬิกาไปทีละ 16 ลูก แล้วอ่านข้อมูลเข้าไปเก็บในชิพรีจิสเตอร์ทีละบิต

หลังจากข้อมูลทั้งหมด และพาริตี้บิตถูกรับเข้ามาหมดแล้ว บิตต่อไปที่ตามมาก็ควรเป็นบิตจบ ถ้าเป็นลอจิก "ต่ำ" UART จะระบุว่าไม่สามารถรับข้อมูลได้อย่างถูกต้อง และแสดงข้อผิดพลาดขึ้นมา เพราะฉะนั้น บ็อดเรตที่เที่ยงตรงทั้งตัวส่งและตัวรับจึงจำเป็นมาก เพราะถ้าผิดพลาดแล้ว จะทำให้การส่งข้อมูลและการรับข้อมูลผิดพลาดทันที

เมื่อตัวรับอ่านข้อมูลได้ 8 หรือ 9 บิต (รวมกับพาริตี้บิต) ได้อย่างถูกต้อง บิตต่อมาก็จะเป็นบิตจบ ซึ่งมีลอจิก "สูง" มีช่วงเวลาการคงสภาวะเป็น 1 , 1.5 หรือ 2 ของช่วงเวลาแต่ละบิต หาก

ถูกต้องหมดทุกอย่าง UART จะทำการตรวจสอบข้อมูลที่รับมาว่า มีพาริตี้บิตถูกต้องหรือเปล่า โดยจะตรวจสอบว่าเป็นพาริตี้คู่ หรือพาริตี้คี่ ถ้าพาริตี้ที่ตรวจพบไม่ถูกต้องตามที่กำหนด แฟลกจะบอกข้อผิดพลาดอีกทันทีเช่นกัน

UART ส่วนมากสามารถรับข้อมูลเข้ามาทีละ 2 ชุด โดยชุดแรกเก็บไว้ในชิฟรีจิสเตอร์ และที่เหลือเก็บไว้ในโฮลดิ้งรีจิสเตอร์(holding register) ข้อมูลที่รับได้ จะเลื่อนจากชิฟรีจิสเตอร์ไปยังโฮลดิ้งรีจิสเตอร์ เพื่อรอให้โปรเซสเซอร์อ่านข้อมูลออกไป เมื่อเลื่อนข้อมูลไปแล้ว ชิฟรีจิสเตอร์ก็จะรับข้อมูลใหม่เข้ามาเพื่อเก็บและเตรียมส่งให้กับโฮลดิ้งรีจิสเตอร์ต่อไป

อย่างไรก็ตาม ถ้าโปรเซสเซอร์ไม่อ่านข้อมูลที่อยู่ในโฮลดิ้งรีจิสเตอร์ก่อนที่ข้อมูลตัวใหม่จะถูกส่งเข้ามา เข้ามูลเก่าก็จะถูกทับโดยข้อมูลตัวใหม่ ทำให้ข้อมูลไม่ครบถ้วน เกิดข้อผิดพลาดในการรับส่งข้อมูล

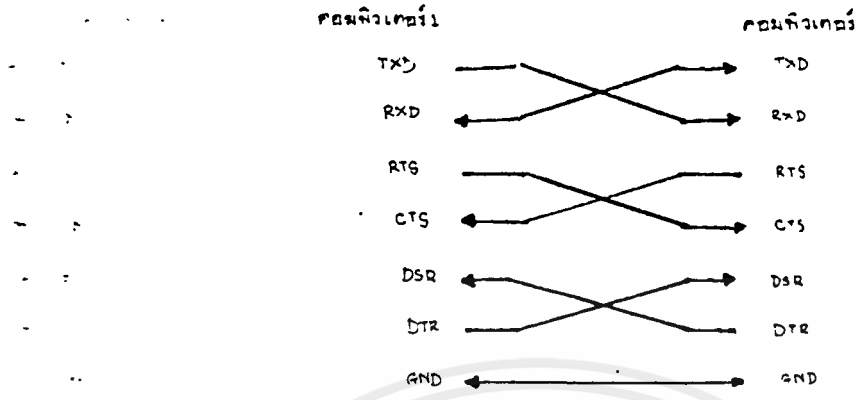
ในปัจจุบันได้มีการพัฒนา UART ที่ดีกว่าเดิม ซึ่งได้รับการออกแบบให้ลดปัญหาเหล่านี้ ชิพไอซีรุ่นใหม่นี้จะประกอบด้วย การเก็บข้อมูลแบบ FIFO (เข้าก่อน ออกก่อน) หมายความว่า จะเก็บข้อมูลเป็นชุด ชุดละ 16 ตัวอักษร พอเต็มจึงจะมีการทับของข้อมูล ช่วยให้โปรเซสเซอร์มีเวลาในการอ่านข้อมูลได้นานขึ้น

สำหรับวิธีอื่นที่จะป้องกันการเสียหายของข้อมูลมีดังนี้

1.ควบคุมอัตราเร็วการส่งข้อมูล คือ ตัวรับข้อมูล สามารถที่จะสั่งให้ตัวส่งหยุดการส่งข้อมูลชั่วคราว เมื่อรับข้อมูลไม่ทัน

2.ควบคุมทางฮาร์ดแวร์ ระบบนี้ควบคุมได้โดยใช้อุปกรณ์ทางอิเลกทรอนิกส์ โดยใช้สัญญาณที่มีอยู่ในพอร์ต RS-232 ซึ่งเป็นพอร์ตสื่อสาร เพื่อบอกความพร้อมของการรับข้อมูล ซึ่งมีชื่อเรียกของสายสัญญาณว่า RTS (Request To Send) และ CTS (Clear To Send) ถ้าเป็นการติดต่อระหว่างคอมพิวเตอร์ 2 เครื่อง RTS ของเครื่องที่ 1 จะต่อกับ CTS อีกเครื่องหนึ่ง ดูได้จากรูปที่ 2.3.4 ลักษณะการโต้ตอบแบบนี้ เรียกว่า hand shaking

เมื่อคอมพิวเตอร์ตัวรับพร้อมที่จะรับข้อมูลจะแจ้งสัญญาณไปที่สาย RTS คอมพิวเตอร์ตัวส่งจะตอบกลับที่สายสัญญาณ CTS นอกจากนี้ ยังมีวิธี hand shaking โดยใช้ซอฟท์แวร์ เรียกว่า XON/XOF hand shaking ซึ่งวิธีคล้ายกัน



รูปที่ 2.3.4 ลักษณะการเชื่อมต่อสัญญาณโต้ตอบเพื่อป้องกันข้อผิดพลาดในการรับส่งข้อมูล

<u>ระบบอินเตอร์เฟส RS-232</u>

ผู้ที่กำหนดมาตรฐาน RS-232 ขึ้นเมื่อกว่า 20 ปีมาแล้วก็คือ สถาบันแห่งหนึ่งในสหรัฐอเมริกา ซึ่งเป็นที่ยอมรับกับ มีชื่อว่า Electronics Industrials Association หรือเรียกสั้น ๆ ว่า EIA โดยได้กำหนดเป็นมาตรฐานสำหรับการเชื่อมต่อคอมพิวเตอร์ (การอินเตอร์เฟส) ซึ่งมาตรฐานนี้เป็นการส่งข้อมูลแบบอะซิงโครนัส มาตรฐานนี้เป็นที่ยอมรับของบริษัทผลิตคอมพิวเตอร์ส่วนใหญ่ และมีความนิยมใช้กันอย่างแพร่หลายในปัจจุบัน

ระบบ RS-232 ได้มีการปรับปรุงแก้ไขในปี 2529 เป็น EIA 232-D1986 (โดยทั่วไปเรียกว่า RS-232D) มาตรฐานนี้จะใช้ขั้วต่อขนาด 25 ขาแบบ D (D-subminiature connector) สำหรับการส่งข้อมูลระหว่างระบบ (RS-232C ไม่ได้เจาะจงชนิดของคอนเนกเตอร์ที่ใช้) ในระบบนี้สัญญาณจะตรงข้ามกับความเป็นจริง คือ ลอจิก "สูง" มีระดับแรงดัน -3 ถึง -25 โวลต์ แต่ส่วนใหญ่จะใช้ -12 โวลต์ และลอจิก "ต่ำ" มีระดับแรงดันตั้งแต่ +3 ถึง +25 โวลต์ ส่วนใหญ่ใช้ +12 โวลต์

การที่ได้กำหนด RS-232 เป็นมาตรฐานในการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ ทำให้มีการติดต่อสื่อสารข้อมูลกันอย่างแพร่หลาย ระบบ RS-232 นี้มีความเร็วในการส่งข้อมูล 20,000 บิตต่อวินาที (20 kbps) และระยะส่งไม่เกิน 50 ฟุต ตามคุณสมบัติ อย่างไรก็ตาม สามารถยอมให้ส่งไกลกว่านี้ได้ โดยจะต้องคำนึงถึงค่าอิมพีแดนซ์ในสาย (X_c) โดยต้องให้มีค่าน้อยที่สุด

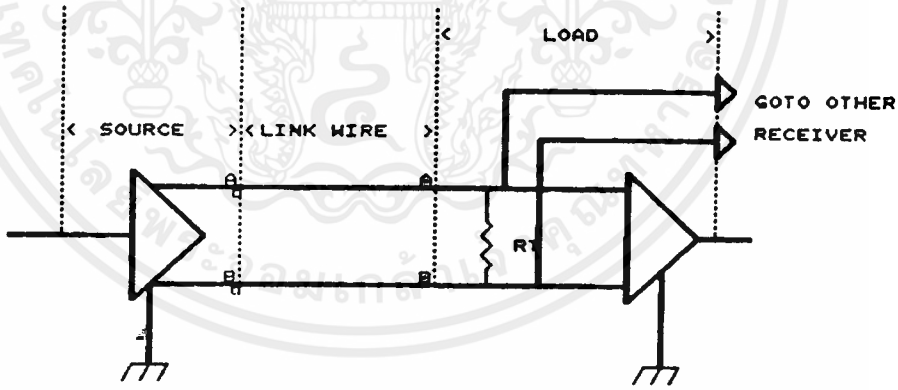
จะเห็นได้ว่า การส่งข้อมูลแบบ RS-232 ค่อนข้างมีข้อจำกัด คือ ส่งได้แค่ 20,000 บิตต่อวินาที ที่ 50 ฟุต ถ้าระยะส่งไกลกว่านี้ ความเร็วในการส่งจะลดลง และมีสัญญาณรบกวนมากขึ้น เนื่องจากสายส่งสัญญาณเป็นแบบอ้างอิงกับกราวด์ ไม่ใช่ระบบสมดุล (unbalance)

ระบบอินเตอร์เฟส RS-422

เมื่อไม่นานมานี้ EIA ก็ได้ออกแบบมาตรฐานชนิดใหม่ โดยพัฒนาจากของเดิม เนื่องมาใช้แทน RS-232 ในส่วนข้อจำกัดที่ RS-232 มี โดยได้ออกแบบให้ RS-422 มีความสามารถส่งข้อมูลได้ไกล ๆ คือ สามารถส่งได้ไกลถึง 4,000 ฟุต และความเร็วในการส่งข้อมูลสูงสุดถึง 10 ล้านบิตต่อวินาที (10 Mbps) (แบบไม่เป็นทางการ) ความสามารถสูงเช่นนี้ เพราะใช้การแยกสายสัญญาณเป็น 2 เส้น ต่อ 1 สัญญาณ โดยไม่ใช้กราวด์ร่วม เรียกว่า เป็นระบบสมดุล (balance) อีกทั้งใช้ระดับสัญญาณที่ +5 โวลต์ จึงไม่จำเป็นต้องใช้แหล่งจ่ายไฟเพิ่มเติมเหมือน RS-232

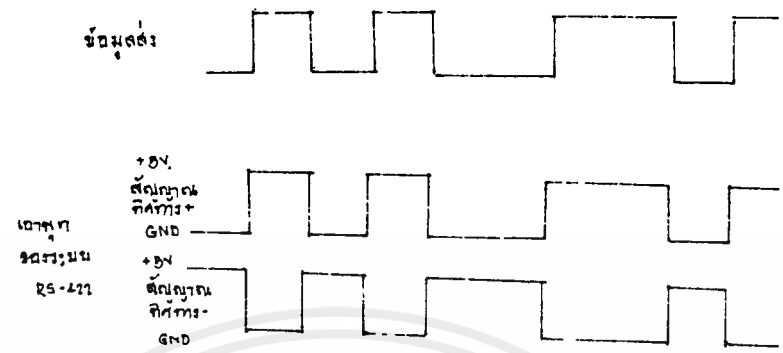
อย่างไรก็ตาม ที่ความเร็ว 90 kbps ยอมให้ส่งไกลเต็มที่ 4,000 ฟุต ความเร็ว 2 Mbps ที่ระยะทาง 200 ฟุต และความเร็ว 10 Mbps ที่ 35 ฟุต

ใน 1 เส้นสัญญาณ สายทั้ง 2 เส้นจะตีเกรียวไปด้วยกัน ทำให้มีอิมพีแดนซ์เท่ากันและกระแสไหลในสายจะเท่ากัน แต่ทิศทางตรงกันข้าม สัญญาณรบกวนต่าง ๆ จะหักล้างกันหมด ส่วนสายกราวด์นั้น จะใช้เพียงการอ้างอิงระดับแรงดันเท่านั้น โดยไม่ได้เป็นทางผ่านของสัญญาณแต่อย่างใด ลักษณะการเชื่อมต่อแบบนี้ แสดงในรูปที่ 2.3.5



รูปที่ 2.3.5 การเชื่อมสัญญาณแบบสมดุลที่ใช้ในระบบ RS-422

ในการส่งสัญญาณแบบนี้นั้น ขณะที่เส้นหนึ่งมีสัญญาณ +5 โวลต์ อีกเส้นที่คู่กันจะเป็น 0 โวลต์ และในทางตรงข้าม เมื่อสายหนึ่งเป็น 0 โวลต์ อีกสายจะเป็น +5 โวลต์ ดังนั้น การสวิงแรงดันที่ปรากฏระหว่าง 2 สาย จะมีค่าเป็น 10 โวลต์ ดังแสดงในรูป 2.3.6



รูปที่ 2.3.6 เปรียบเทียบข้อมูลที่ส่งกับสัญญาณเอาท์พุทของระบบ RS-422

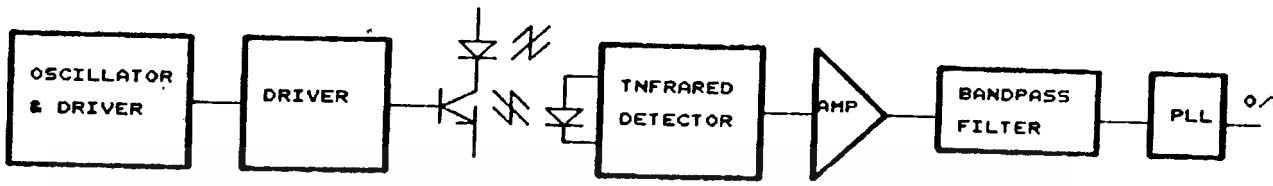
สัญญาณ RS-422 จะแบ่งเป็น 1 คู่สาย ต่อ 1 สัญญาณ เพื่อลดค่าอิมพีแดนซ์ในสาย ทำให้ผลของความไวต่อสัญญาณรบกวนมีน้อยมาก ถึงแม้ RS-422 จะมีคุณสมบัติที่ดีกว่า RS-232 แต่ก็ยังมีความนิยมน้อย ปัจจุบันกำลังเป็นที่นิยมใช้กันในโรงงานอุตสาหกรรมที่เป็นแบบอัตโนมัติ และต้องการความเชื่อถือสูง

2.4 อุปกรณ์ตรวจจับ (SENSOR)

ในปัจจุบันมีการพัฒนาตัวตรวจจับขึ้นมามากมายหลายชนิด ส่วนใหญ่จะมีลักษณะเป็นโมดูลใช้งานง่าย แต่มักจะมีราคาแพง สำหรับโครงงานนี้มีตัวตรวจจับที่ทำขึ้นดังต่อไปนี้

1 สวิทช์ลำแสงอินฟราเรด (Infrared Switch)

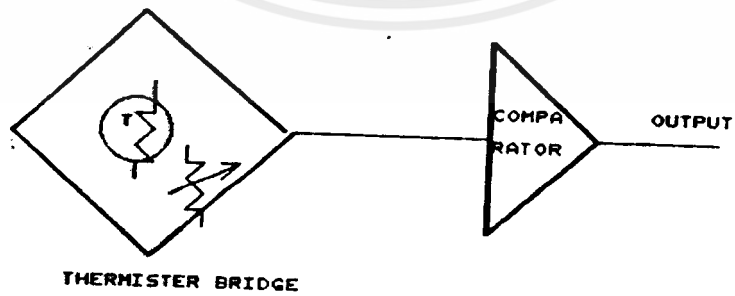
ใช้หลักการส่งลำแสงอินฟราเรดระหว่างตัวส่งและตัวรับ หากมีวัตถุมาขวางลำแสงก็จะตรวจจับได้ จากวงจรในรูป *.* ลำแสงอินฟราเรดจาก Infrared LED ถูกขับด้วยความถี่ประมาณ 100 kHz จากวงจร Astable Multivibrator (IC 555) ภาครับ จะรับได้จากวงจรขยาย ซึ่งดีเทคแสงอินฟราเรด โดย Infrared Diode แล้วเข้าวงจรกรองแถบความถี่แบบแอคทีฟ (Active Bandpass Filter) แล้วเข้าวงจรสวิทช์ต่อไป



รูปแบบ 2.4.1

2 ตัวตรวจจับความร้อน (Heat Detector)

ใช้ตัวเทอมิสเตอร์ (Thermister) ซึ่งเป็นตัวต้านทานที่แปรค่าตามอุณหภูมิ สำหรับที่ใช้ในวงจรนี้เทอร์มิสเตอร์เป็นแบบที่ความต้านทานจะลดลงเมื่ออุณหภูมิสูงขึ้น จากรูป *.* เทอร์มิสเตอร์ต่อเป็นวงจรบริดจ์ ที่อุณหภูมิ 60 องศาเซลเซียสปรับตัวต้านทานแปรค่าได้ให้วงจรบริดจ์มีศักดาประมาณ 0 โวลท์ (Balance Bridg) ที่อุณหภูมิห้อง (ประมาณ 25 องศาเซลเซียส) เทอร์มิสเตอร์ จะมีความต้านทานมากขึ้น ทำให้วงจรบริดจ์ มีศักดาตกคร่อมเป็นบวก (Unbalance Bridg) ซึ่งเป็นภาวะปกติ ลักษณะเช่นนี้ เมื่ออุณหภูมิสูงถึงขีดกำหนด วงจรคอมพาราเตอร์ จะให้เอาท์พุทออกมาเป็น 1

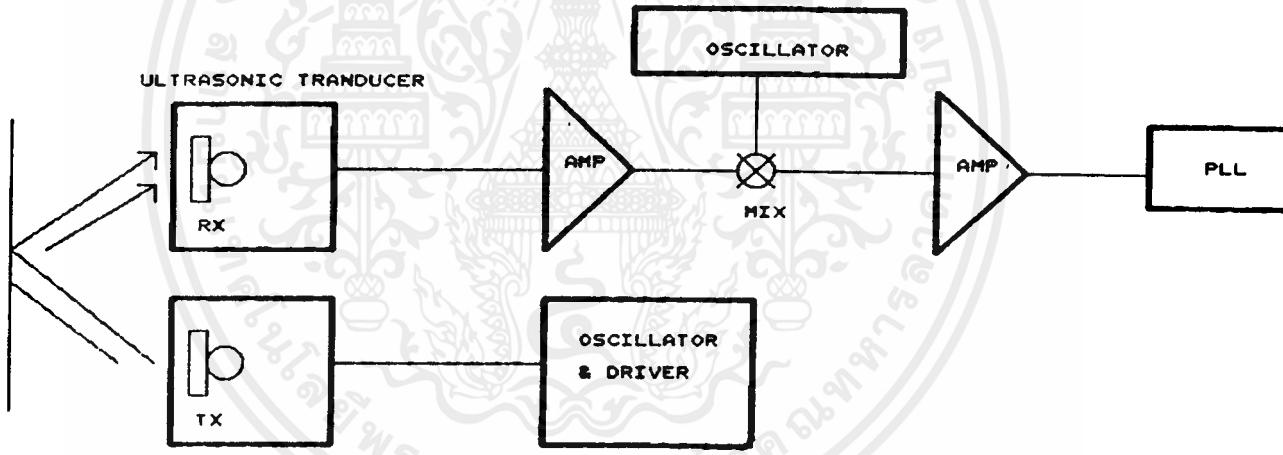


รูปที่ 2.4.2

3 ตัวตรวจจับคลื่นอุลตร้าโซนิค (Ultrasonic Sensor)

วงจรนี้ใช้ตรวจจับความเคลื่อนไหวในบริเวณที่กำหนด ที่สภาวะปกติไม่มีการเคลื่อนไหวใดๆ ใช้หลักการส่งคลื่นอุนตร้าโซนิคสะท้อนไปมาในห้องโดยที่ตัวส่งและรับจะอยู่ใกล้กัน ตัวรับจะนำคลื่นที่รับเข้ามา มาทำการบีทกับความถี่ที่กำเหนิดจากออสซิลเลเตอร์ภายในวงจร หากมีการเคลื่อนไหวจะเกิดปรากฏการณ์ดรอปเปอร์ (Dropper) ทำให้ความถี่บีทที่ได้เปลี่ยนไป ทำให้สามารถตรวจจับได้

จากรูป *.* วงจรภาคส่งใช้วงจรอะเสตเบิ้ลมันติไวเบรเตอร์ (Astablemultivibrator) เป็นตัวขับอุนตร้าโซนิคทรานซ์ดิวเซอร์ (Ultrasonictransducer) ความถี่ 40 kHz สำหรับวงจรตั้งรับ ตัวรับคลื่นอุนตร้าโซนิครับความถี่ที่ส่งเข้ามาแล้วเข้าวงจรขยายสัญญาน จากนั้นนำมาบีทกับวงจรออสซิเลเตอร์



