ปีการศึกษา 2535

บ้านอัตโนมัติ
(HOME AUTOMATIC)

โดย

นาย	ยิ่งยง	ไชยฟู	32-1250
นาย	สถาพร	จิตหัตถะ	32-1343
นาย	ปรีดา	แซ่อื้อ	33-1008

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ พิพัฒน์ เลาหสงคราม

ปริญญานิพนธ์ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า

เจ้าคุณทหาร ลาดกระบัง

เรื่อง บ้านอัตโนมัติ

(HOME AUTOMATIC)

ผู้จัดทำ

1. นาย ยิ่งยง ไชยฟู 32-1250
2. นาย สถาพร จิตหิตถะ 32-1343
3. นาย ปรีดา แซ่อึ้ง 33-1008

.....อาจารย์ที่ปรึกษา

(ผศ. พิพัฒน์ เลาหสงคราม)

บ้านอัตโนมัติ
(HOME AUTOMATIC)

นาย ยิ่งยง ไชยฟู 32-1250

นาย สถาพร จิตหัตถะ 32-1343

นาย ปรีดา แซ่อื้อ 33-1008

อาจารย์ที่ปรึกษา

ผศ. พิพัฒน์ เลาหสงคราม

ปีการศึกษา

2535

บทคัดย่อ

โครงงานนี้ใช้คอมพิวเตอร์ควบคุมอุปกรณ์ไฟฟ้าที่ใช้ในบ้านอย่างอัตโนมัติ ในลักษณะของการตั้งโปรแกรมล่วงหน้าตามเวลาที่ต้องการให้เปิด/ปิดอุปกรณ์ไฟฟ้าในรอบ 1 อาทิตย์ โดยควบคุมผ่านสายไฟเอซี นอกจากนี้ยังมีการตรวจจับคนที่อยู่ภายในห้องด้วยอินฟราเรดเซนเซอร์ เพื่อทำการเปิด/ปิดหลอดไฟได้อีกด้วย

หัวใจของโครงงานนี้ อยู่ที่อุปกรณ์เซมิคอนดัคเตอร์ตัวหนึ่ง ซึ่งเป็นตัวจัดการการทำงานต่าง ๆ เกือบทั้งหมด เรียกว่า ไมโครคอมพิวเตอร์แบบชิพเดี่ยว นอกจากนี้ก็ยังมีชุดที่เชื่อมต่อกับโหลด และชุดอินฟราเรดเซนเซอร์ ที่มีความสำคัญเช่นเดียวกัน

HOME AUTOMATIC

MR.	YINGYONG	CHAIYAFOO	32-1250
MR.	STAPORN	JITHUTTA	32-1343
MR.	PREEDA	SAEEU	33-1008

ADVISOR

PROF. ASIST. PHIPHAT LOAHASONGKRAM

ACADEMIC YEAR

1992

ABSTRACT

This project use the computer to control appliances that are used in the houses automatically. That is we install the advanced program and use the on/off appliance when it is requried within one week. It is controlled by passing AC line.On the other hand , it can detect the man is in the room with infrared sensor. For on/off the lamp.

The center of this project is an equipment , called a semi-conductor. It arranges all the works , called a single chip micro computer. Moreover , It has the load connection unit and infrared sensor unit which is important as well.

สารบัญ

บทที่ 1
บทนำ

ปัจจุบันความเจริญเติบโตทางเทคโนโลยีได้รุดหน้าไปอย่างรวดเร็ว ความสะดวกสบายในชีวิตประจำวันก็มีมากขึ้น เนื่องจากได้นำเอาเทคโนโลยีในสาขาต่างๆมาประกอบเข้าด้วยกัน และประดิษฐ์เป็นเครื่องอำนวยความสะดวกต่างๆ เช่น เครื่องปรับอากาศ เครื่องซักผ้า เตาอบไมโครเวฟ ฯลฯ สิ่งเหล่านี้เกิดจากการผสมผสานกันระหว่างเทคโนโลยีในสาขาต่างๆ แต่สิ่งหนึ่งที่เข้ามามีบทบาทในชีวิตประจำวันของเรามากขึ้นนั่นก็คือไมโครคอมพิวเตอร์ถูกนำมาใช้ควบคุมอุปกรณ์ต่างๆที่กล่าวมาข้างต้น ซึ่งปัจจุบันมี CPU ที่ถูกผลิตขึ้นมาให้มีคุณสมบัติเหมาะสมกับงานด้านต่างๆ เช่น CPU ที่ใช้ในงานคอนโทรล หรือที่เรียกว่า SINGLE CHIP MICROCONTROLLER เป็นต้น

โครงงาน HOME AUTOMATIC เป็นตัวอย่างหนึ่งของการใช้ประโยชน์จาก SINGLE CHIP MICROCONTROLLER ของ INTEL เบอร์ 8031 ซึ่งอยู่ในตระกูล MCS-51 เป็นตัวควบคุมการทำงานตามโปรแกรมที่อยู่ในอีพรอม และใช้สายไฟเอซี ที่เดินอยู่ตามอาคารบ้านเรือนมาใช้เป็นสายนำสัญญาณ ซึ่งคุณสมบัติของการรับ-ส่งย่อมมีลักษณะแตกต่างไปจากสายนำสัญญาณที่ใช้กันอยู่ทั่วไป รูปแบบการส่งข้อมูลเป็นแบบดิจิตอลโดยมีการส่งแบบ Carrier Wave Keying (CWK) และเนื่องจากสายไฟเอซี ที่มีอยู่ภายในอาคาร มีการต่อถึงกันทุกๆจุดภายในอาคารนั้นๆแล้ว ข้อมูลที่ส่งออกจากจุดใดจุดหนึ่งในอาคารย่อมไปปรากฏอยู่ที่ทุกๆจุดของเต้าเสียบในอาคารนั้น ซึ่งจะเห็นชัดว่าทุกๆแห่งที่มีสายไฟเอซี เดินผ่านไปก็จะมีสัญญาณปรากฏอยู่ด้วยเช่นกัน

จากหลักการนี้เอง เราจึงนำมาประยุกต์ใช้ในการรับส่งข้อมูลเพื่อควบคุมการทำงานของอุปกรณ์ต่างๆ ภายในอาคารให้เปิด/ปิดตามเวลาที่กำหนดไว้และสามารถตรวจสอบว่าการทำงานของอุปกรณ์นั้นๆเป็นไปตามที่สั่งไว้หรือไม่

นอกจากนั้นโครงงานนี้ยังประกอบด้วยอินฟราเรดเซนเซอร์ที่มีตัวไพโรอิเล็กตริกเป็นหัวใจสำคัญ เนื่องจากตัวมันถูกออกแบบให้มีความสามารถที่จะรับรู้ได้ว่ามีคนเข้ามาใกล้หรือไม่ ซึ่งอาศัยหลักการตรวจจับการเปลี่ยนแปลงอุณหภูมิ โดยการเปลี่ยนแปลงของรังสีอินฟราเรดที่แผ่ออกมา เนื่องจากการเคลื่อนไหวของคนในรัศมีที่มันสามารถรับรู้ได้

และจากหลักการนี้ จึงนำมาประยุกต์ใช้ในการเปิด/ปิดอุปกรณ์ไฟฟ้าที่เป็นหลอดไฟ ซึ่งอินฟราเรดเซนเซอร์นี้จะทำงานร่วมกับคอนโทรลเลอร์ โดยอินฟราเรดเซนเซอร์จะส่งสัญญาณไปบอกให้คอนโทรลเลอร์ทราบ เพื่อที่จะให้คอนโทรลเลอร์ทำการเปิดหลอดไฟ

ในกรณีที่ตรวจจับได้ว่ามีคนเข้ามาอยู่ในห้องนั้นแล้ว

จากที่กล่าวมาทั้งหมดจะเห็นว่าโครงงาน HOME AUTOMATIC สามารถที่จะนำมาใช้ประโยชน์ในชีวิตประจำวันได้ ซึ่งเป็นจุดมุ่งหมายหลักของโครงงานชุดนี้

บทที่ 2
ทฤษฎีและหลักการ

MCS-51ไมโครคอนโทรลเลอร์แบบชิพเดี่ยว

MCS-51 เป็นอุปกรณ์ที่ออกแบบมาสนองความต้องการของผู้ใช้ คือ มีสายอินพุทและเอาท์พุทภายในตัวเอง และสายควบคุมอื่นๆที่ใช้สำหรับแยกข้อมูลกับแอดเดรสและยังมีชุดคำสั่งเพิ่มขึ้นเป็นพิเศษเพื่อจัดการข้อมูล แถมท้ายด้วยวงจรตั้งเวลากับวงจรนับอีกด้วย (ปกติวงจรนับจะสามารถทำงานเป็นวงจรตั้งเวลาได้ด้วยจึงเรียกควบคู่กันไป คือ วงจรตั้งเวลา/วงจรนับ) ดังมีรายละเอียดของ MCS-51 ดังนี้คือ

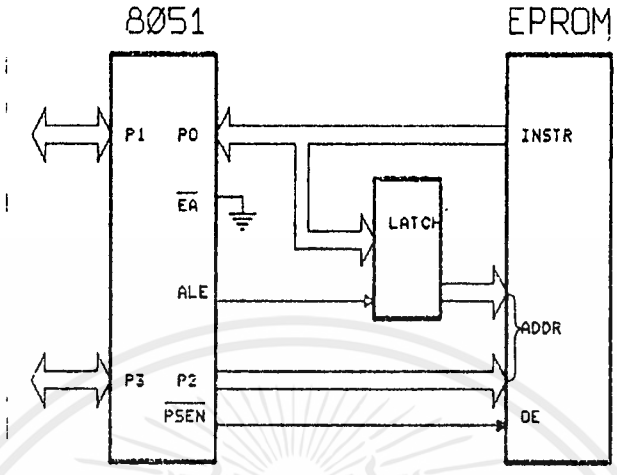
การจัดหน่วยความจำ

ในระบบของ MCS-51 จะมีการแบ่งหน่วยความจำเหมือนกับซีพียูทั่วๆไปคือ จะแบ่งเป็น 2 ลักษณะตามชนิดของข้อมูลที่เก็บดังนี้

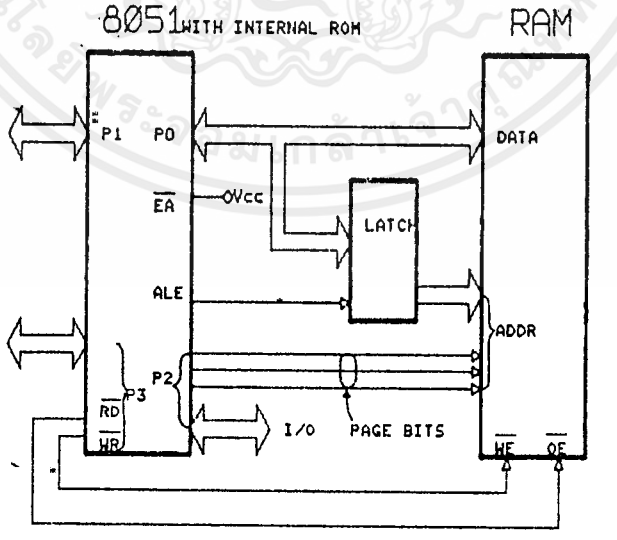
หน่วยความจำข้อมูล (data memory)

หน่วยความจำโปรแกรม (program memory)

ถ้าจะพูดกันง่ายๆ หน่วยความจำข้อมูลก็หมายถึง หน่วยความจำส่วนที่เป็นแรม (RAM) ซึ่งเราสามารถอ่านหรือเขียนข้อมูลเปลี่ยนแปลงได้ตลอดเวลา แต่ไม่สามารถรันโปรแกรมบนหน่วยความจำส่วนนี้ได้ ส่วนหน่วยความจำโปรแกรมจะหมายถึงหน่วยความจำที่อ่านได้อย่างเดียว (ROM) ซึ่งบรรจุโปรแกรมที่จะให้ MCS-51 ทำงาน โดยหน่วยความจำทั้ง 2 ประเภทนี้จะถูกแยกออกจากกันด้วยคำสั่งทางซอฟต์แวร์และลักษณะการติดต่อทางฮาร์ดแวร์ กล่าวคือ จะมีคำสั่งเฉพาะสำหรับการติดต่อกับหน่วยความจำชนิดใดชนิดหนึ่งและจัดสัญญาณสโตรบในการติดต่อกับหน่วยความจำแต่ละชนิดแยกต่างหากกันด้วย ดังรูปที่ 2-1 และ รูปที่ 2-2 ในหน้าถัดไป



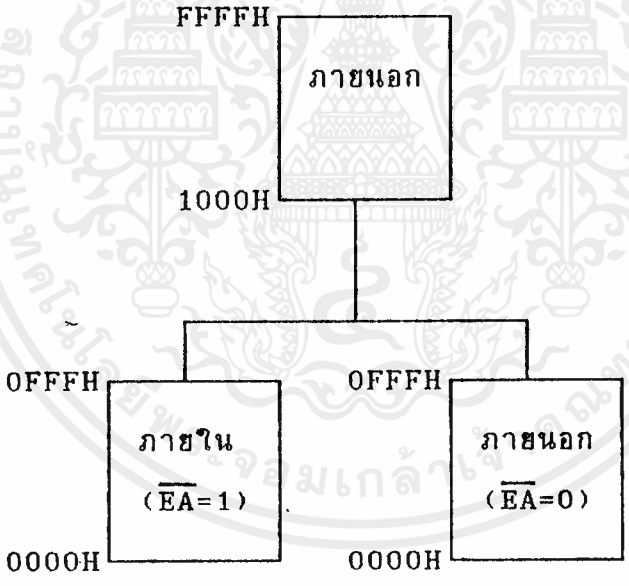
รูปที่ 2-1 การเชื่อมต่อ MCS-51 กับหน่วยความจำโปรแกรม



รูปที่ 2-2 การเชื่อมต่อ MCS-51 กับหน่วยความจำข้อมูล

หน่วยความจำโปรแกรม

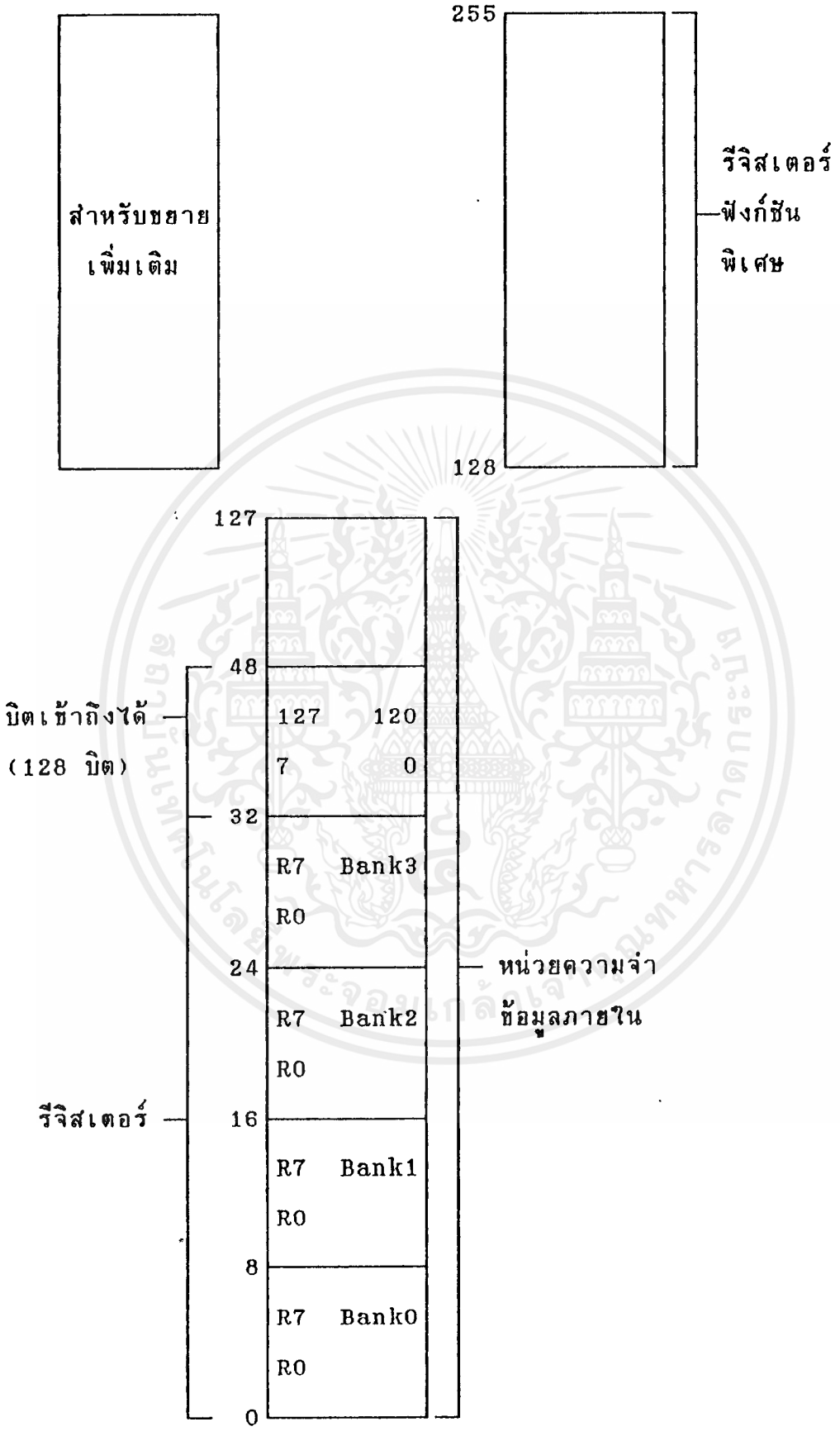
ใน MCS-51 จะแบ่งหน่วยความจำโปรแกรมออกเป็น 2 ส่วน คือ หน่วยความจำโปรแกรมภายในและหน่วยความจำโปรแกรมภายนอก หน่วยความจำโปรแกรมภายในก็คือหน่วยความจำประเภท ROM หรือ EPROM ที่อยู่ภายในตัว MCS-51 เบอร์ 8051/8751 ตามลำดับ ซึ่งมีขนาด 4 กิโลไบต์ ส่วนหน่วยความจำโปรแกรมภายนอกก็คือ หน่วยความจำที่ต่ออยู่ภายนอกตัว MCS-51 โดยอาจจะเป็นการต่อเพื่อขยายหน่วยความจำเพิ่มเนื่องจากหน่วยความจำภายในไม่พอ หรืออาจต่อเป็นหน่วยความจำโปรแกรมภายนอกทั้งหมดเลยก็ได้ ในกรณีของ 8031 ภายในตัวมันไม่มีหน่วยความจำโปรแกรมเหมือนเบอร์ 8051/8751 โดย MCS-51 สามารถเลือกให้รันโปรแกรมในหน่วยความจำโปรแกรมภายในหรือภายนอกก็ได้ โดยควบคุมที่ขา $\overline{EA}$ (ขา 31) แต่สำหรับเบอร์ 8031 จะต้องต่อขา $\overline{EA}$ ลงกราวด์เสมอ การจัดแบ่งพื้นที่หน่วยความจำโปรแกรมภายในและภายนอก แสดงได้ดังในรูปที่ 2-3



รูปที่ 2-3 การจัดแบ่งพื้นที่หน่วยความจำโปรแกรมภายในและภายนอก

หน่วยความจำข้อมูล

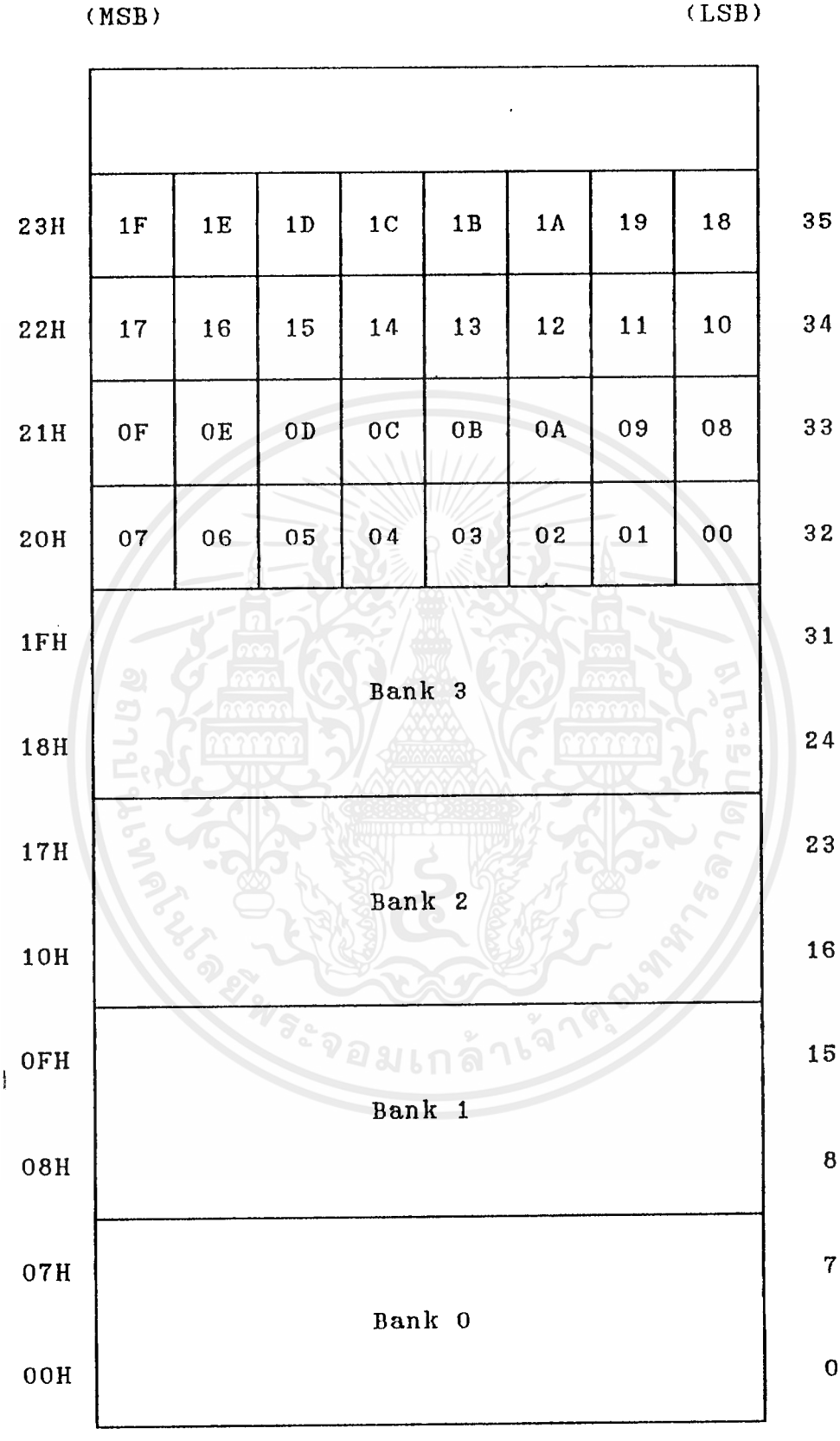
หน่วยความจำข้อมูลของ MCS-51 ถูกแบ่งเป็น 2 ส่วนเหมือนกันคือ หน่วยความจำข้อมูลภายในและภายนอก โดยหน่วยความจำข้อมูลภายนอกจะเข้าถึงได้ก็ด้วยคำสั่ง MOVX เท่านั้น สำหรับหน่วยความจำข้อมูลภายในของ 8031/8051 และ 8751 จะมีทั้งหมด 256 ไบต์ โดยแบ่งเป็น 128 ไบต์ส่วนบน ซึ่งใช้เป็นที่อยู่ของรีจิสเตอร์ฟังก์ชันพิเศษและอีก 128 ไบต์ส่วนล่าง ซึ่งจะถูกใช้งานทั่วไป ดังรูปที่ 2-4 และ รูปที่ 2-5



รูปที่ 2-4 การจัดพื้นที่หน่วยความจำข้อมูล

	(MSB)							(LSB)	
7FH									127
2FH	7F	7E	7D	7C	7B	7A	79	78	47
2EH	77	76	75	74	73	72	71	70	46
2DH	6F	6E	6D	6C	6B	6A	69	68	45
2CH	67	66	65	64	63	62	61	60	44
2BH	5F	5E	5D	5C	5B	5A	59	58	43
2AH	57	56	55	54	53	52	51	50	42
29H	4F	4E	4D	4C	4B	4A	49	48	41
28H	47	46	45	44	43	42	41	40	40
27H	3F	3E	3D	3C	3B	3A	39	38	39
26H	37	36	35	34	33	32	31	30	38
25H	2F	2E	2D	2C	2B	2A	29	28	37
24H	27	26	25	24	23	22	21	20	36

รูปที่ 2-5 พื้นที่การกำหนดตำแหน่งของหน่วยความจำข้อมูลภายใน



รูปที่ 2-5 (ต่อ) พื้นที่การกำหนดตำแหน่งบิตของหน่วยความจำข้อมูลภายใน

จากรูปจะเห็นว่าพื้นที่หน่วยความจำข้อมูลภายในส่วน 128 ไบต์ล่าง (ตำแหน่ง00H-7FH) จะถูกแบ่งเป็น 3 ส่วนคือส่วนของรีจิสเตอร์แบงค์ (00H-1FH),ส่วนพิเศษที่สามารถเข้าถึงตำแหน่งบิตได้โดยตรง (20H-2FH) และส่วนที่ใช้งานทั่วไป (30H-7FH)

ส่วนของรีจิสเตอร์แบงค์มีทั้งหมด 4 แบงค์ แต่เราสามารถใช้ได้ครั้งละแบงค์เท่านั้น การเลือกใช้แบงค์ไหนอยู่ที่เรากำหนดค่าในรีจิสเตอร์ PSW

คำสั่งที่ใช้ในการติดต่อกับหน่วยความจำข้อมูลซึ่งจะแบ่งลักษณะการกำหนดเลขที่อยู่ของหน่วยความจำข้อมูลได้ 4 โหมดด้วยกันคือ

โหมดการกำหนดเลขที่อยู่รีจิสเตอร์ จะเป็นการติดต่อกับข้อมูลที่อยู่ในรีจิสเตอร์โดยตรง ตัวอย่างเช่น

```
MOV A,R0 ; ย้ายค่าใน R0 ไปไว้ที่ ACC
ADD A,R3 ; บวกค่าใน ACC กับค่าใน R3 แล้วนำผลลัพธ์ไปไว้ใน ACC
```

โหมดการกำหนดเลขที่อยู่โดยตรง จะเป็นการติดต่อกับข้อมูลที่ตำแหน่งแอดเดรสที่กำหนดโดยตรง ซึ่งจะต้องอยู่บริเวณตำแหน่งของแรมภายใน 128 ไบต์ ตัวอย่างเช่น

```
MOV A,40H ; ย้ายค่าข้อมูลในหน่วยความจำตำแหน่ง 40H ไปเก็บไว้ใน ACC
ADD A,41H ; บวกค่าข้อมูลในแรมตำแหน่ง 41H กับข้อมูลใน ACC
```

โหมดการกำหนดเลขที่อยู่ข้อมูลโดยทันที จะเป็นการติดต่อกับข้อมูลคงที่ ตัวอย่างเช่น

```
MOV A,#40H ; นำค่า 40H ไปเก็บใน ACC
ADD A,#41H ; บวกค่า 41H เข้ากับค่าใน ACC
```

โหมดการกำหนดเลขที่อยู่รีจิสเตอร์โดยอ้อม จะเป็นการติดต่อกับข้อมูลที่ใช้ค่าในตัวรีจิสเตอร์ R0 หรือ R1 เป็นตัวชี้ตำแหน่ง ตัวอย่างเช่น

```
MOV A,@R0 ; นำค่าข้อมูลในหน่วยความจำที่ถูกชี้ด้วย R0 ไปเก็บใน ACC
ADD A,@R1 ; บวกค่าของข้อมูลในหน่วยความจำที่ถูกชี้ด้วย R1 กับค่าใน ACC
```

ตัวจับเวลา/ตัวนับ (Timer/Counter)

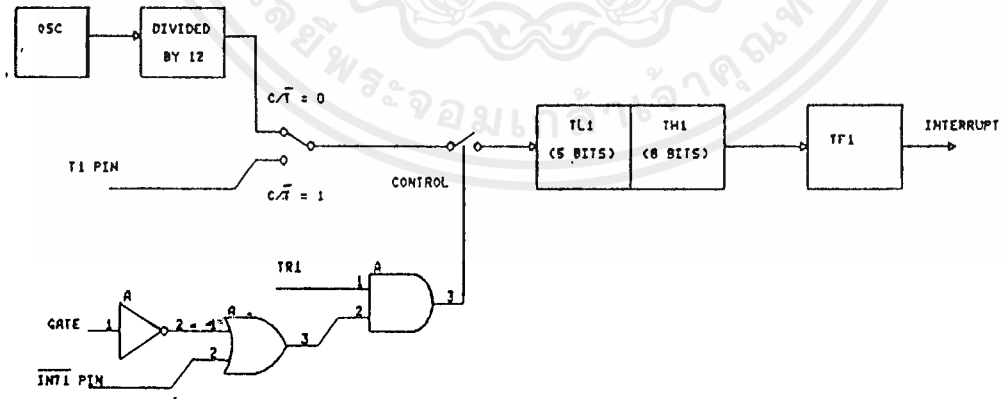
ในเบอร์ 8031/8051 และ 8751 จะมีตัวจับเวลา/ตัวนับขนาด 16 บิต จำนวน 2 ตัว คือ ไทเมอร์/เคาน์เตอร์ 0 และ ไทเมอร์/เคาน์เตอร์ 1 ส่วนเบอร์ 8032 กับเบอร์ 8052 จะมีเพิ่มอีก 1 ตัว โดยแต่ละตัวสามารถที่จะกำหนดให้ทำงานเป็นตัวจับเวลาหรือตัวนับได้โดยการเซตหรือเคลียร์บิต $C/\overline{T}$ ที่ตัวรีจิสเตอร์ควบคุม TMOD ซึ่งอยู่ในกลุ่มรีจิสเตอร์ฟังก์ชันพิเศษ