รูปแบบ 2.4.3

บทที่ 3
การคำนวณและการสร้าง

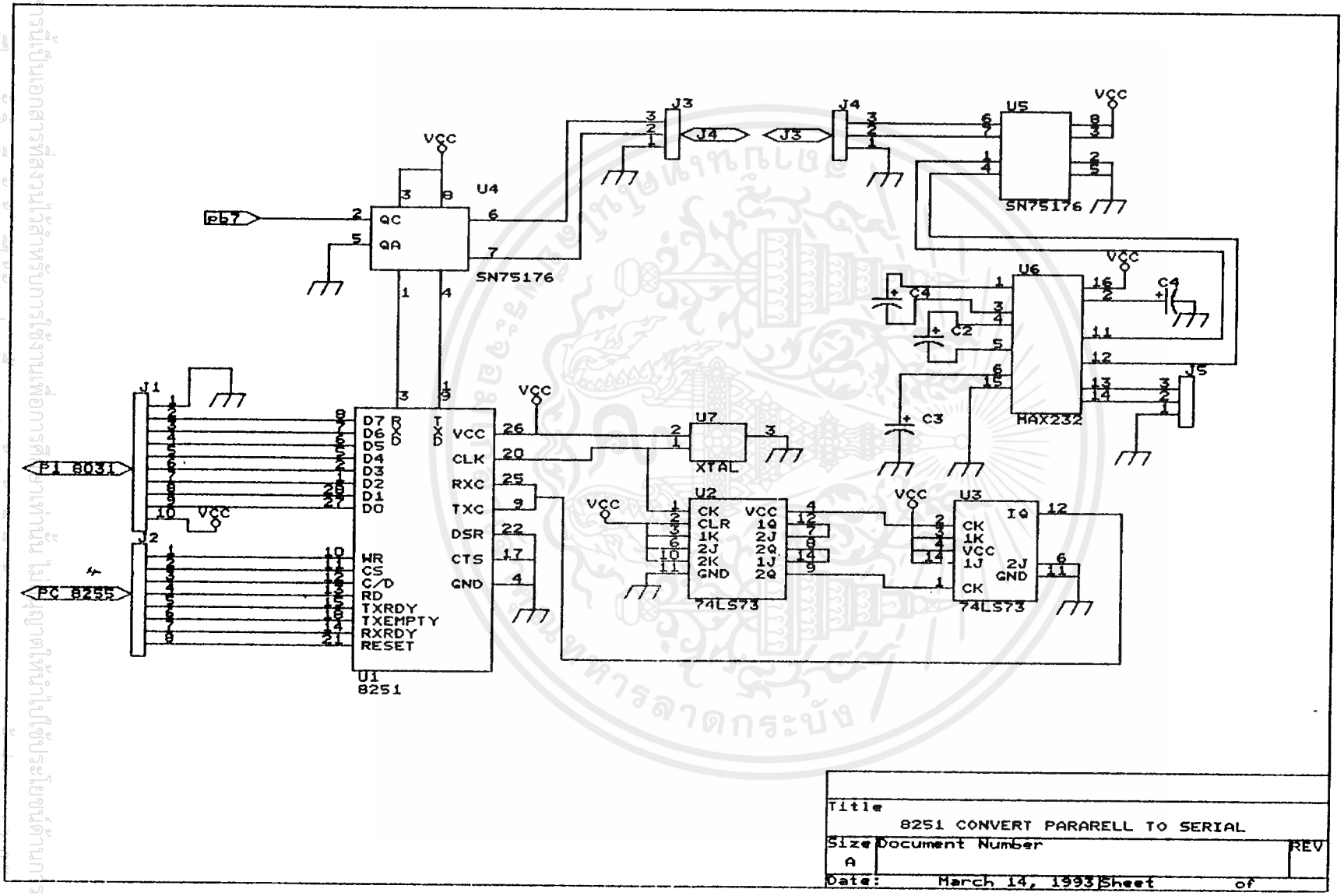
3.1 การเชื่อมต่อระหว่าง USART 8251 กับ บอร์ด controller 8031

เนื่องจาก บอร์ด controller 8031 มี serial port เพียงพอร์ตเดียว ซึ่งบอร์ด controller ระดับชั้น และระดับห้อง ใช้ในการติดต่อกัน ทำให้เราต้องแปลงพอร์ตขนาน(PORT1) ของบอร์ด controller ระดับชั้น เป็น serial port เพื่อทำการติดต่อกับ IBM PC ในการแปลงพอร์ตขนานนี้ เราใช้ USART 8251 การเชื่อมต่อระหว่าง 8251 กับ 8031 เราใช้ port 1 ของขนานสำหรับตัวรับ D0 - D7 ของ 8251 และใช้ port C ของ 8255 ต่อกับขา RD, WR, TxRDY, RxRDY, CS, C/D เพื่อ control 8255 ดังรูป 3.1.1

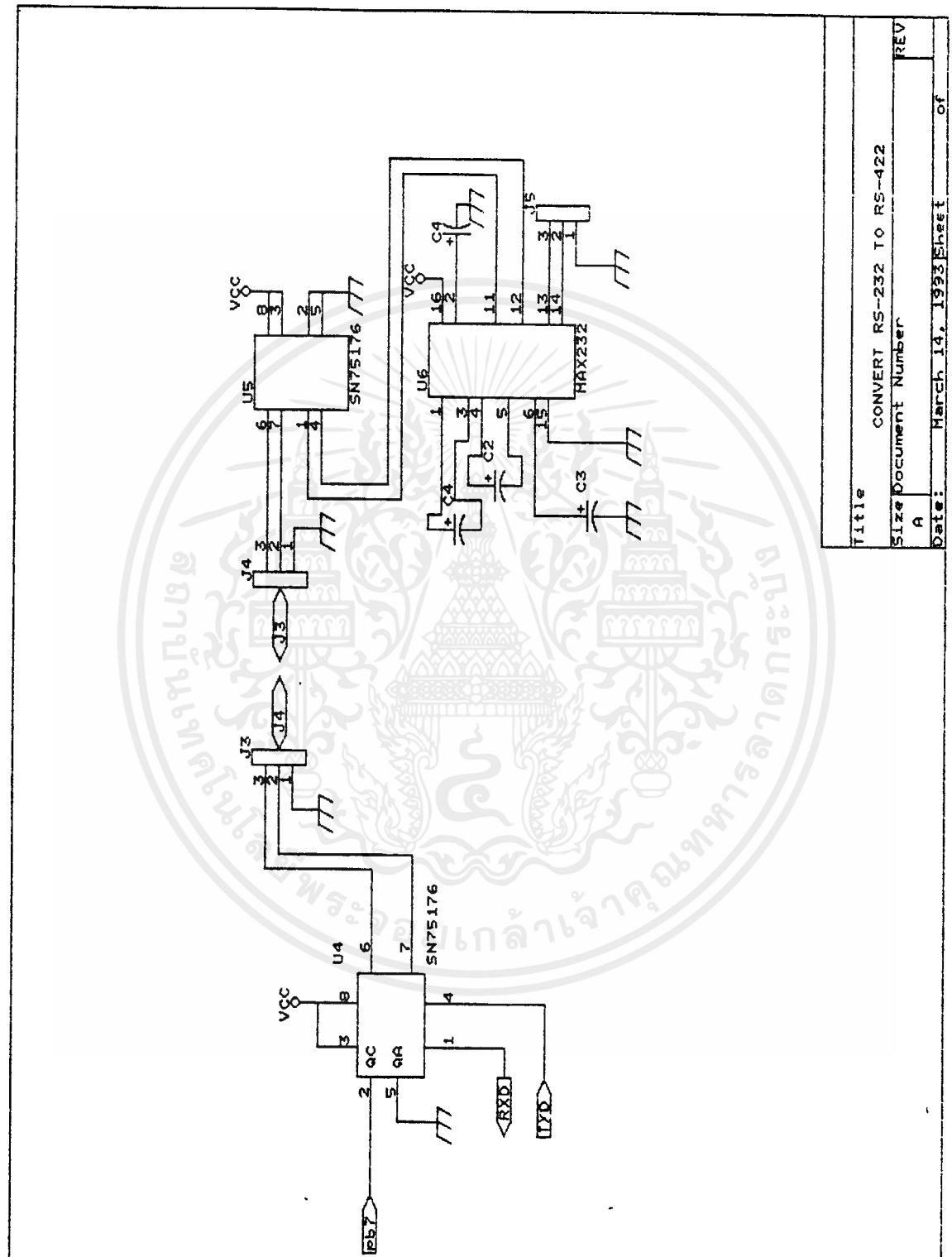
3.2 การแปลง RS-232 เป็น RS- 422

ตามระดับสัญญาณที่ใช้ในวงจร (+5V) นั้นไม่สามารถที่จะส่งไปได้ไกลนัก ดังนั้น ก่อนที่จะส่งข้อมูลไปในสายส่งจะต้องทำการเปลี่ยนระดับแรงดันของสัญญาณใหม่ เพื่อที่สามารถจะส่งสัญญาณได้ไกลขึ้น สำหรับมาตราฐาน RS-232 นี้จะใช้ระดับแรงดันในสายส่งประมาณ $\pm$12V และในการส่งระหว่างตัวประมวลผลสื่อสาร (บอร์ด 8031 ที่ประจำที่ห้อง) กับ IBM PC จะแปลง RS-232 เป็น RS-422

การส่งสัญญาณข้อมูลอนุกรมจาก IBM PC ใช้มาตรฐาน RS-232 แต่ระบบในโครงงานนี้ต้องการ การส่งระยะไกล ๆ และสัญญาณรบกวนต่ำ จึงใช้มาตรฐาน RS-422 ซึ่งเป็นการส่งแบบดิฟเฟอเรนเชียล จากวงจรจะใช้ไอซี MAX232 ในการเปลี่ยนสัญญาณ RS-232 เป็น สัญญาณ TTL และรับสัญญาณ TTL เปลี่ยนเป็นสัญญาณ RS-232 สัญญาณ TTL จะถูกเปลี่ยนเป็นสัญญาณ RS-422 โดยไอซี 74SN176 ดังรูป 3.2.1



รูปที่ 3.1.1 แสดงการเชื่อมต่อ 8251 กับ 8031



รูปที่ 3.2.1 การแปลง RS-232 เป็น RS-422

3.3 ส่วนการแสดงผลของ board 8031 และการเชื่อมต่อ sensor

บอร์ด microcontroller8031จะแสดงผลโดยใช้7-segment 4 หลัก ในสภาวะปกติจะแสดงผลเป็นนาฬิกา แต่เมื่อ sensor ผิดปกติ จะแสดงเบอร์ของ sensor ที่ผิดปกติ ในกรณีที่ sensor ตัวแรกผิดปกติ จะแสดงผล sensor ตัวแรก ต่อมา sensor ตัวที่ 2 จะติด โดยที่ sensor ตัวแรกยังไม่ถูก reset จะแสดงผล sensor ตัวที่ 2 และถ้า sensor ตัวที่ 2 ถูก reset ก่อน ก็จะกลับไปแสดงผล sensor ตัวแรกเช่นเดิม

การเชื่อมต่อ sensor ในบอร์ด microcontroller 8031 ระดับห้อง จะเชื่อมต่อ sensor ได้ทั้งหมด 16 ตัว และควบคุมกลับได้ 4 จุด ส่วนบอร์ด microcontroller ระดับชั้น จะติดตัวควบคุมได้ 8 จุด การแสดงผลตัวควบคุมจะแสดงผลด้วย LED

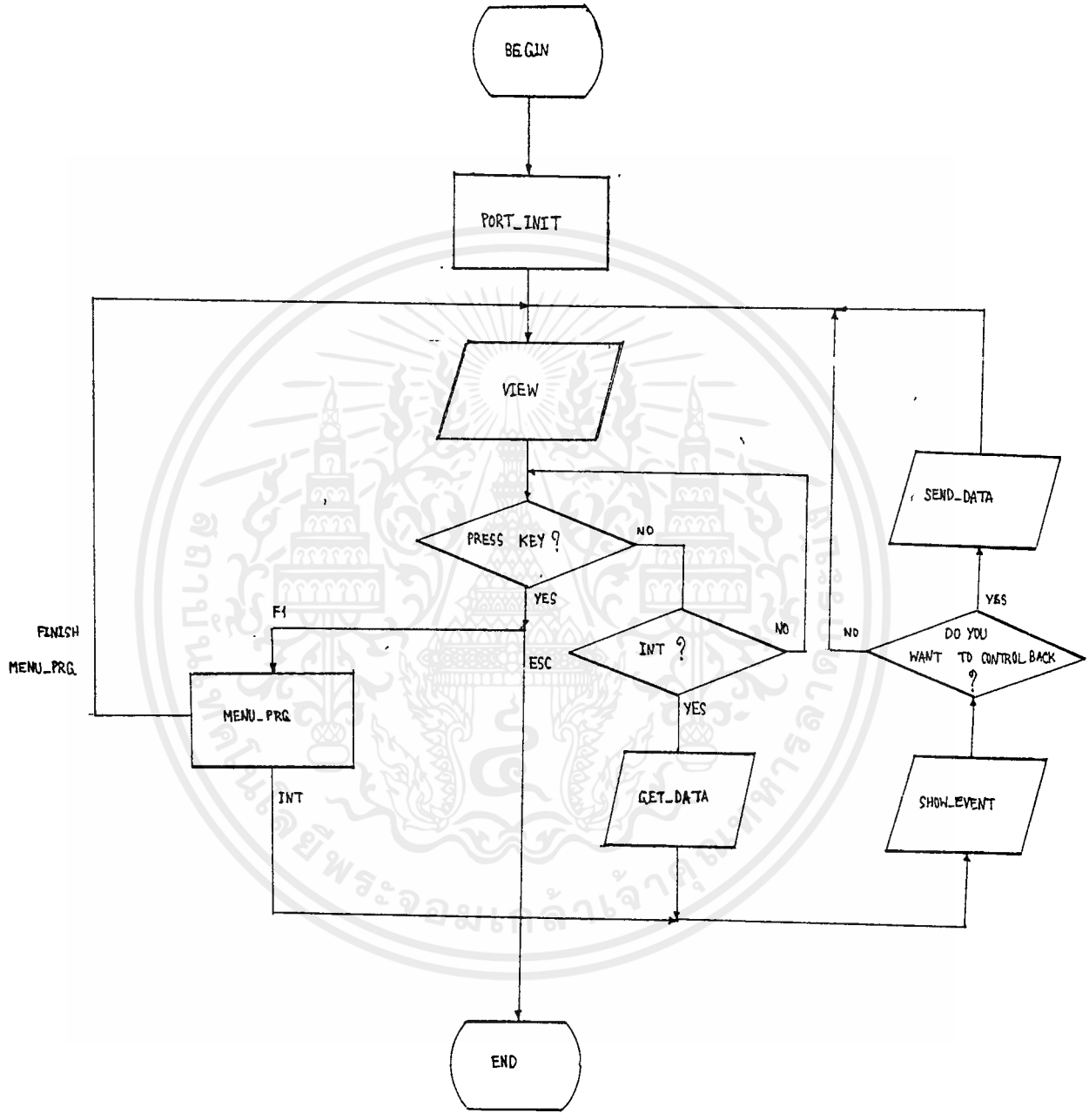
3.4 การทำงานของโปรแกรมใน IBM PC

หน้าที่หลัก

- แสดงผลเหตุการณ์ผิดปกติต่าง ๆ ที่เกิดขึ้น และสามารถควบคุมกลับได้ ทั้งเมื่อเหตุการณ์ปกติหรือ เกิดเหตุการณ์ผิดปกติขึ้น
- สามารถตั้งค่าหรือกำหนดค่าตัวเลือกต่าง ๆ เพื่อกำหนดลักษณะการทำงานได้

ขั้นตอนการทำงานของโปรแกรม

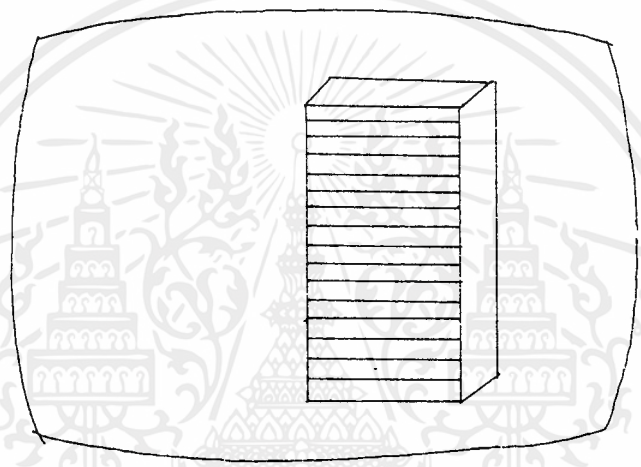
สามารถแสดงได้ดัง flowchart หน้าถัดไป



อธิบายการทำงานเป็นขั้น ๆ ได้ดังนี้ คือ

ขั้นแรก เมื่อเริ่มการทำงานของโปรแกรม จะต้องมีการตั้งสถานะของ port สื่อสารอนุกรมก่อน โดยการใช้ฟังก์ชัน port_init() เป็นการกำหนดรูปแบบของการสื่อสารแบบ อะซิงโครนัส ซึ่งในโปรแกรมนี้ กำหนดค่า baud rate = 4800,มี 1 start bit,มี 1 stop และมี bit 8 bit ข้อมูล

ขั้นตอนที่ 2 แสดงรูปตึก หรืออาคารที่จะรักษาความปลอดภัย โดยใช้ฟังก์ชัน view() ซึ่งในโปรแกรมนี้ ได้กำหนดให้เป็น อาคารที่พักอาศัยสูง 20 ชั้น ดังรูป



รูปที่ 3.4.2 แสดงผลหน้าจอแรกของโปรแกรม

ขั้นตอนที่ 3 ทำการตรวจสอบการกดคีย์

ถ้ากดคีย์ 'ESC' จะออกจากโปรแกรม

ถ้ากดคีย์ 'F1' จะเข้าไปทำงานโปรแกรมพิเศษ ที่ใช้ในการกำหนดค่าต่าง ๆ (จะอธิบายในรายละเอียดต่อไป)

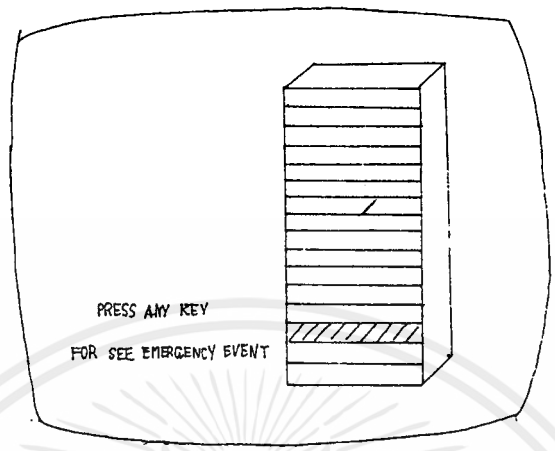
ขั้นตอนที่ 4 ตรวจสอบการส่งอินเตอร์รัพท์ โดยการเช็คสถานะของ portอนุกรม ใช้ฟังก์ชัน check_stat() เพื่อเช็คสถานะเป็นจำนวน 10 ครั้ง ถ้าครบ 10 ครั้งแล้วยังไม่มีการส่งข้อมูลใด ๆ มา ก็จะเลิกการเช็คสถานะ กลับไปทำการตรวจสอบการกดคีย์อีกครั้ง

หากมีการส่งข้อมูลมา ก็จะทำการอ่านข้อมูลโดยใช้ฟังก์ชัน rport() ถ้าค่าที่อ่านได้คือค่า FF(ค่าของการส่งอินเตอร์รัพท์) ก็จะไปทำงานขั้นตอนต่อไป

ขั้นตอนที่ 5 รับข้อมูลที่ส่งมาโดยใช้ฟังก์ชัน get_data() ข้อมูลที่รับได้แก่ หมายเลขชั้น, หมายเลขห้อง และหมายเลขของเหตุการณ์ที่เกิด

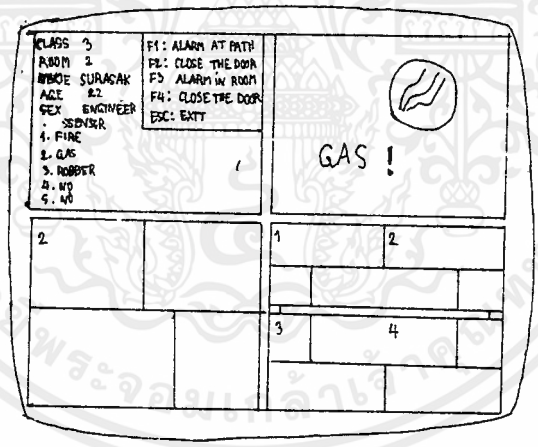
ขั้นตอนที่ 6 ใช้ฟังก์ชัน file_name() เพื่อหาชื่อของไฟล์ ซึ่งเป็นข้อมูลของแต่ละห้อง เช่น ถ้าเป็นชั้นที่ 10 ห้องที่ 2 ชื่อของไฟล์ที่จะได้คือ 10_10.dat ข้อมูลที่เก็บในไฟล์นี้ เช่น ชื่อผู้พักอาศัย,อายุ,อาชีพ และข้อมูลของอุปกรณ์ตรวจจับ(sensor) ณ.จุดต่าง ๆ

ขั้นตอนที่ 7 แสดงสัญญาณเตือนและกระพริบ ณ.ชั้นที่เกิดเหตุการณ์ โดยใช้ฟังก์ชัน alarm() ซึ่งจะได้เป็นดังรูป



รูปที่ 3.4.3 แสดงผลการกระพริบเมื่อเกิดเหตุการณ์

ขั้นตอนที่ 8 เมื่อกดคีย์ใด ๆ ฟังก์ชัน show_event() จะแสดงว่าเกิดเหตุการณ์อะไรที่ไหน และอ่านข้อมูลของห้องนั้น ๆ พร้อมทั้งมีการกระพริบ ณ.ห้องที่เกิดเหตุการณ์นั้นด้วย ดังรูป



รูปที่ 3.4.4 แสดงผลการเกิดเหตุการณ์อย่างละเอียด

ขั้นตอนที่ 9 ตรวจสอบการกดคีย์ และการส่งอินเตอร์รัพท์

ถ้ากดคีย์ 'F1' เป็นการควบคุมกลับเพื่อเปิดสัญญาณเตือนภัยที่ทางเดินของชั้น

ถ้ากดคีย์ 'F2' เป็นการควบคุมกลับเพื่อปิดประตูทางขึ้นลงของชั้น

ถ้ากดคีย์ 'F3' เป็นการควบคุมกลับเพื่อเปิดสัญญาณเตือนภัย ภายในห้องที่เกิดเหตุการณ์

ถ้ากดคีย์ 'F4 เป็นการควบคุมกลับเพื่อล็อคประตูทางเข้าออกของห้องที่เกิดเหตุการณ์

ถ้ากดคีย์ 'ESC' จะออกจากการแสดงผลนี้ และกลับไปแสดงรูปที่ 1 อีกครั้ง

ถ้าไม่มีการกดคีย์ใด ๆ โปรแกรมจะทำการแสดงผลเหตุการณ์ที่เกิดขึ้นต่อไป

หากมีการส่งอินเตอร์รัพท์ (การส่งข้อมูลค่า FF มา) ก็จะทำการแสดงผลเหตุการณ์ที่เกิด

ใหม่นี้ ดังรูปที่ 2 ทันที

และเมื่อต้องการกลับไปดูเหตุการณ์เก่าอีกครั้ง ก็ให้กดคีย์ 'F3'

การทำงานของฟังก์ชันสำคัญ ๆ บางฟังก์ชัน

ฟังก์ชัน port_init()

เป็นฟังก์ชันที่ใช้ในการกำหนดรูปแบบของการส่งข้อมูลอนุกรมแบบอะซิงโครนัส โดยใช้อินเตอร์รัพท์หมายเลข 14 ของ BIOS ฟังก์ชันหมายเลข 0 โดยมีรีจิสเตอร์ AH เป็นตัวผ่านค่าหมายเลขฟังก์ชัน(ในที่นี้คือ 0) รีจิสเตอร์ AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ตอนุกรม โดยมีความหมายของแต่ละบิตดังตาราง

หมายเลขประจำบิต	ความหมาย
7,6,5	อัตรารับส่งข้อมูล 000 = 110 baud 001 = 150 baud 010 = 300 baud 011 = 600 baud 100 = 1200 baud 101 = 2400 baud 110 = 4800 baud 111 = 9600 baud
4,3	พาริตี้ 00 หรือ 10 = ไม่มีพาริตี้ 01 = พาริตี้คี่ 11 = พาริตี้คู่
2	จำนวนของบิตสิ้นสุด 0 = 1 บิตสิ้นสุด 1 = 2 บิตสิ้นสุด

หมายเลขประจำบิต	ความหมาย
1,0	จำนวนบิตในข้อมูล 1 ไบต์ 10 = 7 บิต 11 = 8 บิต

ตัวอย่างการใช้รหัสเพื่อเตรียมสถานะของพอร์ตอนุกรม เช่น ถ้าต้องการตั้งให้พอร์ตมีอัตรา 9600 baud มีพาริตี้คู่ มี 1 บิตสิ้นสุด และใช้ 8 บิตต่อ 1 ไบต์ข้อมูล จะได้รูปแบบของรหัสดังตารางข้างล่างนี้

บิตที่	7	6	5	4	3	2	1	0
รหัส	1	1	1	1	1	0	1	1

เครื่องคอมพิวเตอร์โดยทั่วไป จะมีพอร์ตอนุกรมได้มากถึง 7 พอร์ต โดยหมายเลขพอร์ตที่จะใช้สามารถกำหนดผ่าน รีจิสเตอร์ DX พอร์ตแรกมีหมายเลข 0 พอร์ตต่อไปหมายเลข 1เช่นนี้เรื่อยๆไป

ฟังก์ชัน port_init() มีรายละเอียดดังนี้

```
void port_init(port,code)
int port;
unsigned char code;
{
      union REGS r;

      r.x.dx = port;       /* หมายเลขพอร์ตอนุกรม */
      r.h.ah = 0;          /* เรียกฟังก์ชันหมายเลข 0 ทำหน้าที่เตรียมสถานะ */
      r.h.al = code;       /* รหัสตั้งสถานะเริ่มต้น */
      int86(0x14,&r,&r);
}
```

ฟังก์ชัน check_stat()

ใช้เช็คสถานะของพอร์ตอนุกรม โดยการใช้ฟังก์ชันหมายเลข 3 ของอินเตอร์รัพท์หมายเลข 14 ของ BIOS รีจิสเตอร์ DX ใช้สำหรับหมายเลขพอร์ตที่จะตรวจสอบ สถานะของพอร์ตสามารถแปลความหมายได้จากรหัสที่ปรากฏในรีจิสเตอร์ AH และ AL ดังตาราง

สถานะของสายส่ง(AH)	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Data Ready	0
Overrun Error	1
Parity Error	2
Framing Error	3
Break-Detect Error	4
Transfer hold-register empty	5
Transfer shift-register empty	6
Time-out Error	7
สถานะของโมเด็ม(AL)	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Change in clear-to-send	0
Change in data-set-ready	1
Trailing-edge ring detector	2
Change in line signal	3
Clear-to-send	4

สถานะของโมเด็ม(AL)	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Data-set-ready	5
Ring indicator	6
Line signal detector	7

จะเห็นว่า สัญญาณสถานะต่าง ๆ ส่วนใหญ่จะใช้งานกับโมเด็ม ดังนั้น เมื่อนำมาใช้กับอุปกรณ์อื่น จึงลดความสำคัญลง แต่ยังมีสัญญาณอีกสัญญาณหนึ่งที่มีความสำคัญมากคือ Data Ready ซึ่งจะเป็นตัวบอกว่าเมื่อไหร่ที่ข้อมูลถูกรับเข้ามา และพร้อมที่จะถูกอ่านเข้าไปเก็บ

ฟังก์ชัน check_stat() มีรายละเอียดดังนี้

```
check_stat(port)
int port;      /* หมายเลขพอร์ต */
{
      union REGS r;

      r.x.dx = port;          /* กำหนดหมายเลขพอร์ต */
      r.h.ah = 3;             /* ฟังก์ชันหมายเลข 3 ทำหน้าที่ตรวจสอบสถานะพอร์ตอนุกรม */
      int86(0x14,&r,&r);
      return r.x.ax;
}
```

ฟังก์ชัน rport()

เมื่อเช็คสถานะพอร์ต จนได้ว่า Data Ready แล้ว ก็จะทำการอ่านข้อมูลจากพอร์ตอนุกรม ซึ่งสามารถทำได้โดยเรียกผ่าน BIOS อินเตอร์รัพท์หมายเลข 14 ฟังก์ชันหมายเลข 2 รีจิสเตอร์ DX ใช้สำหรับกำหนดหมายเลขพอร์ต ข้อมูลที่อ่านได้จะเก็บไว้ในรีจิสเตอร์ AL เมื่ออ่านข้อมูลเสร็จแล้ว สถานะ

ของข้อมูลและพอร์ตสามารถดูได้ที่บิตที่ 7 ของรีจิสเตอร์ AH

ฟังก์ชัน rport() ซึ่งจะทำการอ่านข้อมูล 1 ไบต์ มีรายละเอียดดังนี้ คือ

```
rport(port)
int port;      /* หมายเลขพอร์ต */
{
      union REGS r;

      /* รอจนกว่าจะได้รับตัวอักษร */
      while(!(check_stat(PORT)&256))
           if(kbhit())  {     /* ยกเลิกเมื่อมีการกดคีย์ */
               getch();
               exit(1);
           }
      r.x.dx = port;          /* กำหนดหมายเลขพอร์ต */
      r.h.ah = 2;             /* ฟังก์ชันหมายเลข 2 ทำหน้าที่อ่านตัวอักษร */
      int86(0x14,&r,&r);
      if(r.h.ah&128)
         printf("read error detected in serial port");
      return r.h.al;
}
```

ฟังก์ชัน sport()

ทำหน้าที่ส่งข้อมูล 1 ไบต์ออกทางพอร์ตอนุกรม โดยการใช้อินเตอร์รัพท์หมายเลข 14 ฟังก์ชันหมายเลข 1 ของ BIOS หมายเลขของพอร์ตอนุกรมที่จะทำการส่งข้อมูล สามารถกำหนดได้โดยผ่านรีจิสเตอร์ DX และข้อมูลที่ต้องส่งจะอยู่ในรีจิสเตอร์ AL เมื่อทำการส่งเรียบร้อยแล้ว สถานะของการส่งข้อมูลจะปรากฏในรีจิสเตอร์ AH เพื่อใช้สำหรับตรวจดูว่า การส่งข้อมูลถูกต้องหรือไม่

รายละเอียดของฟังก์ชัน sport() มีดังนี้

```
void sport(port,c)
int port;
char c;
{
      union REGS r;

      r.x.dx = port;
      r.h.al = c;
      r.h.ah = 1;
      int86(0x14,&r,&r);
      if(r.h.ah & 128) {
           printf("send error detected in serial port");
           exit(1);
       }
}
```

ถ้าบิตที่ 7 ของรีจิสเตอร์ AH มีค่า 1 เมื่อส่งข้อมูลเสร็จสิ้น แสดงว่าเกิดความผิดพลาดในการส่งข้อมูลขึ้น จะต้องตรวจดูสถานะของพอร์ต เพื่อดูว่าเกิดความผิดพลาดอะไร

ฟังก์ชัน get_data()

ฟังก์ชันนี้ใช้เมื่อ มีการส่งอินเตอร์รัพท์มาจากภายนอก (ค่าการอินเตอร์รัพท์ คือ FF) โดยจะทำการรับข้อมูลหมายเลขชั้นก่อน 1 ไบต์ ส่วนหมายเลขห้องและหมายเลขเหตุการณ์ จะทำการรับเป็นจำนวน 2 ครั้ง แล้วจะตรวจสอบดูว่าเท่ากันหรือไม่ ถ้าเท่ากันจะส่งตอบกลับไปด้วยค่า FF แต่ถ้าไม่เท่ากันจะตอบกลับไปด้วยค่า EE เพื่อแจ้งให้มีการส่งข้อมูลมาใหม่อีกครั้ง

ฟังก์ชัน get_data() มีรายละเอียดดังนี้

```
void get_data(port,dta)
int port;          /* หมายเลขพอร์ต */
char dta[];        /* ใช้เก็บข้อมูลที่อ่านได้ */
{
    char x;
    int i,j,y;

    dta[0] = r_port(port);     /* รับไบต์แรกเป็นหมายเลขชั้น */

    for(i = 0;i < 3;i++)       /* ส่งหมายเลขชั้นกลับไปเพื่อให้ตรวจสอบ */
       sport(port,dta[0]);     /*          ความถูกต้อง              */
    wait(port);                /*  รอจนกว่าจะมีการตอบกลับ           */

    y = 1;
    do  {
        for (j = 1; j <= 4; j++)       /*  อ่านข้อมูล 2 ไบต์ 2 ครั้ง  */
           dta[j] = rport(port);



        if (dta[1] != dta[3] || dta[2] != dta[4])  {
            sport(port,0xee);
            y = 0;                              /* ตรวจสอบความถูกต้อง     */
        }                                       /* ถ้าผิดส่งค่า EE กลับ    */
        else                                    /* ถ้าถูกส่งค่า FF กลับ    */
            sport(port,0xff);
    } while (y == 0);
}
```

ฟังก์ชัน send_data()

ทำหน้าที่

- ส่งข้อมูลเพื่อการควบคุมกลับ ทั้งในกรณีเหตุการณ์ปกติ และในกรณีที่เกิดเหตุการณ์ไม่ปกติขึ้น

- การส่งข้อมูลของเวลา เพื่อการตั้งเวลา ณ.จุดใด ๆ (บอร์ด MCS-51 ใด ๆ) โดยการใช้ฟังก์ชัน settime ซึ่งอยู่ในโปรแกรมพิเศษ

ดังนั้น รูปแบบของการส่งข้อมูลจึงมี 2 แบบ คือ เพื่อการควบคุมกลับ และการตั้งเวลา ซึ่งจะส่งข้อมูลกลับเป็นจำนวน 3 และ 2 ไบต์ตามลำดับ

ฟังก์ชัน send_data() มีรายละเอียดต่าง ๆ ดังนี้

```
void send_data(dta,format)
char dta[4];
int format;          /* รูปแบบการควบคุมกลับ */
{
        int i,j;
        int x;
        char code;
        int byte;

        do  {
            x = 1;
            sport(PORT,dta[0]);              /* ส่งหมายเลขชั้น */
            if (rport(PORT) != dta[0])  {
               sport(PORT,0xee);
               x = 0;
            }
        } while(x==0);

        switch(format) {          /* รูปแบบการควบคุม */
            case 1:
```

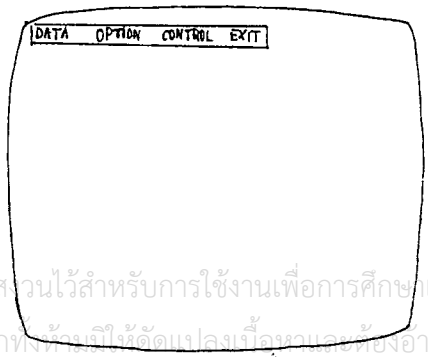
```
                code = 0x07;
                byte = 2;
                break;
            case 2:
                code = 0xf8;
                byte = 3;
                break;
            case 3:
                code = 0x8f;
                byte = 3;
                break;
        }

        sport(PORT,code);          /* ส่งรหัสการควบคุม */
        wait(PORT);

        for (i = 0;i < 2;i++)          /* ส่งข้อมูลที่จะควบคุมทั้งหมด */
            for (j = 0;j < byte;j++)
                sport(PORT,dta[j]);
        wait(PORT);
}
```

<u>การทำงานของโปรแกรมพิเศษ</u>

ขณะที่โปรแกรมกำลังแสดงรูปที่ 1 อยู่นั้น (รูปติก 20 ชั้น) จะมีการตรวจสอบการกดคีย์ และตรวจสอบการอินเตอร์รัพท์ หากคีย์ที่กดเป็นคีย์ 'F1' ก็จะเข้าสู่การทำงานของโปรแกรมพิเศษ ซึ่งจะมีลักษณะเป็น MENU ดังรูป



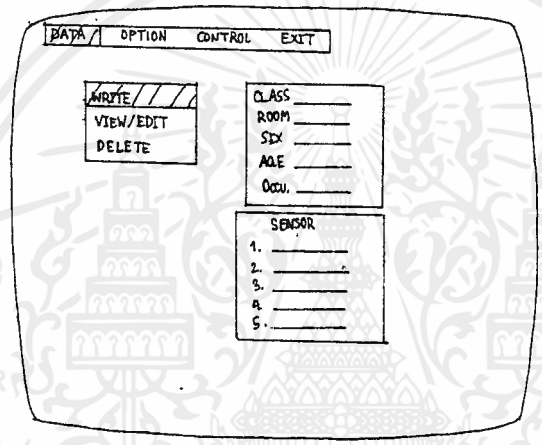
รูปที่ 3.4.5 แสดง menu ของโปรแกรมพิเศษ

แยกอธิบายการทำงานของ MENU ต่าง ๆ ได้ดังนี้

DATA มี 3 ฟังก์ชันการทำงาน ดังนี้ คือ

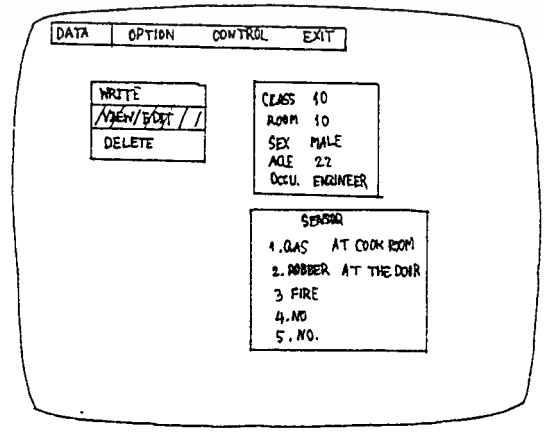
1.WRITE DATA ทำหน้าที่เก็บข้อมูลของผู้เป็นเจ้าของห้องพัก โดยจะมีการใส่ข้อมูลต่าง ๆ เช่น ชื่อของเจ้าของห้องพัก ,อายุ ,อาชีพ และรายละเอียดของการติดตั้งอุปกรณ์การตรวจจับ (sensor) ณ.จุดต่าง ๆ

การทำงานเป็นดังรูป



รูปที่ 3.4.6 แสดงเมื่อเลือกการทำงาน WRITE DATA

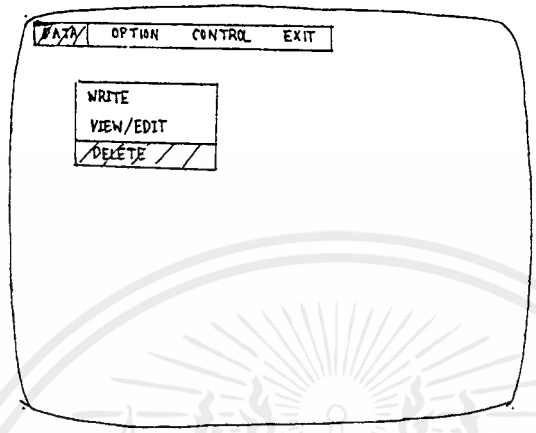
2.VIEW/EDIT ทำหน้าที่เรียกข้อมูลที่เก็บไว้ของแต่ละห้องพักมาดูได้ พร้อมทั้งถ้าต้องการแก้ไขข้อมูลใด ๆ ก็สามารถเลื่อนเคอร์เซอร์ เพื่อทำการแก้ไขได้ทันที ดังรูป



รูปที่ 3.4.7 แสดงเมื่อเลือกการทำงาน VIEW/EDIT

3.DELETE ทำหน้าที่ลบข้อมูลของห้องพักใด ๆ ที่ต้องการ โดยการใส่หมายเลขชั้น และหมายเลขห้อง ข้อมูลของห้องนั้นก็จะถูกลบออกไปพร้อมกับชื่อไฟล์ของข้อมูล

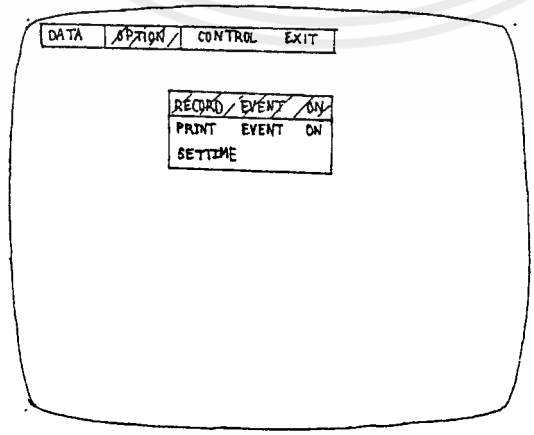
การทำงานเป็นดังรูป



รูปที่ 3.4.8 แสดงเมื่อเลือกการทำงาน DELETE

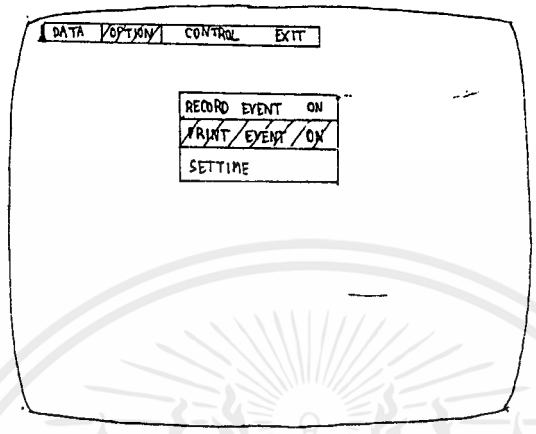
OPTION มี 3 ฟังก์ชันการทำงาน คือ

1.RECORD EVENT เพื่อเลือกว่า เมื่อเกิดเหตุการณ์ใด ๆ ขึ้น ต้องการให้มีการบันทึกข้อมูลลงไฟล์ข้อมูล หรือไม่ เลือกโดยการกด ENTER ให้เป็น RECORD EVENT ON หรือ RECORD EVENT OFF ดังรูป



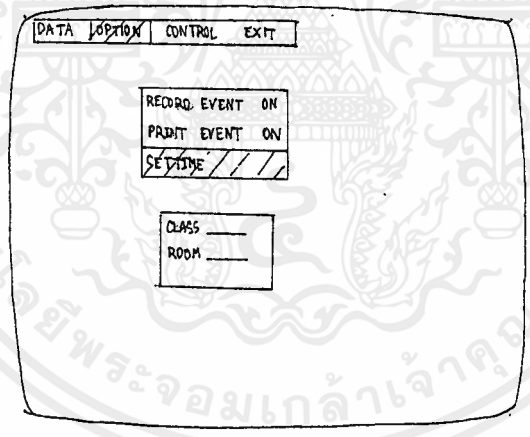
รูป 3.4.9 แสดงเมื่อเลือกการทำงาน RECORD EVENT

2.PRINT EVENT เพื่อทำการเลือกว่า เมื่อเกิดเหตุการณ์ใด ๆ ขึ้น ต้องการให้มีการบันทึกข้อมูล โดยการพิมพ์ไปยัง printer หรือไม่ เลือกโดยการกด ENTER ให้เป็น PRINT EVENT ON หรือ PRINT EVENT OFF ดังรูป



รูป 3.4.10 แสดงเมื่อเลือกการทำงาน PRINT EVENT

3.SETTIME ทำหน้าที่ ส่งข้อมูลเวลา เพื่อตั้งเวลาให้กับบอร์ดควบคุม โดยมีการทำงานดังรูป

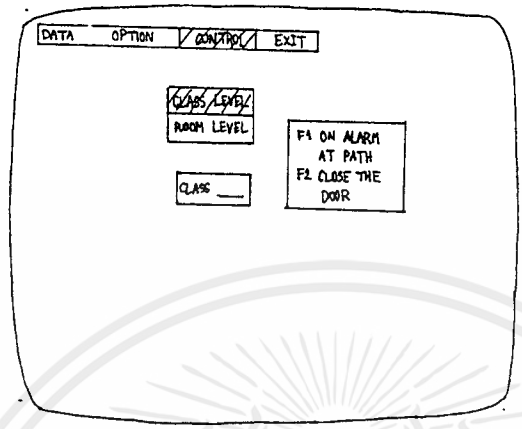


รูป 3.4.11 แสดงเมื่อเลือกการทำงาน SETTIME

ทำหน้าที่ ควบคุมกลับไปยังจุดที่ต้องการ ในขณะที่เหตุการณ์ปกติ โดยจะมีรูปแบบการควบคุมกลับ 2 ลักษณะ คือ

1.การควบคุมกลับระดับชั้น (CLASS LEVEL) จะเป็นการควบคุมกลับ ณ.จุด หรือ บริเวณที่เป็นส่วนรวมของทุกห้องในแต่ละชั้นนั้น เช่น การเปิดสัญญาณเตือนภัยบริเวณทางเดิน, การปิดประตูทางขึ้นลง เป็นต้น

เริ่มต้นเมื่อเลือก MENU "CLASS LEVEL" ก็ทำการใส่ค่าหมายเลขของชั้นที่ต้องการ และจะปรากฏตัวเลือกให้ว่าจะควบคุมไปที่ใด ดังรูป



รูป 3.4.12 แสดงเมื่อเลือกการทำงานการควบคุมระดับชั้น

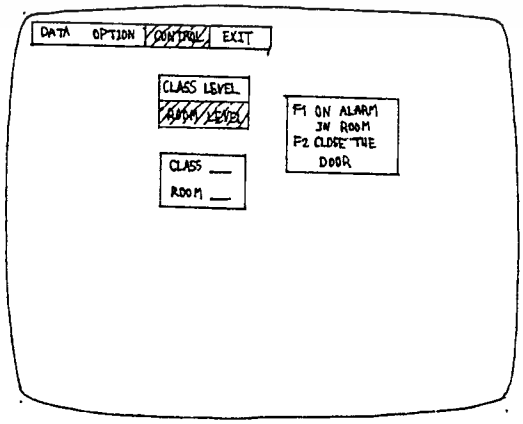
ถ้ากดคีย์ "F1" จะเป็นการควบคุมกลับเพื่อเปิดสัญญาณเตือนภัยบริเวณทางเดินของชั้น

ถ้ากดคีย์ "F2" จะเป็นการควบคุมกลับเพื่อปิดประตูทางขึ้นลงของชั้น

ถ้ากดคีย์ "ESC" จะยกเลิกการแสดงตัวเลือกจุดที่จะควบคุม และทำงานอื่นต่อไป

2.<u>การควบคุมกลับระดับห้อง</u> (ROOM LEVEL) เป็นการควบคุมกลับ ณ.จุดหรือบริเวณภายในห้องใดห้องหนึ่งที่ต้องการ เช่น การเปิดสัญญาณเตือนภัยภายในห้อง, การปิดประตูทางเข้าออกของห้อง เป็นต้น

เริ่มต้นเมื่อเลือก MENU "ROOM LEVEL" ก็ทำการใส่ค่าหมายเลขของชั้น และหมายเลขของห้องที่ต้องการ ต่อมาก็จะปรากฏตัวเลือกของจุดที่จะควบคุม ดังรูป



รูป 3.4.13 แสดงเมื่อเลือกการทำงานการควบคุมระดับห้อง

ถ้ากดคีย์ "F1" จะเป็นการควบคุมกลับเพื่อเปิดสัญญาณเตือนภัยบริเวณทางเดินของชั้น

ถ้ากดคีย์ "F2" จะเป็นการควบคุมกลับเพื่อปิดประตูทางขึ้นลงของชั้น

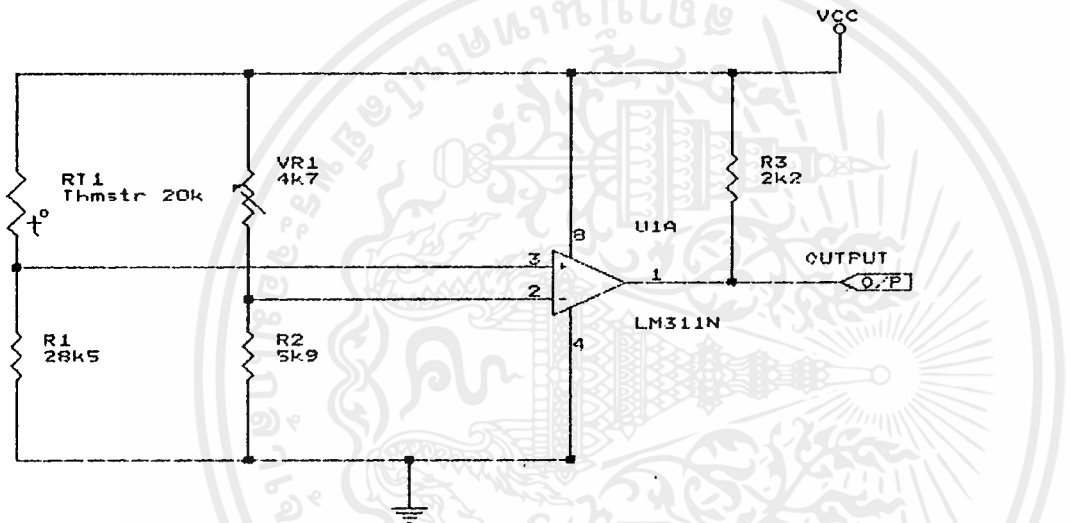
ถ้ากดคีย์ "ESC" จะยกเลิกการแสดงตัวเลือกจุดที่จะควบคุม และทำงานอื่นต่อไป