ในการกำหนดให้ทำงานเป็นตัวจับเวลา ตัวรีจิสเตอร์ TH1 และ TL1 ซึ่งทำหน้าที่เป็นตัวเก็บค่าจำนวนพัลส์ที่เข้ามาจะเพิ่มค่าทุกๆ แมชีนไซเกิล โดยแต่ละแมชีนไซเกิลจะประกอบด้วย 12 คาบออสซิลเลเตอร์ดังนั้นอัตราการนับแต่ละครั้งจะใช้เวลาเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์ ซึ่งส่วนใหญ่จะใช้ในงานอินเตอร์รัพต์ RTC

และถ้าให้ทำงานเป็นตัวนับ รีจิสเตอร์ตัวนับจะเพิ่มค่าทุกครั้งที่มีการเปลี่ยนแปลงสถานะจาก "1" เป็น "0" ที่ขา T0 หรือ T1 โดยอัตราความถี่สูงสุดที่สามารถนับได้ต้องไม่เกิน 1/24 ของความถี่ออสซิลเลเตอร์

ตัวจับเวลา/ตัวนับสามารถโปรแกรมให้มันทำงานได้ต่างกันถึง 4 โหมด โดยการตั้งค่าในรีจิสเตอร์ TMOD ซึ่งจะกล่าวถึงการทำงานของแต่ละโหมดดังนี้

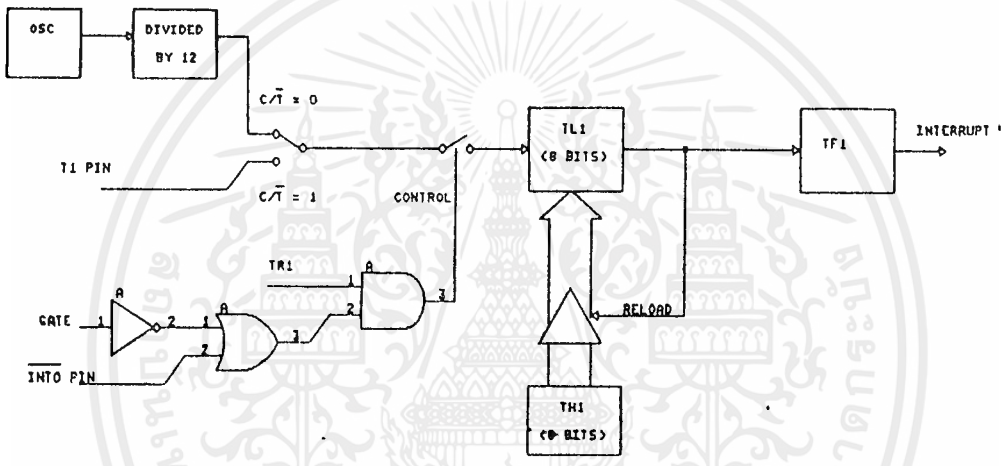
โหมด 0 รีจิสเตอร์ตัวนับจะถูกกำหนดให้มี 13 บิต ประกอบด้วยรีจิสเตอร์TH1 8 บิต และTL1 อีก 5 บิตอันดับต่ำ ซึ่งสามารถกำหนดให้เป็นตัวจับเวลาหรือตัวนับได้โดยเซตหรือเคลียร์ที่บิต $C/\overline{T}$ ในตัวรีจิสเตอร์ TMOD การทำงานของรีจิสเตอร์ตัวนับจะนับขึ้นครั้งละ 1 เมื่อมีสัญญาณเข้ามา 1 ลูกและเมื่อนับจนเป็น "1" หมดทุกบิตก็จะกลับมาเป็น "0" หมดทุกบิตใหม่ ซึ่งจะเป็นการเกิดโอเวอร์โฟลว์ (Overflow) ไปทดแฟลกอินเตอร์รัพต์ TF1 ให้เป็น "1" ลักษณะวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 0 และโหมด 1 เป็นดังรูปที่ 2-6



รูปที่ 2-6 ลักษณะวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 0 และโหมด 1

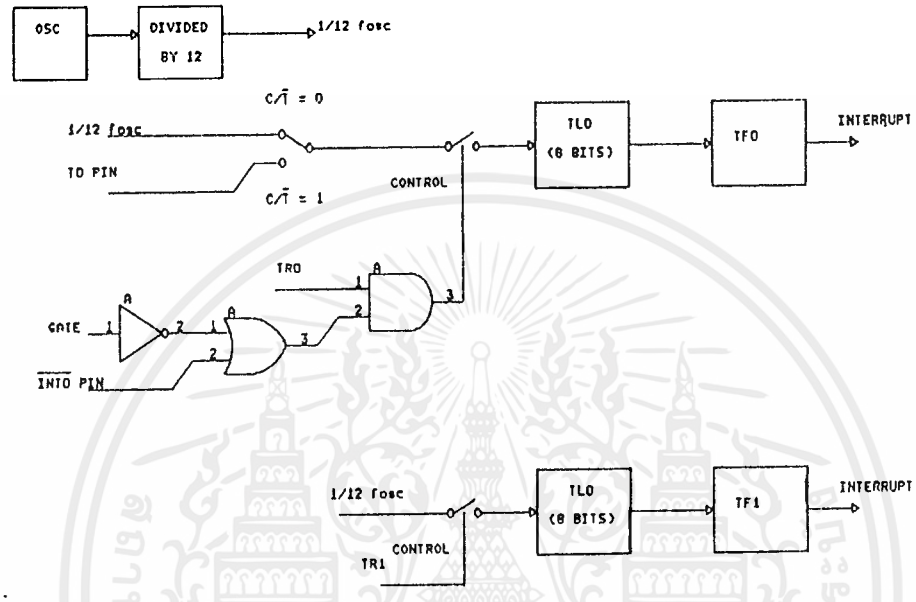
โหมด 1 การทำงานจะเหมือนกับโหมด 0 ทุกอย่างยกเว้นรีจิสเตอร์ตัวนับจะเป็นขนาด 16 บิต

โหมด 2 จะใช้รีจิสเตอร์ TL1 เป็นตัวนับเพียงตัวเดียวและเมื่อ TL1 นับจนเป็น "1" หมดทุกบิต ก็จะมีการโหลดค่าจากรีจิสเตอร์ TH1 เข้าไปไว้ใน TL1 โดยอัตโนมัติ และทำการทดแฟลกอินเตอร์รัพต์ TF1 ให้เป็น "1" ค่าใน TH1 นี้เราสามารถตั้งค่าได้ด้วยซอฟต์แวร์ ลักษณะวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 2 แสดงดังในรูปที่ 2-7



รูปที่ 2-7 ลักษณะวงจรการทำงานของตัวจับเวลา/ตัวนับที่ 1 ในโหมด 2

โหมด 3 เป็นการเพิ่มตัวจับเวลาขึ้นอีก 1 ตัวแต่จะเป็นขนาด 8 บิตทั้งคู่ ซึ่งลักษณะการทำงานอื่นๆ จะเหมือนกับโหมด 0 ซึ่งสามารถแสดงวงจรการทำงานของตัวจับเวลากับตัวนับที่ 0 ในโหมด 3 ได้ดังรูปที่ 2-8



รูปที่ 2-8 ลักษณะวงจรการทำงานของตัวจับเวลากับตัวนับที่ 0 ในโหมด 3

สำหรับรายละเอียดบิตควบคุมในรีจิสเตอร์ TMOD มีดังนี้คือ

TIMER/COUNTER1				TIMER/COUNTER0			
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0

GATE : เซตเป็น "1" จะเป็นการอีนาเบิลตัวจับเวลา/ตัวนับให้ถูกควบคุมด้วยขาINTx ต้องมีสถานะสูงและบิต TRX ใน TCON ต้องเซตเป็น "1" จึงจะเริ่มทำงาน แต่ถ้า GATE เป็น "0" ตัวจับเวลา/ตัวนับจะถูกควบคุมให้เริ่มทำงานด้วยบิต TRX เท่านั้น

C/$\overline{T}$: เป็นบิตควบคุมในการเลือกทำงานเป็นตัวจับเวลาหรือตัวนับ ถ้าเป็น"0" จะทำงาน เป็นตัวจับเวลา ถ้าเป็น "1"จะทำงานเป็นตัวนับ

M1,M0 : เป็นตัวเลือกโหมดการทำงานดังนี้

M1	M0	โหมดการทำงาน
0	0	โหมด 0
0	1	โหมด 1
1	0	โหมด 2
1	1	โหมด 3

หมายเหตุ x หมายถึงเลข 1 หรือเลข 0

จะต้องเคลียร์บิตนี้ด้วย โปรแกรมหลังการรับทุกครั้ง

โหมด 3 จะเหมือนกับโหมด 2 ทุกอย่าง ยกเว้นอัตราบ็อดจะแปรผันตามการตั้งตัวจับเวลาตัวที่1 ซึ่งจะใช้สูตรเดียวกับโหมด 1

การอินเตอร์รัพต์

ในระบบของ MCS-51 จะอินเตอร์รัพต์ได้จาก 5 แหล่ง(ยกเว้นเบอร์ 8032/8052 มี 6 แหล่ง) โดยสามารถตั้งระดับความสำคัญได้ 2 ระดับคือ ระดับสูงและระดับต่ำ

การอินเตอร์รัพต์ความสำคัญต่ำ สามารถที่จะถูกอินเตอร์รัพต์จากอินเตอร์รัพต์ตัวอื่นที่มีระดับความสำคัญสูงได้ แต่ไม่สามารถที่จะถูกอินเตอร์รัพต์จากอินเตอร์รัพต์ตัวอื่นที่มีระดับเดียวกันได้แต่ถ้าเป็นการอินเตอร์รัพต์จากตัวที่มีความสำคัญสูงจะไม่สามารถถูกอินเตอร์รัพต์จากตัวอื่นได้เลย โดยเราสามารถตั้งระดับความสำคัญของแต่ละแหล่งอินเตอร์รัพต์ได้ที่รีจิสเตอร์ IP และถ้าในเหตุการณ์ที่มีการร้องขออินเตอร์รัพต์ในระดับความสำคัญเดียวกันเข้ามาพร้อมกัน ก็จะต้องมีการจัดระดับความสำคัญของแต่ละแหล่ง ดังตารางที่ 2-1

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

ตารางที่ 2-1 การจัดลำดับความสำคัญของแหล่งการอินเตอร์รัพต์ต่าง ๆ

โหมด 2 เป็นการรับส่งข้อมูลขนาด 11 บิตเข้าทางขา RXD และส่งออกทางขา TXD ประกอบด้วย 1 บิตสตาร์ตมีค่า "0",9 บิตข้อมูลและ 1 บิตสต็อป โดยการรับข้อมูลบิตที่ 9 จะถูกนำมาเก็บที่บิต RB8 ใน SCON ส่วนการส่งจะต้องใส่บิตที่ 9 ไว้ใน TB8 ของ SCON ก่อน อัตราบ็อดสามารถเลือกได้ 2 อัตราคือ 1/32 หรือ 1/64ของความถี่ออสซิลเลเตอร์ ซึ่งขึ้นอยู่กับการเซตบิต SMOD ในรีจิสเตอร์ PCON ซึ่งรายละเอียดของรีจิสเตอร์ SCON แสดงได้ดังนี้คือ

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 : ควบคุมอีนาเบิลการใช้โปรเซสเซอร์หลายตัวในการสื่อสารซึ่งกันและกันในโหมด 2 และ 3

REN : ตัวอีนาเบิลอนุกรมการรับเมื่อเซตเป็น "1" และถ้าเป็น "0" เป็นการดิสเอเบิล

TB8 : เป็นตัวเก็บข้อมูลบิตที่ 9 ที่จะส่งในโหมด 2 และ 3

RB8 : เป็นตัวรับข้อมูลบิตที่ 9 ในโหมด 2 และ 3 ส่วนในโหมด 1 จะเป็นสต็อปบิต

TI : เป็นแฟลกอินเตอร์รัพต์การเซตด้วยฮาร์ดแวร์ที่ปลายช่วงของบิตที่ 8 ในโหมด 0 หรือที่จุดเริ่มต้นของบิตสต็อปในโหมดอื่นในการส่งแบบอนุกรมของทุกโหมดจะต้องเคลียร์บิตนี้ด้วยโปรแกรมหลังการส่ง

RI : เป็นแฟลกอินเตอร์รัพต์การรับ เซตด้วยฮาร์ดแวร์ที่ปลายช่วงของบิตที่ 8 ในโหมด 0 หรือจุดครึ่งของช่วงบิตสต็อปในโหมดอื่น ในการรับแบบอนุกรม

ส่วนรายละเอียดของบิตควบคุมที่อยู่ภายในรีจิสเตอร์ TCON มีดังนี้คือ

TF_1	TR_1	TF_0	TR_0	IE_1	IT_1	IE_0	IT_0

TFx : เป็นแฟลกอินเตอร์รัพต์จะถูกเซตด้วยฮาร์ดแวร์เมื่อเกิดโอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับ และจะเคลียร์ตัวเองโดยอัตโนมัติ เมื่อทำงานอินเตอร์รัพต์นั้นเสร็จเรียบร้อยแล้ว

TRx : เป็นบิตควบคุมให้ตัวจับเวลา/ตัวนับเริ่มทำงานโดยการเซตให้เป็น "1" และให้หยุด ทำงานด้วยการเคลียร์ให้เป็น "0" โดยซอฟต์แวร์

IEx : เป็นแฟลกอินเตอร์รัพต์จากสัญญาณภายนอก เซตด้วยฮาร์ดแวร์เมื่อมีสัญญาณขอการอินเตอร์รัพต์ปรากฏที่ขา INTx และจะเคลียร์ตัวเองโดยอัตโนมัติ เมื่อกระโดดไปทำงานบริการอินเตอร์รัพต์ที่ขอมาเรียบร้อยแล้ว

ITx : เป็นบิตควบคุมรูปแบบสัญญาณอินเตอร์รัพต์ภายนอกจะเซต/เคลียร์ด้วยซอฟต์แวร์ โดยถ้าเซตเป็น "1" จะถูกอินเตอร์รัพต์ด้วยสัญญาณขอบขาลง และถ้าเคลียร์เป็น "0" จะถูกอินเตอร์รัพต์ด้วยสัญญาณระดับแรงดันต่ำ

หมายเหตุ x หมายถึงเลข 1 หรือเลข 0

พอร์ตอนุกรม

MCS-51 จะมีพอร์ตอนุกรมเป็นแบบฟูลดูเพล็กซ์ (full duplex) สามารถที่จะส่งและรับข้อมูลได้พร้อมกันเพราะมีบัฟเฟอร์ 2 ตัว ใช้ในการรับตัวหนึ่งและส่งตัวหนึ่ง โดยโครงสร้างของรีจิสเตอร์บัฟเฟอร์ทั้ง 2 ตัวนี้จะแยกกันแต่การติดต่อจะใช้ชื่อเดียวกันคือ SBUF พอร์ตอนุกรมของ MCS-51 สามารถที่จะโปรแกรมให้ทำงานได้แตกต่างกัน 4 โหมด คือ

โหมด 0 ข้อมูลจะเข้าและออกทางขา RXD โดยการเลื่อนสัญญาณนาฬิกาออกที่ขา TXD ข้อมูลจะเป็น 8 บิต โดยจะส่งบิตนัยสำคัญต่ำ (LSB) ก่อน อัตราบ็อดจะคงที่ที่ 1/12 ของความถี่ออสซิลเลเตอร์

โหมด 1 เป็นการรับ/ส่งข้อมูลขนาด 10 บิต โดยการส่งออกทางขา TXD และรับเข้าทางขา RXD รูปแบบบิตจะประกอบด้วย 1 บิตสตาร์ตเป็น "0" ,8 บิตข้อมูล และ 1 สต๊อปบิตเป็น "1" อัตราบ็อด (baud rate) แปรผันได้ตามการตั้งตัวจับเวลาตัวที่ 1 โดยมีสูตรดังนี้

อัตราบ็อด = $2^{SMOD}/32$ x ความถี่ออสซิลเลเตอร์/12 x (256 - TH1)

การถูกขัดจังหวะให้มาทำงานบริการอินเตอร์รัพต์แต่ละแหล่งก็จะต้องมีตำแหน่งที่จะกระโดดไปทำงานที่แน่นอน ซึ่งใน MCS-51 จะกำหนดที่อยู่ของตำแหน่งที่จะกระโดดไปทำงานเมื่อถูกอินเตอร์รัพต์จากแต่ละแหล่ง ดังแสดงในตารางที่ 2-2

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

ตารางที่ 2-2 ตำแหน่งที่จะกระโดดไปทำงานเมื่อถูกอินเตอร์รัพต์จากแต่ละแหล่ง

ส่วนรายละเอียดบิตควบคุมการอินเตอร์รัพต์ในรีจิสเตอร์ IE มีดังนี้คือ

EA	–	ET2	ES	ET1	EX1	ET0	EX0

EA : ถ้า EA = 0 จะดิสเอเบิลการอินเตอร์รัพต์ทั้งหมด ถ้า EA = 1 จะสามารถอินเตอร์รัพต์ได้ โดยแต่ละแหล่งอินเตอร์รัพต์จะมีอิสระในการเซต/เคลียร์ให้อีนาเบิลหรือดิสเอเบิลได้

ETx : จะเป็นบิตควบคุมการยอมรับอินเตอร์รัพต์โอเวอร์โฟลว์ของตัวจับเวลา/ตัวนับ x โดยถ้าเซตเป็น "1" จะยอมรับ แต่ถ้าเป็น "1" จะไม่ยอมรับ

ES : จะเป็นบิตควบคุมการยอมรับอินเตอร์รัพต์พอร์ตอนุกรม โดยถ้า ES = "0" จะยอมรับ แต่ถ้า ES เป็น "0" จะไม่ยอมรับ

EXx : เป็นบิตควบคุมการยอมรับอินเตอร์รัพต์ภายนอกจากขา INTx ถ้าเป็น "1" จะเป็นการยอมรับ แต่ถ้าเป็น "0" จะไม่ยอมรับ

ชุดคำสั่งของ MCS-51

ในหัวข้อนี้จะกล่าวถึงชุดคำสั่งของ MCS-51 ซึ่งถูกแบ่งเป็นลักษณะการทำงานตามฟังก์ชันได้ 5 กลุ่มคือ

กลุ่มการถ่ายเทข้อมูล
กลุ่มคณิตศาสตร์
กลุ่มตรรกศาสตร์
กลุ่มของบูลลีน
กลุ่มของการกระโดด

กลุ่มการถ่ายเทข้อมูล

แบ่งออกเป็น 2 ชุดคำสั่งคือ ชุดคำสั่งถ่ายเทข้อมูลใน RAM ภายใน กับชุดคำสั่งถ่ายเทข้อมูลใน RAM ภายนอก โดยมีรายละเอียดดังนี้

การถ่ายเทข้อมูลใน RAM ภายใน

ชุดคำสั่งของการถ่ายเทข้อมูลใน RAM ภายในนั้นแสดงได้ดังในตารางที่ 2-3

นีโมนิก	การกระทำ	โหมดการแอดเดรส			
		Dir	Ind	Reg	Imm
MOV A,<src>	A = <src>	x	x	x	x
MOV <dest>,A	<dest> = A	x	x	x	
MOV <dest>,<src>	<dest> = <src>	x	x	x	x
MOV DPTR,#data16	DPTR = 16bit immediate const.				x
PUSH <src>	INC SP:MOV"@SP",<src>	x			
POP <dest>	MOV<dest>,"@SP":DEC SP	x			
XCH A,<byte>	ACC and <byte> exchange data	x	x	x	
XCHD A,@Ri	ACCand@Ri exchange low nibble		x		

ตารางที่ 2-3 ชุดคำสั่งของการถ่ายเทข้อมูลใน RAM ภายใน

ซึ่งเวลาที่ใช้ในหนึ่งคำสั่งนั้นจะเป็นเวลาเมื่อขณะใช้ความถี่ในการทำงานของ CPU ที่ความถี่ 12 MHz และรายละเอียดของแต่ละคำสั่งมีดังนี้

MOV จะทำงานในลักษณะการถ่ายเทข้อมูลเป็นขนาดไบต์หรือบิตก็ได้ จากแหล่งกำเนิดเข้าสู่ตัวรับข้อมูลในฟิลด์โอเปอร์แรนด์

PUSH จะทำงานโดยเพิ่มค่าในรีจิสเตอร์ SP ก่อนแล้วจึงถ่ายข้อมูล 1 ไบต์จากแหล่งกำเนิดที่ฟิลด์โอเปอร์แรนด์กำหนดไว้ไปยังบริเวณสแต็กตามตำแหน่งที่รีจิสเตอร์ SP กำหนด

POP การถ่ายเทข้อมูลขนาด 1 ไบต์ จากบริเวณสแต็กตามตำแหน่งที่รีจิสเตอร์ SP กำหนดไปยังรีจิสเตอร์ที่โอเปอร์แรนด์กำหนดและหลังจากนั้นรีจิสเตอร์ SP จะลดค่าลงหนึ่ง

XCH คำสั่งแลกเปลี่ยนไบต์ระหว่างแหล่งกำเนิดโอเปอร์แรนด์กับรีจิสเตอร์ A

XCHD คำสั่งแลกเปลี่ยนขนาดนิบเบิลทางอันดับต่ำของแหล่งกำเนิดโอเปอร์แรนด์กับนิบเบิลอันดับต่ำของแอกคิวมูเลเตอร์

การถ่ายเทข้อมูลใน RAM ภายนอก

คำสั่งในการเคลื่อนย้ายข้อมูลใน RAM ภายนอกนั้นจะดูคำสั่งได้ในตารางที่ 2-4

นีโมนิก	การกระทำ	เวลาการเอ็กซีคิวต์ (μS)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	CALL subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

หมายเหตุ addr คือ address

ตารางที่ 2-4 คำสั่งในการเคลื่อนย้ายข้อมูลใน RAM ภายนอก

ซึ่งจะมีตัวชี้อยู่ 2 แบบคือ ใช้รีจิสเตอร์ R0 และ R1 เป็นตัวชี้และอีกแบบคือ ใช้รีจิสเตอร์ DPTR เป็นตัวชี้ซึ่งถ้าเราใช้ R1(R0 หรือ R1) เป็นตัวชี้นั้นจะชี้ได้เพียง 256 ไบต์เท่านั้น แต่ถ้าใช้ DPTR จะสามารถชี้ได้ถึง 64 กิโลไบต์ทีเดียว

กลุ่มคำสั่งทางคณิตศาสตร์

กลุ่มคำสั่งนี้แสดงได้ดังในตารางที่ 2-5

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=BxA	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

หมายเหตุ ACC คือ Accumulator

ตารางที่ 2-5 กลุ่มคำสั่งทางคณิตศาสตร์

ซึ่งช่วงเวลาการทำงานแต่ละคำสั่งนั้นกำหนดที่ความถี่สัญญาณนาฬิกา 12 MHz คำสั่งทางคณิตศาสตร์ส่วนใหญ่ใช้เวลา 1 μS ยกเว้นคำสั่ง INC DPTR ซึ่งใช้เวลา 2 μS คำสั่งคูณและหารใช้เวลา 4 μS

รายละเอียดการทำงานของแต่ละคำสั่งเป็นดังนี้

INC เป็นการบวกหนึ่งเข้ากับแหล่งกำเนิดโอเปอร์แรนด์และใส่ค่าใหม่กลับเข้าตัวโอเปอร์แรนด์

DEC เป็นการลบหนึ่งออกจากตัวเลขที่อยู่ในแหล่งกำเนิดโอเปอร์แรนด์และนำผลลัพธ์กลับมาเก็บที่ตัวโอเปอร์แรนด์นั้น

ADD เป็นการบวกค่าในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์

ADC เป็นการบวกค่าในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์และบวกกับบิตตัวทดด้วย

SUBB เป็นการนำตัวเลขที่แหล่งกำเนิดโอเปอร์แรนด์ลบออกจากตัวเลขใน A และนำค่าบิตตัวที่ทดมาลบออกอีกและได้ผลลัพธ์ใส่ลงในแอกคิวมูเลเตอร์ A

MUL จะเป็นการคูณแบบไม่คิดเครื่องหมายของตัวเลขที่อยู่ในแอกคิวมูเลเตอร์กับเลขในรีจิสเตอร์ B แล้วได้ผลลัพธ์ 2 ไบต์ นำเก็บไว้ที่ AB โดย A จะรับอันดับต่ำส่วน B จะรับอันดับสูง

DA สำหรับการบวกกันทางระบบตัวเลข BCD เป็นการปรับค่ารวม ซึ่งเป็นผลลัพธ์จากการบวกกันทางไบนารี่ของระบบตัวเลข BCD ขนาด 2 หลักสองจำนวน การปรับค่าตัวเลขผลรวมด้วยการใช้คำสั่ง DA จะได้ผลลัพธ์กลับมาที่แอกคิวมูเลเตอร์

DIV จะเป็นคำสั่งการหารแบบไม่คิดเครื่องหมายที่อยู่ในแอกคิวมูเลเตอร์หารด้วยตัวเลขในรีจิสเตอร์ B แล้วนำผลลัพธ์เก็บไว้ในแอกคิวมูเลเตอร์และเศษเหลือจะอยู่ในรีจิสเตอร์B

กลุ่มคำสั่งทางตรรกศาสตร์

กลุ่มคำสั่งทางตรรกศาสตร์แสดงได้ดังในตารางที่ 2-6

นีโมนิก	การกระทำ	โหมดการแอดเดรส				เวลาเอ็กซีคิวต์(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	ACC only				1
CPL A	A=.NOT.A	ACC only				1
RL A	Rotate ACC Left 1 bit	ACC only				1
RLC A	Rotate Left through C	ACC only				1
RR A	Rotate ACC Right 1 bit	ACC only				1
RRC A	Rotate Right through C	ACC only				1
SWAP A	Swap Nibbles in A	ACC only				1

หมายเหตุ ACC คือ Accumulator

C คือ Carry

ตารางที่ 2-6 กลุ่มคำสั่งทางตรรกศาสตร์

ซึ่งคำสั่งเหล่านี้สามารถที่จะทำงานในลักษณะของบูลลีนได้ทั้งขนาดไบต์หรือเป็นบิตรายละเอียดแต่ละคำสั่งเป็นดังนี้

CPL ด้วยการสั่งกลับค่าหรือคอมพลีเมนต์ (Complement) ข้อมูลในแอกคิวมูเลเตอร์โดยไม่มีผลใดๆต่อค่าแฟลกใน PSW หรือการให้ตำแหน่งแอดเดรสตามบิตนั้นๆ

RL,RLC,RR,RRC,SWAP ทั้ง 5 คำสั่งนี้เป็นการสั่งทำงานการวนบิต บนตัวแอกคิวมูเลเตอร์ซึ่ง RL เป็นการวนซ้าย,RR เป็นการวนขวา,RLC เป็นการวนซ้ายผ่านบิตทด,RRC เป็นการวนขวาผ่านบิตทด และ SWAP เป็นการวนซ้ายสี่ครั้ง

ANL เป็นการ AND กันทางตรรก ระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ ซึ่งจะสั่งให้ทำงานตรรกข้อมูลขนาดเป็นไบต์หรือบิตก็ได้

ORL เป็นการ OR ทางตรรกกันระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ ซึ่งจะสั่งให้ทำงานตรรกข้อมูลขนาดเป็นไบต์หรือบิตก็ได้

XRL เป็นการ XOR กันทางตรรกระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ ซึ่งจะสั่งให้ทำงานตรรกข้อมูลขนาดเป็นไบต์หรือบิตก็ได้

กลุ่มคำสั่งแบบบูลลีน

ชุดคำสั่งเกี่ยวกับบูลลีนแสดงได้ดังในตารางที่ 2-7

นีโมนิก	การกระทำ	เวลาเอ็กซีคิวต์ (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

ตารางที่ 2-7 กลุ่มคำสั่งแบบบูลลีน

ทุกคำสั่งใช้การเข้าถึงข้อมูลโดยตรงในระดับบิต โดยมีการใช้บิตแอดเดรสได้ตั้งแต่ 00-7FH ในพื้นที่ 128 บิต หน่วยความจำข้อมูลภายในและบิตแอดเดรส 80-FFHในบริเวณกลุ่มรีจิสเตอร์ฟังก์ชันพิเศษ (SFR)

กลุ่มของการกระโดด

ตามตารางที่ 2-8

นีโมนิก	การกระทำ	เวลาการเอ็กซีคิวต์ (μS)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	Call subroutine at addr.	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

หมายเหตุ addr คือ address

ตารางที่ 2-8 กลุ่มคำสั่งของการกระโดด

เป็นคำสั่งกระโดดโดยไม่มีข้อแม้ คำสั่ง JMP addr แบ่งเป็น 3 ลักษณะคือSJMP LJMP AJMP ซึ่งแต่ละคำสั่งจะมีข้อแตกต่างของการกระโดดไปยังแอดเดรสไกลสุดต่างกัน คำสั่ง JMP เป็นนีโมนิกที่สามารถจะใช้ได้ถ้าผู้ใช้ไม่คำนึงระยะใกล้ไกลของการกระโดดไป รายละเอียดของการกระโดดแต่ละอย่างมีดังนี้

SJMP จะเป็นการกระโดดแบบย้ายอันดับตำแหน่งเดิม (relative offset) จะกระโดดได้ -128 ถึง +127 ไบต์

AJMP แบบนี้จะกระโดดได้ 2 กิโลไบต์ ซึ่งจะใช้หน่วยความจำเพียง 2 ไบต์ เท่านั้นในการกำหนด

LJMP แบบนี้จะกระโดดได้ถึง 64 กิโลไบต์ ซึ่งจะใช้หน่วยความจำถึง 3 ไบต์ในการกำหนด

JMP @ A + DPTR เป็นการควบคุมให้กระโดดไปยังโปรแกรมที่ต้องการเฉพาะภายในส่วนต่างๆ

CWK (CARRIER WAVE KEYING)

การส่งสัญญาณดิจิตอลที่มีแรงดันต่ำไปตาม AC LINE จะนำคลื่นสัญญาณข้อมูลมามอดูเลทกับสัญญาณพัลซ์ ซึ่งผลที่ได้คือ เอาท์พุทของสัญญาณจะประกอบด้วยสัญญาณพัลซ์ที่มีแอมปลิจูดขึ้นอยู่กับคลื่นสัญญาณข้อมูล โดยที่ความกว้างของพัลซ์และความถี่ยังมีค่าคงเดิม