3.5 การสร้างวงจรในส่วน sensor

วงจรที่ได้ทำการสร้าง สำหรับโครงงานนี้ ได้แก่

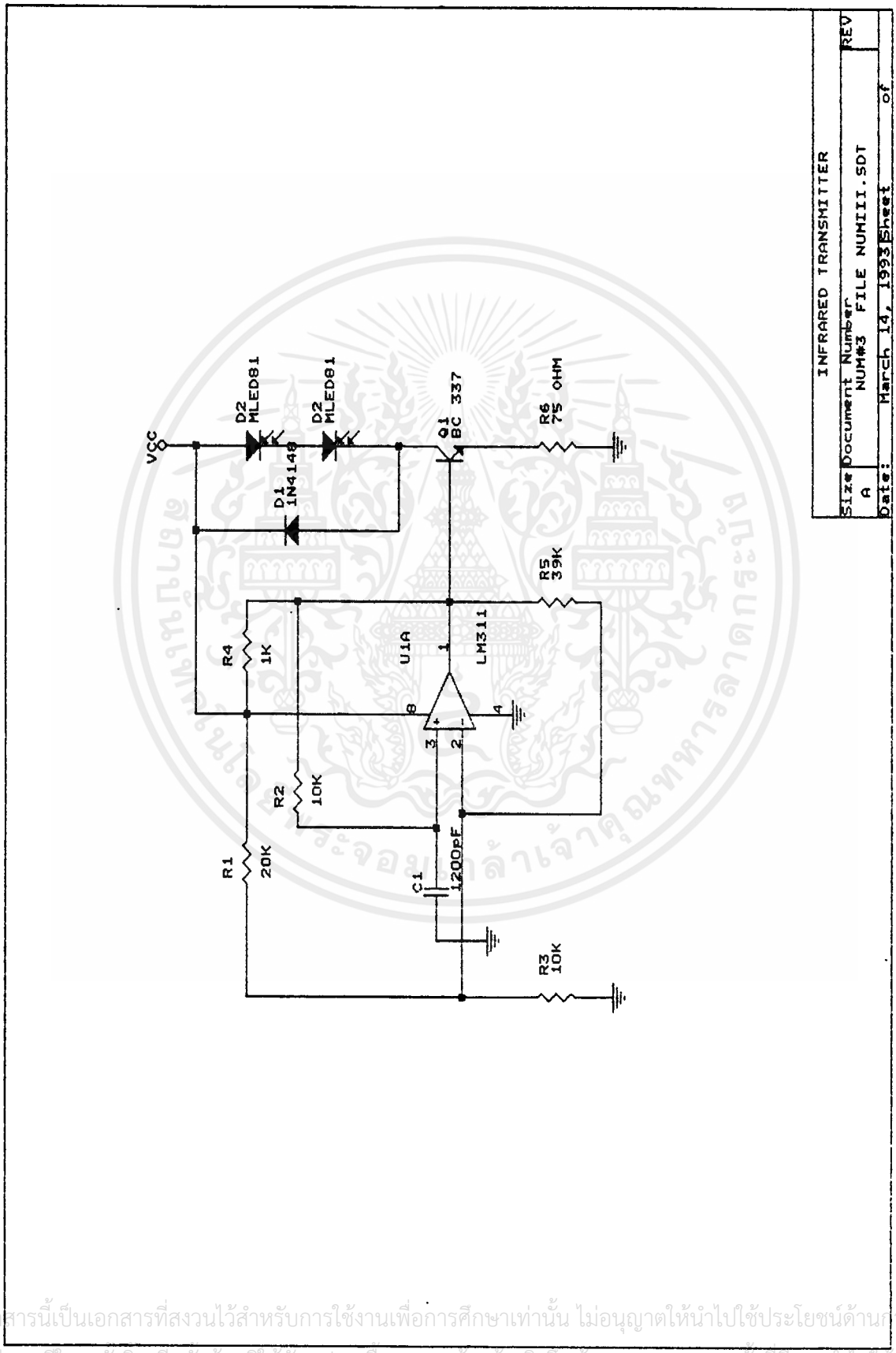
1. วงจรตรวจจับควัน (รูปที่ 3.5.1)
2. วงจร infrared sensor
   - ภาคส่ง (รูปที่ 3.5.2)
   - ภาครับ (รูปที่ 3.5.3)
3. วงจร ultrasonic sensor
   - ภาคส่ง (รูปที่ 3.5.4)
   - ภาครับ (รูปที่ 3.5.5)

ซึ่งวงจรต่าง ๆ เป็นดังรูปตามลำดับ ดังนี้

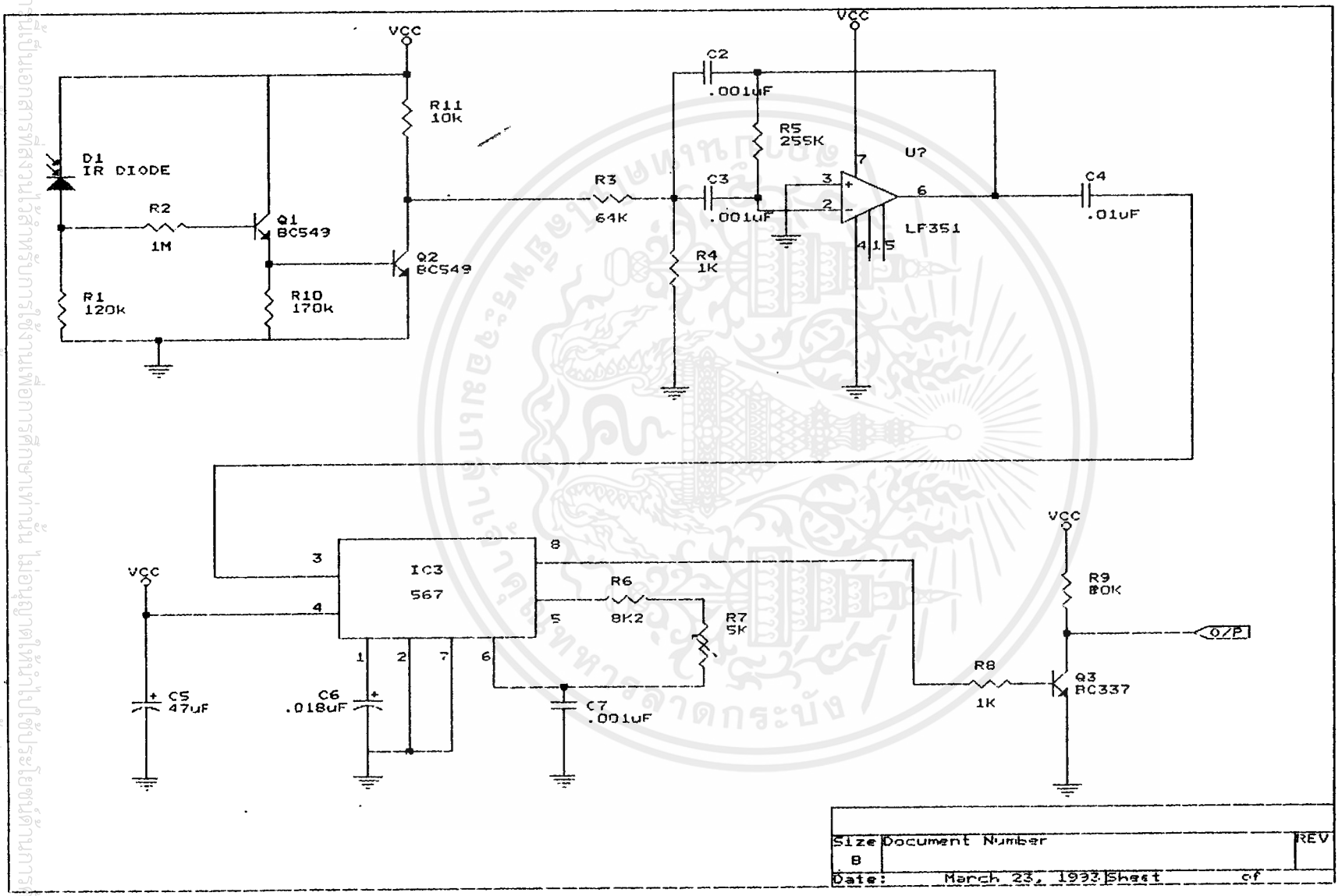


HEAT DETECTOR (FIRE ALARM)		
Size A	Document Number NUM#2 FILE NUMII.SCT	REV
Date: March 23, 1993	Sheet of	

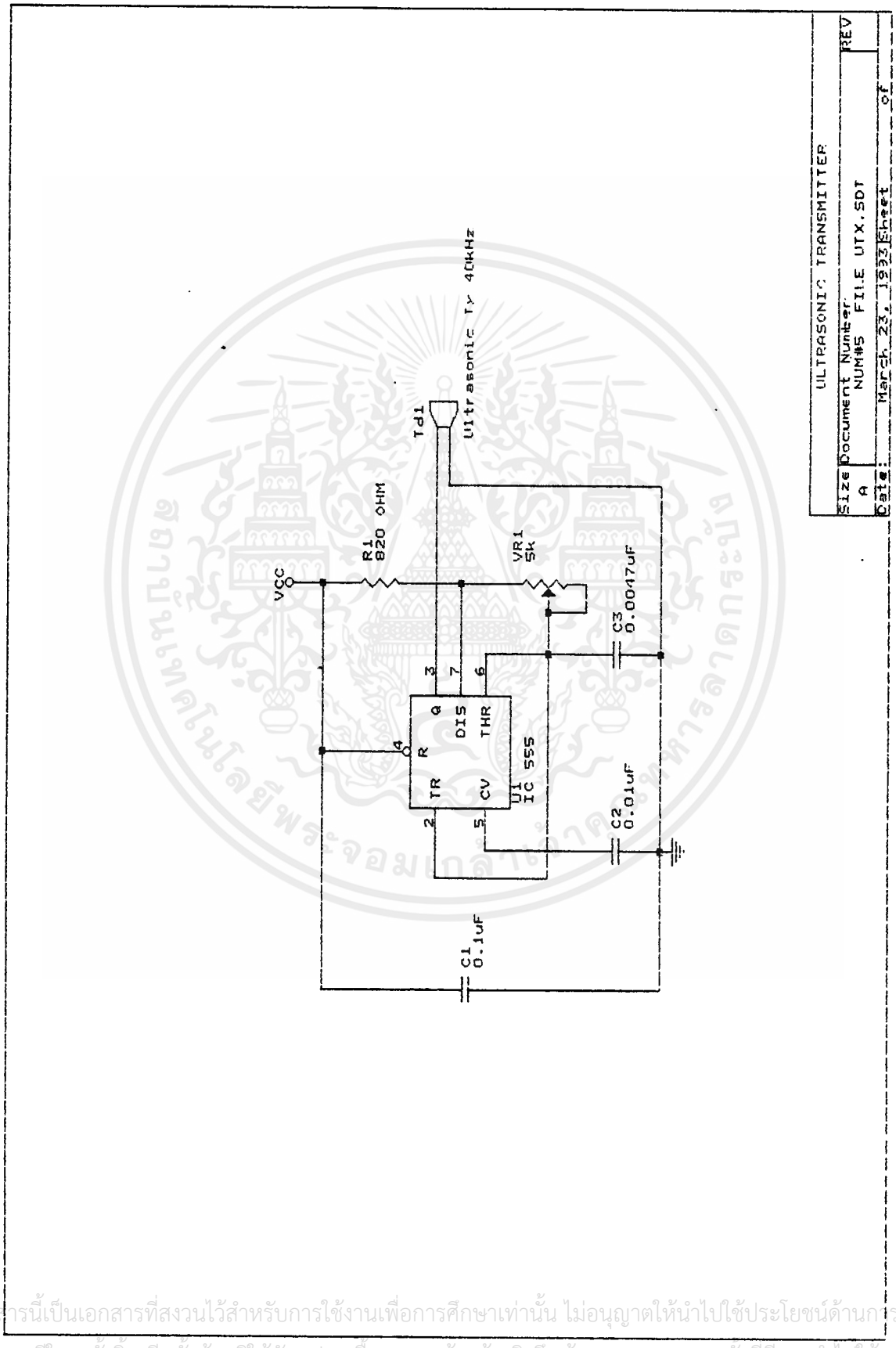
รูปที่ 3.5.1 วงจรตรวจจับความร้อน



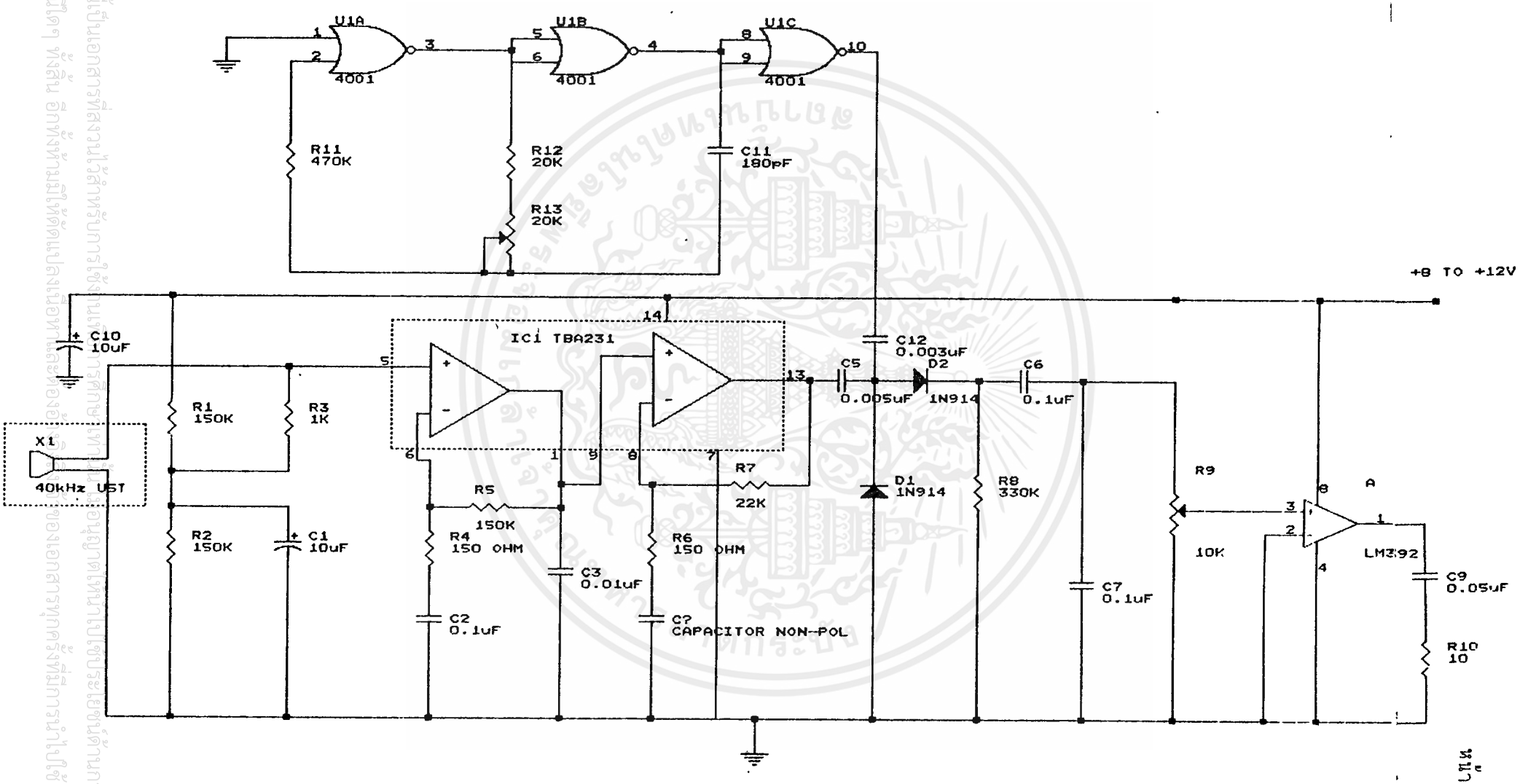
รูปที่ 3.5-2 วงจรภาคส่งสัญญาณ INFRARED SENSOR



รูปที่ 3.5.3 วงจรภาคส่ง INFRARED SENSOR



รูปที่ 3.5.4 วงจรภาคส่ง ULTRASONIC SENSOR



รูปที่ 3.5.5 วงจร ULTRASONIC SENSOR (ภาครับ)

บทที่ 4
การทดลองและผลการทดลอง

ในการติดต่อระหว่าง บอร์ด microcontroller 8031 ระดับชั้นกับบอร์ด microcontroller ระดับห้องนั้น ติดต่อกันทาง serial port ของ CPU 8031 ซึ่งสามารถทำการติดต่อกันได้อย่างถูกต้องแม่นยำ แทบจะไม่มีข้อผิดพลาด

แต่ในการติดต่อระหว่าง บอร์ด microcontroller ระดับชั้นกับ IBM PC นั้น เนื่องจาก 8031 มี serial port เพียง 1 port ซึ่งใช้ติดต่อกับห้องไปแล้ว จึงต้องดัดแปลง parallel port ของ 8031 ให้ส่งแบบ serial แบบ Asynchronous โดยใช้ 8251 USART ทำให้การติดต่อระหว่างชั้น กับ IBM PC มีการผิดพลาดบ่อยครั้ง

ในช่วงแรกของการทดลองรับส่งข้อมูลระหว่าง IBM PC กับ บอร์ด microcontroller นั้น ได้กระทำการรับส่งโดยใช้การรับส่งระหว่าง serial port ของทั้งสองฝ่าย เพื่อทดสอบซอฟท์แวร์ที่ใช้รับส่งใน เครื่อง IBM PC ผลปรากฏว่า ซอฟท์แวร์ที่ใช้สามารถรับส่งข้อมูลได้อย่างดีโดยไม่มีข้อผิดพลาด

แต่เมื่อ บอร์ด microcontroller ใช้ parallel port มาดัดแปลงให้ส่งแบบ serial กับ IBM PC จะทำให้มีข้อผิดพลาดในการรับส่งข้อมูลมาก กล่าวคือ ในบางครั้ง รับส่งข้อมูลได้อย่างถูกต้อง แต่บางครั้งก็ผิดพลาดในบางไบต์ข้อมูล หรือ บางครั้งอาจจะรับส่งไม่ได้เลย

สำหรับการแสดงผลของเหตุการณ์ ทางจอแสดงผลของ IBM PC ถ้าสามารถรับข้อมูลมาได้อย่างถูกต้อง ก็สามารถแสดงผลได้ไม่มีปัญหาอะไร ส่วนการส่งข้อมูลเพื่อควบคุมกลับของ IBM PC ไปยังบอร์ด microcontroller ก็เช่นกัน ถ้าไม่เกิดข้อผิดพลาดในการส่งข้อมูล ก็จะสามารถทำงานเพื่อควบคุมจุดที่ต้องการได้

สำหรับในส่วนของตัวตรวจจับนั้น ผลการทดลองเป็นดังนี้

วงจรตรวจจับความร้อน การทดลองปรากฏว่า ทำงานได้ดี วงจรตรวจจับได้ที่อุณหภูมิประมาณ 40 องศาเซลเซียส ถ้าจะให้ตรวจจับได้ที่อุณหภูมิเร็วกว่านี้ ทำได้โดยเปลี่ยนค่าความต้านทานในวงจร Bridge

สำหรับวงจร ลวิทช์ อินฟราเรดนั้น วงจรในภาคส่ง ส่งด้วยพัลส์ความถี่ประมาณ 100 KHz กระแสประมาณ 60 mA ภาครับจะรับได้ไม่ไว และมีสัญญาณรบกวนเยอะ ต้องออกแบบใหม่และให้มีการชิลด์วงจรที่ดี

วงจร์อุลตราโซนิค ภาครับ รับสัญญาณได้ดี แต่จะมีความไวมาก การทดลองต้องทดลองในห้องที่เงียบสงบ และต้องมีการพัฒนาในส่วนของการดีเทคความถี่ต่อไป

บทที่ 5
สรุปผลและวิจารณ์

ระบบรักษาความปลอดภัยสำหรับอาคารที่ได้สร้างขึ้นนี้ เป็นการนำความรู้ทางด้านอุปกรณ์การตรวจจับ (sensor) ,ความรู้ทางด้านอุปกรณ์ไมโครโปรเซสเซอร์, ความรู้ทางด้านการเขียนโปรแกรมคอมพิวเตอร์ และความรู้ทางด้านการสื่อสารข้อมูลมาประยุกต์ใช้เข้าด้วยกัน

ในส่วนการสร้าง จะมีความยุ่งยากและมีอุปสรรคในส่วนของฮาร์ดแวร์เป็นสำคัญ ซึ่งอยู่ที่การนำส่วนต่าง ๆ มาเชื่อมต่อกัน ให้เป็นระบบเดียวกัน ส่วนในส่วนซอฟท์แวร์นั้นสามารถพัฒนาได้อย่างสะดวกและไม่มีขีดจำกัด โดยเฉพาะในส่วนของ IBM PC

โครงงานชิ้นนี้ เป็นโครงงานที่ได้ออกแบบและพัฒนาขึ้นมาจากแนวความคิด และอาศัยความรู้ต่าง ๆ จึงพัฒนาได้ในระดับเบื้องต้นเท่านั้น มีอีกหลายส่วน ที่ควรได้รับการพัฒนาต่อไป ซึ่งเชื่อว่าหากพัฒนาได้จนอยู่ในระดับที่เหมาะสม จะสามารถนำไปใช้งาน และจะเกิดประโยชน์ได้เป็นอย่างดี

ภาคผนวก

ก.

แสดง FLOWCHART และ SOFTWARE การทำงานของบอร์ด 8031

อธิบายการทำงานของ FLOWCHART

FLOWCHART 1: main program ของห้อง

จะตรวจสอบว่ามี sensor ผิดปกติ หรือมีการควบคุมกลับหรือเปล่า ถ้าตรวจสอบพบ sensor ผิดปกติ จะไปทำงานตาม FLOWCHART 3 และแสดงผล ในกรณีที่ sensor ตัวแรกติดและยังไม่ถูก reset แล้ว sensor ตัวที่ 2 ติด จะแสดง sensor ตัวที่ติดหลังสุด และจะทำการตรวจสอบจากชั้นเรื่อย ๆ ว่าต้องการควบคุมกลับ รวมทั้งตั้งเวลาหรือเปล่า ถ้าพบ จะไปทำงานตาม FLOWCHART 6

FLOWCHART 2: main program ของชั้น

จะคอยตรวจสอบว่ามีการติดต่อจากห้อง (กรณี sensor ผิดปกติ) หรือติดต่อจาก IBM PC (ในกรณีควบคุมกลับ) ถ้าพบว่ามีการติดต่อจากห้อง จะไปทำงานตาม FLOWCHART 4 และถ้ามีการติดต่อจาก IBM PC จะไปทำงานตาม FLOWCHART 5

FLOWCHART 3:

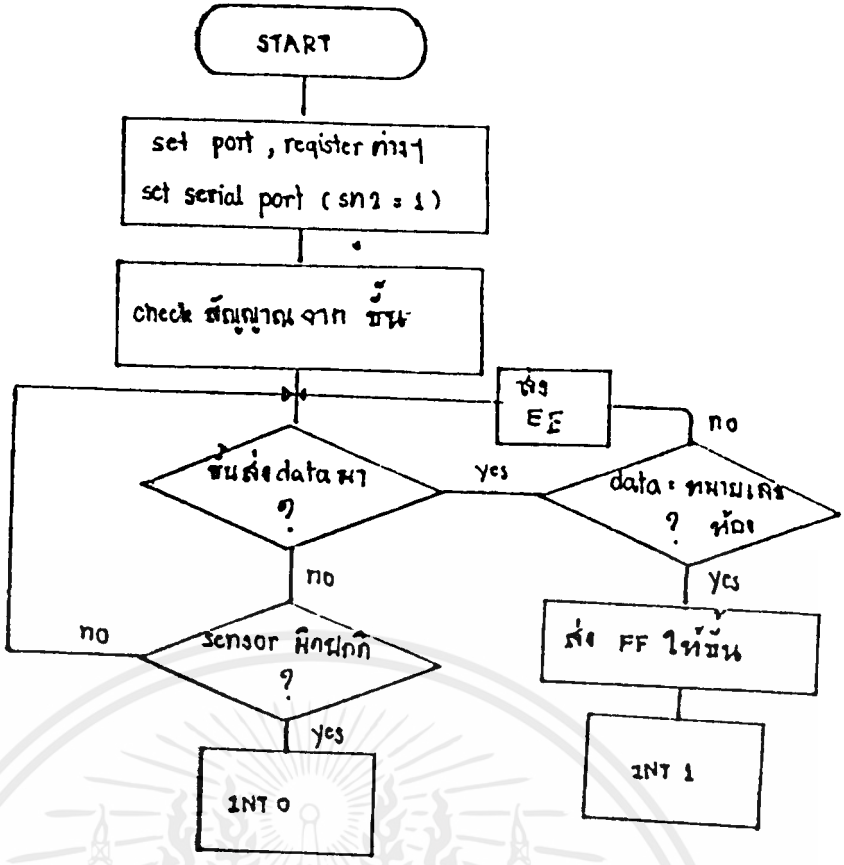
เมื่อห้องพบ sensor ผิดปกติ ก็จะไปทำการตรวจสอบว่า sensor ตัวใดผิดปกติ และจะทำการติดต่อกับชั้น เพื่อส่งหมายเลขห้องกับเหตุการณ์ (หมายเลข sensor ที่ผิดปกติ) ไปให้ชั้น

FLOWCHART 4:

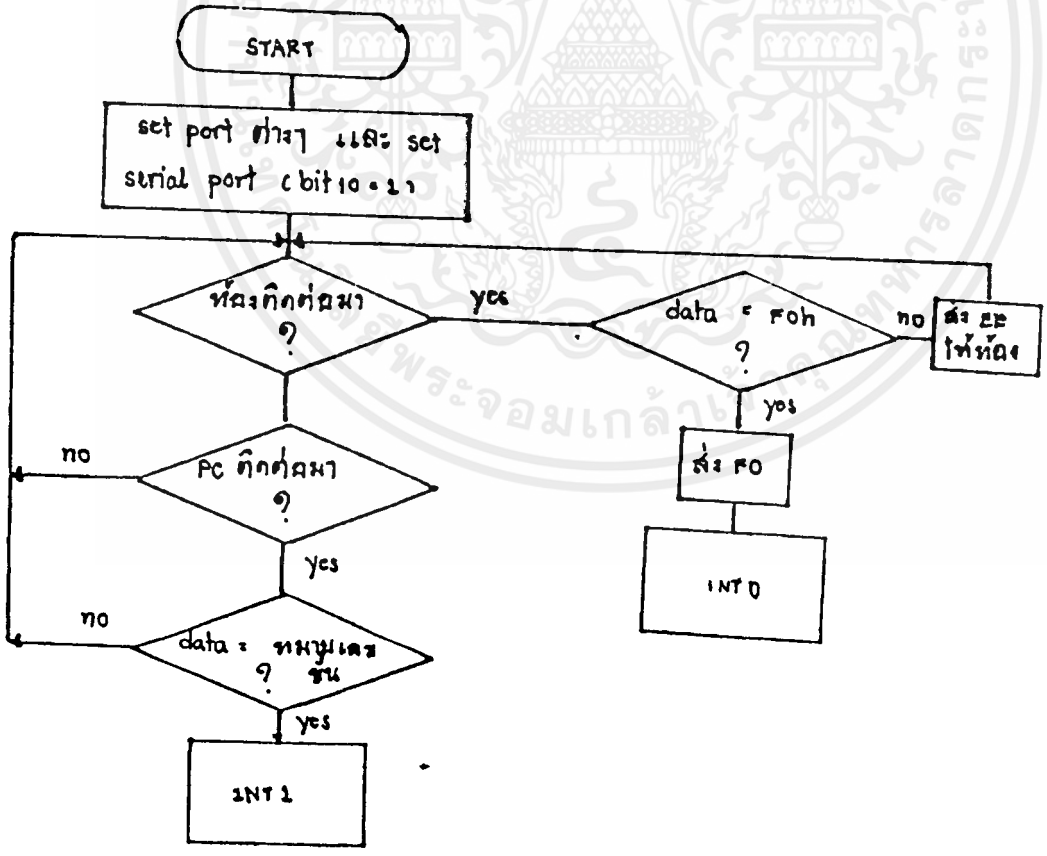
เมื่อห้องติดต่อมา (ตาม FLOWCHART 3) ชั้นจะรับข้อมูลจากห้อง แล้วทำการแจ้งให้ IBM PC ทราบ เพื่อไปแสดงผล

FLOWCHART 5:

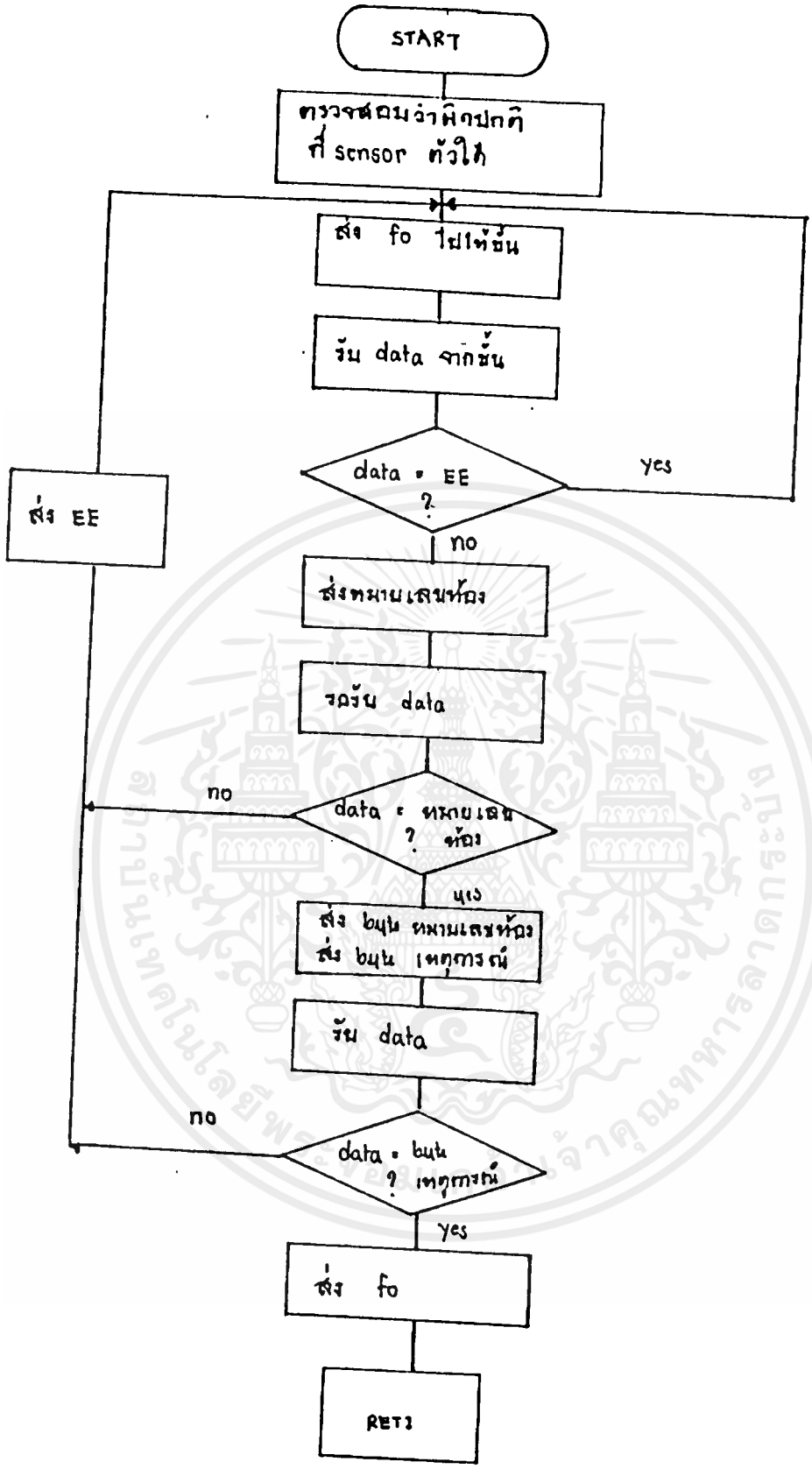
เป็นโปรแกรมระดับห้อง โดยจะตรวจสอบตลอดเวลาว่า ชั้นส่งสัญญาณมาควบคุมหรือไม่ ถ้ามีก็จะ check ว่าต้องการควบคุมออกทางพอร์ตของ 8255 หรือต้องการตั้งเวลา แล้วจะไปทำงานตามที่ชั้นส่งมาให้