หลักการของ CWK ที่ใช้ในโครงงานนี้ประกอบด้วยวงจรแหล่งกำเนิดสัญญาณที่มีความถี่ไม่สูงมากนัก เพื่อนำไปมอดูเลทกับสัญญาณพัลซ์ที่ได้มาจากวงจรสร้างพัลซ์ วงจรมอดูเลทที่ใช้ก็เป็นวงจรริปเปิลเคาน์เตอร์ที่ได้รับการทริกจากวงจรขยายสัญญาณข้อมูล ซึ่งเอาท์พุตที่ได้จากวงจรริปเปิลเคาน์เตอร์จะมีลักษณะแอมปลิจูดของพัลซ์ขึ้นอยู่กับสัญญาณข้อมูล ซึ่งเป็นลักษณะของ CWK ตามที่กล่าวมาแล้วในย่อหน้าแรก

หมายเหตุ สัญญาณที่มอดูเลทแล้วจะถูกฝากไปกับสัญญาณไฟฟ้าภายในบ้านที่มีความถี่ 50Hz โดยจะถูกขยายและผ่านชุดเชื่อมต่อกับเอซีไลน์ก่อนจะส่งออกไป

ส่วนการดีมอดูเลชัน (Demodulation) จะใช้ไดโอดดีเทคและวงจรชมิททริกเกอร์ นั่นคือ ถ้ามีสัญญาณ 100 KHz เข้ามาก็จะมีสัญญาณเอาท์พุตออกไป แต่ถ้าไม่มีสัญญาณ 100KHz เข้ามาแล้วละก็ สัญญาณเอาท์พุตก็จะไม่มี โดยรายละเอียดของสิ่งที่กล่าวมาแล้วข้างต้นสามารถที่จะศึกษาได้จากบทที่ 6

อินฟราเรดเซนเซอร์

อินฟราเรดเซนเซอร์ที่ใช้ในโครงงานนี้อาศัยการเปลี่ยนแปลงของรังสีอินฟราเรดแต่ไม่ใช่แบบวิธีก่อนๆ ที่ใช้ตัวส่งพัลซ์และตัวตรวจจับการเปลี่ยนแปลงของพัลซ์ที่ถูกส่งออกมา แต่จะเป็นชนิดที่เรียกว่า "single end type" คือมีแต่ตัวรับเท่านั้นโดยอาศัยหลักการที่เรียกว่า "passive infrared detector" คือการตรวจจับการเปลี่ยนแปลงความร้อนจากการเปลี่ยนแปลงของรังสีอินฟราเรดที่แผ่ออกมาจากตัวของคนในขณะที่มีการเคลื่อนไหว

โดยปกติในตัวคนจะมีพลังงานความร้อนแผ่ออกมาในปริมาณที่แน่นอนอยู่จำนวนหนึ่ง แต่เมื่อเกิดการเคลื่อนที่หรือเคลื่อนไหว จะเป็นเหตุให้อุณหภูมิในบริเวณนั้นเปลี่ยนแปลงไป ซึ่งคลื่นรังสีความร้อนที่แผ่กระจายออกมาจะครอบคลุมแถบความถี่ในย่าน คลื่นแสงและคลื่นวิทยุประมาณ 0.74-300 ไมโครเมตร ซึ่งปกติเราเรียกแถบความถี่นี้ว่า "แถบอินฟราเรด (infrared region)" พลังงานที่แผ่ออกมาจะถูกตรวจจับด้วยตัวตรวจจับที่เรียกว่า

"pyroelectric" โดยมีเลนส์กรองแสงเฟรชนอลเลนส์ (fresnel lens) เป็นตัวช่วยตัดแสงที่ไม่ต้องการออก และยอมให้แสงในย่านที่ตามองไม่เห็น (อินฟราเรด) ผ่านไปได้ อีกทั้งยังเป็นตัวกำหนดรัศมีของการตรวจจับให้ได้ความไวสูงขึ้นอีกด้วย

เฟรชนอลเลนส์

ทำด้วยวัสดุพลาสติกสังเคราะห์ชนิดพิเศษจะยอมให้เฉพาะแสงที่ตาคนมองไม่เห็น (แสงอินฟราเรด)ผ่านเข้าไปได้เท่านั้น ประกอบด้วยเลนส์ทั้งหมด 24 ชิ้นรวมกัน ทำให้มุมในการรับแสงกว้างถึง 90 องศา

คุณสมบัติ (Specification)
โบโรซิลิเคต พลาสติก
ดัชนีหักเหแสง (Refractive Index) = 1.52 ที่ 10.6 ไมครอน
โฟกัสแสง (Focus) = 2.5 ซ.ม.
ผิวหน้าเรียบหันออกไปยังด้านแสงเข้า
ผิวในโค้งหันเข้าตัวรับแสง

การใช้งาน
- ตัดแสงรบกวนให้ตัวรับแสงอินฟราเรดทุกชนิดจากแสงที่ตามองเห็น
- เพิ่มระยะในการรับหรือตรวจจับของไพโรอิเล็กทริก
- เพิ่มระยะในการรับของตัวรับแสงอินฟราเรด เช่น อินฟราเรดโฟโตทรานซิสเตอร์,อินฟราเรดโฟโตไดโอดและโฟโตโวตาอิค
- เพิ่มระยะในการบังคับของอินฟราเรดรีโมทคอนโทรล เช่น ทีวี,วีดีโอ เปิดปิดประตูหรือโรงรถ หรือระบบกันขโมย
- วางเลนส์ในแนวตรงห่างจากตัวรับ 2.5 ซ.ม. ถ้าต้องการให้มุมกว้างขึ้นสามารถวางเลนส์ในแนวโค้งรัศมี 2.5 ซ.ม.

ไพโรอิเล็กทริก

โครงสร้างภายในของตัวตรวจจับแบบไพโรอิเล็กทริกมีส่วนประกอบที่สำคัญๆ คือ

1. ตัวไวแสงที่ทำจากผลึกของลิเทียมซัลเฟต 2 ชุด และ
2. เฟ็ตอีก 1 ตัว ประกอบเข้าด้วยกันภายในตัวถังแบบ TO-5

ชิ้นของผลึกแร่แต่ละตัวจะมีขนาดประมาณ 2x1 มม. ต่ออนุกรมกันอยู่แต่ต่อกลับขั้วกัน เมื่อมันถูกทำให้ร้อนจะเกิดการประจุไฟฟ้าที่ผิวของทั้ง 2 ด้านที่อยู่ตรงข้ามกัน ดังนั้นเมื่อมีสัญญาณใดๆ (ซึ่งเป็นสัญญาณที่มันจะตอบสนองได้โดยขึ้นอยู่กับคุณสมบัติเฉพาะของตัวมัน) มาตกกระทบตัวมันเข้า*อย่างต่อเนื่อง*จะเป็นเหตุให้สัญญาณลบและบวกถูกผลิตขึ้น ซึ่งสัญญาณที่ถูกผลิตขึ้นนี้จะเกิดการเปลี่ยนแปลงเป็นช่วงกว้างมากจากยอดถึงยอดของสัญญาณ แต่เนื่องจากผลึกทั้ง 2 ต่อกลับขั้วกันอยู่ จึงทำให้ผลรวมของทั้ง 2 สัญญาณหักล้างกันหมดไป ลักษณะดังกล่าวนับเป็นผลดีอย่างมากในอันที่จะป้องกันการทำงานในสภาวะที่เราไม่ต้องการ เช่น

- แสงรบกวนจากภายนอก เช่น แสงแดดซึ่งจัดเป็นแสงที่ต่อเนื่อง
- การเปลี่ยนแปลงอุณหภูมิเนื่องจากสภาวะแวดล้อมซึ่งโดยปกติไม่ได้เปลี่ยนในทันทีทันใด
- การเปลี่ยนแปลงเนื่องจากการสั่นสะเทือน ที่ไม่ได้เกิดจากคน

ตัวไพโรอิเล็กตริกบรรจุอยู่ในตัวถังแบบ TO-5 โดยมีช่องเพื่อให้ลำแสงอินฟราเรดผ่านเข้าไปได้ เมื่อใดก็ตามที่สัญญาณที่ผ่านเข้าไปตกกระทบยังตัวคริสตอลทั้งสองเป็นสัญญาณที่มีการเปลี่ยนแปลงอย่างไม่ต่อเนื่องและไม่สม่ำเสมอจะทำให้เกิดค่าความแตกต่างเกิดขึ้นระหว่างผลึกทั้ง 2 ชิ้น ทำให้ได้สัญญาณจำนวนหนึ่งมาขยายให้แรงขึ้นและป้อนเข้าวงจรรับ-ส่งข้อมูลทางเอซีไลน์ ซึ่งรายละเอียดดูได้ในบทที่ 6

บทที่ 3
ลักษณะของโครงงาน HOME AUTOMATIC

ปกติแล้วชีวิตประจำวันของคนเรา มีการกระทำหลายอย่างเป็นไปในลักษณะของวัฏจักรในรอบวันหรือในรอบสัปดาห์ เช่น การทำงานในวันจันทร์ถึงวันศุกร์และหยุดพักผ่อนในวันเสาร์และวันอาทิตย์ การใช้อุปกรณ์ไฟฟ้าบางอย่างก็มีการใช้งานเป็นวัฏจักรของเวลา เช่น การใช้เครื่องปรับอากาศในสำนักงาน ซึ่งจะเปิดใช้ตั้งแต่วันจันทร์ถึงวันศุกร์และจะเปิดใช้งานก่อนเวลาครึ่งชั่วโมง เพื่อให้เย็นในเวลาเริ่มทำงาน การปิดก็จะปิดก่อนเวลาเลิกประมาณครึ่งชั่วโมง วัฏจักรของการปิดเปิดเครื่องทำความเย็นนี้เป็นวัฏจักรที่ค่อนข้างแน่นอนเช่นเดียวกับวัฏจักรของเครื่องใช้ไฟฟ้าชนิดอื่น ๆ

โครงงานชุดนี้ จึงได้ออกแบบมาเพื่อให้สามารถทำงานแทนวัฏจักรที่ซ้ำซากเหล่านี้ โดยผู้ใช้สามารถตั้งโปรแกรมการทำงานเป็นวัฏจักรใน 1 สัปดาห์ และสามารถเลือกวันที่ต้องการให้ทำงาน, เวลาที่ทำงานละเอียดในหน่วยของนาที โดยกำหนดการปิดเปิดกี่ครั้งก็ได้ในแต่ละวันแต่ก็จะถูกจำกัดด้วยขนาดของหน่วยความจำและจำนวนโปรแกรมที่จะสามารถจะกำหนดใช้งานได้ นอกจากนั้นยังเพิ่มความสะดวกสบายให้แก่ผู้ใช้ในการปิดเปิดอุปกรณ์ไฟฟ้าที่เป็นหลอดไฟ ซึ่งมีการใช้งานไม่ค่อยจะเป็นวัฏจักรมากนัก โดยการทำงานของชุดอินฟราเรดเซนเซอร์

เพราะฉะนั้นโครงงานชุดนี้จึงเหมาะสมกับการนำมาใช้ในบ้านเรือนหรือสำนักงาน โดยได้ออกแบบให้มีขนาดที่เหมาะสมและง่ายต่อการใช้งาน ซึ่งประกอบด้วย

1. ชุดคอนโทรลเลอร์
2. ชุดที่เชื่อมต่อกับโหลด
3. ชุดอินฟราเรดเซนเซอร์

โดยแต่ละชุดจะประกอบด้วยวงจรที่ทำหน้าที่รับและส่งข้อมูลตามเอชีไลน์อยู่ในชุดเดียวกันเพื่อให้ระบบรับและส่งข้อมูลตามเอชีไลน์ในโครงงานชุดนี้เป็นแบบปิด (Close Loop) กล่าวคือ เมื่อมีข้อมูลส่งไปตามเอชีไลน์แล้ว เราสามารถรู้ได้ว่าแต่ละชุดทำงานได้ตามที่ต้องการหรือไม่ทำให้มีความสะดวกในการตรวจสอบ ซึ่งจะมีประโยชน์มากในกรณีที่แต่ละชุดอยู่ไกลจากกันมาก ๆ เช่น อยู่กันคนละชั้นของบ้าน เป็นต้น

แต่ชุดที่พิเศษไปกว่านั้นก็คือชุดอินฟราเรดเซนเซอร์ ซึ่งนอกจากจะประกอบด้วยวงจรที่ทำหน้าที่รับและส่งข้อมูลตามเอซีไลน์แล้วยังประกอบด้วยวงจรเซนเซอร์ที่ทำหน้าที่ตรวจจับว่าในห้องนั้น ๆ ของบ้านมีคนอยู่หรือไม่

ดังนั้นการออกแบบจึงแบ่งเป็นหัวข้อใหญ่ ๆ ได้ 4 หัวข้อด้วยกัน คือ

1. วงจรของ MCS-51 (อยู่ในชุดคอนโทรลเลอร์)
2. ส่วนของโปรแกรมมอนิเตอร์ (อยู่ในชุดคอนโทรลเลอร์)
3. วงจรรับ-ส่งข้อมูลตามเอซีไลน์
4. วงจรเซนเซอร์

การออกแบบวงจรของ MCS-51 จะถูกออกแบบขึ้นมาก่อนให้ตรงกับลักษณะโครงงานที่ต้องการ ซึ่งจะต้องออกแบบการต่อตัวไมโครคอนโทรลเลอร์ กับ อุปกรณ์สนับสนุนต่าง ๆ เพื่อเป็นอินพุตและเอาต์พุต คือ คีย์บอร์ดและภาคแสดงผล กับ เอาต์พุตที่ส่งไปควบคุมอุปกรณ์ภายนอก จากนั้นจึงโปรแกรมคำสั่งให้กับตัวไมโครคอนโทรลเลอร์ โดยเรียกโปรแกรมส่วนนี้ว่า โปรแกรมมอนิเตอร์ (Program Monitor) จึงจะทำให้เครื่องทำงานตามที่เราต้องการได้ แล้วจึงมาออกแบบวงจรรับ-ส่งข้อมูลตามเอซีไลน์และวงจรเซนเซอร์ซึ่งเป็นส่วนที่สำคัญมากเช่นเดียวกัน

บทที่ 4
การออกแบบทางฮาร์ดแวร์

การออกแบบทางฮาร์ดแวร์ที่ดีนั้นควรหาอุปกรณ์ที่เหมาะสมกับลักษณะของวัตถุประสงค์ที่ต้องการและมีความสวยงาม ทนทานในการใช้งานในระยะเวลานาน อีกทั้งต้องประหยัดทางด้านราคาอีกด้วย

ดังนั้นจึงได้สรุปลักษณะของเครื่องว่ามีลักษณะอย่างไร เพื่อให้สามารถออกแบบฮาร์ดแวร์ได้อย่างเหมาะสม ดังนี้

1. เป็นนาฬิกาแสดงเวลาปัจจุบันได้
2. สามารถตั้งเวลาโปรแกรมการเปิด/ปิด อุปกรณ์ไฟฟ้าได้ล่วงหน้า
3. สามารถเปิด/ปิดอุปกรณ์ไฟฟ้า โดยการตรวจจับของเซนเซอร์ว่ามีคนอยู่หรือไม่

จากลักษณะโดยสรุปนี้ อุปกรณ์ที่เหมาะสม คือ อุปกรณ์ประเภทไมโครคอมพิวเตอร์ชิพเดียว ซึ่งทำให้ประหยัดอุปกรณ์เพอริเฟอรัลที่ต้องต่อเติมเพิ่มในกรณีที่ใช้ CPU โดด ๆ เช่น Z80 , 8080 หรือ 8035 กล่าวคือ สามารถประยุกต์อุปกรณ์ดังนี้

-DATA MEMORY

-TIMER/COUNTER

-I/O PORT

ซึ่งจากการศึกษาพบว่าอุปกรณ์ที่เหมาะสมคือ อุปกรณ์ในตระกูล MCS-51 ซึ่งมีมากมายหลายเบอร์ นับตั้งแต่ 8051,8052,8031,8032,8751 จึงเลือก 8031 ซึ่งสามารถต่อหน่วยความจำภายนอกได้และมีหน่วยความจำข้อมูลภายใน 128 x 8 Byte จึงได้เลือกใช้ EPROM 2764 ควบคู่กับ 74LS373 ซึ่งเป็น LATCH สำหรับเป็นหน่วยความจำภายนอก และใช้ STATIC RAM 6264 สำหรับเป็นหน่วยความจำข้อมูลภายนอกสำหรับผู้ใช้ในการโปรแกรมเวลาเปิด/ปิดอุปกรณ์ไฟฟ้า

การใช้งานพอร์ทต่าง ๆ ของ 8031

- P1.0-P1.3 INPUT KEYBOARD
- PORT 2 & PORT 0 ใช้สำหรับอ้างอิงตำแหน่งหน่วยความจำภายนอก
- PORT 3 ใช้เป็นขาควบคุมต่าง ๆ

การใช้งานพอร์ทต่าง ๆ ของ 8255

- PORT A เป็นอินพุทพอร์ทขนาดแปดบิท ใช้รับข้อมูลที่ FEEDBACK สภาวะเปิด/ปิดของโหลด หรือสภาวะการตรวจจับของเซนเซอร์ ว่ามีคนอยู่หรือไม่

- PORT B ว่าง ไม่มีการใช้งาน
- PORT C PC.0-PC.4 การกำหนดหมายเลขอุปกรณ์ไฟฟ้าที่จะควบคุม

  PC.5 เป็นตัวสั่งโหลดว่าให้เปิด/ปิด (1/0)

  PC.6 เป็นตัวบอกว่า สัญญาณที่ส่งไปเป็นการสั่งเปิด/ปิดโหลดหรือถามสภาวะเปิด/ปิดของโหลด

  PC.7 เป็นตัวกำหนดว่า ให้มีการส่งสัญญาณออกไปจากคอนโทรลเลอร์หรือไม่ (ส่ง "0" ; ไม่ส่ง "1")

การทำงานของวงจรในส่วนฮาร์ดแวร์

(การต่อพ่วงอุปกรณ์ภายนอกเข้ากับไมโครคอนโทรลเลอร์)

1. การใช้งานร่วมกับ EPROM และ STATIC RAM ภายนอก

การพัฒนาโปรแกรมเพื่อให้เครื่องต้นแบบที่ใช้ไมโครคอนโทรลเลอร์ทำงานได้ตามข้อกำหนดต่าง ๆ อย่างไม่ผิดพลาด ส่วนใหญ่จะพัฒนาโปรแกรมด้วยการให้ไมโครคอนโทรลเลอร์ทำงานตามโปรแกรมที่เก็บอยู่ภายนอก ซึ่งใช้ EPROM 2764 แต่ระหว่างการพัฒนาจะใช้ EPROM EMULATOR ที่รับส่งข้อมูลอนุกรมกับ IBM PC เพื่อเป็นความสะดวกในการอ่านเขียนและแก้ไขโปรแกรมพร้อมกับการต่อวงจร STATIC RAM (6116) ขนาด 2K Byte ภายนอกเพื่อใช้ในการเก็บข้อมูลโปรแกรมผู้ใช้ (USER PROGRAM)

โดยที่วงจรจะบังคับให้ขา $\overline{EA}$ มีสถานะต่ำ เพื่อเป็นการบอกให้ไมโครคอนโทรลเลอร์ทำงาน โปรแกรมภายนอกซึ่งอยู่ใน EPROM ขนาด 8K x 8 Byte ตัว LATCH 74LS373 จะทำการ LATCH ข้อมูลแอดเดรสไว้ เนื่องจากขา DB0-DB7 เป็นแบบมัลติเพลกซ์ข้อมูลและแอดเดรส ดังนั้นจึงต้อง LATCH แยกเอาค่าแอดเดรสมาแสดงที่ขาเอาท์พุทของตัว 74LS373 ไปถอดรหัสโปรแกรมในตัว EPROM 2764 เพื่อให้ตัวไมโครคอนโทรลเลอร์ สามารถทำงานตามโปรแกรมภายนอกได้อย่างถูกต้อง การใช้โปรแกรมจากภายนอกไมโครคอนโทรลเลอร์จะทำให้ตัวไมโครคอนโทรลเลอร์ มีขาพอร์ทที่น้อยไป คือ ขา P0.0-P0.7 และP2.0-P2.7 ซึ่งโปรแกรมภายนอกนี้จะกำหนดให้อยู่ในตำแหน่ง 0000H-1FFFH โดยใช้สัญญาณจากขา 2Y0 ของวงจรถอดรหัส 74LS156 ซึ่งถอดรหัสจากขาของ 8031 มาทำการกำหนดตำแหน่งที่ขา CE ของ EPROM 2764

ส่วนการเข้าถึงข้อมูล RAM จะใช้ขา RD และ WR เป็นขาควบคุมในการอ่านและเขียนตามลำดับ ซึ่งหน่วยความจำข้อมูลภายนอกนี้จะกำหนดให้อยู่ในตำแหน่ง 2000H-

27FFH โดยใช้สัญญาณจากขา 2Y1 ของวงจรถอดรหัส 74LS156 มาทำการกำหนดตำแหน่งโดยป้อนสัญญาณเข้าที่ขา CE ของ STATIC RAM 6116

2. การขยายพอร์ทเพื่อเพิ่มจำนวนอินพุทและเอาท์พุท

เนื่องจาก PORT 2 และ PORT 0 ของไมโครคอนโทรลเลอร์ ใช้ในการรับส่งข้อมูลและกำหนดตำแหน่งของ EPROM 2764 และ STATIC RAM 6116 ทำให้เหลือพอร์ทในการใช้งานเพียงพอร์ทเดียว (PORT 1) จึงจำเป็นต้องต่อวงจรขยายพอร์ทเพื่อให้เป็นอินพุทและเอาท์พุทให้กับส่วนควบคุมนี้

การต่อขาแอดเดรสของพอร์ท 8255 คือ A0 และ A1ไม่สามารถจัดการกับการมัลติเพลกซ์แอดเดรสกับข้อมูลเองได้ จึงต้องต่อกับขาเอาท์พุทของตัว LATCH 74LS373 ที่ใช้ในการแอดเดรสข้อมูลโปรแกรม EPROM ภายนอกให้ได้ค่าการถอดรหัสแอดเดรสA0,A1 ได้สี่ตำแหน่ง เพื่อใช้ในการส่งข้อมูลออก และเข้าพอร์ททั้งสามของ 8255 และอีกหนึ่งแอดเดรสที่เหลือจะใช้ในการจัดการส่ง CONTROL WORD เข้าที่ 8255 ในการอ้างอิงการกำหนดตำแหน่งของพอร์ท จะกำหนดให้อยู่ในตำแหน่ง 0E0E0H-0E0E3H โดยใช้สัญญาณจากขา Y7 ของวงจรถอดรหัส 74LS138 ซึ่งถอดรหัสจากขา DA5-DA7 มาเข้าขา CS ของ 8255 และใช้สัญญาณจาก 1Y3 ของ 74LS156 ซึ่งถอดรหัสจากขา DA13-DA15 มาเข้าขา G2B ของ 74LS138 ด้วย ซึ่งในการต่อพอร์ททั้งสาม (A,B,C) ไปใช้งานนั้นได้ต่อ LATCH (74LS373) ในพอร์ท A และ B เพื่อใช้ในการรับและส่งข้อมูลและเพื่อป้องกันการผิดพลาดด้วย

ในการส่ง CONTROL WORD เพื่อทำการกำหนดพอร์ททั้งสามนั้นใช้ค่า 90H เพื่อกำหนดให้พอร์ท A เป็นอินพุท โดยใช้เพียงสองบิท คือ

A.6 เป็นตัวบอกว่ามีข้อมูลส่งมาจากโหลดหรือเซนเซอร์ เรียบร้อยแล้ว

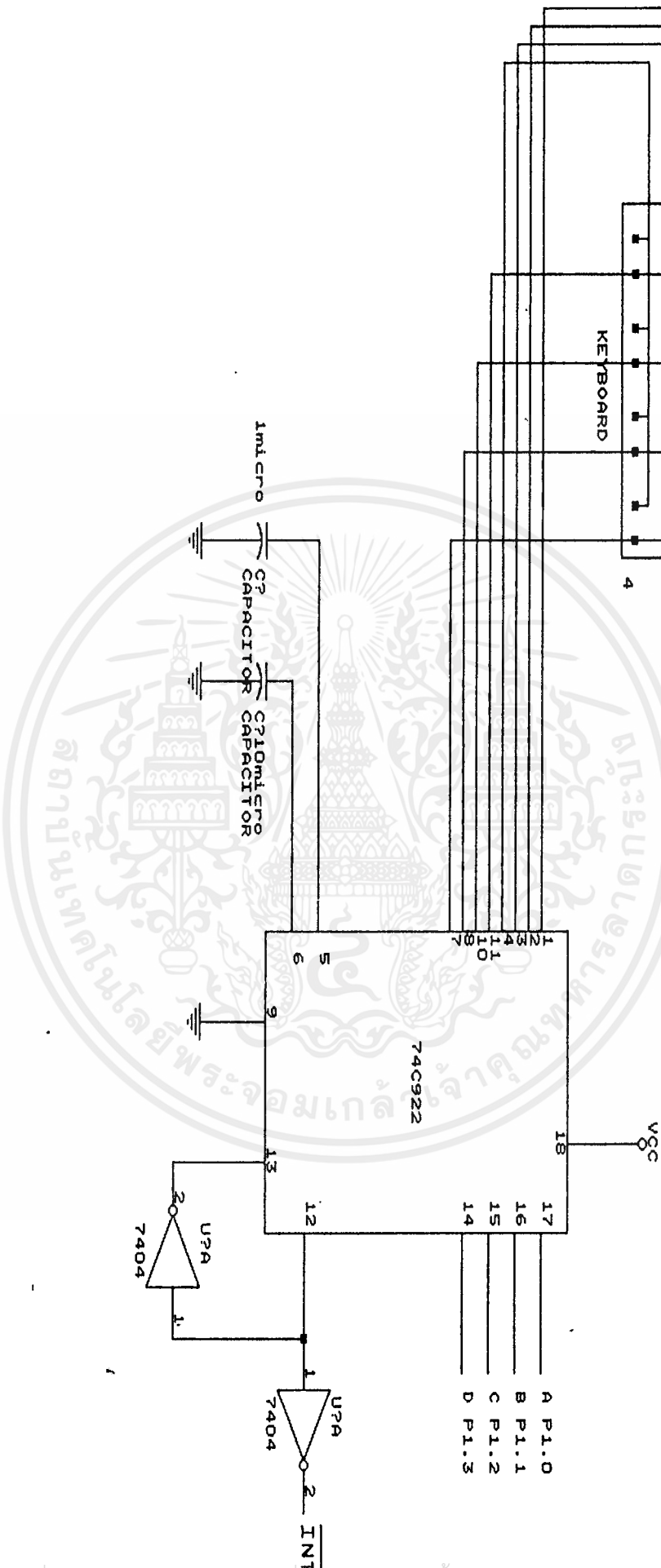
A.5 เป็นตัวบอกว่าโหลดมีการเปิด/ปิดตามคำสั่งหรือยัง หรือบอกสภาวะการตรวจจับของเซนเซอร์ว่ามีคนหรือไม่

และพอร์ท C เป็นเอาท์พุทพอร์ทที่ใช้ในการควบคุมการเปิด/ปิด อุปกรณ์ไฟฟ้า โดยกำหนดให้บิท PC.0-PC.4 เป็นรหัสหมายเลขอุปกรณ์ PC.5 ให้สถานะ "0" เมื่อต้องการปิดอุปกรณ์ไฟฟ้าและ PC.5 ให้สถานะ "1" เมื่อต้องการเปิดอุปกรณ์ไฟฟ้า โดยการส่งจะสัมพันธ์กับ PC.6 ที่เป็น "1" ถ้า PC.6 เป็น "0"แล้วจะเป็นการส่งสัญญาณเพื่อบอกให้โหลดหรือเซนเซอร์ส่ง FEEDBACK มาและ PC.7 จะเป็นบิทที่ทำการ ENABLE ให้มีการส่งสัญญาณเมื่อเป็น "0" และเมื่อเป็น "1" จะเลิกส่งสัญญาณ