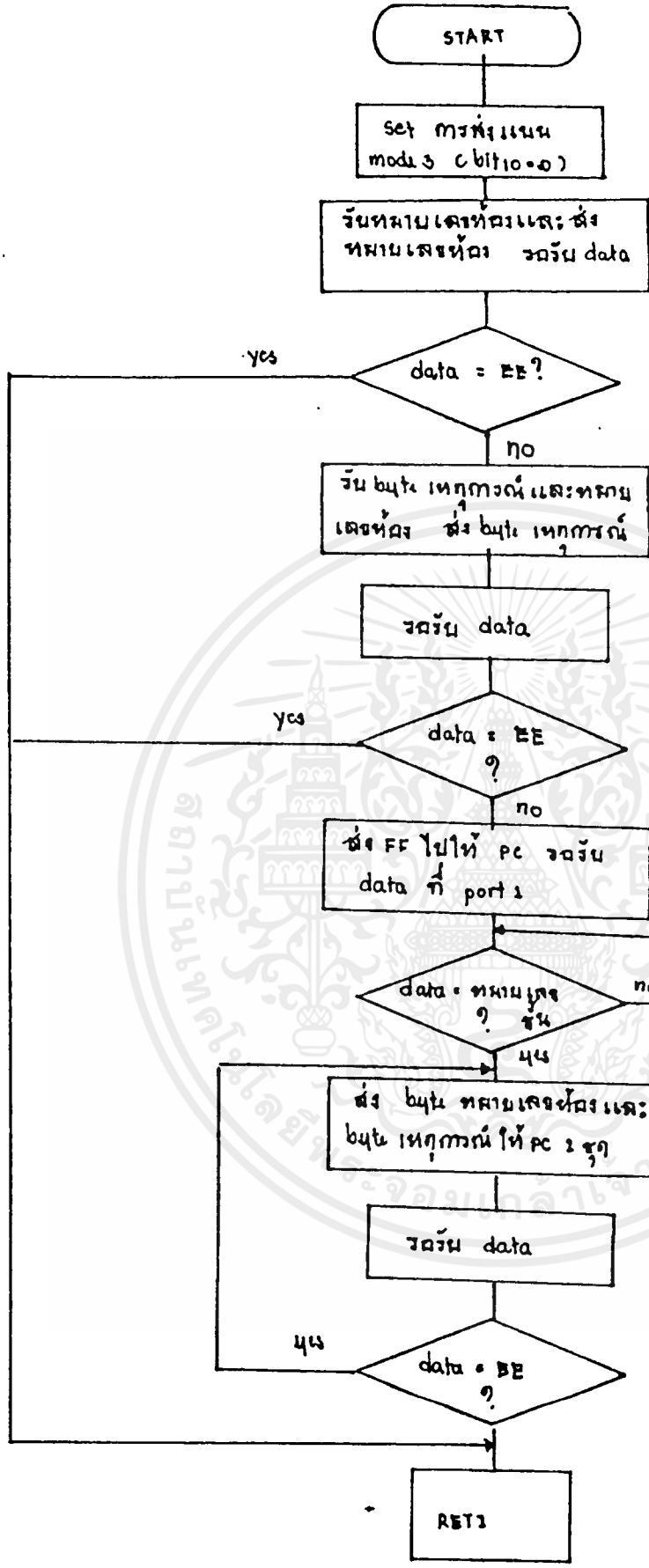
FLOWCHART 1 : main program ของ ห้อง



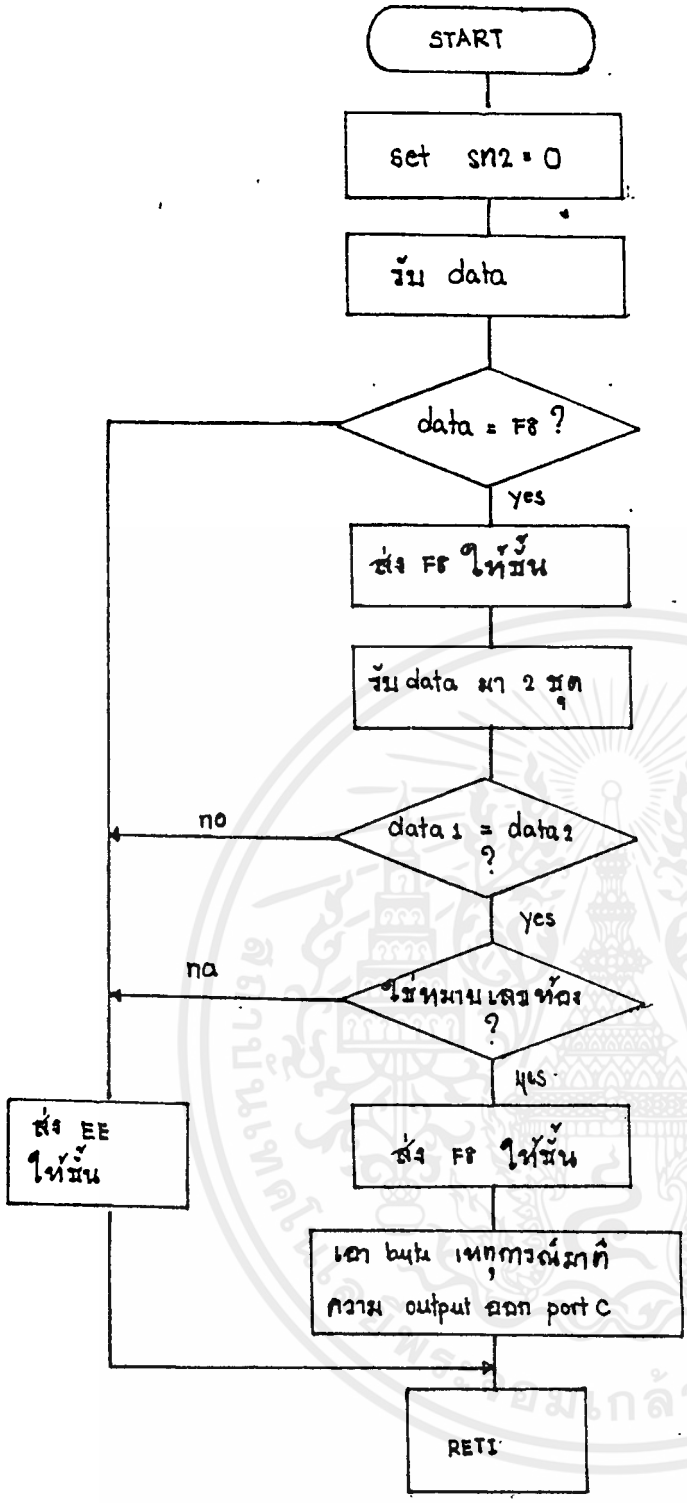
FLOWCHART 2 : main program ของ ขั้น



FLOWCHART 3 : Sensor พิกปกติ แจ้งไปยังชั้น



FLOWCHART 4 : ขั้นรอรับข้อมูลจากห้อง แล้วแจ้งให้ PC ทราบ



FLOWCHART 5 : รับการควบคุมกลับจากมัน

SOFTWARE ของบอร์ด microcontroller

ระดับห้อง

```
;*****************************************
;*                                       *
;*       PROGRAM CONTROL                 *
;*       ROOM N°.1                       *
;*                                       *
;*       PROGRAM BY                      *
;*       A.S. SEBSEEDSUCCESS             *
;*       Electronics Department          *
;*       KMIT L                          *
;*                                       *
;*****************************************

        ORG     0000H
;---------------------------------------
NROOM   EQU     01H                     ;ROOM No.1
CONWORD EQU     92H
PORTA   EQU     0E0E0H
PORTB   EQU     0E0E1H
PORTC   EQU     0E0E2H
CONTRL  EQU     0E0E3H
ONESSF  EQU     064H
ONEHSP  EQU     10H
ONEMSP  EQU     11H
SENPA   EQU     00H
SENPB   EQU     00H
COMPT   EQU     0F0H
ERROR   EQU     0EEH
TIME    EQU     08FH
FSCTRL  EQU     0F8H

;---------------------------------------
;       BIT AREA
;---------------------------------------
ENDOT   EQU     7FH
INTF6   EQU     7EH

;---------------------------------------
;       INTERNAL RAM
;---------------------------------------
RAMIN   EQU     30H
SENA    EQU     RAMIN
SENB    EQU     RAMIN+1
SEN0    EQU     RAMIN+2
SEN1    EQU     RAMIN+3
SEN2    EQU     RAMIN+4
SEN3    EQU     RAMIN+5
SEN4    EQU     RAMIN+6
SEN5    EQU     RAMIN+7
SEN6    EQU     RAMIN+8
SEN7    EQU     RAMIN+9
SEN8    EQU     RAMIN+10
SEN9    EQU     RAMIN+11
SEN10   EQU     RAMIN+12
SEN11   EQU     RAMIN+13
SEN12   EQU     RAMIN+14
```

```
SEN13   EQU     RAMIN+15
SEN14   EQU     RAMIN+16
SEN15   EQU     RAMIN+17
SENMC   EQU     RAMIN+18        ;18-33
KEEPSEN EQU     RAMIN+34
SENNUM  EQU     RAMIN+35
SENDSEN EQU     RAMIN+36
HEX     EQU     RAMIN+37
BCD     EQU     RAMIN+41
DSP     EQU     RAMIN+45
GENPUR  EQU     RAMIN+49
TENMS   EQU     RAMIN+50
SEC     EQU     RAMIN+51
MIN     EQU     RAMIN+52
HOUR    EQU     RAMIN+53
TMSET   EQU     RAMIN+54
BLDOT   EQU     RAMIN+55
CONBYTE EQU     RAMIN+56        ;KEEP CONTROL BYTE FROM USER
;---------------------------------
        LCALL   DELAY
        LCALL   DELAY
        LCALL   DELAY
        SJMP    START


        ORG     000BH
        LJMP    CLOCK

START:  MOV     R7,#00H                 ;DELAY TIME START
        DJNZ    R7,$
        DJNZ    R7,$
        DJNZ    R7,$
        MOV     A,#CONWORD
        MOV     DPTR,#CONTRL
        MOVX    @DPTR,A
        MOV     A,#PORTC
        MOV     A,#00H
        MOVX    @DPTR,A
        ANL     TMOD,#0FH
        ORL     TMOD,#20H
        MOV     SCON,#0F8H              ;SET SM2=1,TB8=1
        MOV     TH1,#0FAH               ;BAUDRATE 4800
        ANL     PCON,#7FH
        SETB    EA
        SETB    ET0
        SETB    TR1

        MOV     TENMS,#100D
        MOV     HOUR,#12D
        MOV     MIN,#34D
        MOV     SEC,#00D
        LCALL   STTIME
        SETB    ENDDT
        CLR     INTFB
        MOV     SENA,#00H
```

```
        MOV     SENB,#00H
        MOV     SENNUM,#0FH             ;#####

CHECK:  SETB    TB8
        SETB    SM2
        JB      RI,CKROOM               ;CHECK CONTROL BACK &&

CKSEN:  JB      INTFG,OLDSEN            ;FGINTO = 1,SENSOR
        MOV     DPTR,#PORTA             ;ABNORMAL CHECK OLD SENSOR
        MOVX    A,@DPTR                 ;FGINTO = 0,SENSOR
        CJNE    A,#00H,OLDSEN           ;NORMAL
        MOV     DPTR,#PORTB             ;CHECK SENSOR
        MOVX    A,@DPTR                 ;SENSOR NORMAL
        CJNE    A,#00H,OLDSEN           ;DISPLAY CLOCK
        LCALL   DSPCLK
        SJMP    CHECK

CKROOM: LCALL   XCEV                    ;CHECK OWN ROOM?
        CJNE    A,#NROOM,CHECK          ;NO,GOTO CHECK
        CLR     TB8                     ;YES,SEND 'FF'
        CLR     SM2                     ;TO CLASS
        MOV     A,#0FFH                 ;CALL ROOMCTRL
        LCALL   XMIT
        LCALL   RMCTRL
        SJMP    CHECK

OLDSEN: LCALL   DSPSEN                  ;SENSOR STILL ABNORMAL
        LCALL   ABSEN                   ;DISPLAY SENSOR
        SJMP    CHECK

;***************************************
;
;       INTO SENSOR ABNORMAL
;
;***************************************
ABSEN:  MOV     DPTR,#PORTA             ;COMPARE PORTA WITH
        MOVX    A,@DPTR                 ;OLD SENA
        CJNE    A,SENA,READA            ;SENA??
        MOV     DPTR,#PORTB             ;COMPARE PORTB WITH
        MOVX    A,@DPTR                 ;OLD SENB
        CJNE    A,SENB,READB            ;SENB???
        LJMP    EXITOLD

READA:  XRL     A,SENA                  ;MOV PORTA IN SENA
        MOV     R0,#0FFH
        CLR     C

SEEKA:  INC     R0                      ;CLEAR BYTE NOT
        RRC     A                       ;SAME BEFORE
        JNC     SEEKA                   ;PORTA
        MOV     A,#SEN0
        ADD     A,R0
        MOV     R0,A
        MOV     @R0,#00H                ;CLEAR
        MOVX    A,@DPTR
```

```
        MOV     SENA.A                  ;KEEP IN SENA
        SJMP    ZERO

READB:  XRL     A.SENB                  ;MOV PORTB IN SENB
        MOV     R0.#0FFh
        CLR     C

SEEKB:  INC     R0                      ;CLEAR BYTE NOT
        RRC     A                       ;SAME BEFORE
        JNC     SEEKB                   ;PORTB
        MOV     A.#SENB
        ADD     A.R0
        MOV     R0.A
        MOV     @R0.#00H
        MOVX    A.@DPTR
        MOV     SENB.A                  ;KEEP IN SENB

ZERO:   MOV     A.SENA                  ;COMPARE PORTA AND PORTB
        JNZ     READ                    ;WITH 00
        MOV     A.SENB
        JZ      CLRSEN                  ;YES, GOTO CLRSEN

READ:   MOV     DPTR.#PORTB             ;KEEP VALUE OF SENSOR
        MOVX    A.@DPTR                 ;IN SEN0-SEN15
        MOV     R1.#SEN15
        MOV     R2.#08H                 ;KEEP PORTB

TRANB:  CLR     C
        RLC     A
        PUSH    ACC
        MOV     A.#00H
        RLC     A
        ADD     A.@R1
        MOV     @R1.A
        DEC     R1
        POP     ACC
        DJNZ    R2,TRANB
        MOV     DPTR.#PORTA
        MOVX    A.@DPTR
        MOV     R1.#SEN7                ;KEEP PORTA
        MOV     R2.#08H

TRANA:  CLR       C
        RLC     A
        PUSH    ACC
        MOV     A.#00H
        RLC     A
        ADD     A.@R1
        MOV     @R1.A
        DEC     R1
        POP     ACC
        DJNZ    R2.TRANA
        MOV     R2.#16D                 ;REFINE OUT BYTE00
        MOV     R1.#SEN1F               ;
        MOV     R3.#00H
```

```
NONZERO:MOV     A,@R1
        DEC     R1
        JZ      JUMP
        MOV     A,R1
        INC     A
        PUSH    ACC
        INC     R3

JUMP:   DJNZ    R2,NONZERO
        MOV     A,R3
        MOV     R4,A                    ;R4 KEEP BYTE NONZERO
        MOV     R0,#SENNO               ;BYTE NONZERO KEEP

KEEP:   POP     ACC
        MOV     @R0,A                   ;IN SENNO>
        INC     R0
        DJNZ    R3,KEEP
        MOV     R0,#SENNO
        MOV     KEEPSEN,@R0

GETNO:  MOV     R5,KEEPSEN              ;R5 KEEP MININUM BYTE

MINUS:  CJNE    R4,#01H,DECRE
        SJMP    RESULT

DECRE:  DEC     R4
        CLR     C
        MOV     R1,KEEPSEN              ;MOV    KEEPSEN,@R0
        MOV     A,@R1                   ;MOV    A,@R1
        INC     R0
        MOV     KEEPSEN,@R0
        MOV     R1,KEEPSEN
        SUBB    A,@R1
        JC      MINUS
        SJMP    GETNO

RESULT: MOV     A,R5
        INC     A
        MOV     R1,#SEN0
        SUBB    A,R1
        MOV     SENNUM,A                ;DISPLAY SENSOR
        MOV     SENDSEN,A               ;SENDSEN FOR SEND
        SETB    INTFG                   ;EVENT BYTE
        LJMP    EVENT                  ; ### LJMP    EVENT

CLRSEN: MOV     R1,#SEN0
        MOV     R2,#16D
        CLR     A

CLEAR:  MOV     @R1,A                   ;INTFG
        INC     R1
        DJNZ    R2,CLEAR
        CLR     INTFG
EXITOLD:LJMP    EXITSEN
```

```
EVENT:  SETB    TB8
        SETB    SM2
        MOV     A,#COMPT                ;SEND  F0  TO CLASS
        CLR     SM2
        LCALL   XMIT
        CLR     TB8
        LCALL   XCEV
        ANL     A,#11H                  ;CHECK CLASS SEND
        JZ      EVENT                   ;ERROR HAPPEN
        MOV     A,#NROOM                ;GOTO EVENT
        LCALL   XMIT                    ;YES.SEND ROOM No.

ROOM?:  MOV     A,#00H                  ;
        LCALL   XCEV
        CJNE    A,#NROOM,SENDE0         ;CHECK  ERROR
        MOV     A,#COMPT                ;NO.GOTO SENDE0
        LCALL   XMIT                    ;YES.
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        MOV     A,#NROOM                ;SEND ROOM NO.
        LCALL   XMIT                    ;
        NOP
        NOP
        NOP
        NOP
        MOV     A,SENDSEN               ;SENDSEN KEEP SENSOR NO.
        LCALL   XMIT                    ;SEND EVENT
        MOV     A,#00H
        LCALL   XCEV                    ;RECEIVE SENDSEN
        CJNE    A,SENDSEN,SENDE0        ;A = SENDSEN
        MOV     A,#COMPT
        LCALL   XMIT
        SJMP    EXITSEN

SENDE0: MOV     A,#ERROR                ;ERROR SEND ERROR
        LCALL   XMIT
        SJMP    EVENT

EXITSEN:SETB    TB8
        SETB    SM2
        RET
```

```
;****************************************;
;                                        ;
;       SUBROUTINE RMCTRL                ;
;       CONTROL ROOM                     ;
;       FROM PC                          ;
;                                        ;
;****************************************;

RMCTRL: PUSH    02H
        PUSH    03H
        PUSH    04H


        CLR     SM2
        CLR     TB8
        LCALL   XCEV                    ;RECEIVE FGCTRL
        CJNE    A,#FGCTRL.SETTIME       ;FGCTRL = 'FB'?
        MOV     A,#FGCTRL               ;YES.SEND FGCTRL
        LCALL   XMIT
        LCALL   XCEV
        MOV     R2,A                    ;R2 = ROOM NO.(1)
        LCALL   XCEV
        MOV     R3,A                    ;R3 = EVENT BYTE(1)
        LCALL   XCEV
        MOV     R4,A                    ;R4 = ROOM NO.(2)
        LCALL   XCEV                    ;A =EVENT BYTE(2)
        XRL     A,R3                    ;EVENT BYTE(1) = (2) ?
        JNZ     SENDE1                  ;NO.
        MOV     A,R4                    ;YES,
        XRL     A,R2                    ;ROOM NO.(1) = (2) ?
        JNZ     SENDE1                  ;NO.
        MOV     A,R4                    ;YES,

        CJNE    A,#NROOM,SENDE1         ;ROOM NO. = OWN ROOM ?
        MOV     A,#FGCTRL               ;NO.GOTO
        LCALL   XMIT                    ;YES,
        MOV     A,R3                    ;SEEK CONTROL
        ANL     A,#00000111B
        MOV     R0,#conbyte
        LCALL   CTRLPC                  ;TABLE FOR CONTROL
        MOV     A,CONBYTE               ;CONBYTE KEEP CONTROL
        MOV     DPTR,#PORTC             ;PORTC =
        MOVX    @DPTR,A                 ;PORTCONTROL
        SJMP    EXITCON                 ;EXIT SUBROUTINE

SETTIME:CJNE    A,#TIME,SENDE2          ;CHECK USER WANT
        LCALL   XMIT                    ;TO SET TIME ?
        LCALL   XCEV                    ;NO.GOTO SENDE2
        MOV     R2,A                    ;R2 = HOUR(1)
        LCALL   XCEV
        MOV     R3,A                    ;R3 = MINUTE(1)
        LCALL   XCEV
        MOV     R4,A                    ;R4 = HOUR(2)
        LCALL   XCEV                    ;A = MINUTE(2)
        XRL     A,R3                    ;MINUTE(1) = MINUTE(2)
```

```
        JNZ     SENDE1                  ;NO.GOTO SENDE1
        MOV     A,#TIME
        LCALL   XMIT
        MOV     A,R4                    ;SET TIME
        MOV     HOUR,A                  ;SET HOUR
        MOV     A,R3
        MOV     MIN,A                   ;SET MINUTE
        SJMP    EXITCON                 ;EXIT SUBROUTINE

SENDE1: MOV     A,#ERROR                ;ERROR
        LCALL   XMIT
        SJMP    EXITCON

SENDE2: MOV     A,#ERROR                ;ERROR
        LCALL   XMIT
        LJMP    RMCTRL

;---------------------------------------
;       TABLE CONTROL PORT C
;---------------------------------------
CTRLPC: PUSH    ACC
        MOV     dptr,#ctrlnum
        MOVC    A,@A+DPTR
        MOV     @r0,A
        POP     ACC
        RET
CTRLNUM:DB      00000000B
        DB      00000100B
        DB      00001000B
        DB      00010000B
        DB      00100000B
        DB      01000000B
        DB      10000000B

EXITCON:POP     04H                     ;EXIT SUBROUTINE
        POP     03H
        POP     02H
        SETB    TBB
        SETB    SM2
        RET

;*****************************************
;       DISPLAY CLOCK
;*****************************************
DSPCLK: MOV     HEX,#00H
        MOV     HEX+1,HOUR
        LCALL   HTOD
        MOV     GENPUR,BCD+1
        MOV     HEX,#00H
        MOV     HEX+1,MIN
        LCALL   HTOD
        MOV     BCD,GENPUR
        LCALL   UNPACK
        LCALL   DTOSEG
        SETB    ENDOT
```

```
        LCALL   SCANK
        RET
;***************************************
;       DISPLAY SENSOR
;***************************************
DSPSEN: MOV     HEX,#00H
        MOV     HEX+1,SENNUM
        LCALL   HTOD
        LCALL   UNPACK
        LCALL   DTOSEG
        MOV     DSP,#6DH                ;CHARACTER 'S'
        MOV     DSP+1,#40H              ;CHARACTER  -'
        CLR     ENDOT
        LCALL   SCANK
        RET

;***************************************
;       HTOD
;***************************************
HTOD:   CLR     C
        MOV     BCD,#00H
        MOV     BCD+1,#00H
        MOV     R2,#10H
HTOD1:  MOV     R0,#HEX+1
        MOV     R3,#02H
HTOD2:  MOV     A,@R0
        RLC     A
        MOV     @R0,A
        DEC     R0
        DJNZ    R3,HTOD2
        MOV     R0,#BCD+1
        MOV     R3,#02H
HTOD3:  MOV     A,@R0
        ADDC    A,@R0
        DA      A
        MOV     @R0,A
        DEC     R0
        DJNZ    R3,HTOD3
        DJNZ    R2,HTOD1
        RET

;***************************************
;       UNPACK
;***************************************
UNPACK: MOV     R0,#BCD
        MOV     R1,#DSP
        MOV     R3,#02H
PACK1:  MOV     A,@R0
        ANL     A,#0F0H
        SWAP    A
        MOV     @R1,A
        INC     R1
        MOV     A,@R0
        ANL     A,#0FH
        MOV     @R1,A
```

```
        INC     R0
        INC     R1
        DJNZ    R5.PACK1
        RET

;*****************************************
;       DTOSEG
;*****************************************
DTOSEG: MOV     R4.#04H
        MOV     DPTR.#SEGCODE
        MOV     R1.#DSP
DTOSEG1:MOV     A.@R1
        MOVC    A.@A+DPTR
        MOV     @R1.A
        INC     R1
        DJNZ    R4.DTOSEG1
        RET
SEGCODE:DB      3FH.06H.5BH.4FH.66H.6DH.7DH.07H.7FH.6FH.6DH.40H

;*****************************************
;       SCANK
;*****************************************
SCANK:  MOV     R2.#04H
        MOV     R4.#00H
        MOV     R0.#DSP
SCANK1: LCALL   SCANS
        INC     R4
        DJNZ    R2.SCANK1
        RET
SCANS:  MOV     DPTR.#PORTC
        movx    a.@dptr
        anl     a.#11111100b
        orl     A.R4
        MOVX    @DPTR.A
        MOV     A.@R0
        CJNE    R4.#01H.UNDOT
        JNB     ENDOT.UNDOT
        MOV     BLDOT.SEC
        ANL     BLDOT.#01H              :DOT OPEN/SHUT
        MOV     R5.BLDOT
        CJNE    R5.#00H.UNDOT
DOT:    ORL     A.#80H
UNDOT:  MOV     P1.A
        LCALL   DELAY
        CLR     A
        MOV     P1.A
        INC     R0
        RET

DELAY:  MOV     R6.#05H
DELAY1: MOV     R7.#00H
DELAY2: DJNZ    R7.DELAY2
        DJNZ    R6.DELAY1
        RET
```

```
;##########################################
;       TIMER0 INTERRUPT
;##########################################
CLOCK:  PUSH    ACC
CHKMS:  MOV     A,TENMS
        CJNE    A,#00H,DE10MS
        MOV     TENMS,#100D
CHKSEC: MOV     A,SEC
        CJNE    A,#59D,INCSEC
        MOV     SEC,#00H
CHKMIN: MOV     A,MIN
        CJNE    A,#59D,INCMIN
        MOV     MIN,#00H
CHKHR:  MOV     A,HOUR
        CJNE    A,#23D,INCHR
        MOV     HOUR,#00H
        SJMP    EXITCLK
DE10MS: DEC     A
        MOV     TENMS,A
        SJMP    EXITCLK
INCSEC: INC     A
        MOV     SEC,A
        SJMP    EXITCLK
INCMIN: INC     A
        MOV     MIN,A
        SJMP    EXITCLK
INCHR:  INC     A
        MOV     HOUR,A
EXITCLK:POP     ACC
        LCALL   STTIME
        RETI

;---------------------------------------
;       SET TIMER0
;---------------------------------------
STTIME: MOV     TL0,#00H
        MOV     TH0,#0DCH
        ANL     TMOD,#0F0H
        ORL     TMOD,#01H
        SETB    TR0
        RET

;---------------------------------------
;       XMIT SUBROUTINE
;---------------------------------------
XMIT:   MOV     SBUF,A
WAIT:   JNB     TI,WAIT
        CLR     TI
        RET

;---------------------------------------
;       XCEV    SUBROUTINE
;---------------------------------------
XCEV:   JNB     RI,XCEV
        MOV     A,SBUF
```

```
CLR     R1
RET

END
```

SOFTWARE ของบอร์ด microcontroller

ระดับชั้น

```
;#########################################
;#   program control class No.1          #
;#                                       #
;#                                       #
;#   written by                          #
;#                                       #
;#   kalo basisangsuporn                 #
;#   Electronics Department              #
;#   KMIT L                              #
;#                                       #
;#########################################
       portA   equ     0e0e0h          ;control
       portb   equ     0e0e1h          ;control rs-422
       portC   equ     0e0e2h          ;control 8251
       ctriw   equ     0e0e3h
       de_lay  equ     0eeh            ;set delay 1 ms
       nclass  equ     01h             ;class No.
       error   equ     0eeh            ;error happen
;---------------------------------------
;-          internal ram               -
;---------------------------------------
       ramin   equ     60h
       data0   equ     ramin           ;keep data for compare
       data1   equ     ramin+1
       data2   equ     ramin+2
       data3   equ     ramin+3
       data4   equ     ramin+4
       data5   equ     ramin+5
       ;---------
       ;roomsen -
       ;---------
       rmno.   equ     ramin+6         ;keep data
       rmevt   equ     ramin+7         ;keep event
       datadc  equ     ramin+8         ;keep data from pxcev
;---------------------------------------
       org     0000h
       LCALL   DELAY
       LCALL   DELAY
       sjmp    start


start: mov     r7,#00h
       djnz    r7,$
       djnz    r7,$
       djnz    r7,$
       mov     scon,#0f8h              ;set serial port mode 3.sm2=1
       mov     th1,#0fah               ;baudrate = 4800
       anl     pcon,#7fh               ;smod=0
       orl     tmod,#20h
       setb    tr1
       LCALL   SET51

revl:  LCALL   READ
       mov     p1,#00h
```

```
rev:    nop
        setb    tb8
        setb    sm2
        jb      ri,ckroom               :check receive data from room
        mov     dptr,#port1             :check receive data from pc

revi:   MOVX    A,@DPTR
        ANL     A,#01000000B
        CJNE    A,#01000000B,REV
        lcall   pxcev
        NOP
        lcall   pxcev
        NOP
        lcall   pxcev
        LCALL   DELAY
        LCALL   DELAY
        mov     a,datapc

pcint:  cjne    a,#nclass,rev
        lcall   ctribak                 :goto ctribak
        Ljmp    revi

ckroom: clr     tb8
        clr     sm2
        mov     a,sbuf
        clr     ri
        cjne    a,#0f0h,rmerr           :check data from room
        lcall   sxmit                   :send f0h return to data complete
        lcall   roomsen                 :goto roomsen
        lcall   delay
        lcall   delay
        Ljmp    revi

rmerr:  mov     a,#0eeh
        lcall   sxmit
        Ljmp    rev                     :send error

:*****************************************
:*     subroutine roomsen                *
:*     class receive from room           *
:*     send to pc                        *
:* rmno. - keep room No.                 *
:* rmevt - keep event-byte               *
:*****************************************
roomsen:push    acc
        push    dph
        push    dpl
        clr     tb8                     :tb8 = 0
        clr     sm2                     :sm2 = 0
        lcall   sxcev
        mov     rmno.,a                 :rmno. keep room no.
        lcall   sxmit
        lcall   sxcev
        cjne    a,#0eeh,happen          :check error
        ajmp    exitsen
```

```
nabben: lcall   sxcev                   ;receive event-byte and
        mov     rmno..a                 ;room No.
        lcall   sxcev                   ;r1 keep event-byte
        mov     rmevt.a                 ;r0 keep room No.
        lcall   sxmit                   ;send event byte
        lcall   sxcev                   ;receive flag
        cine    a.#0eeh.combc           ;data = eeh?
        ajmp    exitsen

;------- start communicate with pc -----
combc:  lcall   E_422                   ;enable rs-422
        lcall   write
        lcall   delay

combc1: mov     a.#0ffh                 ;send ffh to pc
        lcall   pxmit
        NOP
        NOP

combc3: mov     a.#',
        lcall   pxmit
        mov     a.#nclass               ;###send class No.
        lcall   pxmit

combc3: NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        lcall   read
        lcall   pxcev                   ;receive flag from pc
        mov     a.datapc                ;r3 keep data from pc
        lcall   pxcev
        mov     a.datapc
        lcall   pxcev                   ;receive flag from pc
        mov     a.datapc                ;r3 keep data from pc
        lcall   pxcev
        mov     a.datapc
        lcall   pxcev                   ;receive flag from pc
        mov     a.datapc                ;r3 keep data from pc
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay

        xri     a.#nclass               ;###nclass
        jz      sendpc                  ;RECEIVED DATA = NCLASS?
        ljmp    nopeu
sendpc: lcall   write
```

```
        mov     a,#'.
        lcall   oxmit
        nop
        nop
        nop
        mov     a,#0ffh
        lcall   oxmit
        nop
senoocl:nop
        nop
        mov     a,#'.
        lcall   oxmit
        nop
        nop
        nop
        mov     a,rmno.                 :send(1) room No. to pc
        lcall   oxmit
        nop
        nop
        nop
        nop
        nop
        mov     r7,#00h
        dinz    R7,$
        mov     a,rmevt                 :send(1) event-byte to pc
        lcall   oxmit
        nop
        nop
        nop
        nop
        nop
        dinz    r7,$
        mov     a,rmno.                 :send(2) room No. to pc
        lcall   oxmit
        nop
        nop
        nop
        nop
        nop
        djnz    r7,$
        mov     a,rmevt                 :send(2) event-byte to pc
        lcall   oxmit

exitsen:lcall   D_422                   :disable rs-422
        setb    tb8
        setb    sm2
        ooo     odi
        ooo     odh
        ooo     acc
        ret

noeou:  lcall   write
        mov     a,# .
        lcall   oxmit
        mov     a,#0eeh                 :error happen send eeh to pc
```

```
        lcall   oxmit
        ljmp    combc3                  ;try receive data again

;*****************************************
;*      subroutine-ctrlbak               *
;*      class and room is controled      *
;*      by pc                            *
;*****************************************
ctrlbak:push    acc
        push    dph
        push    dpl

;-------- class communicate with pc ----

        lcall   E_422                   ;enable rs-422
        lcall   write
        mov     a,#','                  ;send class No. to pc
        lcall   oxmit
        mov     r7,#00h
        djnz    r7,$
        mov     a,#nclass               ;send class No. to pc
        lcall   oxmit
        djnz    r7,$
        mov     a,#nclass
        lcall   oxmit
        lcall   read
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   oxcev
        nop
        nop
        nop
        nop
        mov     a,datapc
        cjne    a,#0eeh,ck07            ;check flag ?
        sjmp    exit1

ck07:   cjne    a,#07h,ck18
        lcall   ctclass                 ;pc want control class
        sjmp    exit1

ck18:   cjne    a,#0f8h,ck8f
        lcall   ctroom                  ;pc want control room
        sjmp    exit1

ck8f:   cjne    a,#8fh,unall
        lcall   stime                   ;pc want set room s time
```

```
        sjmp    exit1

unail:  mov     a,#0eeh                 ;incorrect code try again
        lcall   dxmit
        ljmp    ck07

exit1:  lcall   d_422
        setb    tb8
        setb    sm2
        pop     dpl
        pop     dph
        pop     acc
        ret

;---------------------------------------
; subroutine - ctclass                  -
; control class                         -
;---------------------------------------
ctclass:lcall   delay
        lcall   write
        mov     a,#',                   ;
        lcall   dxmit

cclass: lcall   read
        lcall   dxcev
        nop
        nop
        nop
        nop
        lcall   dxcev
        nop
        nop
        nop
        nop
        lcall   dxcev
        nop
        nop
        nop
        nop
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        mov     r1,#datadc
        lcall   ctrldA
        mov     dptr,#portA             ;control to portA
        mov     a,datadc
        movx    @dptr,a
        lcall   delay
        ret

ctrldA: push    acc
        mov     a,@r1
        mov     dptr,#concode
        movc    a,@a+dptr
```

```
        mov     @r1,a
        pop     acc
        ret
concode:db      00000000b
        db      00000001b
        db      00000010b
        db      00000100b
        db      00001000b
        db      00010000b
        db      00100000b
        db      01000000b
        db      10000000b

noteou: lcall   write
        mov     a,#'.
        lcall   pxmit
        nop
        nop
        mov     a,#0eeh                 ;error happen send een to pc
        lcall   pxmit
        ljmp    pclass                  ;try receive data again

;---------------------------------------
;  subroutine - control room            -
;                                       -
;  data0 = room No.                     -
;  data1 = control point                -
;---------------------------------------
ctroom: lcall   delay
        lcall   write
        mov     a,#'.
        lcall   pxmit

droom:  lcall   read
        lcall   pxcev
        nop
        nop
        nop
        nop
        lcall   pxcev
        nop
        nop
        nop
        nop
        lcall   pxcev
        nop
        nop
        nop
        nop
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        mov     data0,datapc            ;data0 = room No.
        lcall   write
```

```
        mov     a,#'.'
        lcall   sxmit
        lcall   read
        lcall   sxcev
        nop
        nop
        nop
        nop
        lcall   sxcev
        nop
        nop
        nop
        nop
        lcall   sxcev
        nop
        nop
        nop
        nop
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        mov     data1,datadc            ;data1 = control point
        lcall   D_422
                                        ;*****************************
                                        ;   link with room
                                        ;*****************************

sroom:  setb    tb8
        mov     a,data0
        lcall   sxmit                   ;send room No.
        clr     tb8
        clr     sm2
        lcall   sxcev
        cjne    a,#0ffh,sroom           ;flag correct?
        mov     a,#0f8h                 ;send f8h to room
        lcall   sxmit
        lcall   sxcev
        cjne    a,#0eeh,sdataR          ;error happen try again
        sjmp    sroom

sdataR: mov     a,data0                 ;send room No.(1)
        lcall   sxmit
        mov     a,data1                 ;send event-byte(1)
        lcall   sxmit
        mov     a,data0                 ;send room No.(2)
        lcall   sxmit
        mov     a,data1                 ;send event-byte(2)
        lcall   sxmit
        lcall   sxcev
        cjne    a,#0eeh,outrm
        sjmp    sroom

uneouR: mov     a,#0eeh
        lcall   sxmit
```

```
        ljmp    oroom

outrm:  ret

;---------------------------------------
;  subroutine - set time in room        -
;---------------------------------------
stime:  lcall   delay
        lcall   write
        mov     a,#'.
        lcall   oxmit

dtime:  lcall   read
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        mov     data0,datadc            ;data0 = room No.
        lcall   write
        mov     a,#','
        lcall   oxmit
        lcall   read
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   oxcev
        nop
        nop
        nop
        nop
        lcall   delay
        lcall   delay
        lcall   delay
```

```
        lcall   delay
        mov     data1,datapc            ;data1 = new hour
        lcall   write
        mov     a,# .
        lcall   sxmit
        lcall   reao
        lcall   sxcev
        nop
        nop
        nop
        nop
        lcall   sxcev
        nop
        nop
        nop
        nop
        lcall   sxcev
        nop
        nop
        nop
        nop
        lcall   delay
        lcall   delay
        lcall   delay
        lcall   delay
        mov     data2,datapc            ;data2 = new minute
        lcall   D_422
                                        ;************************
                                        ;    set time room
Troom:  setb    tb8                     ;
        mov     a,data0                 ;************************
        lcall   sxmit                   ;send room No.
        clr     tb8
        clr     sm2
        lcall   sxcev
        cjne    a,#0ffh,Troom           ;flag correct?
        mov     a,#08Fh                 ;send 8Fh to room
        lcall   sxmit
        lcall   sxcev                   ;receive 8fh
        cjne    a,#0eeh,sdataT
        sjmp    Troom

sdataT: mov     a,data1                 ;send hour(1)
        lcall   sxmit
        mov     a,data2                 ;send minute(1)
        lcall   sxmit
        mov     a,data1                 ;send hour(2)
        lcall   sxmit
        mov     a,data2                 ;send minute(2)
        lcall   sxmit
        lcall   sxcev
        cjne    a,#0eeh,outt
        sjmp    Troom

uneouT: mov     a,#0eeh
```

```
        lcall   dxmit
        ljmp    dtime

outt:   ret

;---------------------------------------
; subroutine - E_422                    -
; enable rs-422                         -
; output - bit6 of portB                -
;---------------------------------------
E_422:  push    dph
        push    dpl
        push    acc
        mov     dptr,#portB
        mov     a,#80h
        movx    @dptr,a
        pop     acc
        pop     dpl
        pop     dph
        ret

;---------------------------------------
; subroutine - D_422                    -
; disable rs-422                        -
; output - bit6 of portB                -
;---------------------------------------
D_422:  push    dph
        push    dpl
        push    acc
        mov     dptr,#portB
        mov     a,#00h
        movx    @dptr,a
        pop     acc
        pop     dpl
        pop     dph
        ret

;---------------------------------------
; subroutine - dxmit                    -
; transmit data pass 8251               -
; input -  a                            -
; output - a                            -
;---------------------------------------
dxmit:  push    acc
        mov     dptr,#portc
dwait:  movx    a,@dptr
        anl     a,#10h
        cjne    a,#10h,dwait
        pop     acc
        mov     b1,a
        mov     b1,a
        mov     b1,a
        mov     b1,a
        mov     a,#08h                  ;set write
        movx    @dptr,a
```

```
        mov     a,#09h                  ;clear write
        movx    @dptr,a
        mov     dl,#0ffh
        ret

;---------------------------------------
; subroutine - oxcev                    -
; receive data pass 8251                -
; input - a                             -
; output - dataoc                       -
;---------------------------------------
oxcev:  push    doh
        push    dol
        mov     dptr,#portc

RxRDY:  movx    a,@dptr
        anl     a,#01000000b
        cjne    a,#01000000B,RxRDY
        mov     a,#01h                  ;set rd
        movx    @dptr,a
        nop
        nop
        nop
        mov     a,#00h
        nop
        mov     a,dl
        mov     dataoc,a
        mov     a,#09h                  ;clear rd to normal
        movx    @dptr,a
        nop
        nop
        mov     dl,#0ffh
        pop     dol
        pop     doh
        ret

;---------------------------------------
; subroutine - sxmit                    -
; transmit data pass serial port        -
; input - a                             -
; output - a                            -
;---------------------------------------
sxmit:  mov     sbuf,a
swait:  jnb     ti,swait
        clr     ti
        ret

;---------------------------------------
; subroutine - sxcev                    -
; receive data pass serial port         -
; input - a                             -
; output - a                            -
;---------------------------------------
sxcev:  jnb     ri,sxcev
        mov     a,sbuf
```

```
        clr     ri
        ret

;---------------------------------------
;       set 8251
;---------------------------------------
set51:  push    dph
        push    dpl
        mov     a,#88h                  ;set port 8255 portC(0-3)o\p (4-7)i\p
        mov     dptr,#ctrlw
        movx    @dptr,a                 ;set portA,B output  portC in/out
        mov     dptr,#portc
        mov     a,#0dh                  ;set control mode (pc.2 = c\!d)
        movx    @dptr,a
        mov     dl,#0ffh
        mov     dl,#04FH                ;set asynchronous mode
        mov     a,#0ch                  ;set   write
        movx    @dptr,a
        nop
        mov     a,#0dh
        movx    @dptr,a
        nop
        pop     dpl
        pop     dph
        ret

;---------------------------------------
;       set write
;---------------------------------------
wrtie:  push    acc
        mov     dptr,#portc
        mov     dl,#0ffh
        mov     dl,#33h         ;set  command
        mov     a,#0ch          ;set write
        movx    @dptr,a
        nop
        movx    a,#0dh
        movx    @dptr,a
        mov     a,#01h          ;read status
        movx    @dptr,a
        nop
        mov     a,#09h          ;set data mode
        movx    @dptr,a
        nop
        nop
        pop     acc
        ret

;---------------------------------------
;       set read
;---------------------------------------
read:   push    acc
        mov     dptr,#portc
        mov     dl,#0ffh
        mov     dl,#36h         ;set command
```

```
        mov     a,#0cn          ;set write
        movx    @dptr,a
        nop
        mov     a,#00h
        movx    @dptr,a
        mov     a,#01h          ;read status
        movx    @dptr,a
        nop
        mov     a,#09h          ;set data mode
        movx    @dptr,a
        nop
        nop
        pop     acc
        ret

;--------------------------------
;                               -
;       delay                   -
;                               -
;--------------------------------
delay:  push    06h
        push    07h
        mov     r6,#30h
delay1: mov     r7,#00h
delay2: djnz    r7,delay2
        djnz    r6,delay1
        pop     07h
        pop     06h
        ret

        end
```

ภาคผนวก

ข.

SOFTWARE การทำงานใน PC

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <graphics.h>
#include <bios.h>
#include <stdlib.h>
#include <string.h>
#define PORT 1
#define F1 59
#define F2 60
#define F3 61
#define F4 62
#define ESC 27

void building();
void port_init(), sport();
void get_data(),alarm();
void send_data(),showdata(),save_option();
void rec_to_file(),rec_to_prnt();
void rec_opt(),prn_opt(),wait();
void save_video(),restore_video();
void write_video();
void write_string(),write_char();
void display_menu(),draw_border();
void read_option(),write_data();
void get_s(),file_name(),save_file();
void delete(),set_time();

int rec_var = 0;
int prn_var = 0;
char data[5];
char filename[20];
main()
{
      char *dat[2],a[2][10];
      unsigned char x;
      unsigned char y;
      union inkey {
        char ch[2];
        int i;
      } c;
      int k;

      building();
      for(k = 0;k < 2;k++)
         dat[k] = a[k];
      port_init(PORT,195);
      do {
         if(bioskey(1)) {
            c.i = bioskey(0);
            if(c.ch[1] == F1) {
               closegraph();
               menu_pro();
               main();
            }
```

```
            if(c.ch[0] == 27) {
               closegraph();
               exit(1);
            }
         }
      } while (rportw(PORT) != 255);
      get_data(PORT,data);
      alarm(data);
      itoa(data[0],dat[0],10);
      itoa(data[1],dat[1],10);
      file_name(dat,filename);
      show_event(data);
      main();
}
void building()
{
      int gmode;
      int gdriver = DETECT;
      int x1,y1;
      int a1,b1;
      int lx,ly;
      int i;

      initgraph(&gdriver,&gmode,"");
      x1 = getmaxx();
      y1 = getmaxy();
      rectangle(x1/2,y1/3,(3*x1)/4,y1-10);
      line(x1/2,y1/3,(x1/2)+50,(y1/3)-50);
      line((x1/2)+50,(y1/3)-50,(3*x1)/4+50,(y1/3)-50);
      line((3*x1)/4+50,(y1/3)-50,(3*x1)/4,y1/3);
      line((3*x1)/4,y1-10,(3*x1)/4+50,(y1-10)-50);
      line((3*x1)/4+50,(y1-10)-50,(3*x1)/4+50,(y1/3)-50);
      a1 = x1/2;
      b1 = y1/3;
      lx = (3*x1)/4-x1/2;
      ly = (y1-10)-y1/3;
      for(i = 1;i < 20;i++)
         line(a1,b1+(i*ly)/20,a1+lx,b1+(i*ly)/20);
}
void port_init(port,code)
int port;
unsigned char code;
{
      union REGS r;

      r.x.dx = port;
      r.h.ah = 0;
      r.h.al = code;
      int86(0x14,&r,&r);
}

void get_data(port,dta)
int port;
char dta[];
{
```

```
    union REGS r;
    unsigned char x;
    int i,j,y;

    y = 1;
    do {
       wait(port);
       dta[0] = rport(PORT);
       delay(30);
       for(i = 0;i < 5;i++)
          sport(port,dta[0]);
       wait(port);
       x = rport(port);
       if(x == 0xee)
          y = 0;
    } while(y == 0);
    wait(port);
    y = 1;
    do  {
        for (j = 1;j <= 4; j++) {
           dta[j] = rport(port);
        }
        if (dta[1] != dta[3] || dta[2] != dta[4])  {
            for(i = 0;i < 5;i++)
               sport(port,0xee);
            y = 0;
            wait(port);
        }
    } while (y == 0);
}
void alarm(dta)
char dta[];
{
      int x,y;
      int ly;
      int dy;
      int a,b;
      int class;

      x = getmaxx();
      y = getmaxy();
      ly = (y-10)-y/3;
      a = x/2;
      b = y-10;
      dy = ly/20;

      (int) class = dta[0];
      settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
      do {
          setfillstyle(SOLID_FILL,RED);
          floodfill(a+2,b-(class*dy)+2,WHITE);
          setcolor(WHITE);
          outtextxy(65,y-90,"press any key ");
          outtextxy(10,y-40,"for see emergency event !");
          sound(800);
```

```
            delay(400);
            setfillstyle(HATCH_FILL,BLACK);
            floodfill(a+2,b-(classtdv)+2,WHITE);
            setcolor(BLACK);
            outtextxy(65,y-90,"press any key ");
            outtextxy(10,y-40,"for see emergency event !");
            nosound();
            delay(400);
         } while(!kbhit());
         closegraph();
}
void send_data(dta,format)
unsigned char dta[];
int format;
{
        int i,j,k,l;
        int x;
        char code;
        char a[2];
        int byte;
        unsigned char y;

        x = 1;
        do {
           for(i = 0;i < 3;i++)
              sport(PORT,dta[0]);
           wait(PORT);
           for(i = 0;i < 2;i++)
              a[i] = rport(PORT);
           delay(15);
           if (a[1]  != dta[0])  {
                 for(j = 0;j < 3;j++)
                    sport(PORT,0xee);
                 x = 0;
           }
        } while(x==0);

        switch(format) {
        case 1:
           code = 0x07;
           byte = 2;
           break;
        case 2:
           code = 0xf8;
           byte = 3;
           break;
        case 3:
            code = 0x8f;
                byte = 4;
            break;
        }
        delay(15);
        for(i = 0;i < 3;i++)
           sport(PORT,code);
        delay(20);
```

```
        wait(PORT);
        delay(15);
        for(i = 0;i < 5;i++)
           sport(PORT,dta[1]);
        if(byte == 3) {
           delay(20);
           wait(PORT);
           for(i = 0;i < 5;i++)
              sport(PORT,dta[2]);
        }
        if(byte == 4) {
           delay(10);
           for(i = 0;i < 2;i++) {
              wait(PORT);
              for(j = 0;j < 5;j++)
                 sport(PORT,dta[i+2]);
           }
        }
        delay(10);
    }
    rportw(port)
    int port;
    {
        union REGS r;
        int i;

        i = 0;
        while(!(check_stat(PORT)&256) &&(++i != 50))
             ;;
        if(i != 50) {
            r.x.dx = port;
            r.h.ah = 2;
            int86(0x14,&r,&r);
            if(r.h.ah & 128)
            printf("read error detection in serial port");
            return r.h.al;
        }
        else
            return 1;
    }
    rport(port)
    int port;
    {
        union REGS r;

        while(!(check_stat(PORT)&256) )
        ;;
        r.x.dx = port;
        r.h.ah = 2;
        int86(0x14,&r,&r);
        if(r.h.ah & 128)
            printf("read error detection in serial port");
        return r.h.al;
    }
    void sport(port,c)
```

```
int port;
char c;
{
     union REGS r;

     r.x.dx = port;
     r.h.al = c;
     r.h.ah = 1;
     int86(0x14,&r,&r);
     if (r.h.ah & 128)  {
        printf("Send error detected in serial port ");
        exit(1);
     }
}
check_stat(port)
int port;
{
    union REGS r;

    r.x.dx = port;
    r.h.ah = 3;
    int86(0x14,&r,&r);
    return r.x.ax;

}
void wait(port)
int port;
{
       while(rport(port) != ',')
           ;;
}
void rec_to_file(dt)
char dt[];
{
      FILE *fp;
      struct time now;
      int yr;char d_date[2];
      struct date today;
      char *event[3] = {"FIRE","ROBBER","GAS"};
      unsigned char d_ti[3];
      int ev;
      int i;

      if((fp = fopen("event.dat","at")) == NULL) {
          fprintf(stderr,"Cannot open file %s\n","event.dat");
          exit(1);
      }
      getdate(&today);
      yr = today.da_year;
      d_date[0] = today.da_day;
      d_date[1] = today.da_mon;
      fprintf(fp,"\n   DATE: ");
      for(i = 0;i < 2;i++)
         fprintf(fp,"%d/",d_date[i]);
      fprintf(fp,"%d",yr);
```

```
        gettime(&now);
        d_ti[0] = now.ti_hour;
        d_ti[1] = now.ti_min;
        d_ti[2] = now.ti_sec;
        fprintf(fp,"   TIME");
        for(i = 0;i < 3;i++)
           fprintf(fp,":%d",d_ti[i]);
        ev = (int) dt[2];
        fprintf(fp,"\nEVENT : %s\n",event[ev-2]);
        fprintf(fp,"CLASS : %d ",(int) dt[0]);
        fprintf(fp,"ROOM : %d\n",(int) dt[1]);
        fclose(fp);
}
void rec_to_prnt(dt)
char dt[];
{
        struct time now;
        char d_ti[3];
        struct date today;
        int yr;char d_date[2];
        char *event[3] = {"FIRE","ROBBER","GAS"};
        int ev;
        int i;

        getdate(&today);
        yr = today.da_year;
        d_date[0] = today.da_day;
        d_date[1] = today.da_mon;
        fprintf(stdprn,"\n   DATE: ");
        for(i = 0;i < 2;i++)
           fprintf(stdprn,"%d/",d_date[i]);
        fprintf(stdprn,"%d",yr);
        gettime(&now);
        d_ti[0] = now.ti_hour;
        d_ti[1] = now.ti_min;
        d_ti[2] = now.ti_sec;
        fprintf(stdprn,"TIME:");
        for(i = 0;i < 3;i++)
           fprintf(stdprn,":%d",d_ti[i]);
        ev = (int) dt[2];
        fprintf(stdprn,"\nEVENT : %s\n",event[ev-2]);
        fprintf(stdprn,"CLASS : %d ",(int) dt[0]);
        fprintf(stdprn,"ROOM : %d\n",(int) dt[1]);
}
show_event(dta)
char dta[];
{
        char *option_choice[3],opt[3][20];
        FILE *fi;
        int maxx,maxy;
        int half_x,half_y;
        int x1,y1,x2,y2,x3,y3;
        int dx,dy;
        int dx1,dy1;
        int i,j,k,m,n,o;
```

```
  char dt[50];
  int xv[][2] = {{10,10},{10,30},{10,50},{10,70},{10,90},
      {10,110},{10,130},{10,150},{10,170},{10,190},{10,210},
      {10,230} };
  int x = 100,y = 200;
  int room = 0 ;
  int event = 0;
  char *dat[2],a[2][10];

  char tree[30][80] = {
"                           a  ",
"                           a  ",
"                           aa ",
"      a                    aaaaa",
"       a                  aaaaaaa",
"       a                  aaaaaaaa",
"      aa                  aaaaaaa",
"      aaa                  aaaaaaa",
"     aaaa                   aaaaaaa                     a",
"      aaaaa                 aaaaaaaa                    aa",
"    aaaaaa                 aaaaaaaaa                    aa",
"    aaaaaaaa              aaaaaaaaaa                    aaa",
"  aaaaaaaaa                aaaaaaaaaa                  aaaa ",
"    aaaaaaaa             aaaaaaaaa                  aaaa ",
"   aaaaaaaaa             aaaaaaaaaaa                aaaaa ",
"  aaaaaaaaaaaa           aaaaaaaaaaaaa              aaaaaa   ",
"   aaaaaaaaaaaaa           aaaaaaaaaaaaaa           aaaaaa",
"    aaaaaaaaaaaa           aaaaaaaaaaaaaaa          aaaaaaa",
"    aaaaaaaaaaaaa        aaaaaaaaaaaaaaaa          aaaaaaaa",
"   aaaaaaaaaaaaaaaa     aaaaaaaaaaaaaaaaaa       aaaaaaaaa",
"   aaaaaaaaaaaaaaaa     aaaaaaaaaaaaaaaaaaa    aaaaaaaaaaaaa",
"    aaaaaaaaaaaaaaaa   aaaaaaaaaaaaaaaaaaa   aaaaaaaaaaaaaaaa ",
"   aaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa  aaaaaaaaaaaaaaaaaa",
"    aaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaa  aaaaaaaaaaaaaaaaaaaaa",
"  aaaaaaaaaaaaaaaaaa  aaaaaaaaaaaaaaaaaaa   aaaaaaaaaaaaaaaaaaaaa",
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"};

char gas[40][55] = {
"        aa               aa            aa",
"        aa               aa            aa",
"        aa               aa            aa",
"       aa               aa            aa",
"      aa               aa            aa",
"      aa              aa            aa",
"    aa               aa            aa",
"    aa              aa            aa",
"   aa              aa            aa",
"   aa             aa            aa",
"   aa            aa            aa",
"   aa           aa            aa",
"   aa          aa            aa",
"   aa         aa            aa",
"    aa        aa           aa",
"     aa       aa           aa",
```

```
      ää        ää         ää".
       ää        ää        ää".
       ää        ää        ää",
       ää        ää        ää".
        ää      ää          ää",
         ää    ää         äää".
          ää    ää       ää",
          ää    ää       ää".
           ää     ää      ää".
          ää     ää      ää".
         ää     ää      ää".
        ää     ää      ää".
       ää     ää      ää",
      ää     ää      ää",
     ää     ää      ää".
    ää     ää      ää",
   ää     ää      ää",
   ää  ää    ää",
   ää ää  äää",
   ääää äää"};
char rob[25][70] = {
                  ääääääääää",
                ääääääääääääääääää",
              ääääääääääääääääääääääää",
            ääääääääääääääääääääääääääää  ",
          ääääääääääääääääääääääääääääääää ",
        ääääääääää      ää     ääääääääääää    ",
       äääääääää                  ääääääääää   ",
     ääääääääää   äääääää     ääääää  äääääääää   ",
     äääääääää          ä ä          äääääää   ",
     ääääää       äää    ä  ä   äää       ääääää  ",
       ää         äää   ä    ä  äää          ää  ",
       ää           ä      ä                ää  ",
       ää           ä       ä               ää  ",
        ää        äääääääää                 ää      ",
        ää       äääääää   äääääääää        ää      ",
         ää    äääääääää   äääääääääää     ää",
          ää  ää  äääääääääääääääääää  ää    ää",
           ää       ä             ä         ää",
             ää      äääääääääääää        ää".
               ää                        ää",
                 ää                    ää ",
                   ää äääääääääääää  ää",
                      äääääääääääää ",
                         äääääää",
                         äää  äää"};
char ctrl_bk[6][30] = {"  control class level",
                       "F1:on alarm at bath",
           "F2:close the door ",
                       " control room level",
                       "F3:on alarm in room ",
```

```
          "F4:close the door "};

        union inkey {
         char ch[2];
         int i;
        } c;

        int gmode,gdriver = DETECT;

        for(i = 0;i < 3;i++)
           option_choice[i] = opt[i];
        read_option(option_choice);
        if(rec_var == 1)
           rec_to_file(dta);
        if(prn_var == 1);
           rec_to_prnt(dta);
        (int) room = dta[1];
        (int) event = dta[2];

        initgraph(&gdriver,&gmode,"");
        maxx = getmaxx();
        maxy = getmaxy();
        half_x = maxx/2;
        half_y = maxy/2;

        /* begin section bottom right */
        x1 = maxx/2+5;
        y1 = 2*maxy/5+20;
        x2 = maxx-5;
        y2 = maxy-5;
        dx = x2-x1;
        dy = y2-y1;
        setviewport(x1,y1,x2,y2,0);
        rectangle(0,0,dx,dy);
        rectangle(0,dy/2-10,dx,dy/2+10);
        rectangle(20,dy/2-10,20,dy/2+10);
        line(dx/2.0,dx/2,dy/2-10);
        line(dx/2,dy/2+10,dx/2,dy);

        settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
        outtextxy(10,15,"1");
        outtextxy(dx/2+10,15,"2");
        outtextxy(10,dy/2+25,"3");
        outtextxy(dx/2+10,dy/2+25,"4");

        x3 = x1+5;    /* value for send to fn. blink */
        y3 = y1+5;
        dx1 = dx;
        dy1 = dy;
        /* end section bottom right */
        /* begin section bottom left */
        x1 = 5;
        y1 = half_y+25;
        x2 = half_x-5;
        y2 = maxy-5;
```

```
dx = x2-x1;
dy = y2-y1;

setviewport(x1,y1,x2,y2,0);
rectangle(0.0,dx,dy);
line(0.dy/2,dx,dy/2,);
switch(room)  {
case 1:
    line(dx/2.0.dx/2.dy/2);
    line(dx/3.dy/2.dx/3,dy);
    outtextxy(10,10,"1");
    rectangle(dx/2-15,dy-5,dx/2+15,dy);
    break;
case 2:
    line(dx/2.0,dx/2,dy/2);
    line(.2*dx)/3,dy/2,(2*dx)/3,dy);
    outtextxy(10,10,"2");
        rectangle(dx/2-15,dy-5,dx/2+15,dy);
    break;
case 3:
    line(dx/2,dy/2,dx/2,dy);
    line(dx/3,0,dx/3,dy/2);
    outtextxy(10,10,"3");
    rectangle(dx/2-15,0,dx/2+15,5);
    break;
case 4:
    line(dx/2.dy/2,dx/2,dy);
    line((2*dx)/3,0,(2*dx)/3,dy/2);
    outtextxy(10.10,"4");
    rectangle(dx/2-15,0,dx/2+15,5);
    break;
}
setviewport(0.0,0,0,0);
/* end section bottom left */

/* begin section top left */
x1 = 5;
y1 = 5;
x2 = half_x-5;
y2 = half_y+15;
rectangle(x1,y1,x2,y2);
if((fi = fopen(filename,"rt"))== NULL)  {
   fprintf(stderr,"Can't open file %s \n ",filename);
   exit(1);
}
settextstyle(SMALL_FONT,HORIZ_DIR,4);
i = 0;
while(fgets(dt,sizeof(dt),fi),dt[0] != '/')  {
     outtextxy(x1+xy[i++][0],y1+xy[i][1],dt);
}
rectangle(maxx/4+10,5,half_x-5,150);
for(i = 0;i < 6;i++)
   outtextxy(maxx/4+15,i*25+10,ctrl_bk[i]);