3. การใช้งานอินเตอร์รัพ INT0 และ INT1

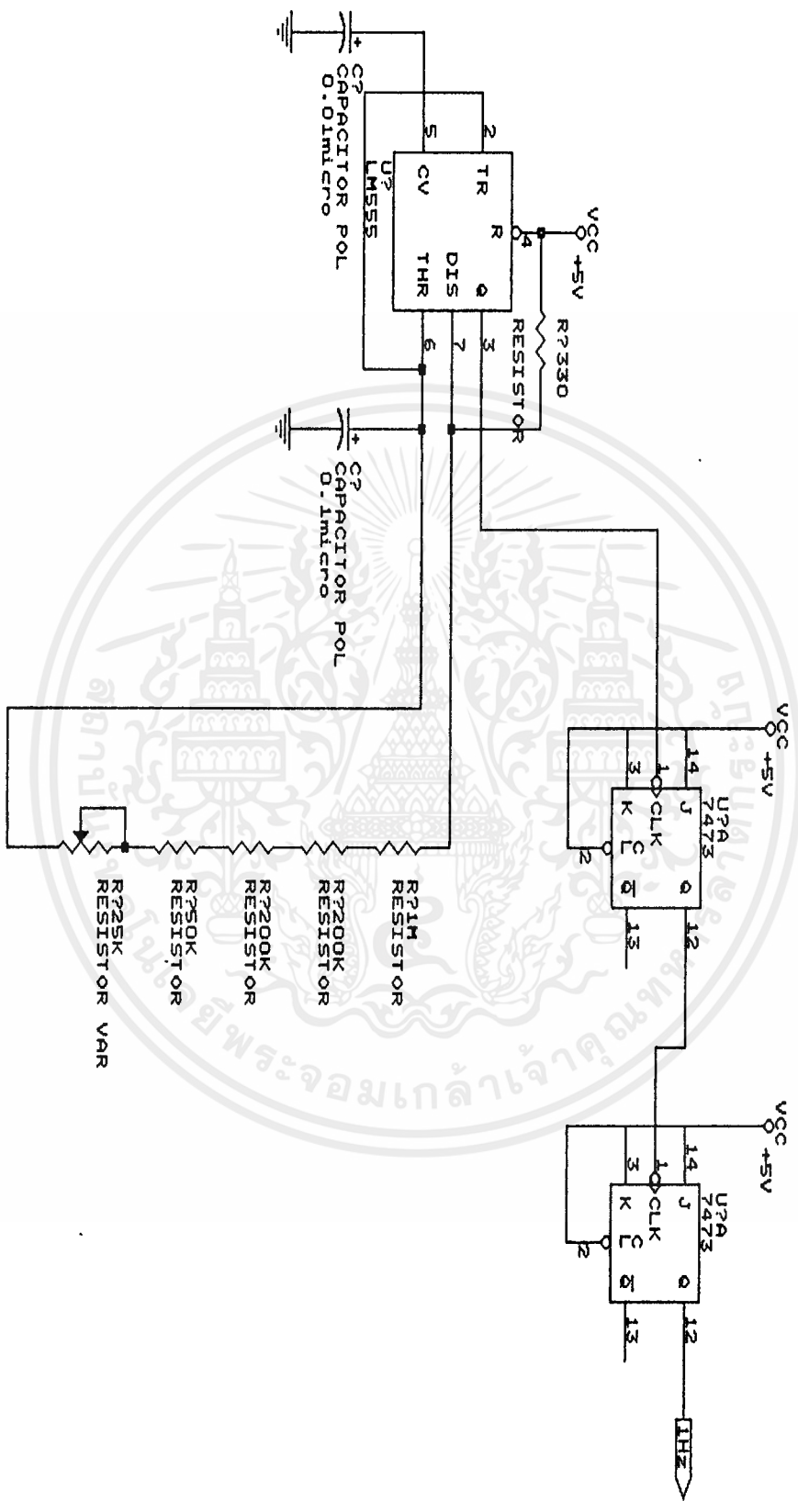
เมื่อมีการร้องขอการอินเตอร์รัพจากการต่อใช้วงจรอินเตอร์รัพหลายแห่ง ตัว MCS-51 สามารถที่จะจัดการการตอบรับการอินเตอร์รัพของแหล่งต่าง ๆ ได้ จากการกำหนดความสัมพันธ์ของการอินเตอร์รัพโดยให้ IP มีค่า 05H เพื่อให้การอินเตอร์รัพจาก INT0 มีความสำคัญกว่า INT1 และเนื่องจากสัญญาณการอินเตอร์รัพจาก INT0 และ INT1 มีลักษณะเป็น PULSE จึงกำหนดให้การตอบสนองการกระตุ้นการอินเตอร์รัพเป็นแบบขอบขาลง โดยให้ TCON มีค่า 0DH

การทำงานของ INT1 นั้น เป็นการทำงานในส่วนของการต่อวงจรคีย์บอร์ดแบบ MATRIX ขนาด 4x4 มาทำการแปลงรหัสโดยผ่านวงจรถอดรหัส 74C922 เพื่อทำการถอดรหัสขนาด 8 บิทให้เหลือ 4 บิท แล้วส่งข้อมูลจากคีย์บอร์ดขนาด 4 บิท เข้าทาง P1.0-P1.3 พร้อมกับส่งสัญญาณ ENABLE มายังขา INT1 เมื่อมีการกดคีย์หนึ่งครั้ง เพื่อแจ้งให้ไมโครคอนโทรลเลอร์ทราบที่ตำแหน่ง 0013H ของ PROGRAM MONITOR เพื่อทำการบันทึกข้อมูลลงคีย์บอร์ด ก่อนจะกระโดดไปทำโปรแกรมเดิม ตามรูปที่ 4-1



รูปที่ 4-1 การใช้งานอินเตอร์รัพต์ INT 1 (วงจร KEYBOARD INTERRUPT)

ส่วนการทำงานของ INTO เป็นการทำงานในส่วนของวงจรฐานเวลา (BASE TIMER) ในการสร้างวงจรนาฬิกา ให้กับส่วนควบคุมนี้ โดยการผลิตสัญญาณฐานเวลามาตรฐานขนาด 1Hz (1 วินาที) ซึ่งได้จากวงจร ASTABLE MULTIVIBRATOR ที่ต่อด้วย TIMER IC 555 มาทำการอินเตอร์รัพที่ขา INTO ทุก ๆ หนึ่งวินาที เพื่อแจ้งให้ไมโครคอนโทรลเลอร์ทราบ และการทำงานของโปรแกรมก็จะกระโดด ไปทำโปรแกรมเพิ่มค่าเวลา (INCREMENT) ที่ตำแหน่ง 0013H ของ PROGRAM MONITOR เพื่อทำการบันทึกค่าเวลาลงใน TIMER BUFFER และตรวจสอบสภาวะต้นนาที เพื่อใช้ในการโปรแกรมเปิด/ปิด อุปกรณ์ไฟฟ้าก่อนจะกระโดดไปทำโปรแกรมเดิม ตามรูปที่ 4-2



รูปที่ 4-2 การใช้งานอินเตอร์รัพต์ INT 0 (วงจร TIMER INTERRUPT)

4. การต่อใช้งานวงจรแสดงผลแบบ LCD

ในการออกแบบได้เลือกใช้ DOT MATRIX LCD MODULE ขนาด 7x5 DOT จำนวน 16 ตัวอักษร ซึ่งในการใช้งาน LCD นั้นสามารถใช้งานได้ง่าย เนื่องจากภาคแสดงผล LCD กับวงจรควบคุมการทำงาน LCD HD 44780 นั้นอยู่บนโมดูลเดียวกันสามารถใช้งานได้โดยการส่งข้อมูลจาก MCS-51 ได้โดยตรงโดยการอ้างอิงตำแหน่งของการติดต่อที่ A0, A1 เข้ากับขา R/W และ RS ตามลำดับ พร้อมทั้งสามารถควบคุมความสว่างโดยการปรับ VR1 ที่ต่อกับขา Vo ของ LCD MODULE

ในการต่อขาแอดเดรสของ LCD HD 44780 คือ A0 และ A1 ไม่สามารถจัดการการมัลติเพลกซ์แอดเดรสกับข้อมูลเองได้ จึงต้องต่อกับขาเอาท์พุทตัว LATCH 74LS373 ที่ใช้ในการแอดเดรสข้อมูลโปรแกรม EPROM ภายนอกให้ได้ค่าการถอดรหัสแอดเดรส A0,A1 ได้สี่ตำแหน่งเพื่อใช้ในการส่งข้อมูลเข้า LCD, การตรวจสอบความพร้อมของ LCD ก่อนทำการส่งข้อมูลและการควบคุมการทำงานของ LCD ตามฟังก์ชันที่ต้องการโดยส่ง CONTROL WORD

ซึ่งในการอ้างอิงการกำหนดตำแหน่งของ LCD (HD44780) จะกำหนดให้อยู่ในตำแหน่ง 0E0C0H-0E0C3H โดยใช้สัญญาณจากขา Y6 ของวงจรถอดรหัส 74LS138 ซึ่งถอดรหัสจากขา DA5-DA7 มาเข้าขา E ของ LCD และสัญญาณจากขา G2B ของ 74LS138 ที่ได้จากขา 1Y3 ของ 74LS156 ซึ่งถอดรหัสจากขา DA13-DA15 ด้วย

ส่วนประกอบอุปกรณ์ต่าง ๆ

การต่อพ่วงอุปกรณ์ภายนอกของไมโครคอนโทรลเลอร์ทั้งสี่ส่วนที่กล่าวมานี้สามารถทำการศึกษาทีละส่วนได้ และ เมื่อนำมาประกอบกัน ก็จะสามารถทราบได้ ถึงการทำงานโดยรวมของวงจรส่วนควบคุมทั้งหมด ซึ่งในการประกอบวงจรทั้งหมด ได้แบ่งเป็นสองส่วนดังนี้

1. ส่วนของอุปกรณ์หลัก ประกอบด้วย

- MICROCONTROLLER	8031
- EPROM	2764
- STATIC RAM	6116
- PORT	8255
- DECODE	74LS138, 74LS156
- LATCH	74LS373

- REGULATOR LM7805
- X'TAL 11.059 MHz

2. ส่วนของอุปกรณ์สนับสนุน ประกอบด้วย

- LATCH (PORT A & C) 74LS373 x 2
- J-K FLIP FLOP 74LS73
- NOT GATE 74LS04
- TIMER 555
- DECODE (KEYBOARD) 74C922
- LCD MODULE HD44780
- KEYBOARD SW.1-SW.

บทที่ 5
โปรแกรมมอนิเตอร์

โปรแกรมมอนิเตอร์ เป็นส่วนสำคัญส่วนหนึ่งของเครื่องเพราะเป็นส่วนควบคุมให้เครื่องทำงานได้ตามต้องการ ส่วนของโปรแกรมนี้เขียนขึ้นโดยมีความสัมพันธ์กับระบบทางฮาร์ดแวร์ต่าง ๆ และโปรแกรมมอนิเตอร์นี้จะเก็บไว้ในส่วนของหน่วยความจำโปรแกรม (EPROM 2764) คือเมื่อเริ่มเปิดเครื่องให้ทำงาน มีการ Reset CPU ก็จะเริ่มทำงานตามโปรแกรมส่วนนี้

โครงสร้างของโปรแกรม

โปรแกรมมอนิเตอร์นี้สามารถแบ่งออกได้เป็น 4 ส่วนใหญ่ ๆ คือ

- INITIAL PROGRAM
- MAIN PROGRAM
- TIMER INTERRUPT PROGRAM
- KEYBOARD INTERRUPT PROGRAM

INITIAL PROGRAM (initial)

เป็นโปรแกรมที่เตรียมค่าเริ่มต้นต่าง ๆ ภายในหน่วยความจำเพื่อการทำงาน ,สภาวะเริ่มต้นเพื่อการทำงานสำหรับ LCD display,กำหนดการใช้งานของ PORT 8255 ให้ PORT A เป็น input PORT C เป็น output และให้ค่าเวลาเริ่มต้นเป็น SUN 00:00:00 และ set ค่าในหน่วยความจำภายนอกเป็น #00h ทั้งหมด แล้วจะกระโดดไปทำ MAIN PROGRAM

MAIN PROGRAM (main)

CPU ปรกติแล้วจะทำงานใน MAIN PROGRAM ตลอดเวลา โดยนำค่า วัน, ชั่วโมง,นาที และวินาทีใน TIMER BUFFER มาแสดงเวลาปัจจุบันโดยเรียกโปรแกรมย่อย transfer และ display,การลดค่าทุกวินาทีจากการกำหนดเวลาเมื่อมีการใช้ SENSOR ในการตรวจจับ และตรวจสอบว่ามีการกด KEYBOARD เรียกใช้ FUNCTION ใดหรือไม่ จากนั้นจะตรวจสอบว่ามีการสั่งให้ RUN โปรแกรมไหนหรือไม่ (1-4) ถ้าไม่มีก็จะทำงานตามสภาวะ SENSOR ซึ่งการทำงานตอนเปิดเครื่องครั้งแรก ก็จะเป็นการทำงานตามสภาวะ SENSOR

TIMER INTERRUPT PROGRAM (walk)

ฐานเวลาได้จากวงจร Astable multivibrator โดยใช้ TIMER IC 555 สร้างเป็นสัญญาณความถี่ 1 Hz เข้าที่ขา INT 0 เมื่อมีสัญญาณ INTERRUPT จะเพิ่มค่าเวลา วินาที,นาที,ชั่วโมง และวันใน TIMER BUFFER และทำการ set ค่า bit 7 ของ newsm ทุก 1 วินาทีสำหรับใช้ในการแสดงเวลา set ค่า bit 0 ทุก 1 นาที สำหรับใช้ในการตรวจสอบ USER PROGRAM

KEYBOARD INTERRUPT PROGRAM

เป็นโปรแกรมที่ทำงานเมื่อได้รับสัญญาณ INTERRUPT จาก KEYBOARD ที่ขา INT 1 จะรับข้อมูลจาก PORT 1 และ set ค่า bit 7 เพื่อบอกว่ามีการกดคีย์และเก็บลง kbd

นอกจากนี้ยังมีโปรแกรมย่อยต่าง ๆ ประกอบ ที่ช่วยสนับสนุนการทำงานของโปรแกรมหลัก จะอธิบายโดยย่อเรียงตามลำดับไปจากต้นโปรแกรมดังนี้

clear: ทำการลบหน้าจอ LCD และ CURSOR อยู่ที่ตำแหน่งซ้ายสุด

waitbf: รอการทำงานทาง Hardware ของ LCD เพื่อพร้อมในการรับหรือส่งข้อมูล

write: เขียนตัวอักษรที่ต้องการออกยังจอ LCD

ddram: กำหนดตำแหน่งจอ LCD ที่จะเขียนตัวอักษร

transfer: ทำการแปลงรหัสเวลาวินาที,นาที,ชั่วโมง และวัน จากเลขฐานสิบหกเป็นรหัส ASCII และเก็บลง Display buffer (address 40h-4fh)

forward: เปลี่ยนรหัสตัวเลขที่รับจาก Keyboard สองตัวเป็น ASCII เตรียมเพื่อแสดงที่จอ LCD

store: ทำการเก็บรหัส ASCII ลง Display buffer

digit: อ่านตัวเลขสองตัวจาก table เป็นรหัส ASCII และเก็บลง Display buffer

colon: เก็บ ":" ลง Display buffer

space: เก็บ " " ลง Display buffer

display: นำตัวอักษรทั้ง 16 ตัวจาก Display buffer แสดงออกที่ LCD

chkkbd: เช็คว่า Keyboard ใดถูกกดและจะทำงานตาม Function การทำงานนั้น

pro: จะถามให้ User ให้เลือกการทำงานเมื่อปุ่ม [PRO] ถูกกด

not: ทำการเช็คว่ามี Program number นั้นอยู่หรือไม่ เมื่อมีการสั่งให้ run หรือ scan และจะบอกให้ User ทราบถ้าไม่มี Program นั้นอยู่

number: ให้ค่า Address เริ่มต้นของ Program number ที่ถูกเลือกแก่ dptr

dayaddr: ให้ค่า Address เริ่มต้นของวันที่ถูกเลือกแก่ dptr

userdata: เป็นโปรแกรมที่ทำหน้าที่รับข้อมูลจาก User ในการตั้งเวลาให้ Load ทำงานในวันต่าง ๆ

kd: ให้ User ป้อนข้อมูลกำหนดวันและแสดงออกจอ LCD

kl: ให้ User ป้อนข้อมูลกำหนด Load number และแสดงออกจอ LCD

lc: รับค่าตัวเลขจาก Keyboard และแสดงออกจอ LCD โดยจำกัดค่าที่รับไม่ให้เกิน 24 และไม่รับค่า #00

clram: ให้ค่า #00 กับ External RAM จนถึง Address ที่ dptr มีค่า dph เท่ากับ r5 และ dpl เท่ากับ r4

decimal: เปลี่ยนรหัสตัวเลขสองตัวเป็น ASCII และเขียนออกจอ LCD

kept: เก็บข้อมูลการตั้งเวลา เปิด-ปิด โหลดลง data input buffer

input: เป็นโปรแกรมจัดการเกี่ยวกับการแสดงตัวอักษรที่จอ LCD เพื่อถามให้ User ป้อนข้อมูลตั้งเวลา เปิด-ปิด โหลดแต่ละตัวและเก็บข้อมูลลง data input buffer

form: เป็นการแสดงอักษรทั้งสิบหกที่กำหนดไว้ ก่อนหน้าจะเรียกโปรแกรมย่อยนี้ออกจอ LCD และแสดง Load number ที่เป็นอยู่ขณะนั้น

codein: เก็บข้อมูลลง data input buffer ในกรณีสั่งให้ทำงานตามสภาวะ SENSOR หรือไม่สั่ง ซึ่งเป็นการเก็บค่าเดียวกันทั้ง 4 ตำแหน่งของ buffer

scan: เป็นการนำ User data จาก RAM ออกมาแสดงในกรณีที่มีการเรียกใช้ฟังก์ชัน SCAN ของ Program number ใด และจะบอกให้ทราบถ้าไม่มีโปรแกรมนั้นอยู่

prong: ทำการลดค่า dptr ลง 1 เนื่องจากไม่มีคำสั่ง dec dptr ใช้

pai: เพิ่มค่า dptr สี่ครั้ง ใช้เพื่อข้ามไปอ่าน User data ชุดต่อไปของ SCAN

klab: ย้อนกลับไปอ่านข้อมูลของ Load number ที่ผ่านมาแล้วและถ้า User แก้ไขข้อมูล ก็จะนำข้อมูลใหม่ไปเก็บ

nosung: แสดงตัวอักษร No dictate ออกจอ LCD ให้ User ทราบ เมื่อไม่มีการสั่งงานใด ๆ กับ Load number นั้นไว้

pull: เป็นการอ่านค่าเวลาเปิดและปิดของ User data จาก External RAM มาแสดงออกจอ LCD

delete: ให้ค่า #00 กับ External RAM ตั้งแต่ตำแหน่งแรกถึงตำแหน่งสุดท้ายของ Program ในกรณีที่เรียกใช้ฟังก์ชัน CLEAR PROGRAM

runner: ทำหน้าที่ ปิด-ปิด โหลดตามโปรแกรมที่ตั้งไว้เมื่อเรียก Program number ใดทำงาน (RUN) และจะบอกให้ทราบถ้าไม่มีโปรแกรมนั้นอยู่

sense: ทำการ เปิด-ปิด โหลดตามสภาวะของ SENSOR

schedule: ทำการ เปิด-ปิด โหลดตามเวลาที่ตั้งไว้ใน User data

subsched: ทำการตรวจเช็คว่าเวลาในขณะนั้นตรงกับเวลาที่สั่งไว้ใน User data หรือไม่และถ้าตรงกันก็จะทำการ เปิด-ปิด โหลด ถูกเรียกใช้โดย schedule

compare: ทำการการเปรียบเทียบเวลาว่าตรงกันหรือไม่ ถูกเรียกใช้โดย subsched

delay: หน่วงเวลา ใช้ในการรอให้ Feed back ที่ส่งมาจากโหลด หรือ Sensor เสร็จเรียบร้อย ก่อนที่จะมีการส่งสัญญาณใด ๆ จาก Controller ไปใน AC-Line ถ้าไม่เช่นนั้นสัญญาณจะรบกวนกัน ทำให้การรับส่งข้อมูลผ่าน AC Line ขัดข้อง

working: แสดงตัวอักษร Dictating load ออกจอ LCD ในช่วงหน่วงเวลาที่มีการรับ-ส่ง ข้อมูลผ่าน AC-Line

on: ทำหน้าที่ส่งสัญญาณเข้า AC-Line ไปเปิดโหลด

off: ทำหน้าที่ส่งสัญญาณเข้า AC-Line ไปปิดโหลด

state: ทำหน้าที่ส่งสัญญาณเข้า AC-Line ไปบอกให้โหลด หรือ Sensor ส่ง Feed back และรอรับสัญญาณ สัญญาณที่รับได้จะเก็บใน F0 ของ PSW Register ถูกเรียกใช้โดย on และ off

load: เป็นการทำงานเมื่อมีการเรียกใช้ฟังก์ชัน LOAD โดยเรียกสภาวะของ Load number ใด ๆ ขึ้นมาดูและให้ User สั่ง เปิด-ปิด โหลดโดยตรงได้

situ: เป็นการอ่านสภาวะของโหลดและแสดงออกจอ LCD ถูกเรียกใช้โดย load

tell: เป็นการเขียนอักษร on หรือ off ออกจอ LCD ถูกเรียกใช้โดย load และ situ

sensor: เป็นการควบคุมการ เปิด-ปิด โหลดตามสภาวะของ Sensor เมื่อมีการเรียกใช้ฟังก์ชัน SENSOR

settime: เป็นโปรแกรมที่ให้ User ทำการตั้งเวลานาฬิกา ในกรณีที่เรียกใช้ฟังก์ชัน SETTIME

din: ให้ User กำหนดวันในการตั้งเวลาและแสดงออกจอ LCD

kr: ทำหน้าที่รับค่าจาก Keyboard โดยจำกัดค่าไม่ให้เกิน 8 และไม่ให้เป็น 0 ถูกเรียกใช้โดย din หรือโปรแกรมอื่นที่รับค่าวัน

sh: แสดงวันออกจอ LCD ถูกเรียกใช้โดย din หรือโปรแกรมอื่นที่แสดงวัน

hrin: รับค่าการตั้งเวลาชั่วโมงจาก Keyboard และแสดงออกจอ LCD

mosin: รับค่าการตั้งเวลานาที หรือวินาทีจาก Keyboard และแสดงออกจอ LCD

pattern: เป็นการนำอักษรที่กำหนดไว้ 16 ตัว โดยคำสั่ง .db (define byte)

ที่กำหนดไว้ก่อนหน้าที่จะเรียกโปรแกรมย่อยนี้เก็บเป็นรหัส ASCII ลง display buffer (address #40h-4fh)

wait: รอให้มีการกด Keyboard และเก็บค่าลง kbd

select: รอให้มีการกดปุ่ม [CLS] หรือ [RET] ถ้าปุ่ม [CLS] ถูกกดจะ set สภาวะ bit 7 ให้ทราบ

sixty: รับค่าจาก Keyboard โดยจำกัดค่าไม่ให้เกิน 59 และยอมรับค่า #00 ด้วย ถูกเรียกใช้โดย mosin

tfback: แปลงค่าตัวเลขสองตัวที่รับจาก Keyboard ให้เป็นเลขฐานสิบหก

cutout: ทำหน้าที่ปิดโหลดทุกตัว ในกรณีที่เรียกใช้ฟังก์ชัน CUTOUT

ตำแหน่งหน่วยความจำภายนอกและอุปกรณ์ต่าง ๆ

1. PROGRAM MONITOR (EPROM #2764)

หน่วยความจำโปรแกรมภายนอกขนาด 8K byte มี address อยู่ที่ 0000h ถึง 1FFFh

2. USER PROGRAM (#6116)

หน่วยความจำข้อมูลภายนอกขนาด 2k byte มี address อยู่ที่ 2000h ถึง 27FFh จะสามารถเก็บ USER PROGRAM ได้ 1 โปรแกรมซึ่งจะใช้เนื้อที่ 2017 byte (7e1h) ใน USER PROGRAM การเก็บค่าสั่ง ON/OFF 1 ครั้งจะใช้เนื้อที่ 4 byte เก็บเวลาชั่วโมงและนาทีในการ ON 2 byte ชั่วโมงและนาทีในการ OFF 2 byte Load number 1-8 สั่งได้ 1 ครั้ง Load number 9-24 สั่งได้ 4 ครั้ง ดังนั้น Load ทั้ง 24 ตัว สำหรับการสั่งงาน 1 วันจะใช้พื้นที่หน่วยความจำ 288 byte และ ทั้ง 7 วันจะใช้พื้นที่หน่วยความจำทั้งหมด 2016 byte และจะใช้ 1 byte สำหรับการตรวจเช็คว่ามี USER PROGRAM หรือไม่โดยมีรายละเอียดดังนี้

แต่ละชุด 4 byte มีการเก็บเป็น

- hour code ┐ ON
- min code ┘
- hour code ┐ OFF
- min code ┘

Program number 1

- byte 1 ใช้ตรวจเช็คว่ามี USER PROGRAM หรือไม่

วันอาทิตย์

- byte 2-5 เก็บข้อมูลของ Load number 1
- byte 6-9 เก็บข้อมูลของ Load number 2

.
.
.

- byte 30-33 เก็บข้อมูลของ Load number 8
- byte 34-49 เก็บข้อมูลของ Load number 9
- byte 50-65 เก็บข้อมูลของ Load number 10

.
.
.

- byte 274-289 เก็บข้อมูลของ Load number 24

.
.
.

วันเสาร์

- byte 1730-1733 เก็บข้อมูลของ Load number 1
- byte 1734-1737 เก็บข้อมูลของ Load number 2

.
.
.

- byte 1758-1761 เก็บข้อมูลของ Load number 8
- byte 1762-1777 เก็บข้อมูลของ Load number 9
- byte 1778-1793 เก็บข้อมูลของ Load number 10

.
.
.

- byte 2002-2017 เก็บข้อมูลของ Load number 24

3. DOT MATRIX LCD MODULE (#HD 44780)

เป็น LCD ขนาด 5x7 dot จำนวน 16 ตัวอักษร สามารถใช้งานได้โดยอ้างอิงตำแหน่งอุปกรณ์ภายนอกได้โดยตรงดังนี้

```
command    .equ 0e0c0h ; Read/Write register
readbusy   .equ 0e0c1h ; Read busy flag and Address
writedata  .equ 0e0c2h ; Write Character
Readdata   .equ 0e0c3h ; Read data from DDRAM
```

4. EXTERNAL PORT (#8255)

เป็น PORT ภายนอกที่มี INPUT/OUTPUT 3 PORT สามารถใช้งานได้โดยอ้างอิงตำแหน่งอุปกรณ์ภายนอกได้โดยตรงดังนี้

```
porta      .equ 0e0e0h ; INPUT PORT (รับ feed back)
portb      .equ 0e0e1h ; OUTPUT PORT (ไม่ได้ใช้งาน)
portc      .equ 0e0c2h ; OUTPUT PORT (ส่งสัญญาณให้ Load)
control    .equ 0e0c3h ; ควบคุมการใช้งาน PORT ทั้ง 3
```

ลักษณะการใช้หน่วยความจำภายในตำแหน่งต่าง ๆ

```
equate sec      30h : timer buffer 1
equate min      31h : timer buffer 2
equate hour     32h : timer buffer 3
equate day      33h : timer buffer 4
equate newsm    34h : บอกสภาวะต้นวินาที และต้นนาทีโดย set bit 7,0
                35h : timer input buffer 1 ┐ sec
                36h : timer input buffer 2 ┘
                37h : timer input buffer 3 ┐ min
                38h : timer input buffer 4 ┘
                39h : timer input buffer 5 ┐ hour
                3ah : timer input buffer 6 ┘
                3bh : timer input buffer 7   day
```

```
equate fnday   3ch : เก็บวันตาม User กำหนด ┐ ใช้ใน userdata
equate spday   3dh : เก็บวันก่อน User กำหนด ┘      และ scan
equate kbd     3eh : keyboard buffer
equate pno     3fh : เก็บค่า Program number
equate prld    40h : เก็บ Load number ตาม User กำหนด ┐ ใช้ใน scan
equate bfld    41h : เก็บ Load number ก่อน User กำหนด ┘ ,userdata
               42h : input buffer 1 ┐ ของ Load number
               43h : input buffer 2 ┘
equate hcon    44h : data input buffer 1 > hour code on
equate mcon    45h : data input buffer 2 > minute code on
equate hcoff   46h : data input buffer 3 > hour code off
equate mcoff   47h : data input buffer 4 > minute code off
equate sr1     48h : sensor 1 ┐
equate sr2     49h : sensor 2 │
equate sr3     4ah : sensor 3 │
equate sr4     4bh : sensor 4 ├ ใช้เป็นตัวตั้งเวลาให้กับ controller ใน
equate sr5     4ch : sensor 5 │ การตรวจเช็คสภาวะของ sensor แต่ละตัว
equate sr6     4dh : sensor 6 │ และจะถูกลดค่าทุกวินาทีใน main program
equate sr7     4eh : sensor 7 │
equate sr8     4fh : sensor 8 ┘
               50h : display buffer  1
               51h : display buffer  2
               52h : display buffer  3
               53h : display buffer  4
               54h : display buffer  5
               55h : display buffer  6
               56h : display buffer  7
               57h : display buffer  8
               58h : display buffer  9
               59h : display buffer 10
```

5ah : display buffer 11

5bh : display buffer 12

5ch : display buffer 13

5dh : display buffer 14

5eh : display buffer 15

5fh : display buffer 16

60h : load buffer 1

61h : load buffer 2

62h : load buffer 3

63h : load buffer 4

64h : load buffer 5

65h : load buffer 6

66h : load buffer 7

67h : load buffer 8

68h : load buffer 9

69h : load buffer 10

6ah : load buffer 11

6bh : load buffer 12

6ch : load buffer 13

6dh : load buffer 14

6eh : load buffer 15

6fh : load buffer 16

70h : load buffer 17

71h : load buffer 18

72h : load buffer 19

73h : load buffer 20

74h : load buffer 21

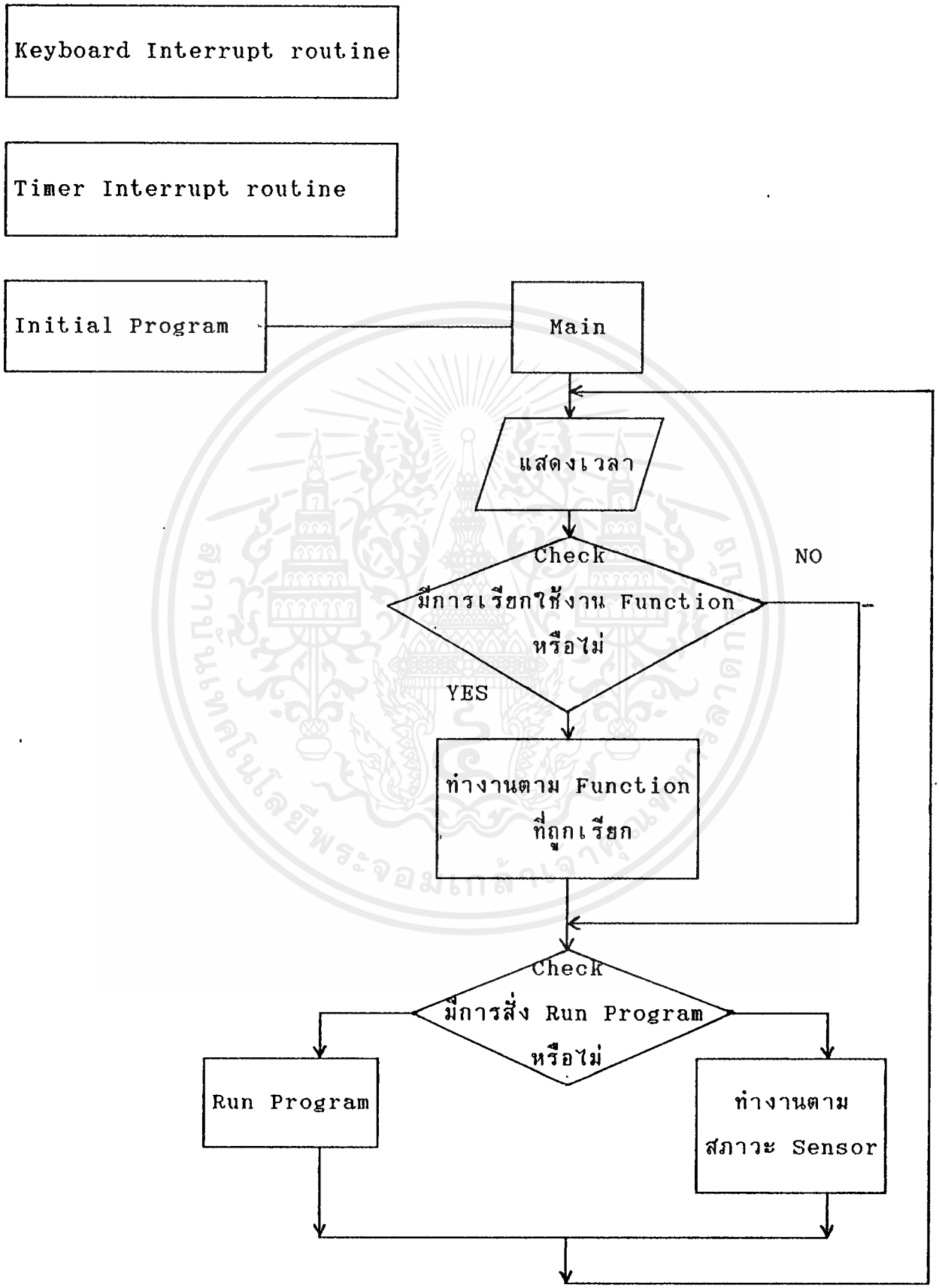
75h : load buffer 22

76h : load buffer 23

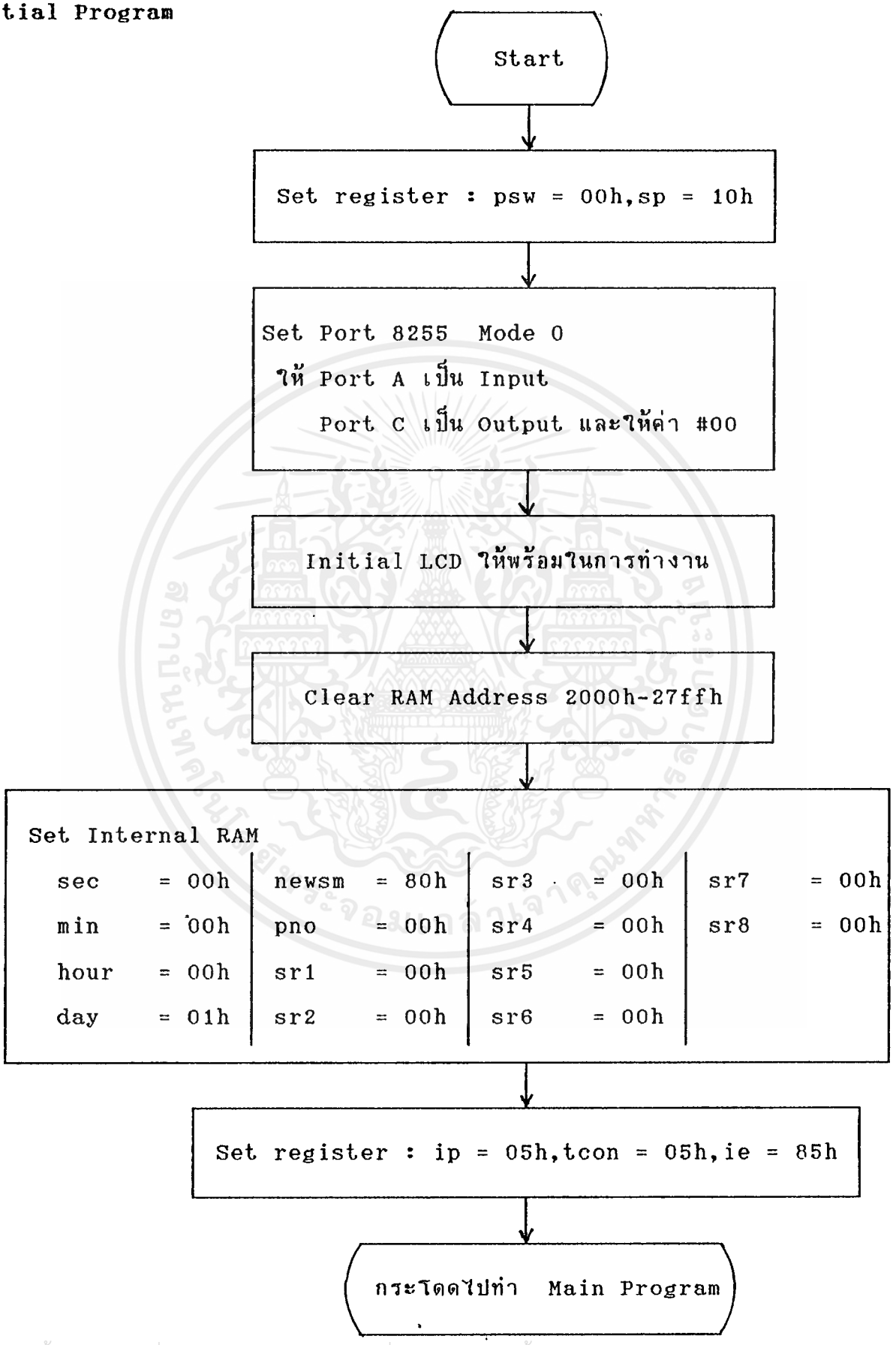
77h : load buffer 24

ใช้เก็บสภาวะ on/off ของโหลด

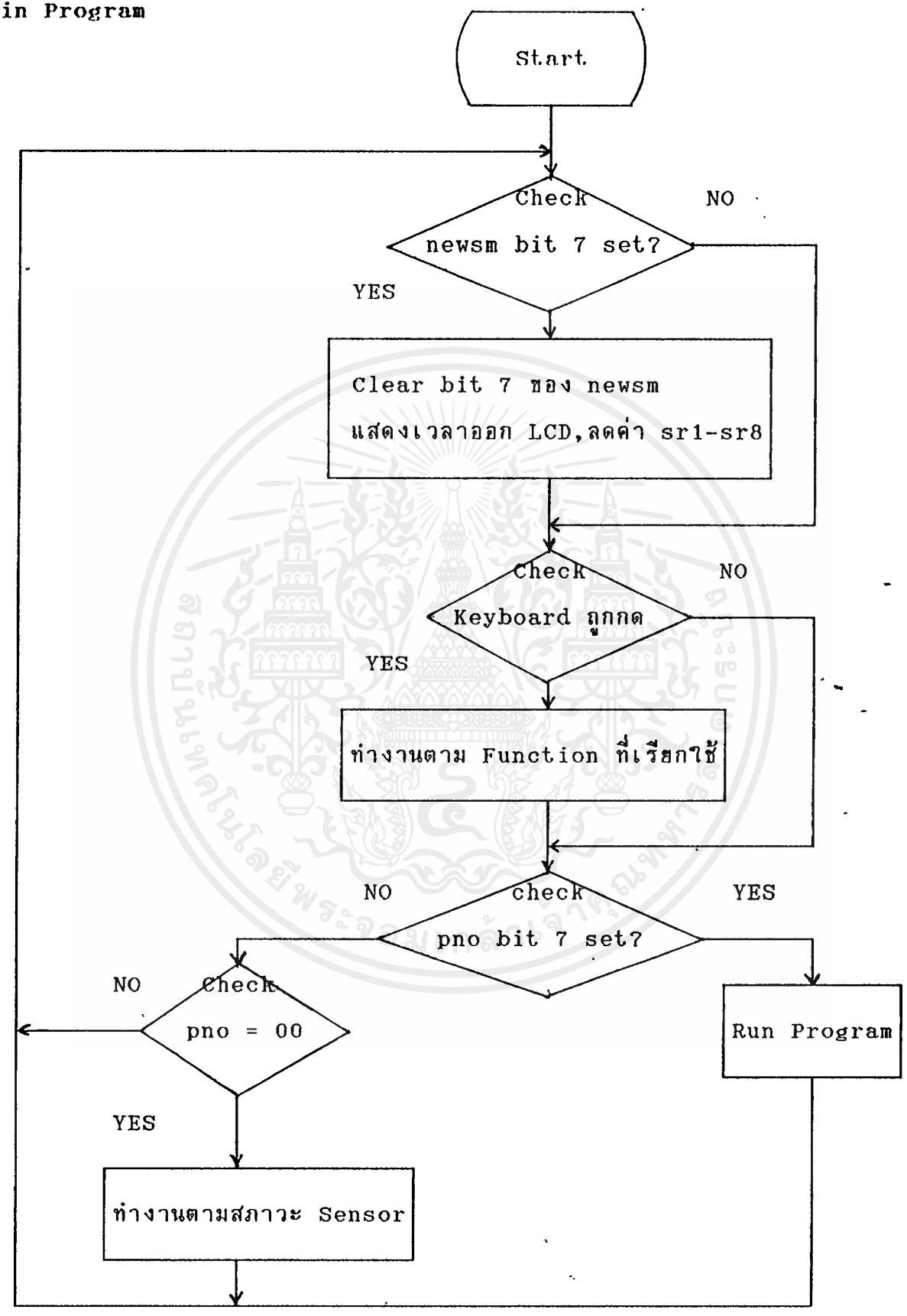
ผังการทำงานของโปรแกรมมอนิเตอร์



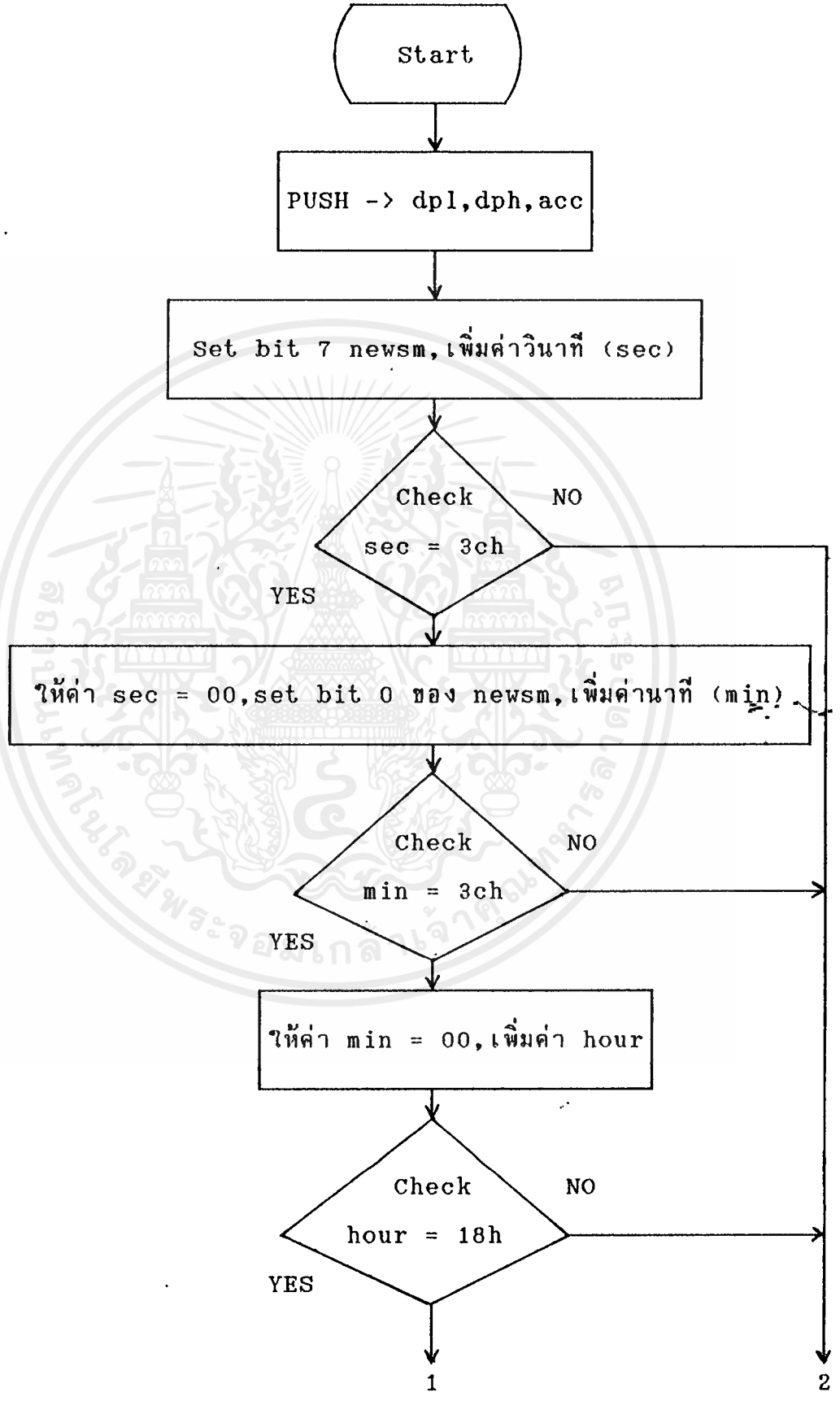
Initial Program

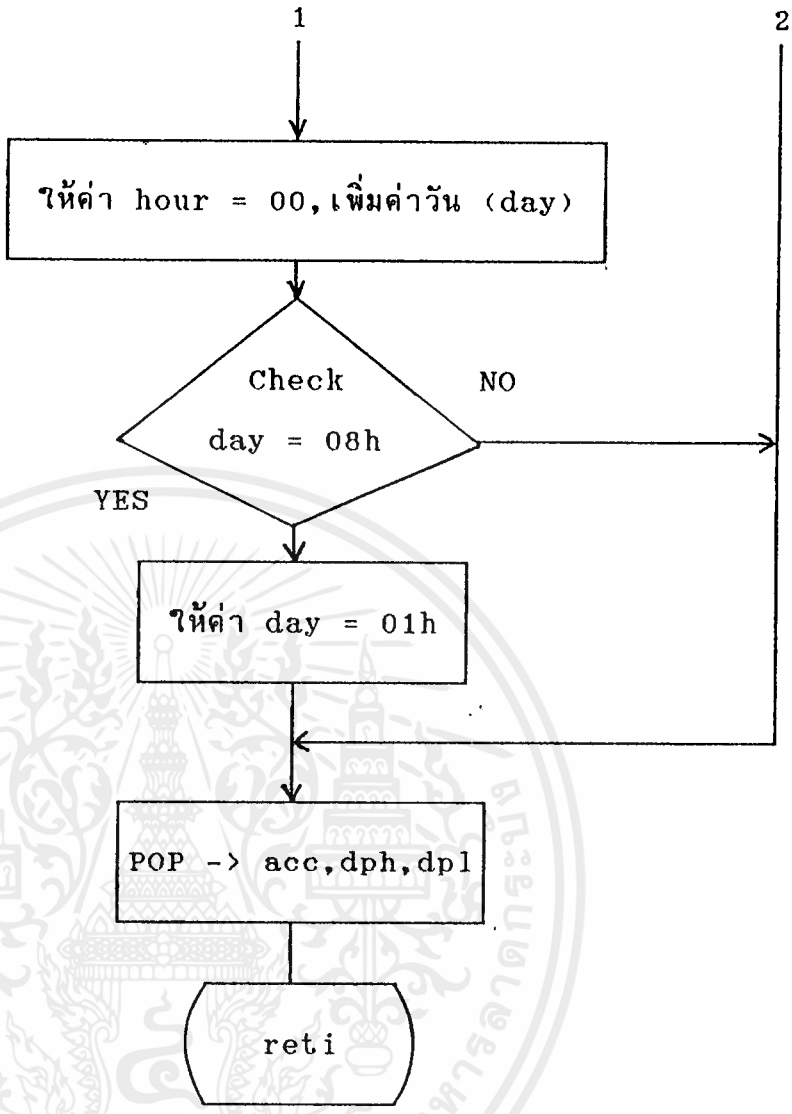


Main Program

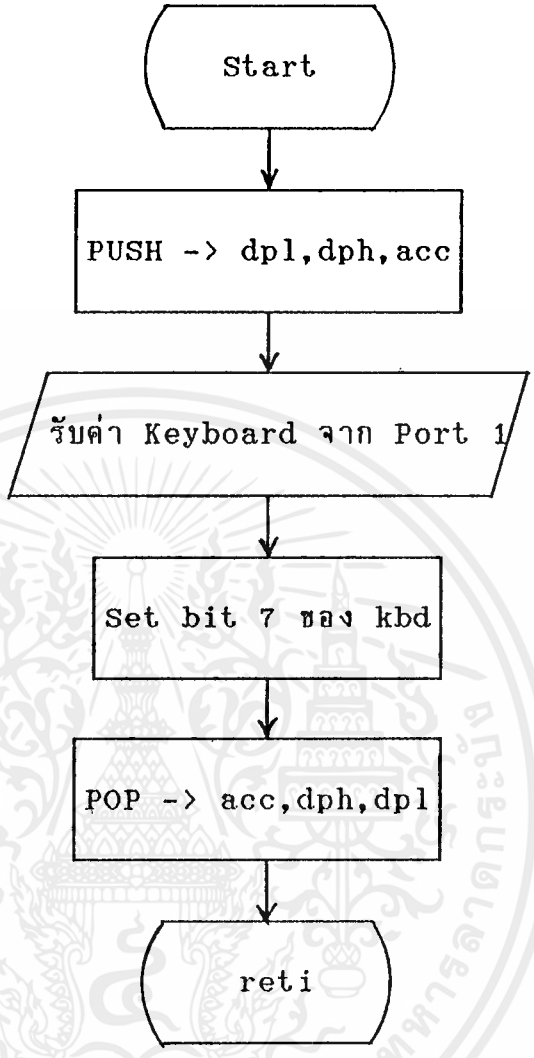


Timer Interrupt routine (walk)