/* end section top left */
```

```
/* begin section top right */
x1 = half_x+5;
y1 = 5;
x2 = maxx-5;
y2 = maxy/2-40;
rectangle(x1,y1,x2,y2);
setviewport(x1,y1,x2,y2,0);
settextstyle(SANS_SERIF_FONT,HORIZ_DIR,6);
setcolor(YELLOW);
circle(250,60,40);
switch(event)  {
   case 2:
     y = 40;
     for(i = 0;i < 35;i++) {
        x = 215;
        for (k = 0;k < sizeof(tree[i]);k++) {
          if (tree[i][k] == 'a')
             putpixel(x,y,RED);
          x++;
        }
        y++;
     }
     setcolor(RED);
     circle(250,50,36);
     setfillstyle(SOLID_FILL,RED);
     floodfill(250,80,RED);
     setcolor(YELLOW);
     circle(250,50,38);
     outtextxy(20,y2/2,"FIRING !");
     break;
  case 3:
     y = 40;
     for(i = 0;i < 35;i++) {
        x = 225;
        for(k = 0;k < sizeof(rob[i]);k++) {
           if(rob[i][k] == 'a')
              putpixel(x,y,WHITE);
           x++;
        }
        y++;
     }
     outtextxy(20,y2/2,"ROBBERING!");
     break;
case 4:
     y = 30;
     for(i = 0;i < 40;i++) {
        x = 235;
        for(k = 0;k < sizeof(gas[i]);k++) {
           if(gas[i][k] ==  a')
              putpixel(x,y,YELLOW);
           x++;
        }
        y++;
     }
     outtextxy(20,y2/2," GAS '");
```

```
            break;
        default:
            exit(1);
        }
        /* end section top right */
        /* start to show blink */
        m = room;
        n = 0;
        if (m <= 2)
        m = m-1;
        if (m > 2)  {
        m = m-3;
        n = 1;
        }
        setviewport(0,0,0,0,0);
        do{
           blink(m,n,x3,y3,dx1,dy1);
           if(bioskey(1)) {
             c.i = bioskey(0);
             if(c.ch[1] == F1) {
               data[2] = 1;
               send_data(data);
             }
             if(c.ch[1] == F2) {
                data[2] = 2;
                send_data(data);
             }
             if(c.ch[0] == ESC)
                return 1;
           }
        } while(roortw(PORT) != 255);

         for(k = 0;k < 2;k++)
           dat[k] = a[k];
         get_data(PORT,data);
         alarm(data);
         itoa(data[0],dat[0],10);
         itoa(data[1],dat[1],10);
         file_name(dat,filename);
         show_event(data);
         main();
        /* end blink */
  }
  blink(a,b,x,y,dx2,dy2)
  int a,b,x,y,dx2,dy2;
  {
        setfillstyle(SOLID_FILL,RED);
        floodfill(x+a*(dx2/2)+10,y+b*(dy2/2)+10,WHITE);
        delay(400);
        setfillstyle(SOLID_FILL,BLACK);
        floodfill(x+a*(dx2/2)+10,y+b*(dy2/2)+10,WHITE);
        delay(400);
  }
/#define BORDER 1
 #define MAX_FRAME 15
```

```
#define REV_VID 0x70
#define NORM_VID 7

char far *vid_mem;
struct menu_frame  {
    int startx,starty,endx,endy;
    unsigned char *p;
    char **menu;
    char *keys;
    int border,count;
    int active;
}  frame[MAX_FRAME];

char *main_menu[] = {
      "Data",
      "Options",
      "Control",
      "Exit"};
char *data_choice[] = {
      "Write    ",
      "View/Edit",
      "Delete   " };

char *room_data[]  = {
      "Class              ",
      "Room               ",
      "Name               ",
      "Sex                ",
      "Age                ",
      "Occu.              "};
char *add_sensor[]  = {
      "      sensor                ",
      "  1.",
      "  2.",
      "  3.",
      "  4.",
      "  5."};
char *ask_room[] = {
      " class:       ",
      " room :       "};

char *level[] = {
      " class level ",
      " room  level "};
char *ask_class[] = {
      " class:     ",
      " point:     "};
char *ask_time[] = {
      "class:     ",
      "room :     ",
      "hour :     ",
      "min. :     "};
char *ask_point[] = {
      " class:       ",
      " room :       ",
```

```
      " point:       "};
void class_level();

menu_org()
{
      char *option_choice[3],v[3][20];
      int i;

      make_menu(0,main_menu,"depe",4,0,0,BORDER);
      make_menu(1,data_choice,"wve",3,1,3,BORDER);
      for(i = 0;i < 3;i++)
         option_choice[i] = v[i];
      read_option(option_choice);
      make_menu(2,option_choice,"rps",3,15,4,BORDER);
      make_menu(3,room_data,"nsaos",6,15,4,BORDER);
      make_menu(4,add_sensor,"n12345",6,12,12,BORDER);
      make_menu(5,ask_room,"cr",2,20,9,BORDER);
      make_menu(6,level,"cr",2,20,4,BORDER);
      make_menu(7,ask_time,"crhm",4,20,11,BORDER);
      make_menu(8,ask_class,"cp",2,40,9,BORDER);
      make_menu(9,ask_point,"crp",3,35,9,BORDER);
      pd_driver();
      return 1;
}
void read_option(x)
char *x[];
{
      FILE *fp;
      int i,j,k,l,m;
      int var[3];
      char y[20];
      char *compare[2] =  {"Record event  on " ,"Print event   on "};

      if((fp = fopen("save.opt","rt")) == NULL)  {
         fprintf(stderr,"Cannot open file %s\n","save.opt");
         exit(1);
      }
      for(i = 0;i < 2;i++) {
         j = 0;
         while((y[j++] = fgetc(fp)) != '\n')
           ;;
        strcpy(x[i],y);
        x[i][16] = ' ';
        x[i][17] = '\0 ';
      }
      fclose(fp);
      for(k = 0;k < 2;k++) {
         if(strcmp(x[k],compare[k]) == 0)
            var[k] = 1;
         else
            var[k] = 0;
      }
      rec_var = var[0];
      prn_var = var[1];
      strcpy(x[2],"Settime         ");
```

```
}
pd_driver()
{
     int choice1,choice2,choice3;
     int i;

     while((choice1 = pulldown(0)) != -1)  {
      switch(choice1)  {
         case 0 :
         while((choice2 = pulldown(1)) != -1) {
              switch(choice2) {
                 case 0:
                   write_data();
                   break;
                 case 1:
                   view_edit();
                   break;
                 case 2:
                   delete();
                   break;
              }
         }
         restore_video(1);
         break;

         case 1 :
         while((choice2 = pulldown(2)) != -1) {
              switch(choice2) {
                 case 0:
                    rec_opt(rec_var);
                    break;
                 case 1:
                    prn_opt(prn_var);
                    break;
                 case 2:
                    set_time();
                    break;
              }
         }
         save_option();
         restore_video(2);
         break;

         case 2 :
         while((choice3 = pulldown(6)) != -1) {
              switch(choice3) {
                 case 0:
                    class_level();
                    break;
                 case 1:
                    room_level();
                    break;
              }
         }
         restore_video(6);
```

```
            break;

            case 3:
               return 1;
          }
        }
        restore_video(0);
    }
    void write_data()
    {
        char *data1[6],*data2[5];
        char x[6][30],y[5][30];
        int i,j;
        char *fname,z[20];

        for(i = 0;i < 6;i++)
            data1[i] = x[i];
        for(j = 0;j < 5;j++)
           data2[j] = y[j];
        pulldown(3);
        get_s(24,6,data1,6);
        fname = z;
        file_name(data1,fname);
        save_file(fname,16,6,40,11,0);
        pulldown(4);
        get_s(18,15,data2,5);
        save_file(fname,15,14,50,19,1);
        if( getch() == ESC) {
           restore_video(3);
           restore_video(4);
        }
    }
    #define BKSP 8
    #define UP 72
    #define LEFT 75
    #define LIGHT 77
    #define DOWN 80
    #define F2 60

    view_edit()
    {
        union inkey {
        char ch[2];
        int i;
        } c;
        char *data[2],a[2][20];
        char *fname,b[20];
        int m;
        int x,y;
        char ch;

        for(m = 0;m < 2;m++)
           data[m] = a[m];
        pulldown(5);
        get_s(30,11,data,2);
```

```
        restore_video(5);
        fname = b;
        file_name(dta,fname);
        showdata(fname);
        x = 23;y = 7;
        gotoxy(x,y);
        for(;;)  {
           while(!bioskey(1))
            ;;
           c.i = bioskey(0);
           if(c.ch[0]) {
              if((c.ch[0] >= 48 && c.ch[0] <= 57) ||
                 (c.ch[0] >= 65 && c.ch[0] <= 90) ||
                 (c.ch[0] >= 97 && c.ch[0] <= 122))
                 putchar(c.ch[0]);
              switch(c.ch[0]) {
                   case '\r' :
                 y++;
                 gotoxy(x,y);
                 break;
                   case ' ' :
                 x++;
                 putch(' ');
                 gotoxy(x,y);
                 break;
                   case ESC :
                 restore_video(11);
                 return  1;
                   case BKSP:
                 x--;
                 gotoxy(x,y);
                 putch(' ');
                 gotoxy(x,y);
                 break;
           }
        }
        else {
           switch(c.ch[1])  {
           case UP :
                 y--;
                 gotoxy(x,y);
                 break;
           case LEFT:
                 x--;
                 gotoxy(x,y);
                 break;
           case LIGHT:
                 x++;
                 gotoxy(x,y);
                 break;
           case DOWN:
                 y++;
                 gotoxy(x,y);
                 break;
           case F2:
```

```
                save_file(fname,16,7.37,18,0);
                break;
            }
        }
        }
}
void delete()
{
     char *dt[2],a[2][20];
     char *fname,b[20];
     int m;

     for(m = 0;m < 2;m++)
        dt[m] = a[m];
     pulldown(5);
     get_s(30,11,dt,2);
     restore_video(5);
     fname = b;
     file_name(dt,fname);
     if((remove(fname) != 0))
       printf("Cannot delete the file!\n");
}
void get_s(x,y,addr,cnt)
char *addr[];
int x,y;
int cnt;
{
     int i;

     gotoxy(x,y);
     for(i = 0;i < cnt;i++)  {
        gets(addr[i]);
        y++;
        gotoxy(x,y);
     }
}
void file_name(addr,n)
char *addr[];
char *n;
{
     char *m,x[20];
     char *extend = ".dat";
     int len1,len2,len3;
     int i,j;

     m = x;
     m = addr[0];
     len1 = strlen(addr[0]);
     m += len1 ;
     *(m)++ = '_';
     len2 = strlen(addr[1]);
     for(i = 0;i < len2;i++)
        *(m)++ = *(addr[1])++;
     addr[1] -= len2;
     len3 = strlen(extend);
```

```
        for(j = 0;j < len3;j++)
           *(m)++ = *extend++;
        *m = '\0';
        m -= (len1+len2+len3+1);
        strcpy(n,m);
    }
    void rec_opt(var)
    int var;
    {
        if(var == 0) {
          rec_var = 1;
          frame[2].menu[0] = "Record event  on ";
        }
        if(var == 1) {
          rec_var = 0;
          frame[2].menu[0] = "Record event off ";
        }
    }
    void save_option()
    {
        FILE *fp;
        int i;

        if((fp = fopen("save.opt","wt")) == NULL)  {
           fprintf(stderr,"Cannot open file %s\n","save.opt");
           exit(1);
        }
        for(i = 0;i < 3;i++)
           fprintf(fp,"%s\n",frame[2].menu[i]);
        fclose(fp);
    }
    void prn_opt(var)
    int var;
    {
        if(var == 0) {
           prn_var = 1;
           frame[2].menu[1] = "Print event   on ";
        }
        if(var == 1) {
          prn_var = 0;
          frame[2].menu[1] = "Print event  off ";
        }
    }
    void set_time()
    {
        unsigned char dt[4];
        unsigned char *ask[4],a[4][4];
        int i;

        for(i = 0;i < 4;i++)
           ask[i] = a[i];
        pulldown(7);
        get_s(30,13,ask,4);
        for(i = 0;i < 4;i++)
           dt[i] = atoi(ask[i]);
```

```
        send_data(dt,3);
        restore_video(7);
    }
    void class_level()
    {
        unsigned char *dat[2],a[2][4];
        unsigned char dt[2];
        union inkey {
          char ch[2];
          int i;
        } c;
        int k,l;

        for(k = 0;k < 2;k++)
           dat[k] = a[k];
        pulldown(8);
        get_s(50,11,dat,2);
        for(l = 0;l < 2;l++)
           dt[l] = atoi(dat[l]);
        send_data(dt,1);
        restore_video(8);
    }
    room_level()
    {
        unsigned char *dat[3],a[3][3];
        unsigned char dt[3];
        union inkey {
          char ch[2];
          int i;
        } c;
        int k,l;
        int y;

        for(k = 0;k < 2;k++)
           dat[k] = a[k];
        pulldown(9);
        get_s(45,11,dat,3);
        for(l = 0;l < 3;l++)
           dt[l] = atoi(dat[l]);
        send_data(dt,2);
        restore_video(9);
    }
    void showdata(finame)
    char *finame;
    {
        FILE *fp;
        int i,num = 10;
        char *data[12],c[12][40];
        char x[40];
        int a = 15,b = 5;

        if((fp = fopen(finame,"r")) == NULL)  {
           fprintf(stderr,"Cannot open file %s\n",finame);
           exit(1);
        }
```

```
    for(i = 0;i < 12;i++)  {
      data[i] = c[i];
      fgets(x,sizeof(x),fp);
      x[20] = '\0';
      strcpy(data[i],x);
    }
    make_menu(num,data,"abc",12,a,b,BORDER);
    pulldown(num);
}
void save_file(finame,startx,starty,endx,endy,form)
char *finame;
int startx,starty,endx,endy;
int form;
{
    FILE *fp;
    union REGS r;
    register x,y;
    char *mode;
    char end = '\0';

    if(form == 0)
      mode = "w";
    else
      mode = "a";

    if((fp = fopen(finame,mode)) == NULL)  {
      fprintf(stderr,"Cann't open file %s\n",finame);
      exit(1);
    }
    for(y = starty;y <= endy;y++) {
      for(x = startx+1;x < endx;x++)  {
        gotoxy(x,y);
        r.h.ah = 8;
        r.h.bh = 0;
        int86(0x10,&r,&r);
        putc(r.h.al,fp);
      }
      fprintf(fp,"%c\n",end);
    }
    if(form == 1)
      fprintf(fp,"%c",'/ );
    if((fclose(fp)) != NULL)  {
      fprintf(stderr,"Cannot close file: %s\n",finame);
      exit(1);
    }
}
int v_choice[MAX_FRAME];
int x_choice[MAX_FRAME];

int pulldown(num)
int num;
{
    int vmode,choice;

    vmode = video_mode();
```

```
    if((vmode != 2) && (vmode != 3) && (vmode != 7))  {
       printf("video must be in 80 column text mode");
       exit(1);
    }
    if (vmode == 7)
       vid_mem = (char far *) 0xB0000000;
    else
       vid_mem = (char far *) 0xB8000000;

    if(!frame[num].active)  {
       save_video(num);
       frame[num].active = 1;
       x_choice[num] = 0;
       y_choice[num] = 0;
    }
    if (frame[num].border)
       draw_border(num);

    display_menu(num);
    if(num != 3 && num != 4 && num != 5 &&
        num != 7 && num != 8 && num != 9 && num != 10)
        return get_resp(num);
}
make_menu(num,menu,keys,count,x,y,border)
int num;
char *menu[];
char *keys;
int count;
int x,y;
int border;
{
     register int i,len;
     int endx,endy,choice,vmode;
     unsigned int *p;

     if(num > MAX_FRAME)  {
       printf("Too many menus  !\n");
       return 0;
     }

     if ((x>79) || (x<0) || (y>24) || (y<0))  {
        printf("Range menu error !\n");
        return 0;
     }

    / len = 0;
     for(i = 0;i < count; i++)
        if(strlen(menu[i]) > len)
           len = strlen(menu[i]);
     if (num == 0)  {
        endx = 4*(len+x+3);
        endy = y+2;
     }
     else  {
        endx = len+3+x;
```

```
            endy = count+1+y;
        }
        if((endx>80) || (endy>25))  {
            printf("menu won't fit \n");
            return 0;
        }
        p = (unsigned int *) malloc(2*(endx-x+1) * (endy-y+1));
        if (!p)
            exit(1);
        frame[num].startx = x;
        frame[num].starty = y;
        frame[num].endx = endx;
        frame[num].endy = endy;
        frame[num].p = p;
        frame[num].menu = (char**) menu;
        frame[num].border = border;
        frame[num].keys = keys;
        frame[num].count = count;
        frame[num].active = 0;
        return 1;
}
void display_menu(num)
int num;
{
    register int x,y,i,j;
    char **m;

    x = frame[num].startx+1;
    y = frame[num].starty+1;
    m = frame[num].menu;

    if (num == 0)
       for (i = 0;i < frame[num].count;i++,x += frame[num].endx/4)
          write_string(x,frame[num].starty+1,m[i],NORM_VID);
    else
    for (j = 0;j  < frame[num].count;j++,y++)
          write_string(frame[num].startx+1,y,m[j],NORM_VID);
}
void draw_border(num)
int num;
{
    register int i;
    char far *v,far *t;

    v = vid_mem;
    t = v;

    for(i = frame[num].startx+1;i < frame[num].endx;i++)  {
        v += frame[num].endy*160 + (i*2);
        *v++ = 186;
        *v = NORM_VID;
        v = t;
        v += frame[num].starty*160 + (i*2);
        *v++ = 186;
        *v = NORM_VID;
```

```
        v = t;
    }

    for(i = frame[num].startv+1;i < frame[num].endy;i++)  {
        v += i*160 + (frame[num].endx*2);
        *v++ = 179;
        *v =NORM_VID;
        v = t;
        v += (i * 160) + frame[num].startx*2;
        *v++ = 179;
        *v = NORM_VID;
        v = t;
    }
    write_char(frame[num].startx,frame[num].starty,218,NORM_VID);
    write_char(frame[num].startx,frame[num].endy,192,NORM_VID);
    write_char(frame[num].endx,frame[num].starty,191,NORM_VID);
    write_char(frame[num].endx,frame[num].endy,217,NORM_VID);
}
get_resp(num)
int num;
{
     union inkey {
       char ch[2];
       int i;
     } c;
     int x_arrow;
     int y_arrow;
     int key_choice;
     int x,y;

     x_arrow = x_choice[num];
     y_arrow = y_choice[num];

     x = frame[num].startx+1;
     y = frame[num].starty+1;

     if(num == 0)
        write_string(x+(frame[num].endx/4)*x_arrow,y,
           frame[num].menu[x_arrow],REV_VID);
     else
         write_string(x,y+y_arrow,
         frame[num].menu[y_arrow],REV_VID);

     for (;;)  {
     while (!bioskey(1))
       ;
     c.i = bioskey(0);

     if (num == 0)
          write_string(x+(frame[num].endx/4)*x_arrow,y,
         frame[num].menu[x_arrow],NORM_VID);
     else
          write_string(x,y+y_arrow,
         frame[num].menu[y_arrow],NORM_VID);
```

```
      if (c.ch[0])  {
         key_choice = is_in(frame[num].keys,tolower(c.ch[0]));
         if (key_choice)
            return key_choice - 1;
         switch(c.ch[0])  {
          case '\r  :
                        x_choice[num] = x_arrow;
              y_choice[num] = y_arrow;
              if(num == 0)
                 return  x_arrow;
              else
                 return y_arrow;
          case ' '  : y_arrow++;
                 break;
          case ESC  : return -1;
         }
       }
       else  {
         switch(c.ch[1])  {
              case 75 : x_arrow--;
                        break;
              case 77 : x_arrow++;
                        break;
              case 72 : y_arrow--;
                        break;
              case 80 : y_arrow++;
                        break;
         }
       }

       if (x_arrow == frame[num].count)
          x_arrow = 0;
       if (x_arrow < 0)
          x_arrow = frame[num].count-1;
       if (y_arrow == frame[num].count)
          y_arrow = 0;
       if (y_arrow < 0)
          y_arrow = frame[num].count-1;

       if (num == 0)
           write_string(x+(frame[num].endx/4)*x_arrow,y,
            frame[num].menu[x_arrow],REV_VID);
       else
           write_string(x,y+y_arrow,
            frame[num].menu[y_arrow],REV_VID);
      }
   }
   void write_string(x,y,p,attrib)
   int x,y;
   char *p;
   int attrib;
   {
        register int i;
        char far *v;
```

```
        v = vid_mem;
        v += y*160 + x*2;
        for (i = x; *p:i++)  {
           *v++ = *p++;
           *v++ = attrib;
        }
}
void write_char(x,y,ch,attrib)
int x,y;
char ch;
int attrib;
{
     register int i;
     char far *v;

     v = vid_mem;
     v += y*160 + x*2;
     *v++ = ch;
     *v   = attrib;
}

void save_video(num)
int num;
{
     register int i,j;
     char *buf_ptr;
     char far *v,far *t;

     buf_ptr = frame[num].p;
     v = vid_mem;
     for (i = frame[num].starty;i < frame[num].endy+1; i++)
       for (j = frame[num].startx;j < frame[num].endx+1; j++)  {
          t =  v + (i*160) + (j*2);
          *buf_ptr++ = *t++;
          *buf_ptr++ = *t;
          *(t-1) = ' ';
       }
}
void restore_video(num)
int num;
{
     register int i,j;
     char far*v,far *t;
     char *buf_ptr;

     buf_ptr = frame[num].p;
     v = vid_mem;
     t = v;
     for (i = frame[num].starty; i < frame[num].endy+1; i++)
        for (j = frame[num].startx; j < frame[num].endx+1; j++)  {
           v = t;
           v += (i*160) +(j*2);
           *v++ = *buf_ptr++;
           *v   = *buf_ptr++;
        }
```

```
        frame[numj].active = 0;
}
video_mode()
{
    union REGS r;

    r.h.ah = 15;
    return int86(0x10,&r,&r) & 255;
}

is_in(s,c)
char *s,c;
{
    register int i;

    for (i = 0;i < *s; i++)
    if (*s++ == c)
        return i+1;
    return 0;
}
```

เอกสารอ้างอิง

1.Kenneth J. Ayala , "The 8051 Microcontroller Architecture,Programming and Application", West Publishing Company.

2.ทินกร ดุ๊ก และคณะ , " การใช้งาน Z80 ", สำนักพิมพ์พิสิกส์เซ็นเตอร์

3.คิวัฒน์ คิวะบวร,พรชัย จักรธำรงและจิรศักดิ์ ชัยวิริยะกุล," การประยุกต์ใช้งานภาษาซี ", (บทที่ 1: เมนูแบบ pop-up pull-down,บทที่ 6: ระบบแลนและการรับส่งข้อมูลด้วยพอร์ตอนุกรม) สำนักพิมพ์ซีเอ็ดยูเคชั่น

กิตติกรรมประกาศ

ขอขอบพระคุณอย่างสูงต่อ รศ.ดร. มนัส สังวรคิลป์ ในฐานะอาจารย์ที่ปรึกษา ทางผู้จัดทำได้รับ คำปรึกษาและแนะนำ พร้อมทั้งดูแลเอาใจใส่ระหว่างการทำปริญญานิพนธ์ฉบับนี้ ตั้งแต่ต้นจนจบการศึกษา ให้สำเร็จเรียบร้อยด้วยดี

และเพื่อนนักศึกษาทุกคนที่ได้ให้ความช่วยเหลือ ในการทำปริญญานิพนธ์ฉบับนี้