Keyboard Interrupt routine (keyboard)



```
.org 0000h
.equ base,0e0h
.equ command,0c0h
.equ readbusy,0c1h
.equ writedata,0c2h
.equ readdata,0c3h
.equ porta,0e0h
.equ portb,0e1h
.equ portc,0e2h
.equ control,0e3h
.equ sec,30h
.equ min,31h
.equ hour,32h
.equ day,33h
.equ newsm,34h
.equ fnday,3ch
.equ spday,3dh
.equ kbd,3eh
.equ pno,3fh
.equ prld,40h
.equ bfld,41h
.equ hcon,44h
.equ mcon,45h
.equ hcoff,46h
.equ mcoff,47h
.equ sr1,48h
.equ sr2,49h
.equ sr3,4ah
.equ sr4,4bh
.equ sr5,4ch
```

```
        .equ sr6,4dh
        .equ sr7,4eh
        .equ sr8,4fh
        ajmp  initial
        .org  0003h
        ajmp  walk
        .org  0013h
        ajmp  keyboard

walk:   push  dpl           ;timer interrupt
        push  dph           ;routine
        push  acc
        mov   a,newsm       ;to show
        setb  acc.7         ;be interrupted
        mov   newsm,a       ;clear after
        mov   a,sec         ;write time to LCD
        inc   a
        mov   sec,a
        xrl   a,#3ch
        jnz   set
        mov   sec,#00h
        mov   a,newsm
        setb  acc.0
        mov   newsm,a
        mov   a,min
        inc   a
        mov   min,a
        xrl   a,#3ch
        jnz   set
        mov   min,#00h
```

```
        mov   a,hour
        inc   a
        mov   hour,a
        xrl   a,#18h
        jnz   set
        mov   hour,#00h
        mov   a,day
        inc   a
        mov   day,a
        xrl   a,#08h
        jnz   set
        mov   day,#01h

set:    pop   acc
        pop   dph
        pop   dpl
        reti

keyboard: push  dpl             ;keyboard
        push  dph               ;interrupt routine
        push  acc
        mov   a,p1
        anl   a,#0fh
        setb  acc,7
        mov   kbd,a
        pop   acc
        pop   dph
        pop   dpl
        reti
```

```
initial:  mov   psw,#00h      ;8 bytes debugger
          mov   sp,#0fh       ;protect SP
          mov   dph,#base     ;overlap debugger
          mov   dpl,#control
          mov   a,#90h        ;set 8255 mode #0
          movx  @dptr,a       ;A <= IN,C => OUT
          mov   dpl,#portc
          mov   a,#00h        ;set 8255
          movx  @dptr,a       ;portc #00h
          mov   dpl,#command
          mov   a,#38h        ;LCD function set
          movx  @dptr,a
          acall waitbf
          mov   a,#0fh        ;display control
          movx  @dptr,a
          acall waitbf
          mov   a,#6h         ;entry mode set
          movx  @dptr,a
          acall waitbf
          acall clear
          mov   dptr,#2000h   ;clear RAM addr.
          mov   r5,#3fh       ;of program 1 - 4
          mov   r4,#84h
          lcall clram
          mov   sec,#00h
          mov   min,#00h
          mov   hour,#00h
          mov   day,#01h
          mov   newsm,#80h
          mov   pno,#00h
```

```
        mov   sr1,#00h
        mov   sr2,#00h
        mov   sr3,#00h
        mov   sr4,#00h
        mov   sr5,#00h
        mov   sr6,#00h
        mov   sr7,#00h
        mov   sr8,#00h
        mov   ip,#05h          ;INT priority
        mov   tcon,#05h        ;falling edge INT
        mov   ie,#85h          ;interrupt enable
        ljmp  main
clear:  push  dpl
        push  dph
        push  acc
        mov   dph,#base
        mov   dpl,#command
        mov   a,#1
        movx  @dptr,a
        acall waitbf
        pop   acc
        pop   dph
        pop   dpl
        ret
waitbf: push  dpl              ;wait for LCD
        push  dph              ;hardware
        push  acc              ;operation
        mov   dph,#base        ;(check busy
        mov   dpl,#readbusy    ; flag)
```

```
rdy:    movx  a,@dptr
        jb    acc.7,rdy
        pop   acc
        pop   dph
        pop   dpl
        ret

write:  push  dpl             ;write character
        push  dph             ;to LCD
        mov   dph,#base       ;a is parameter
        mov   dpl,#writedata
        movx  @dptr,a
        acall waitbf
        pop   dph
        pop   dpl
        ret

ddram:  push  dpl             ;set DD RAM addr.
        push  dph             ;of LCD
        push  acc             ;r3 is address
        mov   dph,#base       ;parameter
        mov   dpl,#command
        mov   a,r3
        movx  @dptr,a
        acall waitbf
        pop   acc
        pop   dph
        pop   dpl
        ret
```

```
          ;change time code
          ;write to LCD
transfer: mov   r0,#30h        ;address sec
          mov   r1,#35h        ;transfer buffer
change:   acall forward
          cjne  r0,#33h,change ;day not done
          mov   a,@r0
          mov   b,#03h
          mul   ab
          add   a,#09h
          mov   @r1,a
          mov   r2,a           ;next character
          mov   r0,#50h        ;point to buffer
          mov   dptr,#table
          acall space
          acall space
char:     mov   a,r2
          movc  a,@a+dptr      ;write day char
          acall store          ;to display buffer
          inc   r2
          cjne  r0,#55h,char
          acall space
loop:     acall digit          ;write time to
          acall colon          ;display buffer
          cjne  r1,#35h,loop
          dec   r0
          acall space
          acall space
          ret
```

```
table:    .db   "0"
          .db   "1"
          .db   "2"
          .db   "3"
          .db   "4"
          .db   "5"
          .db   "6"
          .db   "7"
          .db   "8"
          .db   "9"
          .db   ":"
          .db   " "
          .db   "SUN"
          .db   "MON"
          .db   "TUE"
          .db   "WED"
          .db   "THU"
          .db   "FRI"
          .db   "SAT"
.
.
forward:  mov   a,@r0
          mov   b,#0ah
          div   ab              ;change for
          mov   @r1,b           ;display at LCD
          inc   r1
          mov   @r1,a
          inc   r0
          inc   r1
          ret
```

```
store:    mov   @r0,a
          inc   r0
          ret

          ;kept decimal No.
          ;ASCII (2 digit)
digit:    mov   dptr,#table    ;in LCD buffer
          mov   r4,#02h
repeat:   dec   r1
          mov   a,@r1
          movc  a,@a+dptr
          acall store
          djnz  r4,repeat
          ret

          ;kept ":"
          ;in LCD buffer
colon:    mov   dptr,#table
          mov   a,#0ah
          movc  a,@a+dptr
          acall store
          ret

          ;kept " "
          ;in LCD buffer
space:    mov   dptr,#table
          mov   a,#0bh
          movc  a,@a+dptr
          acall store
          ret
```

```
          ;write character
          ;to LCD (address-
          ;40h=>4fh)
display:  mov   psw,#08h
          acall clear
          mov   r3,#80h        ;r3 set addr. LCD
          acall ddram
          mov   r0,#50h
half1:    mov   a,@r0
          acall write
          inc   r0
          cjne  r0,#58h,half1
          mov   r3,#0c0h
          acall ddram
half2:    mov   a,@r0
          acall write
          inc   r0
          cjne  r0,#60h,half2
          mov   psw,#00h
          ret
main:     mov   a,newsm        ;write LCD only
          jnb   acc.7,one      ;one time in each
          clr   acc.7          ;timer interrupt
          mov   newsm,a        ;for ==> LCD
          acall transfer       ;not ripple
          acall display
          mov   r1,#48h
aoik:     mov   a,@r1
          jz    yip
          dec   @r1
```

```
yip:      inc   r1
          cjne  r1,#50h,aoik
one:      acall chkkbd
          mov   a,pno
          jb    acc.7,run
          jz    matic
          sjmp  main              ;delete in FACT !
matic:    lcall sensor
          sjmp  main
run:      lcall runner
          sjmp  main
chkkbd:   mov   a,kbd             ;check 4 function
          jnb   acc.7,checked     ;first key
          anl   a,#0fh            ;keyboard pressed
          mov   kbd,a
          xrl   a,#0ah            ;check PRO key
          jnz   func2
          lcall pro
          ajmp  checked
func2:    mov   a,kbd
          xrl   a,#0bh            ;check LD key
          jnz   func3
          lcall load
          ajmp  checked
func3:    mov   a,kbd
          xrl   a,#0ch            ;check S/S key
          jnz   func4
          mov   pno,#00h
          ajmp  checked
```

```
func4:    mov   a,kbd
          xrl   a,#0dh          ;check TIM key
          jnz   func5
          lcall settime
          ajmp  checked
func5:    mov   a,kbd
          xrl   a,#0eh          ;check CLS key
          jnz   checked
          lcall cutout
checked:  ret

pro:      mov   dptr,#askno
          lcall pattern
          lcall display
wrong:    mov   r3,#0c4h
          lcall ddram
          mov   r5,#05h         ;not over 4
          lcall wait
          mov   pno,a
          mov   dptr,#table
          movc  a,@a+dptr
          lcall write
          lcall select
          jb    acc.7,wrong
          mov   dptr,#what
          lcall pattern
          lcall display
```

```
subfunc:  mov   r5,#10h
          lcall wait
          xrl   a,#0ah
          jnz   sub2
          acall userdata        ;program
          sjmp  finish
sub2:     mov   a,kbd
          xrl   a,#0ch
          jnz   sub3
          acall scan            ;scan old program
          sjmp  finish
sub3:     mov   a,kbd
          xrl   a,#0eh
          jnz   sub4
          lcall delete          ;delete data
          sjmp  finish          ;user program
sub4:     mov   a,kbd
          xrl   a,#0fh
          jnz   subfunc
          mov   a,pno
          setb  acc.7
          mov   pno,a
finish:   ret

askno:    .db   "Program No.     "
what:     .db   "Function call ? "
```

```
not:      movx  a,@dptr
          jb    acc.7,yes
          mov   dptr,#told
          lcall pattern
          lcall display
          mov   r3,#0c3h
          acall ddram
          mov   a,pno
          clr   acc.7
          mov   pno,a
          mov   dptr,#table
          movc  a,@a+dptr
          acall write
false:    lcall select
          jb    acc.7,false
          clr   acc.7
yes:      ret

told:     .db   "No program      "

number:   mov   a,pno
          clr   acc.7
          mov   r2,a
          dec   r2              ;(prog. No. - 1)
          mov   dptr,#2000h     ;Emulator use 2000
          mov   a,#00h
          xrl   a,r2            ;check prog.1
          jz    prog1
```

```
other:    mov   a,#0e1h       ;>
          clr   c             ;>
          add   a,dpl         ;>
          mov   dpl,a         ;>(prog. No. - 1)*
          mov   a,#07h        ;>(7e1h) --> 2017d
          addc  a,dph         ;>
          mov   dph,a         ;>
          djnz  r2,other      ;>
prog1:    ret

dayaddr:  mov   r1,#00h       ;r1 -> dpl
          mov   r2,#00h       ;r2 -> dph
          mov   a,fnday
          clr   c
          subb  a,spday       ;(fnday - spday)
          jz    atit
          mov   r0,a
otday:    mov   a,#20h        ; ]
          clr   c             ; ]
          add   a,r1          ; ]
          mov   r1,a          ; ] (diff.)*120
          mov   a,#01h        ; ]
          addc  a,r2          ; ]
          mov   r2,a          ; ]
          djnz  r0,otday      ; ]
          mov   a,r1          ; >
          clr   c             ; >
          add   a,dpl         ; > set dptr
          mov   dpl,a         ; > to loop
          mov   a,r2          ; > at that day
```

```
            addc   a,dph          ; >
            mov    dph,a          ; >
atit:       ret

            ;lcall cutout
userdata:   mov    spday,#01h
            mov    fnday,#01h
            acall  number
            setb   acc.7
            movx   @dptr,a
            inc    dptr
nextday:    push   dpl
            push   dph
            mov    dptr,#ut1
            acall  kd
            pop    dph
            pop    dpl
            mov    bfld,#01h
            mov    prld,#01h
            mov    a,fnday
            xrl    a,spday
            jz     ncr
            push   dpl
            push   dph
            acall  dayaddr
            mov    r4,dpl
            mov    r5,dph
            pop    dph
            pop    dpl
            acall  clram
```

```
            mov   spday,fnday
ncr:        push  dpl
            push  dph
            mov   dptr,#ut2
            acall kl
            pop   dph
            pop   dpl
            mov   r1,#43h
            lcall tfback
            mov   prld,a
            xrl   a,bfld
            jz    notcr
notreach:   mov   a,bfld
            clr   c
            subb  a,#09h
            jnc   more8
            mov   r4,#04h
turn4:      mov   a,#00h
            movx  @dptr,a
            inc   dptr
            djnz  r4,turn4
            inc   bfld
            mov   a,bfld
            cjne  a,prld,notreach
            sjmp  notcr
more8:      mov   r4,#10h
turn16:     mov   a,#00h
            movx  @dptr,a
            inc   dptr
            djnz  r4,turn16
```

```
            inc   bfld
            mov   a,bfld
            cjne  a,prld,more8
notcr:      mov   a,prld
            clr   c
            subb  a,#09h
            jnc   since9
            acall input
            acall kept
            sjmp  together
since9:     mov   r6,#04h
s9dc:       acall input
            acall kept
            djnz  r6,s9dc
together:   inc   prld
            inc   bfld
            mov   a,prld
            clr   c
            subb  a,#19h
            jnc   krobld
            ajmp  ncr
krobld:     inc   fnday
            inc   spday
            mov   a,fnday
            clr   c
            subb  a,#08h
            jnc   krobday
            ajmp  nextday
krobday:    ret
```

```
ut1:    .db   "day program     "
ut2:    .db   "Load number     "

kd:     lcall pattern
        lcall display
        mov   r3,#0c4h
        lcall ddram
        lcall sh
        lcall select
        jnb   acc.7,unc
uc:     mov   r3,#0c4h
        lcall ddram
        lcall din
        jb    acc.7,uc
unc:    ret

kl:     lcall pattern
        lcall display
        mov   r3,#0c4h
        lcall ddram
        mov   r0,#40h
        mov   r1,#42h
        acall decimal
        lcall select
        jnb   acc.7,lnc
        acall lc
lnc:    ret
```

```
lc:     mov   r3,#0c4h
        lcall ddram
        mov   r5,#03h
        lcall wait
        mov   dptr,#table
        mov   r1,#43h
        mov   @r1,a
        dec   r1
        movc  a,@a+dptr
        lcall write
        mov   a,43h
        xrl   a,#02h
        jnz   noov
        mov   r5,#05h
        lcall wait
        sjmp  ldok
noov:   mov   r5,#0ah
        lcall wait
ldok:   mov   @r1,a
        movc  a,@a+dptr
        lcall write
        lcall select
        jb    acc.7,lc
        ret

clram:  mov   a,#00h
        movx  @dptr,a
        inc   dptr
        mov   a,dph
        xrl   a,r5
```

```
          jnz   clram
          mov   a,dpl
          xrl   a,r4
          jnz   clram
          ret

decimal:  acall forward
          dec   r1
          mov   r4,#02h
          mov   dptr,#table
two:      mov   a,@r1
          movc  a,@a+dptr
          acall write
          dec   r1
          djnz  r4,two
          ret

kept:     mov   r0,#44h
kpdt:     mov   a,@r0
          movx  @dptr,a
          inc   r0
          inc   dptr
          cjne  r0,#48h,kpdt
          ret

input:    push  dpl
          push  dph
again:    mov   dptr,#ton
          acall form
          acall ddram
```

```
        mov   r5,#10h
        lcall wait
        mov   r1,#2ah
        mov   @r1,a
        mov   a,prld
        clr   c
        subb  a,#09h
        jnc   ndict
        mov   a,@r1
        xrl   a,#0ch
        jnz   ndict
        mov   a,#0c0h
        acall codein
        mov   dptr,#sstb
        mov   r1,#06h
wrss:   movx  a,@dptr
        acall write
        inc   dptr
        djnz  r1,wrss
        lcall select
        jb    acc.7,again
        ajmp  ldrdy
ndict:  mov   a,@r1
        xrl   a,#0fh
        jnz   hm
        mov   a,#00h
        acall codein
        ajmp  ldrdy
```

```
hm:       mov   a,@r1
          clr   c
          subb  a,#03h
          jnc   again
          mov   a,@r1
          mov   dptr,#table
          movc  a,@a+dptr
          acall write
          mov   a,@r1
          xrl   a,#02h
          jnz   ldplbm
          mov   r5,#04h
          lcall wait
          sjmp  hmok
ldplbm:   mov   r5,#0ah
          lcall wait
hmok:     dec   r1
          mov   @r1,a
          movc  a,@a+dptr
          lcall write
          lcall select
          jb    acc.7,again
          dec   r1
mon:      mov   r3,#0c5h
          lcall mosin
          jb    acc.7,mon
          mov   r1,#3ah
          lcall tfback
          setb  acc.7
          mov   hcon,a
```

```
                lcall tfback
                mov   mcon,a
                mov   dptr,#toff
                acall form
hof:            mov   r3,#0c1h
                lcall hrin
                jb    acc.7,hof
mof:            mov   r3,#0c5h
                lcall mosin
                jb    acc.7,mof
                mov   r1,#3ah
                lcall tfback
                setb  acc.7
                mov   hcoff,a
                lcall tfback
                mov   mcoff,a
ldrdy:          pop   dph
                pop   dpl
                ret

ton:            .db   "   on  h  :m  "
toff:           .db   "    off h  :m  "
sstb:           .db   "SENSOR"

form:           lcall pattern
                lcall display
                mov   r3,#80h
                lcall ddram
                lcall sh
                mov   r0,#40h
```

```
          mov   r1,#42h
          acall decimal
          mov   r3,#0c1h
          ret

codein:   mov   r1,#44h
lic:      mov   @r1,a
          inc   r1
          cjne  r1,#48h,lic
          ret

scan:     lcall number
          lcall not
          jb    acc.7,mesi
          ljmp  sejvan
mesi:     inc   dptr
          mov   spday,#01h
          mov   fnday,#01h
torvan:   push  dpl
          push  dph
          mov   dptr,#sc1
          lcall kd
          pop   dph
          pop   dpl
          mov   bfld,#01h
          mov   prld,#01h
          mov   a,fnday
          clr   c
          subb  a,spday
          jnc   pokti
```

```
            lcall number
            inc   dptr
            mov   spday,#01h
            lcall dayaddr
            sjmp  cont
pokti:      mov   a,fnday
            xrl   a,spday
            jz    cont
            lcall dayaddr
cont:       mov   spday,fnday
torld:      push  dpl
            push  dph
            mov   dptr,#sc2
            lcall kl
            pop   dph
            pop   dpl
            mov   r1,#43h
            lcall tfback
            mov   prld,a
            clr   c
            subb  a,bfld
            jnc   kieng
tee:        mov   a,bfld
            clr   c
            subb  a,#0ah
            jnc   noi
            mov   r4,#04h
pik:        lcall prong
            djnz  r4,pik
            sjmp  mac
```

```
noi:      mov   r4,#10h
pik16:    lcall prong
          djnz  r4,pik16
mac:      dec   bfld
          mov   a,bfld
          cjne  a,prld,tee
          sjmp  pak
kieng:    mov   a,prld
          xrl   a,bfld
          jz    pak
nong:     mov   a,bfld
          clr   c
          subb  a,#09h
          jnc   pue
          lcall pai
          sjmp  harn
pue:      mov   r4,#10h
sibhok:   inc   dptr
          djnz  r4,sibhok
harn:     inc   bfld
          mov   a,bfld
          cjne  a,prld,nong
pak:      mov   a,prld
          clr   c
          subb  a,#09h
          jnc   thong
          movx  a,@dptr
          jnb   acc.7,hme
          jnb   acc.6,toi
          push  dpl
```

```
            push  dph
            mov   dptr,#sc3
            lcall form
            lcall ddram
            pop   dph
            pop   dpl
            acall pai
            sjmp  centrate
toi:        mov   r0,#44h
            acall pull
            sjmp  centrate
hme:        acall nosung
centrate:   acall klab
            sjmp  vit
thong:      mov   r7,#04h
vachr:      movx  a,@dptr
            jnb   acc.7,poo
            mov   r0,#44h
            acall pull
            sjmp  pong
poo:        acall nosung
pong:       acall klab
            djnz  r7,vachr
vit:        inc   prld
            inc   bfld
            mov   a,prld
            clr   c
            subb  a,#19h
            jnc   sejld
            ljmp  torld
```

```
sejld:    inc   fnday
          inc   spday
          mov   a,fnday
          clr   c
          subb  a,#08h
          jnc   sejvan
          ljmp  torvan
sejvan:   ret

sc1:      .db   "Day?           "
sc2:      .db   "Load number?   "
sc3:      .db   "         SENSOR "
sc4:      .db   "     No dictate "
sc5:      .db   "      on    :  "
sc6:      .db   "      off   :  "

prong:    mov   a,dpl
          xrl   a,#00h
          jz    high
          dec   dpl
          sjmp  angkor
high:     dec   dph
          mov   dpl,#0ffh
angkor:   ret

pai:      mov   r4,#04h
see:      inc   dptr
          djnz  r4,see
          ret
```

```
klab:     mov   r3,#0c7h
          lcall ddram
          lcall select
          jnb   acc.7,maikae
          mov   r4,#04h
lod:      acall prong
          djnz  r4,lod
          lcall input
          lcall kept
maikae:   ret

nosung:   push  dpl
          push  dph
          mov   dptr,#sc4
          lcall form
          lcall ddram
          pop   dph
          pop   dpl
          acall pai
          ret

pull:     movx  a,@dptr
          anl   a,#3fh
          mov   @r0,a
          inc   dptr
          inc   r0
          cjne  r0,#48h,pull
          push  dpl
          push  dph
          mov   dptr,#sc5
```

```
            lcall form
            lcall ddram
            mov   r0,#44h
            mov   r1,#42h
            lcall decimal
            mov   r3,#0c5h
            lcall ddram
            mov   r1,#42h
            lcall decimal
mairub:     lcall select
            jb    acc.7,mairub
            mov   dptr,#sc6
            lcall form
            lcall ddram
            mov   r0,#46h
            mov   r1,#42h
            lcall decimal
            mov   r3,#0c5h
            lcall ddram
            mov   r1,#42h
            lcall decimal
            pop   dph
            pop   dpl
            ret

delete:     mov   a,pno
            clr   acc.7
            mov   pno,a
            lcall number
            push  dpl
```

```
            push  dph
            inc   pno
            lcall number
            mov   r5,dph
            mov   r4,dpl
            pop   dph
            pop   dpl
            lcall clram
            ret

runner:     lcall number
            lcall not
            jnb   acc.7,np
            inc   dptr
            mov   bfld,#01h
            mov   prld,#01h
            mov   spday,#01h
            mov   fnday,day
            lcall dayaddr
            mov   r0,#60h        ;load buffer addr.
            mov   r2,#00h        ;load pointer
            mov   a,newsm
            jnb   acc.0,uv
            lcall schedule
            ajmp  np
uv:         mov   a,hour
            clr   c
            subb  a,#12h
            jc    midnight
            ajmp  auto
```

```
midnight: mov   a,hour
          clr   c
          subb  a,#06h
          jnc   np
auto:     movx  a,@dptr
          jnb   acc.7,icr4
          jb    acc.6,icr4
          mov   r1,#32h
          anl   a,#3fh
          clr   c
          subb  a,@r1
          jz    cmfr
          jnc   icr4
          inc   dptr
          sjmp  thb
cmfr:     dec   r1
          inc   dptr
          movx  a,@dptr
          anl   a,#3fh
          clr   c
          subb  a,@r1
          jz    thb
          jnc   icr3
thb:      mov   r1,#32h
          inc   dptr
          movx  a,@dptr
          anl   a,#3fh
          clr   c
          subb  a,@r1
          jz    cmbk
```

```
            jnc   during
            sjmp  icr2
cmbk:       dec   r1
            inc   dptr
            movx  a,@dptr
            anl   a,#3fh
            clr   c
            subb  a,@r1
            jz    icr1
            jnc   only1
            sjmp  icr1
icr4:       inc   dptr
icr3:       inc   dptr
icr2:       inc   dptr
icr1:       inc   dptr
            acall sense
            sjmp  outtime
during:     inc   dptr
only1:      inc   dptr
outtime:    inc   r0
            inc   r2
            cjne  r2,#08h,auto
np:         ret

sense:      mov   a,r0
            clr   c
            subb  a,#28h
            mov   r1,a
            mov   a,@r1
            jnz   pid
```

```
            mov   a,r2
            add   a,#18h
            mov   b,a
            push  dpl
            push  dph
            lcall state
            pop   dph
            pop   dpl
            acall delay
            mov   a,psw
            jb    acc.5,light
            mov   @r1,#07h
            mov   a,@r0
            jnb   acc.5,pid
            lcall off
            sjmp  pid
light:      mov   @r1,#25h
            mov   a,@r0
            jb    acc.5,pid
            lcall on
pid:        ret

schedule:   mov   a,newsm
            clr   acc.0
            mov   newsm,a
eight:      movx  a,@dptr        ;check user sched.
            jnb   acc.7,onsen    ;this LD. dictate?
            jnb   acc.6,simple   ;sensor or dictate
onsen:      inc   dptr
            inc   dptr
```

```
          inc   dptr
          inc   dptr
          sjmp  nextld
simple:   acall subsched       ;time control
nextld:   inc   r0             ;pointnextLDbuffer
          inc   r2             ;pointer next load
          cjne  r2,#08h,eight  ;first 8 load
allload:  movx  a,@dptr
          jnb   acc.7,by       ;this LD. dictate?
          mov   r4,#04h
fourtime: acall subsched       ;dictate and
          djnz  r4,fourtime    ;one load scan
          sjmp  passed
by:       mov   r4,#10h        ;not dictate so
stepup:   inc   dptr           ;increment dptr
          djnz  r4,stepup      ;to next load
passed:   inc   r0             ;inc LD. buffer
          inc   r2             ;inc LD. pointer
          cjne  r2,#18h,allload
          ret

subsched: mov   r1,#32h        ;address hour
          lcall compare
          inc   dptr
          jnz   comoff         ;jump if not hour
          lcall compare
          jnz   comoff         ;jump if not min.
          lcall on             ;ON load
```

```
comoff:   inc   dptr
          mov   r1,#32h        ;address hour
          lcall compare
          inc   dptr
          jnz   nxcom          ;jump if not hour
          lcall compare
          jnz   nxcom          ;jump if not min.
          lcall off            ;OFF load
nxcom:    inc   dptr
          ret

compare:  movx  a,@dptr
          anl   a,#3fh         ;clear bit 7&6
          mov   r6,a           ;rest user data
          mov   a,@r1          ;mov hour or min.
          xrl   a,r6           ;time checked
          dec   r1
          ret

delay:    mov   r5,#05h
siki:     mov   r6,#7dh
soko:     mov   r7,#63h
saka:     djnz  r7,saka
          djnz  r6,soko
          djnz  r5,siki
          ret

working:  mov   dptr,#bok
          lcall pattern
          lcall display
```

```
            ret
bok:        .db   "Dictating load  "

on:         push  dpl
            push  dph
            mov   dph,#base
            mov   dpl,#portc
            mov   a,r2          ;load No.
            orl   a,#0a0h       ;set bit 7&5 "ON"
            mov   b,a
            movx  @dptr,a
            acall state         ;if LD. not ON
            mov   a,psw
            anl   a,#20h
            mov   @r0,a         ;kept state
            acall delay         ;in load buffer
            pop   dph
            pop   dpl
            ret

off:        push  dpl
            push  dph
            mov   dph,#base
            mov   dpl,#portc
            mov   a,r2          ;load No.
            setb  acc.7         ;set bit 7
            mov   b,a
            movx  @dptr,a       ;clear bit 5 "OFF"
            acall state         ;if LD. not OFF
            mov   a,psw
```

```
                anl   a,#20h
                mov   @r0,a          ;kept state
                acall delay          ;in load buffer
                pop   dph
                pop   dpl
                ret

state:          acall working
                mov   dph,#base
                mov   dpl,#portc
                mov   a,b
                setb  acc.7          ;set bit 7&6
                setb  acc.6
                movx  @dptr,a        ;read "flag"
                mov   a,b
                clr   acc.7
                setb  acc.6
                movx  @dptr,a
                mov   r7,#10h
drop:           mov   r3,#7dh
cop:            mov   r4,#63h
carry:          djnz  r4,carry
                djnz  r3,cop
                djnz  r7,drop
                mov   dpl,#porta
                movx  a,@dptr
                jnb   acc.6,roy
                jnb   acc.5,nothing
                setb  psw.5
                sjmp  roy
```

```
nothing:  clr   psw.5
roy:      mov   dpl,#portc
          mov   a,b
          setb  acc.7
          movx  @dptr,a
          clr   acc.7
          movx  @dptr,a
          ret

load:     mov   dptr,#ut2
          lcall pattern
          lcall display
          lcall lc
          mov   r1,#43h
          lcall tfback
          mov   prld,a
vivat:    mov   dptr,#l1
          lcall pattern
          lcall display
          mov   r3,#84h
          lcall ddram
          mov   r0,#40h
          mov   r1,#42h
          lcall decimal
          acall situ
          mov   kbd,#20h
          mov   r5,#07h
in3:      mov   r6,#00h
in2:      mov   r7,#00h
in1:      mov   a,kbd
```

```
        jnb   acc.7,w
        clr   acc.7
        xrl   a,#01h
        jnz   f
        mov   r3,#0c5h
        lcall ddram
        mov   dptr,#l2
        acall tell
        lcall select
        jb    acc.7,vivat
        mov   a,prld
        add   a,#5fh
        mov   r0,a
        mov   a,prld
        dec   a
        mov   r2,a
        lcall on
        ajmp  vivat
f:      mov   a,kbd
        clr   acc.7
        xrl   a,#00h
        jnz   w
        mov   r3,#0c5h
        lcall ddram
        mov   dptr,#l3
        acall tell
        lcall select
        jb    acc.7,vivat
        mov   a,prld
        add   a,#5fh
```

```
        mov   r0,a
        mov   a,prld
        dec   a
        mov   r2,a
        lcall off
        ajmp  vivat
w:      djnz  r7,in1
        djnz  r6,in2
        djnz  r5,in3
        ret

l1:     .db   "Ld.      "
l2:     .db   " on"
l3:     .db   "off"

situ:   mov   a,prld
        add   a,#5fh
        mov   r0,a
        mov   r3,#0c0h
        lcall ddram
        mov   a,@r0
        jnb   acc.5,x
        mov   dptr,#l2
        acall tell
        sjmp  n
x:      mov   dptr,#l3
        acall tell
n:      ret
```

```
tell:     mov   r3,#03h
roab:     movx  a,@dptr
          lcall write
          inc   dptr
          djnz  r3,roab
          ret

sensor:   mov   a,hour
          clr   c
          subb  a,#12h
          jc    bfdawn
          sjmp  night
bfdawn:   mov   a,hour
          clr   c
          subb  a,#06h
          jnc   shine
night:    mov   r0,#60h
          mov   r2,#00h
meik:     acall sense
          inc   r0
          inc   r2
          cjne  r2,#08h,meik
shine:    ret
settime:  mov   dptr,#stab    ;function for
          lcall pattern       ;set time
          lcall display
d:        mov   r3,#80h       ;set day
          lcall din
          jb    acc.7,d
          mov   day,fnday
```

```
h:      mov   r3,#85h       ;set hour
        lcall hrin
        jb    acc.7,h
m:      mov   r3,#0c1h      ;set min
        mov   r1,#38h
        lcall mosin         ;received set min
        jb    acc.7,m       ;RET or CLS
s:      mov   r3,#0c5h      ;set sec
        mov   r1,#36h
        lcall mosin         ;received set sec
        jb    acc.7,s       ;RET or CLS
        mov   r1,#3ah
        lcall tfback        ;change keyin
        mov   hour,a        ;to time code
        lcall tfback
        mov   min,a
        lcall tfback
        mov   sec,a
        ret

stab:   .db   "d    h  :m  :s  "

din:    acall kr
        acall sh
        acall select
        ret

kr:     lcall ddram
nz:     mov   r5,#08h       ;key received
        lcall wait          ;not over 7
```

```
        xrl   a,#00h          ;key received
        jz    nz              ;is not 0
        mov   a,kbd
        mov   fnday,a         ;kept day set
        ret

sh:     mov   a,fnday
        mov   b,#03h
        mul   ab
        add   a,#09h
        mov   r2,a
        mov   r4,#03h
wrtd:   mov   a,r2
        mov   dptr,#table     ;show set
        movc  a,@a+dptr       ;at LCD
        lcall write
        inc   r2
        djnz  r4,wrtd
        ret

hrin:   lcall ddram
        mov   r5,#03h         ;key received
        lcall wait            ;not over 2
        mov   r1,#3ah
        mov   @r1,kbd
        dec   r1
        mov   dptr,#table
        movc  a,@a+dptr
        lcall write
        mov   a,3ah           ;check if first
```

```
          xrl   a,#02h          ;input is 2
          jnz   noplbm          ;second input
          mov   r5,#04h         ;not over 3
          lcall wait
          sjmp  ok
noplbm:   mov   r5,#0ah         ;key received
          lcall wait            ;not over 9
ok:       mov   @r1,kbd
          dec   r1
          movc  a,@a+dptr
          lcall write
          lcall select          ;RET or CLS
          ret

mosin:    lcall ddram           ;received
          lcall sixty           ;min or sec
          ret

pattern:  mov   psw,#01h
          mov   r0,#50h         ;kept 16 character
sixteen:  movx  a,@dptr         ;to LCD buffer
          lcall store           ;(address 40h-4fh)
          inc   dptr
          cjne  r0,#60h,sixteen
          mov   psw,#00h
          ret

wait:     mov   a,kbd           ;wait for keyin
          jnb   acc.7,wait      ;and limit keyin
          anl   a,#0fh          ;not over
```

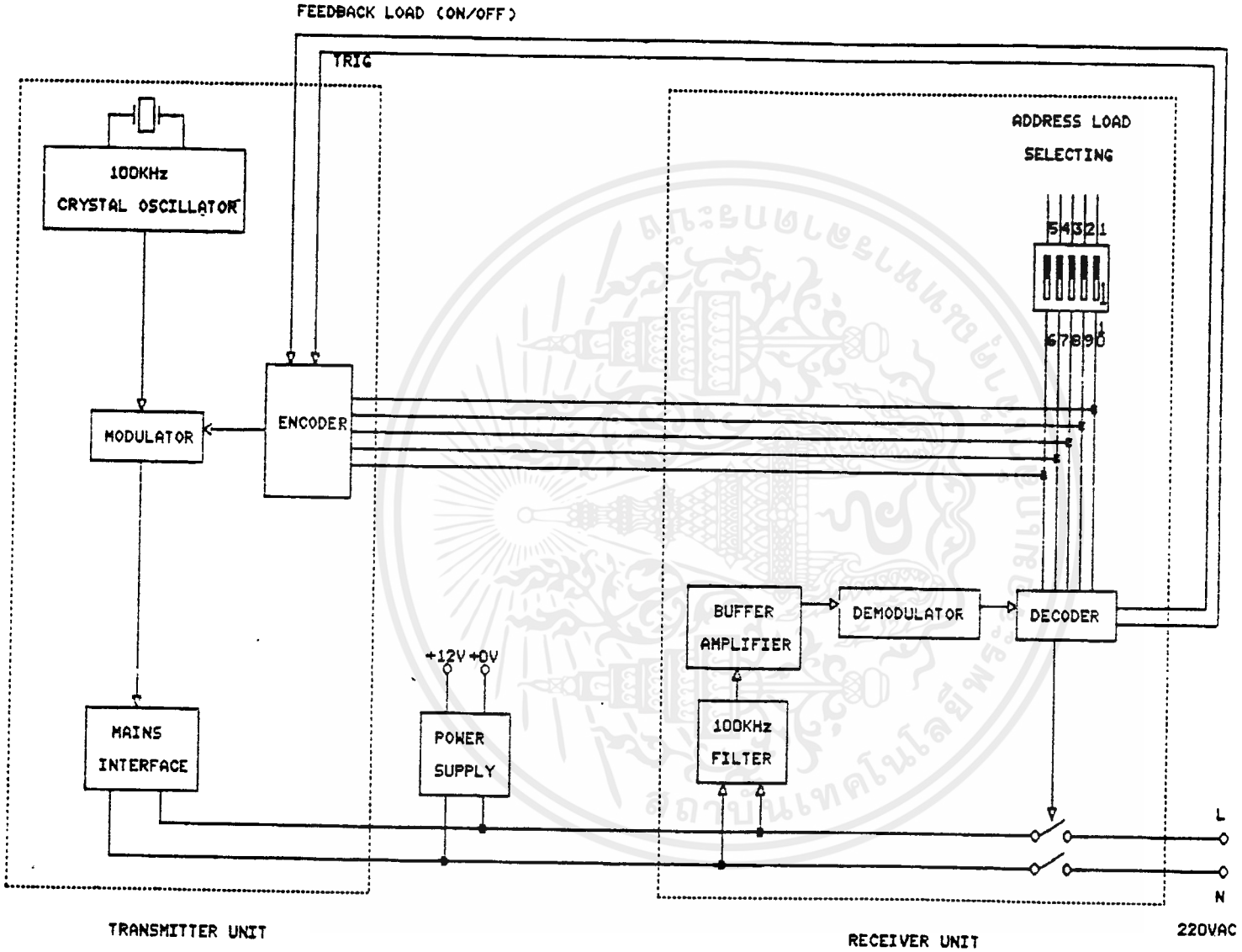
```
                mov   kbd,a          ;r5 is parameter
                clr   c
                subb  a,r5
                jnc   wait
                mov   a,kbd          ;to show at LCD
                ret
select:         mov   r5,#10h        ;check keyin is
                acall wait           ;RET or CLS
                xrl   a,#0fh         ;check RET
                jz    return
                mov   a,kbd
                xrl   a,#0eh         ;check CLS
                jnz   select
                setb  acc.7          ;show state keyin
return:         ret                  ;is CLS
sixty:          mov   r5,#06h        ;receive keyin
                acall wait           ;and limit keyin
                mov   @r1,kbd        ;not over 59
                dec   r1
                mov   dptr,#table
                movc  a,@a+dptr
                lcall write
                mov   r5,#0ah
                acall wait
                mov   @r1,kbd
                dec   r1
                movc  a,@a+dptr
                lcall write
                acall select
                ret
```

```
tfback:  mov   a,@r1          ;change keyin back
         mov   b,#0ah         ;to time code
         mul   ab
         dec   r1
         add   a,@r1
         dec   r1
         ret
         .
cutout:  mov   a,pno
         clr   acc.7
         mov   pno,a
         mov   r0,#60h
         mov   r2,#00h
ct:      lcall off
         inc   r2
         inc   r0
         cjne  r2,#18h,ct
         ret
         .end
```

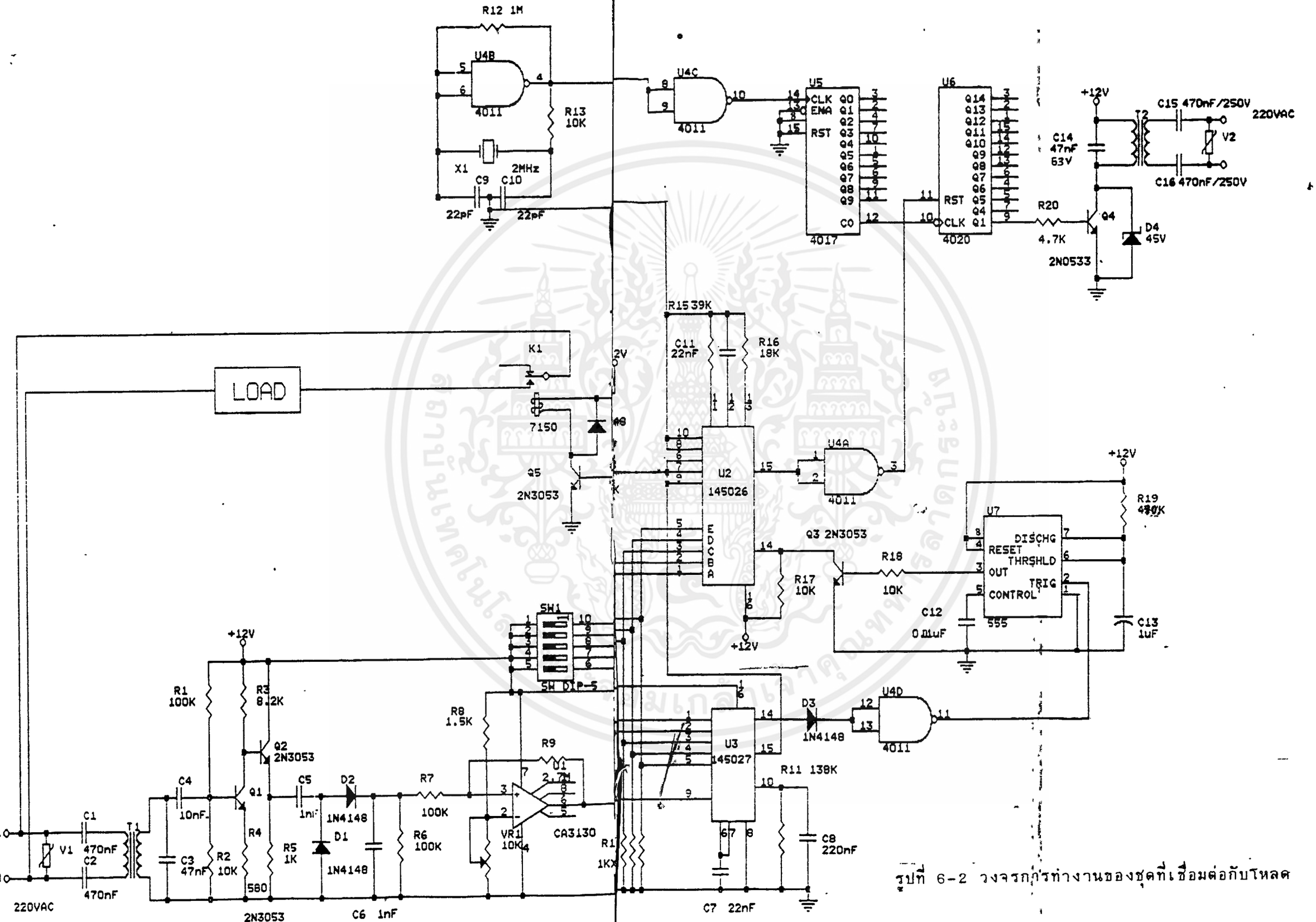
บทที่ 6
การทำงานทางฮาร์ดแวร์

ในบทนี้จะอธิบายการทำงานของชุดที่เชื่อมต่อกับโหลดและชุดอินฟราเรดเซนเซอร์ ว่ามีการทำงานอย่างไร โดยดูได้จากรูปที่ 6-1 ถึง รูปที่ 6-4 ซึ่งมีลักษณะของวงจรคล้ายกันมาก เพียงแต่ว่าชุดที่เชื่อมต่อกับโหลดจะมีส่วนของรีเลย์ที่ใช้ควบคุมโหลด

แต่ชุดอินฟราเรดเซนเซอร์จะไม่มีส่วนที่เป็นรีเลย์แต่จะมีส่วนของวงจรเซนเซอร์เพิ่มเข้ามาแทน



รูปที่ 6-1 บล็อคไดอะแกรมแสดงหน่วยรับและหน่วยส่งข้อมูลของชุดที่เชื่อมต่อกับโหลด



รูปที่ 6-2 วงจรการทำงานของชุดที่เชื่อมต่อกับโหลด

หลักการทำงานของวงจร

สัญญาณพัลส์ความถี่ 100 KHz ที่ส่งมาตามเอซีไลน์ จะผ่านตัวเก็บประจุ 47 นาโนฟารัด ทั้งสองตัว ซึ่งทำหน้าที่ป้องกันความถี่ 50 Hz จากเอซีไลน์และสัญญาณความถี่นี้จะถูกคัปปลิ้งโดย T_1 มายังขดปฐมภูมิ ซึ่งมีตัวเก็บประจุ 47 นาโนฟารัดต่อขนานอยู่ ซึ่งจะเป็นวงจรจูนที่รีโซแนนต์ความถี่ 100 KHz ทำให้มีอิมพีแดนซ์สูงซึ่งมีผลให้แอมปลิจูดของสัญญาณมีขนาดใหญ่ขึ้น แต่ที่ความถี่อื่น ๆ จะทำให้วงจรจูนมีอิมพีแดนซ์ต่ำมาก หรือกล่าวอีกนัยหนึ่งคือ เป็นวงจรฟิลเตอร์ความถี่ 100 KHz (100 KHz Filter)

โดยแอมปลิจูดของสัญญาณที่ได้รับนี้จะขึ้นอยู่กับ

- ความยาวของสายไฟระหว่างชุดคอนโทรลเลอร์กับชุดที่เชื่อมต่อกับโหลด
- จำนวนเครื่องใช้ไฟฟ้าที่ต่ออยู่

เพราะฉะนั้นสัญญาณพัลส์ความถี่ 100 KHz ที่ฟิลเตอร์แล้วจะต้องถูกคัปปลิ้งผ่านตัวเก็บประจุ 10 นาโนฟารัด ไปยังทรานซิสเตอร์ BC 237 ตัวแรก ซึ่งมีอัตราขยาย 10 เท่า จากนั้นก็จะส่งไปยังทรานซิสเตอร์ BC 237 ตัวที่สอง โดยจะได้เอาท์พุทไปขับไดโอด 1N4148 ซึ่งทำหน้าที่ Demodulator สัญญาณพัลส์ความถี่ 100 KHz

สัญญาณพัลส์ความถี่ 100 KHz ที่ผ่านไดโอด 1N4148 มาแล้วจะนำมาเข้าวงจรชมิททริกเกอร์โดย IC 3130 เพื่อทำให้เป็นสัญญาณสี่เหลี่ยม

เอาท์พุทที่ได้จาก IC 3130 จะเป็นข้อมูลดิจิตอลแบบอนุกรม ซึ่งเป็นอินพุทของวงจรถอดรหัส (DECODER) โดยจะส่งเข้าที่ขา 9 ของ IC MC 145027 เพื่อทำการถอดรหัสสัญญาณที่ตรงกับสวิทช์เลือก S_1-S_4 ที่ตั้งไว้ ถ้ารหัสตรงกัน ข้อมูลตรงขา 15 และข้อมูลตรงขา 14 ของ IC MC145027 ก็จะแสดงค่าตามข้อมูล F และ G ตามลำดับ

โดยสัญญาณที่ขา 15 จะใช้ในการขับทรานซิสเตอร์ 2N3053 เพื่อที่จะไปควบคุมรีเลย์ให้ทำการเปิด/ปิดโหลด

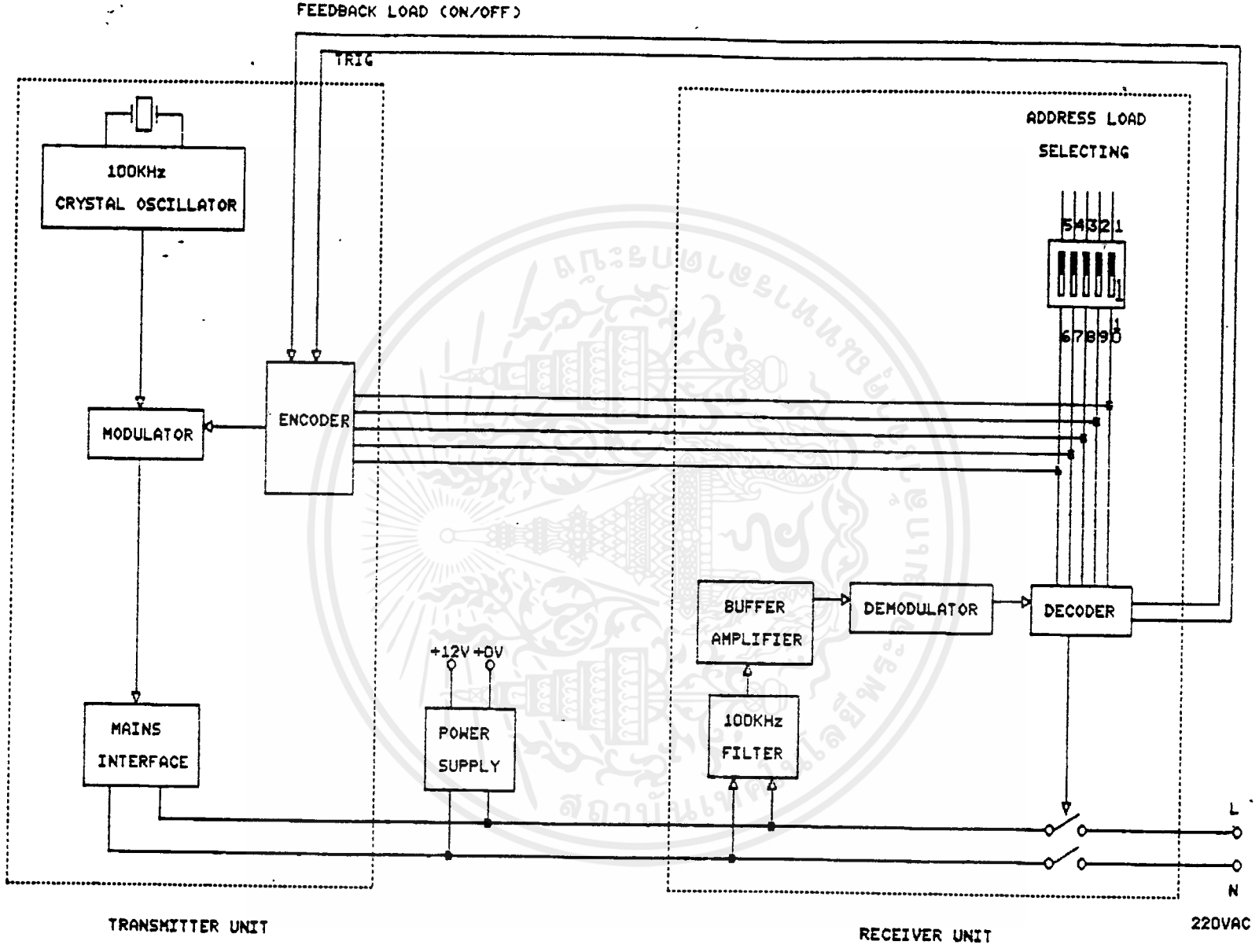
ส่วนสัญญาณที่ขา 14 จะใช้ในการบอกวงจรส่งข้อมูลของโหลดให้ส่งข้อมูลออกไปว่าตอนนี้โหลดได้ถูกเปิด/ปิดเรียบร้อยแล้ว

ในส่วนของโปรแกรมควบคุมในตัวคอนโทรลเลอร์ นั้นจะทำการส่งข้อมูลสั่งให้โหลดทำการเปิด/ปิดก่อน เสร็จแล้วถึงจะส่งข้อมูลอีกชุดเพื่อสั่งให้วงจรส่งข้อมูลของโหลดส่งค่าสภาวะของโหลดมาว่าถูกเปิด/ปิดแล้วหรือยังกลับมาบอกตัวคอนโทรลเลอร์อีกที

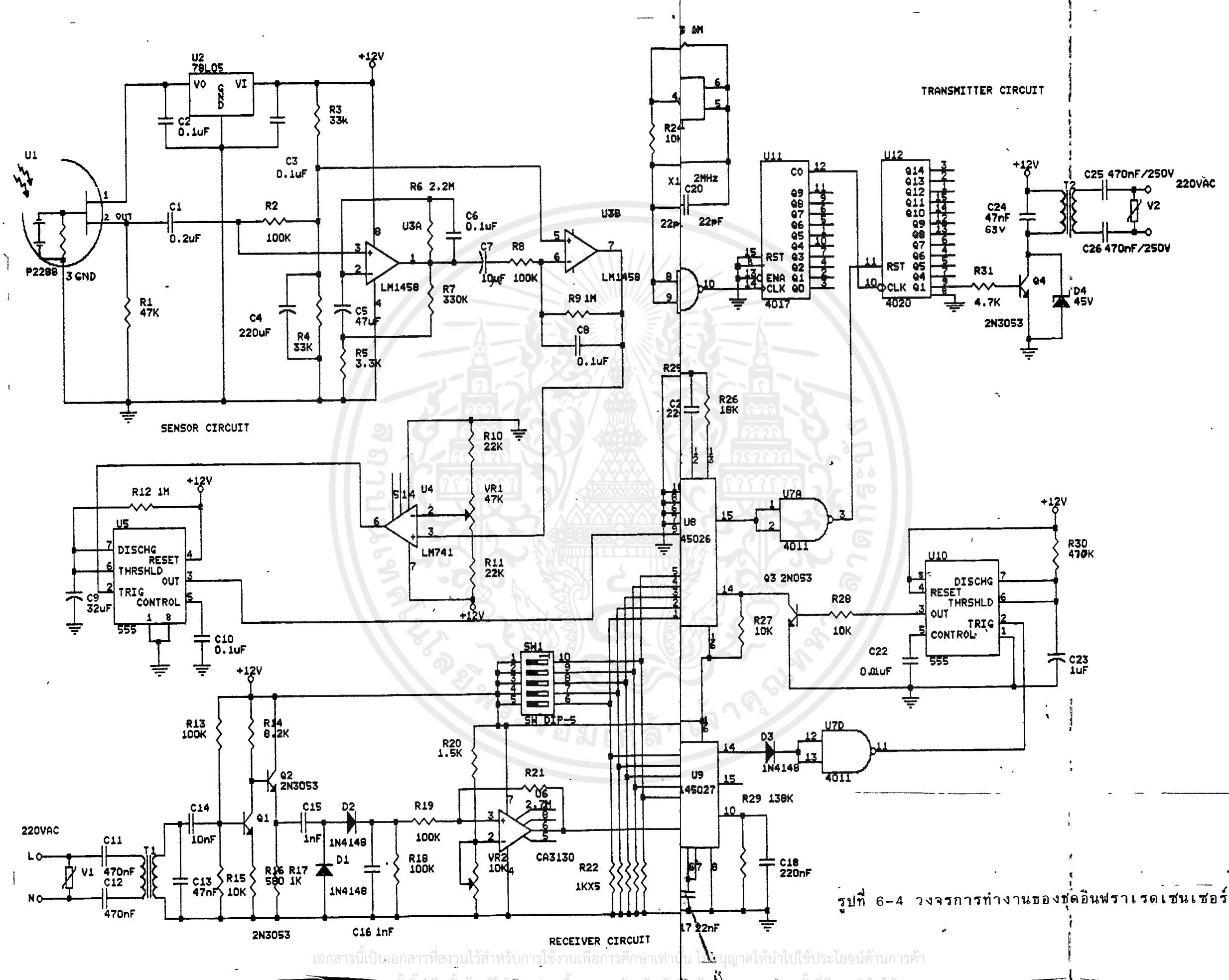
ซึ่งค่าสภาวะของโหลดเราดูจากค่าตรงขาคอลเลคเตอร์ของทรานซิสเตอร์2N3053 โดยกรณีที่โหลดยังปิดอยู่จะไม่มีกระแสไหลผ่านตัวทรานซิสเตอร์ 2N3053 ทำให้แรงดันตรงจุดนี้มีค่าเท่ากับแรงดันจากแหล่งจ่ายที่เราจ่ายให้

อีกกรณีหนึ่งคือโหลดถูกเปิดซึ่งในกรณีนี้จะมีกระแสไหลผ่านตัวทรานซิสเตอร์2N3053 ข้อมูลที่ได้จะต่อเข้ากับขา 9 หรือข้อมูลบิทที่ 7 หรือที่เราเรียกว่าข้อมูล H นั่นเอง

ในการส่งข้อมูลของวงจรส่งข้อมูลของโหลดนี้ เราจะเห็นว่าข้อมูลที่ส่งออกมาจะมีผลต่อวงจรรับข้อมูลของโหลดเช่นเดียวกับที่ส่งออกมาจากตัวคอนโทรลเลอร์ ทำให้เราต้องน่าค่าที่ส่งมาจากตัวคอนโทรลเลอร์ ว่าต้องการเปิด/ปิดโหลดใส่ให้กับวงจรส่งข้อมูลของโหลดก่อนที่จะส่งข้อมูลกลับไป โดยเราจะเอาค่าสภาวะจากขา 15 ของ IC MC145026 ที่วงจรหน่วยรับข้อมูลของโหลดใส่ให้แก่ขา F หรือขาที่ 6 ของวงจรส่งข้อมูลของโหลดก่อนที่จะส่งข้อมูลออกตามเอซีไลน์



รูปที่ 6-1 บล็อคไดอะแกรมแสดงหน่วยรับและหน่วยส่งข้อมูลของชุดที่เชื่อมต่อกับโหลด



รูปที่ 6-4 วงจรการทำงานของชุดอินฟราเรดเซนเซอร์

หลักการทำงานของวงจร

จากวงจร ไพโรอิเล็กตริกเซ็นเซอร์ จะมีความต้านทาน 47 กิโลโอห์ม ต่อเป็นโหลด (load) และบัฟเฟอร์ IC 78L05 ทำหน้าที่รักษาแรงดันให้ได้ 5 โวลต์คงที่ป้อนให้กับ ไพโรอิเล็กตริกเซนเซอร์ ซึ่งตามปกติจะใช้กับแรงดันในช่วง 2.2-10 โวลต์

ส่วน IC ทั้งสองตัวที่ต่อจากไพโรอิเล็กตริกเซนเซอร์ จะทำหน้าที่เป็นวงจรขยายสัญญาณที่ได้จากไพโรอิเล็กตริกให้แรงขึ้นโดย IC ตัวแรกต่อเป็นวงจรขยายแบบไม่กลับเฟส แต่ IC ตัวที่สองต่อเป็นวงจรขยายแบบกลับเฟส ทำให้เมื่อเวลามีสัญญาณผ่านเข้ามาทางอินพุต จะทำให้แรงดันที่ขา 7 ของ IC ตัวที่สองลดลงซึ่งการจัดวงจรแบบนี้จะทำให้มีคุณสมบัติเป็นวงจรแบนด์พาสฟิลเตอร์ด้วย โดยมี ตัวเก็บประจุ 0.1 ไมโครฟารัด ทั้งสองตัวช่วยลดทอนสัญญาณความถี่สูง ๆ IC LM741 เป็นวงจรทริกเกอร์ซึ่งต่อเป็นวงจรเปรียบเทียบแรงดัน (comparator) โดยมี VR 47 กิโลโอห์ม เป็นตัวตั้งระดับความแรงของสัญญาณที่จะให้วงจรเริ่มทำงาน ซึ่งก็เป็นการปรับความไวของเครื่องนั่นเอง

ถ้าหากมีสัญญาณเข้ามาจะทำให้แรงดันที่ขา 3 ของ IC LM741 ตกลง หากตกลงจนมีค่าน้อยกว่าแรงดันที่ขา 2 จะทำให้แรงดันที่ขา 6 ตกเป็น "0" ไปเข้าวงจรรับ-ส่งข้อมูล ซึ่งมีลักษณะคล้ายกับวงจรรับ-ส่งข้อมูลของชุดที่เชื่อมต่อกับโหลดเพียงแต่ว่าข้อมูลที่ใช้เป็นข้อมูล H หรือข้อมูลบิทที่ 7 ซึ่งจะป้อนเข้าที่ขา 9 ของ IC MC145026 โดยถ้ามีสภาวะเป็น "0" แสดงว่ามีคนอยู่ แต่ถ้ามีค่าเป็น "1" แสดงว่าไม่มีคนอยู่ และที่แตกต่างกันอีกอย่างคือบิทที่ 5ที่ใช้สำหรับการสั่งปิดเปิดอุปกรณ์นั้นจะไม่ใช้ในชุดอินฟราเรดเซนเซอร์

บทที่ 7
คุณสมบัติและการใช้งานของ HOME AUTOMATION

คุณสมบัติ

- วัฏจักรโปรแกรม	1 สัปดาห์
- การแสดงเวลา	เป็นตัวเลข 6 หลักสำหรับ ชั่วโมงมนาที, วินาที และตัวอักษร 3 หลักสำหรับวัน
- จำนวน OUTPUT	24 สำหรับโหลด และ 8 สำหรับ SENSOR
- Load 1-8	สามารถเลือกให้ทำงานตามเวลาที่ตั้ง โดยสั่งเปิด-ปิดได้ 1 ครั้ง ใน 1 วัน หรือทำงานตามสภาวะของ SENSOR ที่ตรวจจับการมีอยู่ของคน
- Load 9-24	สามารถตั้งเวลา เปิด-ปิด ได้ 4 ครั้งใน 1 วัน
- PROGRAM	โดยการใช้ RAM 6116 สามารถตั้งได้ 1 โปรแกรม และสามารถตั้งได้ถึง 4 โปรแกรมถ้าใช้ RAM ขนาด 4K byte

การใช้งาน HOME AUTOMATION

การใช้งาน HOME AUTOMATION สามารถแบ่งการใช้งานได้ 5 ลักษณะคือ 1. PROGRAM เป็นฟังก์ชันในการตั้งเวลา เปิด-ปิด อุปกรณ์ไฟฟ้า หรือ สั่งให้ทำงานตามสภาวะของ SENSOR สำหรับ Load 1-8 นอกจากนั้นแล้วยังสามารถตรวจสอบแก้ไข, ลบโปรแกรม และเลือกโปรแกรมที่จะให้ทำงานได้ด้วย

ค่าอธิบาย	การเสดงผล
1.1 กด [PRO] 1.2 กดตัวเลข 1-4 และ [RET]	1.1 Program number x 1.2 Function call?

หลังจากเลือก Program number แล้ว มีฟังก์ชันเลือกใช้งานได้ 4 อย่างคือ กดปุ่ม [PRO],[S/S],[CLS] และ [RET] ดังนี้

[PRO] การโปรแกรม (PROGRAM) เช่น

ตั้งเวลาเปิด 10.00 น. เวลาปิด 11.30 น.

คำอธิบาย	การแสดงผล
1. กด [PRO]	1. day program SUN
2. กด [RET]	2. Load number 01
3. กด [RET]	3. SUN01 on h :m
4. ป้อนเวลาชั่วโมง และ [RET]	4. SUN01 on 10:m
5. ป้อนเวลานาที และ [RET]	5. SUN01 off h :m
6. ป้อนเวลาชั่วโมง และ [RET]	6. SUN01 off 11:m
7. ป้อนเวลานาที และ [RET]	7. SUN02 on h :m

ก่อนกด [RET] ในแต่ละครั้งสามารถกด [CLS] เพื่อเปลี่ยน วัน,Load number และเวลาได้ สำหรับวันและ Load number ที่ถูกข้ามไปจะไม่มีการสั่งงานใด ๆ ทั้งสิ้น สำหรับ Load 1-8 สามารถให้ทำงานตามสภาวะ SENSOR โดย

คำอธิบาย	การแสดงผล
4. กด [S/S]	4. SUN01 SENSOR
5. กด [RET]	5. SUN02 on h :m

และ Load 9-24 จะมีขั้นตอนให้ป้อนเวลา เปิด-ปิด 4 ครั้งโดย Load ทุกตัวถ้าไม่ต้องการสั่งงานในขั้นตอนใดก็สามารถกด [RET] ให้ข้ามไปได้

[S/S] เป็นการเรียกโปรแกรมที่ USER ป้อนไว้ออกมาดู (SCAN)

คำอธิบาย	การแสดงผล
1. กด [S/S]	
*- ถ้าไม่มี Program number x อยู่	*- No program x
- กด [RET] จะกลับไปแสดงเวลา	
*- ถ้ามี Program number x อยู่	*- Day? SUN
- กด [RET]	- Load number? 01
- กด [RET]	- SUN01 on 10:00
- กด [RET]	- SUN01 off 11:30

จะแสดงเวลาที่ USER ป้อนไว้ และถ้าต้องการแก้ไขก็สามารถทำได้โดยกด [CLS] ในขณะที่แสดงเวลา off เช่น

คำอธิบาย	การแสดงผล
	- SUN01 off 11:30
- กด [RET]	- SUN01 on h :m

แล้วทำการป้อนโปรแกรมเหมือนเดิม

[RET] เป็นการเรียกโปรแกรมขึ้นมาทำงาน (RUN)

คำอธิบาย	การแสดงผล
1. กด [RET]	
ถ้าไม่มี Program number x อยู่	1. No program x
2. กด [RET]	2. จะกลับไปแสดงเวลา

ถ้ามี Program number x อยู่ก็จะกลับไปแสดงเวลาเลยและเริ่มทำงานตามโปรแกรมนั้น

[CLS] เป็นการลบ Program number x ทิ้ง (DELETE)

คำอธิบาย	การแสดงผล
1. กด [CLS]	1. -

จะกลับไปแสดงเวลาตามเดิมและ Program number x ทั้งหมดจะหายไป

2. LOAD เป็นฟังก์ชันที่เรียกสภาวะของอุปกรณ์นั้น ๆ ออกมาดูว่าเป็นอย่างไร และสามารถสั่งเปิด-ปิดได้

คำอธิบาย	การแสดงผล
2.1 กด [LD] 2.2 ป้อนเบอร์โหลดและกด [RET]	2.1 Load number xx 2.2 Ld. 01 off

ถ้าไม่มีการสั่งงานใด ๆ จะแสดงสภาวะอยู่ประมาณ 10 วินาทีแล้วจะแสดงเวลาตามเดิม ถ้าต้องการสั่งงานก็สามารถทำได้โดย

คำอธิบาย	การแสดงผล
2.3 กดเลข "1" (on) เมื่อ Load number 1เปิดแล้วจะแสดง	2.3 Ld. 01 off on Ld. 01 on

ในทางกลับกันถ้าต้องการสั่งปิดก็กดเลข "0"

3. TIME เป็นการตั้งเวลาให้ตรงกับเวลาปัจจุบัน (SETTIME) โดย

ค่าอธิบาย	การแสดงผล
3.1 กด [TIM]	3.1 d h :m :s
3.2 ป้อนวันด้วยเลข 1-7 (SUN-SAT) และ [RET	3.2 SUN h :m :s
3.3 ป้อนเวลาชั่วโมงและกด [RET]	3.3 SUN 08 :m :s
3.4 ป้อนเวลานาทีและกด [RET]	3.4 SUN 08 :30 :s
3.5 ป้อนเวลาวินาทีและ [RET]	3.5 SUN 08 :30 :45

แล้วนาฬิกาจะเดินต่อไปตามปกติ

4. SENSOR เป็นการสั่งให้ปิดอุปกรณ์ทุกตัว แต่อุปกรณ์ที่มี SENSOR จะ เปิด-ปิด ตามสภาวะของ SENSOR ทำโดยกด [S/S]

5. CUTOUT เป็นการสั่งปิดอุปกรณ์ทุกตัว ทำโดยกด [CLS]

บทที่ 8
สรุปผลของโครงงานและข้อเสนอแนะ

โครงงานชุดนี้เป็นการส่งข้อมูลในลักษณะของแรงดัน ซึ่งระดับแรงดันที่ส่งไปตามเอซีไลน์ จะขึ้นอยู่กับจำนวนเครื่องใช้ไฟฟ้าที่ต่ออยู่ ทำให้ระดับแรงดันเปลี่ยนแปลงไป ในบางครั้งก็อาจจะทำให้หน่วยรับข้อมูลไม่ทำงานเลยก็ได้ ฉะนั้นถ้าต้องการให้การรับ-ส่งมีประสิทธิภาพมากขึ้น จึงขอแนะนำให้ใช้วิธีการรับ-ส่งข้อมูลในรูปของกระแส ซึ่งจะทำให้ระดับกระแสที่ส่งไปตามเอซีไลน์ ไม่ขึ้นอยู่กับจำนวนเครื่องใช้ไฟฟ้าที่ต่ออยู่ รายละเอียดศึกษาได้จากข้อมูลของไอซีเบอร์ LM 1893 / LM 2893 ของ NATIONAL SEMICONDUCTOR ELECTRONICS

และในโครงงานนี้อาจจะเพิ่มการใช้คอมพิวเตอร์ในการติดต่อกับตัวคอนโทรลเลอร์ โดยติดต่อผ่านทาง RS-232 ซึ่งการนำคอมพิวเตอร์มาใช้นั้นก็เพื่อความสะดวกในการโปรแกรมการเปิด/ปิดอุปกรณ์ไฟฟ้าทำให้การใช้งานง่ายขึ้น

มาถึงตรงนี้ผู้จัดทำหวังว่าผู้สนใจจะปรับปรุงโครงงานชุดนี้ให้สมบูรณ์ยิ่งๆขึ้นไป เพื่อที่จะนำมาใช้ประโยชน์ในชีวิตประจำวันให้ได้สูงสุดตามเป้าหมายของโครงงานชุดนี้

National Semiconductor

MM54C922/MM74C922 16-Key Encoder
MM54C923/MM74C923 20-Key Encoder

general description

These CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. The keyboard scan can be implemented by either an external clock or external capacitor. These encoders also have on-chip pull-up devices which permit switches with up to 50 kΩ on resistance to be used. No diodes in the switch array are needed to eliminate ghost switches. The internal debounce circuit needs only a single external capacitor and can be defeated by omitting the capacitor. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released, even if another key is depressed. The Data Available will return high to indicate acceptance of the new key after a normal debounce period; this two key roll over is provided between any two switches.

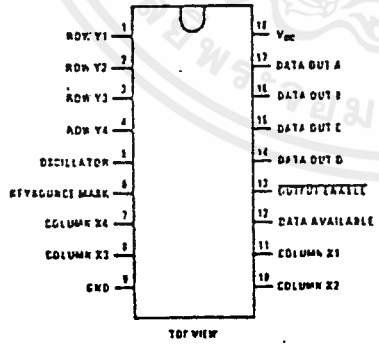
An internal register remembers the last key pressed even after the key is released. The TRI-STATE® outputs provide for easy expansion and bus operation and are LPTTL compatible.

features

- 50 kΩ maximum switch on resistance
- On or off chip clock
- On chip row pull-up devices
- 2 key roll-over
- Keybounce elimination with single capacitor
- Last key register at outputs
- TRI-STATE outputs LPTTL compatible
- Wide supply range 3V to 15V
- Low power consumption

connection diagrams

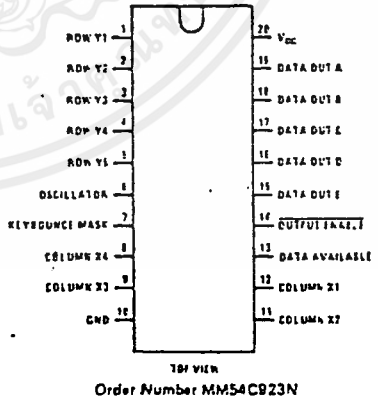
Dual-In-Line Package



TOP VIEW

Order Number MM54C922N
or MM74C922N
See Package 20

Dual-In-Line Package



TOP VIEW

Order Number MM54C923N
or MM74C923N
See Package 20A

absolute maximum ratings

Voltage at Any Pin	V_{CC} − 0.3V to V_{CC} + 0.3V	Package Dissipation	500 mW
Operating Temperature Range		Operating V_{CC} Range	3V to 15V
MM54C922, MM54C923	−55°C to +125°C	V_{CC}	18V
MM74C922, MM74C923	−40°C to +85°C	Lead Temperature (Soldering, 10 seconds)	300°C
Storage Temperature Range	−65°C to +150°C		

dc electrical characteristics

Min/max limits apply across temperature range unless otherwise noted

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
CMOS TO CMOS						
V_{T+}	Positive-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq$ 0.7 mA	3	3.6	4.3	V
		V_{CC} = 10V, $I_{IN} \geq$ 1.4 mA	6	6.8	8.6	V
		V_{CC} = 15V, $I_{IN} \geq$ 2.1 mA	9	10	12.9	V
V_{T-}	Negative-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq$ 0.7 mA	0.7	1.4	2	V
		V_{CC} = 10V, $I_{IN} \geq$ 1.4 mA	1.4	3.2	4	V
		V_{CC} = 15V, $I_{IN} \geq$ 2.1 mA	2.1	5	6	V
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V	3.5	4.5		V
		V_{CC} = 10V	8	9		V
		V_{CC} = 15V	12.5	13.5		V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V		0.5	1.5	V
		V_{CC} = 10V		1	2	V
		V_{CC} = 15V		1.5	2.5	V
I_{rp}	Row Pull-Up Current at Y1, Y2, Y3, Y4 and Y5 Inputs	V_{CC} = 5V, V_{IN} = 0.1 V_{CC}		−2	−5	μA
		V_{CC} = 10V		−10	−20	μA
		V_{CC} = 15V		−22	−45	μA
$V_{OUT(1)}$	Logical "1" Output Voltage	V_{CC} = 5V, I_O = −10μA	4.5			V
		V_{CC} = 10V, I_O = −10μA	9			V
		V_{CC} = 15V, I_O = −10μA	13.5			V
$V_{OUT(0)}$	Logical "0" Output Voltage	V_{CC} = 5V, I_O = 10μA			0.5	V
		V_{CC} = 10V, I_O = 10μA			1	V
		V_{CC} = 15V, I_O = 10μA			1.5	V
R_{on}	Column "ON" Resistance at X1, X2, X3 and X4 Outputs	V_{CC} = 5V, V_O = 0.5V		500	1400	Ω
		V_{CC} = 10V, V_O = 1V		300	700	Ω
		V_{CC} = 15V, V_O = 1.5V		200	500	Ω
I_{CC}	Supply Current	V_{CC} = 5V, Osc at 0V		0.55	1.1	mA
		V_{CC} = 10V		1.1	1.9	mA
		V_{CC} = 15V		1.7	2.6	mA
$I_{IN(1)}$	Logical "1" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 15V		0.005	1.0	μA
$I_{IN(0)}$	Logical "0" Input Current at $\overline{\text{Output Enable}}$	V_{CC} = 15V, V_{IN} = 0V	−1.0	−0.005		μA
CMOS/LPTTL INTERFACE						
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V	V_{CC} − 1.5			V
		74C, V_{CC} = 4.75V	V_{CC} − 1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	54C, V_{CC} = 4.5V			0.8	V
		74C, V_{CC} = 4.75V			0.8	V
$V_{OUT(1)}$	Logical "1" Output Voltage	54C, V_{CC} = 4.5V, I_O = −360μA	2.4			V
		74C, V_{CC} = 4.75V, I_O = −360μA	2.4			V
$V_{OUT(0)}$	Logical "0" Output Voltage	54C, V_{CC} = 4.5V, I_O = −360μA			0.4	V
		74C, V_{CC} = 4.75V, I_O = −360μA			0.4	V

dc electrical characteristics (con't)

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
OUTPUT DRIVE (See 54C/74C Family Characteristics Data Sheet)					
I_{SOURCE} Output Source Current (P-Channel)	V_{CC} = 5V, V_{OUT} = 0V, T_A = 25°C	−1.75	−3.3		mA
I_{SOURCE} Output Source Current (P-Channel)	V_{CC} = 10V, V_{OUT} = 0V, T_A = 25°C	−8	−15		mA
I_{SINK} Output Sink Current (N-Channel)	V_{CC} = 5V, V_{OUT} = V_{CC}, T_A = 25°C	1.75	3.6		mA
I_{SINK} Output Sink Current (N-Channel)	V_{CC} = 10V, V_{OUT} = V_{CC}, T_A = 25°C	8	16		mA

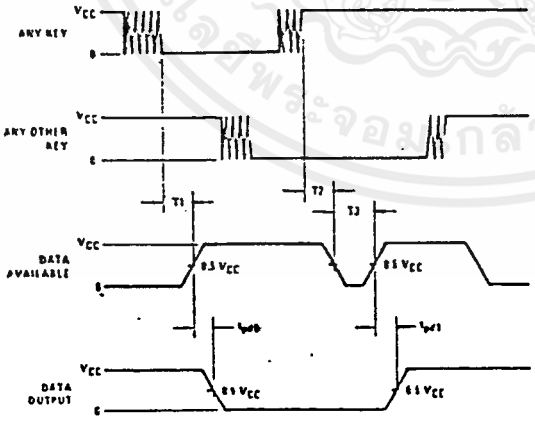
ac electrical characteristics T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
t_{pd0},t_{pd1} Propagation Delay Time to Logical "0" or Logical "1" from D.A.	C_L = 50 pF, *(Figure 1)*				
	V_{CC} = 5V		60	150	ns
	V_{CC} = 10V		35	80	ns
	V_{CC} = 15V		25	60	ns
t_{0H},t_{1H} Propagation Delay Time from Logical "0" or Logical "1" into High Impedance State	R_L = 10k, C_L = 5 pF, *(Figure 2)*				
	V_{CC} = 5V R_L = 10k		80	200	ns
	V_{CC} = 10V C_L = 10 pF		65	150	ns
	V_{CC} = 15V		50	110	ns
t_{H0},t_{H1} Propagation Delay Time from High Impedance State to a Logical "0" or Logical "1"	R_L = 10k, C_L = 50 pF, *(Figure 2)*				
	V_{CC} = 5V R_L = 10k		100	250	ns
	V_{CC} = 10V C_L = 50 pF		55	125	ns
	V_{CC} = 15V		40	90	ns
C_{IN} Input Capacitance	Any Input, (Note 2)		5	7.5	pF
C_{OUT} TRI-STATE Output Capacitance	Any Output, (Note 2)		10		pF

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Capacitance is guaranteed by periodic testing.

switching time waveforms



T1 = T2 = RC, T3 = 0.7 RC where R = 10k and C is external capacitor at KBM input.

FIGURE 1

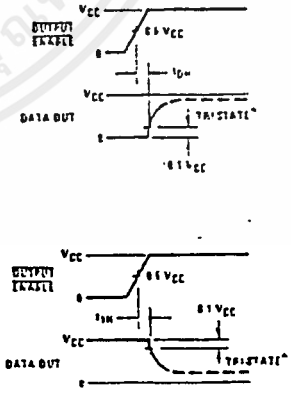
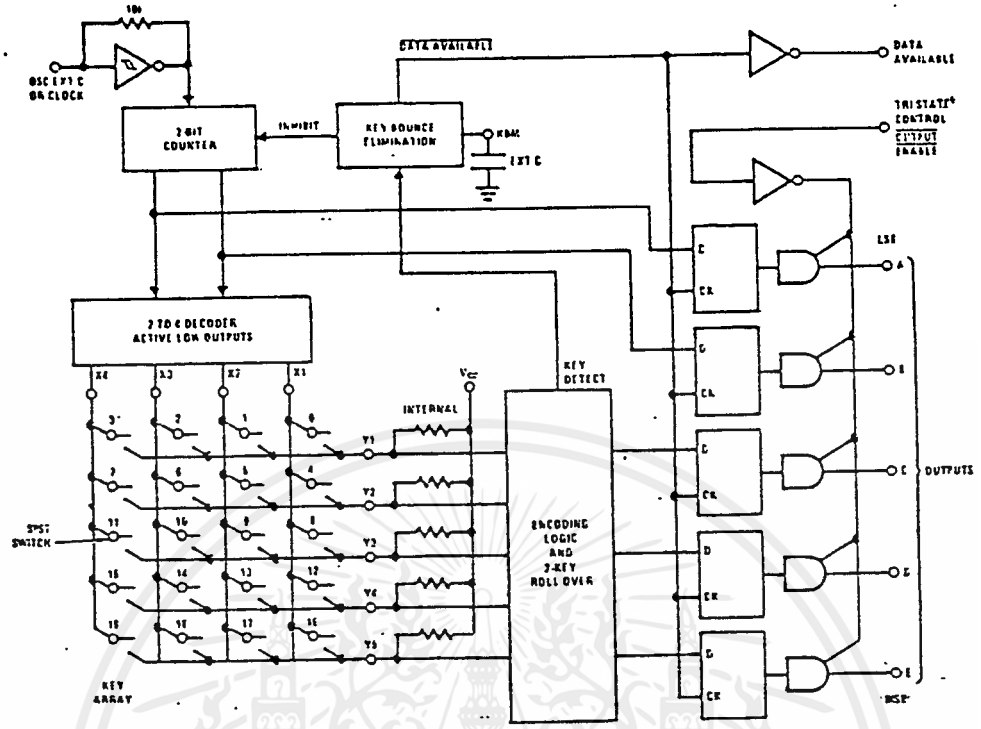


FIGURE 2

block diagram



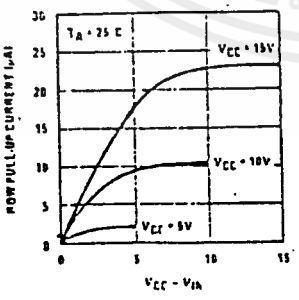
truth table

SWITCH POSITION	0 Y1,X1	1 Y1,X2	2 Y1,X3	3 Y1,X4	4 Y2,X1	5 Y2,X2	6 Y2,X3	7 Y2,X4	8 Y3,X1	9 Y3,X2	10 Y3,X3	11 Y3,X4	12 Y4,X1	13 Y4,X2	14 Y4,X3	15 Y4,X4	16 Y5*,X1	17 Y5*,X2	18 Y5*,X3	19 Y5*,X4
DATA	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
STB	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
A2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
A1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
A0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

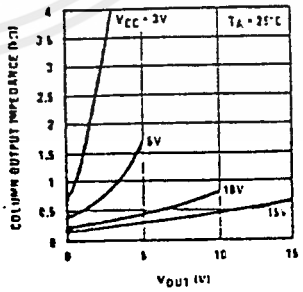
*Omit for MM54C922/MM74C922

typical performance characteristics

Typical I_{rp} vs V_{IN} at Any Y Input



Typical R_{on} vs V_{OUT} at Any X Output



MOTOROLA

MC145026
MC145027
MC145028

Advance Information

CMOS MSI
(LOW-POWER COMPLEMENTARY MOS)

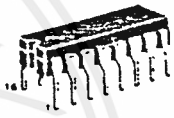
REMOTE CONTROL
ENCODER/DECODER PAIRS

MC145026 ENCODER, MC145027/MC145028 DECODERS

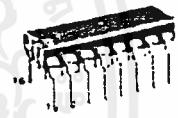
The MC145026 will encode nine bits of information and serially transmit this information upon receipt of a transmit enable, $\overline{TE}$, (active low) signal. Nine inputs may be encoded with trinary data (0, 1, open) to allow 3^9 (19,683) different codes.

Two decoders are presently available. Both use the same transmitter — the MC145026. The decoders will receive the 9-bit word and will interpret some of the bits as address codes and some as data. The MC145027 will interpret the first five transmitted bits as address and the last four bits as data. The MC145028 will treat all nine bits as address. If no errors are received, the MC145027 will output the four data bits when the transmitter sends address codes that match that of the receiver. A valid transmission output will go high on both decoders when they recognize an address that matches that of the decoder. Other receivers can be produced with different address/data ratios.

- May be Addressed in either Binary or Trinary
- Trinary Addressing Maximizes Number of Codes
- Interfaces with RF, Ultrasonic, or Infrared Transmission Medias
- Double Transmissions for Error Checking
- 4.5 V to 18 V Operation
- On-Chip R/C Oscillator; No Crystal Required
- High External Component Tolerance; Can use 5% Components
- Standard B-Series Input and Output Characteristics



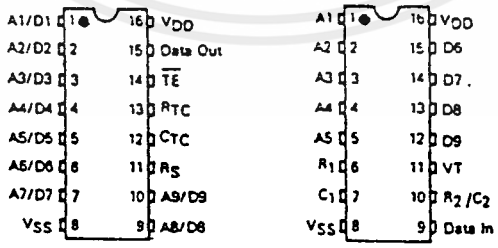
L SUFFIX
CERAMIC PACKAGE
CASE 620-...



P SUFFIX
PLASTIC PACKAGE
CASE 648

MC14XXXX — Suffix — Denotes
- L — Ceramic Package
- P — Plastic Package

PIN ASSIGNMENTS



MC145026 Encoder | MC145027 Decoder | MC145028 Decoder

This is advance information and specifications are subject to change without notice.

MAXIMUM RATINGS (Voltages Referenced to V_{SS})

Rating	Symbol	Value	Unit
DC Supply Voltage	V_{DD}	− 0.5 to + 18	V
Input Voltage, All Inputs	V_{in}	− 0.5 to V_{DD} + 0.5	V
DC Current Drain Per Pin	I	10	mA
Operating Temperature Range	T_A	− 40 to + 85	°C
Storage Temperature Range	T_{stg}	− 65 to + 150	°C

ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	V_{DD}	− 40°C		25°C			+ 85°C		Unit
		V	Min	Max	Min	Typ	Max	Min	Max	
Output Voltage "0" Level V_{in} = V_{DD} or 0	V_{OL}	5.0	–	0.05	–	0	0.05	–	0.05	V
		10	–	0.05	–	0	0.05	–	0.05	
		15	–	0.05	–	0	0.05	–	0.05	
"1" Level V_{in} = 0 or V_{DD}	V_{OH}	5.0	4.95	–	4.95	5.0	–	4.95	–	V
		10	9.95	–	9.95	10	–	9.95	–	
		15	14.95	–	14.95	15	–	14.95	–	
Input Voltage "0" Level (V_O = 4.5 or 0.5 V) (V_O = 9.0 or 1.0 V) (V_O = 13.5 or 1.5 V)	V_{IL}	5.0	–	1.5	–	2.25	1.5	–	1.5	V
		10	–	3.0	–	4.50	3.0	–	3.0	
		15	–	4.0	–	6.25	4.0	–	4.0	
"1" Level (V_O = 0.5 or 4.5 V) (V_O = 1.0 or 9.0 V) (V_O = 1.5 or 13.5 V)	V_{IH}	5.0	3.5	–	3.5	2.75	–	3.5	–	V
		10	7.0	–	7.0	5.50	–	7.0	–	
		15	11.0	–	11.0	8.25	–	11.0	–	
Output Drive Current Source (V_{OH} = 2.5 V) (V_{OH} = 4.6 V) (V_{OH} = 9.5 V) (V_{OH} = 13.5 V)	I_{OH}	5.0	− 2.5	–	− 2.1	− 4.2	–	− 1.7	–	mA
		5.0	− 0.52	–	− 0.44	− 0.88	–	− 0.36	–	
		10	− 1.3	–	− 1.1	− 2.25	–	− 0.9	–	
		15	− 3.6	–	− 3.0	− 8.8	–	− 2.4	–	
Sink (V_{OL} = 0.4 V) (V_{OL} = 0.5 V) (V_{OL} = 1.5 V)	I_{OL}	5.0	0.52	–	0.44	0.88	–	0.36	–	mA
		10	1.3	–	1.1	2.25	–	0.9	–	
		15	3.6	–	3.0	8.8	–	2.4	–	
Input Current – TE (MC145026, Pullup Device)	I_{in}	5.0	–	–	3.0	4.0	7.0	–	–	μA
		10	–	–	16	20	26	–	–	
		15	–	–	35	45	55	–	–	
Input Current R_S (MC145026) Data In (MC145027, MC145028)	I_{in}	15	–	± 0.3	–	± 0.00001	± 0.3	–	± 1.0	μA
Input Current A1/D1-A9/D9 (MC145026) A1-A5 (MC145027) A1-A9 (MC145028)	I_{in}	5.0	–	–	–	± 55	± 80	–	–	μA
		10	–	–	–	± 300	± 340	–	–	
		15	–	–	–	± 650	± 725	–	–	
Input Capacitance (V_{in} = 0)	C_{in}	–	–	–	–	5.0	7.5	–	–	pF
Quiescent Current – MC145026	I_{DD}	5.0	–	–	–	0.0050	0.10	–	–	μA
		10	–	–	–	0.0100	0.20	–	–	
		15	–	–	–	0.0150	0.30	–	–	
Quiescent Current – MC145027, MC145028	I_{DD}	5.0	–	–	–	30	50	–	–	μA
		10	–	–	–	60	100	–	–	
		15	–	–	–	90	150	–	–	
Total Supply Current – MC145026 (f_c = 20 kHz)	I_T	5.0	–	–	–	100	200	–	–	μA
		10	–	–	–	200	400	–	–	
		15	–	–	–	300	600	–	–	
Total Supply Current – MC145027, MC145028 (f_c = 20 kHz)	I_T	5.0	–	–	–	200	400	–	–	μA
		10	–	–	–	400	800	–	–	
		15	–	–	–	600	1200	–	–	

SWITCHING CHARACTERISTICS (C_L = 50 pF, T_A = 25°C)

Characteristic	Symbol	V_{DD}	Min	Typ	Max	Unit
Output Rise and Fall Time	t_{TLH} t_{THL}	5.0 10 15	– – –	100 50 40	200 100 80	ns
Data In Rise and Fall Time (MC145027, MC145028)	t_{TLH} t_{THL}	5.0 10 15	– – –	– – –	15 15 15	μs
Encoder Clock Frequency	f_{cl}	5.0 10 15	0 0 0	– – –	2 5 10	MHz
Maximum Decoder Frequency (Referenced to Encoder Clock) (See Figure 9)	f_{cl}	5.0 10 15	– – –	– – –	240 410 450	kHz
$\overline{TE}$ Pulse Width	t_{WTL}	5.0 10 15	650 300 200	– – –	– – –	ns
System Propagation Delay ($\overline{TE}$ to Valid Transmission)	–	–	–	182	–	Clock Cycles
Tolerance on Timing Components ($\Delta R_{TC} = \Delta C_{TC} = \Delta R_1 = \Delta C_1$) ($\Delta R_2 = \Delta C_2$)	–	– –	– –	– –	±25 ±25	%

OPERATING CHARACTERISTICS

MC145026

The encoder will serially transmit nine bits of trinary data as defined by the state of the A1/D1-A9/D9 input pins. These pins can be in either of three states (0, 1, open) allowing 3^9 = 19683 possible codes. The transmit sequence will be initiated by a low level of the $\overline{TE}$ input pin. Each time the $\overline{TE}$ input is forced low the encoder will output two identical data words. This redundant information is used by the receiver to reduce errors. If the $\overline{TE}$ input is kept low, the encoder will continuously transmit the data words. The transmitted words are self-completing (two words will be transmitted for each $\overline{TE}$ pulse).

Each transmitted data bit is encoded into two data pulses. A logic zero will be encoded as two consecutive short pulses, a logic one by two consecutive long pulses, and an open as a long pulse followed by a short pulse. The input state is determined by using a weak output device to try to force each input first low, then high. If only a high state results from the two tests, the input is assumed to be hard wired to V_{DD}. If only a low state is obtained, the input is assumed to be hard wired to V_{SS}. If both a high and a low can be forced at an input, it is assumed to be open and is encoded as such.

The transmit sequence is enabled by a logic zero on the $\overline{TE}$ input. This input has an internal pullup device so that a simple switch may be used to force the input low. While $\overline{TE}$ is high the encoder is completely disabled, the oscillator is inhibited and the current drain is reduced to quiescent current. When $\overline{TE}$ is brought low, the oscillator is started, and an internal reset is generated to initialize the transmit sequence. Each input is then sequentially selected and a determination is made as to input logic state. This information is serially transmitted via the Data Out output pin.

MC145027

The decoder will receive the serial data from the encoder, check it for errors and output data if valid. The transmitted data consisting of two identical data words is examined bit by bit as it is received. The first five bits are assumed to be address bits and must be encoded to match the address inputs at the receiver. If the address bits match, the next four (data) bits are stored and compared to the last valid data stored. If this data matches, the VT pin will go high on the 2nd rising edge of the 9th bit of the first word. Between the two data words no signal is sent for three data bit times. As the second encoded word is received, the address must again match, and if it does, the data bits are checked against the previously stored data bits. If the two words of data (four bits each) match, the data is transferred to the output data latches and will remain until new data replaces it. At the same time, the Valid Transmission output pin is brought high and will remain high until an error is received or until no input signal is received for four data bit times.

Although the address information is encoded in trinary fashion, the data information must be either a one or a zero. A trinary (open) will be decoded as a logic one.

MC145028

This receiver operates in the same manner as the MC145027 except that nine address bits are used and no data output is available. The Valid Transmission output is used to indicate that a valid signal has been received.

Although address information normally is encoded in trinary, the designer should be aware that, for the MC145028, the ninth address bit (A9) must be either a one or a zero. This part, therefore, can accept only $2 \times 3^8 = 13,122$ different codes. A trinary (open) A9 will be interpreted as a logic 1. However if the transmitter sends a trinary (or logic 1) and the receiver address is a logic 1 (or trinary) respectively, the valid transmission output will be shortened to the R1 × C1 time constant.

DOUBLE TRANSMISSION DECODING

Although the encoder sends two words for error checking, a decoder does not necessarily wait for two transmitted words to be received before issuing a valid transmission output. Refer to the flowcharts in Figures 7 and 8.

PIN DESCRIPTION

MC145026 Encoder

A1/D1-A9/D9 — These inputs will be encoded and the data serially output from the encoder.

V_{SS} — The most negative supply (usually ground).

R_S, C_{TC}, R_{TC} — These pins are part of the oscillator section of the encoder. If an external signal source is used instead of the internal oscillator it should be connected to the R_S input and the R_{TC} and C_{TC} pins should be left open.

$\overline{TE}$ — This Transmit-Enable (active low) input will initiate transmission when forced low. A pullup device will keep this input high normally.

Data Out — This is the output of the encoder that will present the serially encoded signals.

V_{DD} — The most positive supply.

MC145027 Decoder

A1-A5 — These are the address inputs that must match the encoder inputs A1/D1-A5/D5 in order for the decoder to output data.

D6-D9 — These outputs will give the information that is presented to the encoder inputs A6/D6-A9/D9. Note: only binary data will be acknowledged, a trinary open will be decoded as logic one.

R_1, C_1 — These pins accept a resistor and capacitor that are used to determine whether a narrow pulse or a wider pulse has been encoded. The time constant $R_1 \times C_1$ should be set to 1.72 transmit clock periods: $R_1C_1 = 3.95\ R_{TC}C_{TC}$.

R_2/C_2 — This pin accepts a resistor to V_{SS} and a capacitor to V_{SS} that are used to detect both the end of an encoded word and the end of transmission. The time constant $R_2 \times C_2$ should be 33.5 transmit clock periods (four data bit periods). This time constant is used to determine that the Data In input has remained low for four data bit times (end of transmission). A separate comparator looks at a voltage equivalent two data bit times (0.4 R_2C_2) to detect the dead time between transmitted words. $R_2C_2 = 77\ R_{TC}C_{TC}$.

Valid Transmission, VT — This output will go high when the following conditions are satisfied:

1. the transmitted address matches the receiver address, and
2. the transmitted data matches the last valid data received.

VT will remain high until either a mismatch is received, or no input signal is received for four data bit times.

V_{DD} — The most positive supply.

V_{SS} — The most negative supply (usually ground).

FIGURE 1 — ENCODER BLOCK DIAGRAM: MC145026*

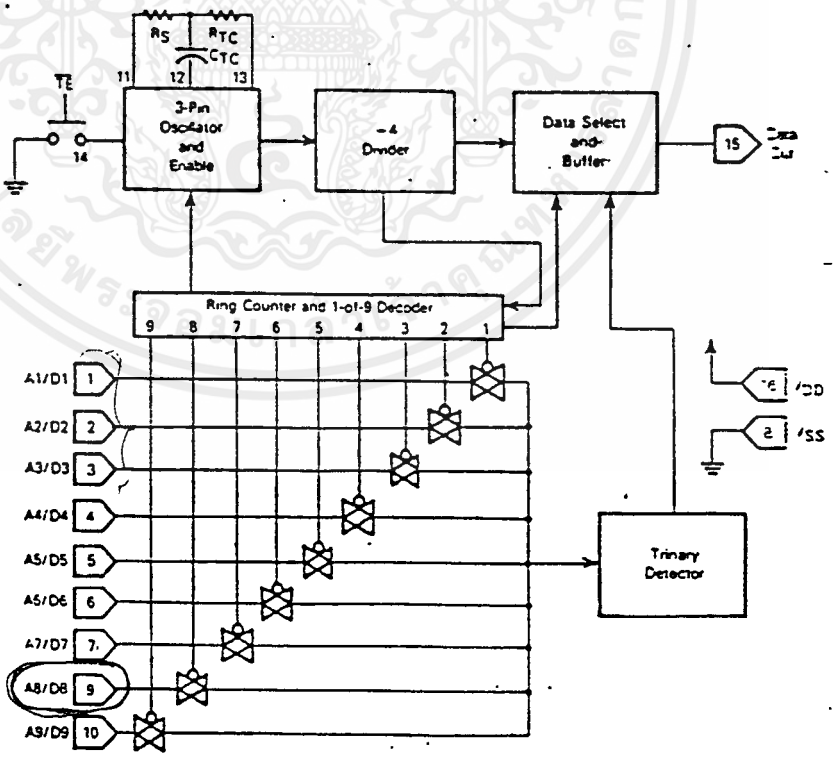


FIGURE 2 – DECODER BLOCK DIAGRAM MC145027

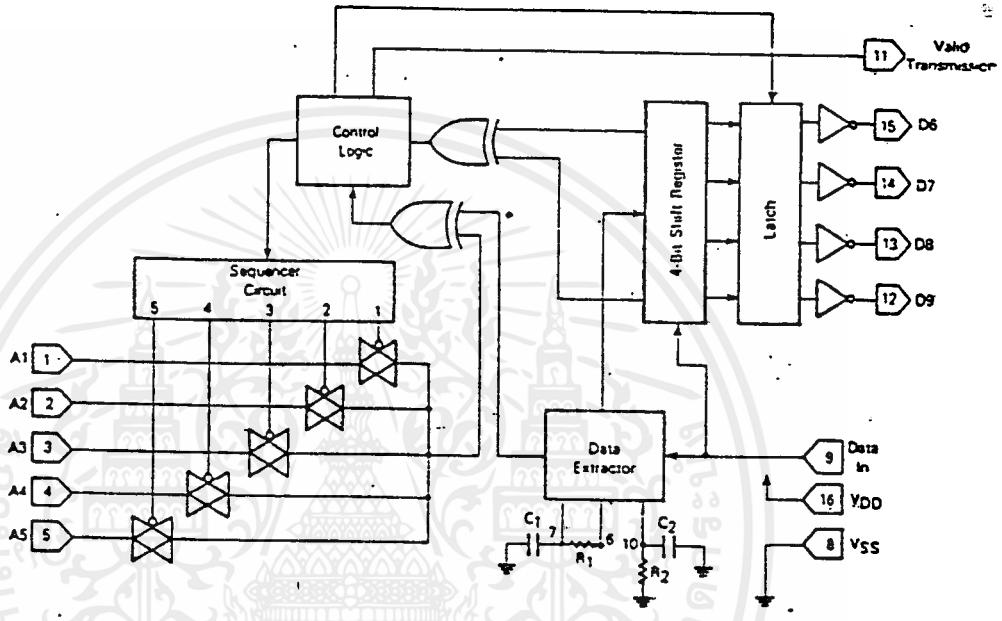


FIGURE 3 – DECODER BLOCK DIAGRAM MC145028

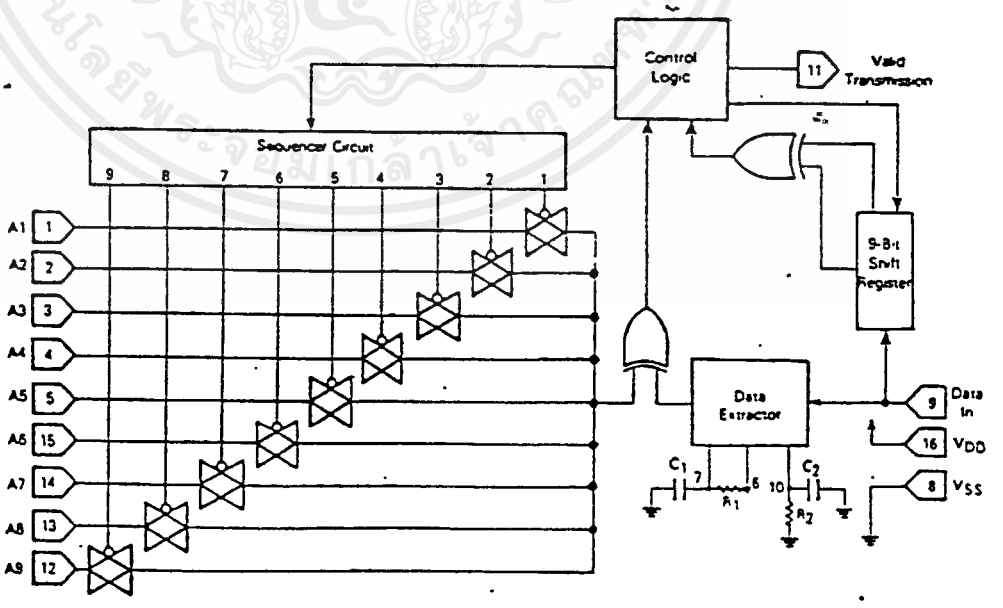
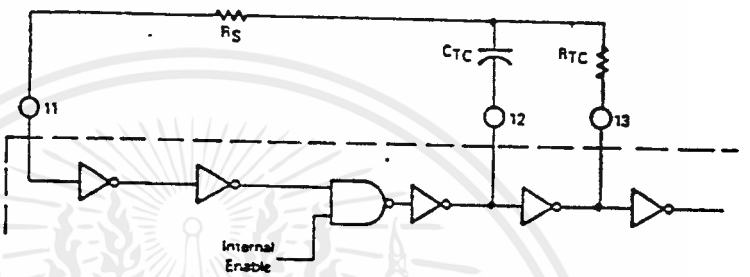


FIGURE 4 — ENCODER OSCILLATOR INFORMATION



This oscillator will operate at a frequency determined by the external RC network; i.e.,

$$f \approx \frac{1}{2.3\, R_{TC}\, C_{TC}'} \text{ (Hz)}$$

for 1 kHz ≤ f ≤ 400 kHz

where: $C_{TC}' = C_{TC} + C_{layout} + 12$ pF

$R_S = 2\, R_{TC}$

$R_S \geq 20$ k

$R_{TC} \geq 10$ k

400 pF < C_{TC} < 15 pF

The value for R_S should be chosen to be about 2 times R_{TC}. This range will ensure that current through R_S is insignificant compared to current through R_{TC}. The upper limit for R_S must ensure that R_S + 5 pF (input capacitance) is small compared to $R_{TC} + C_{TC}$.

For frequencies outside the indicated range, the formula will be less accurate. The actual oscillation range of this circuit is from less than 1 Hz to over 1 MHz.

FIGURE 5 — ENCODER/DECODER TIMING DIAGRAM

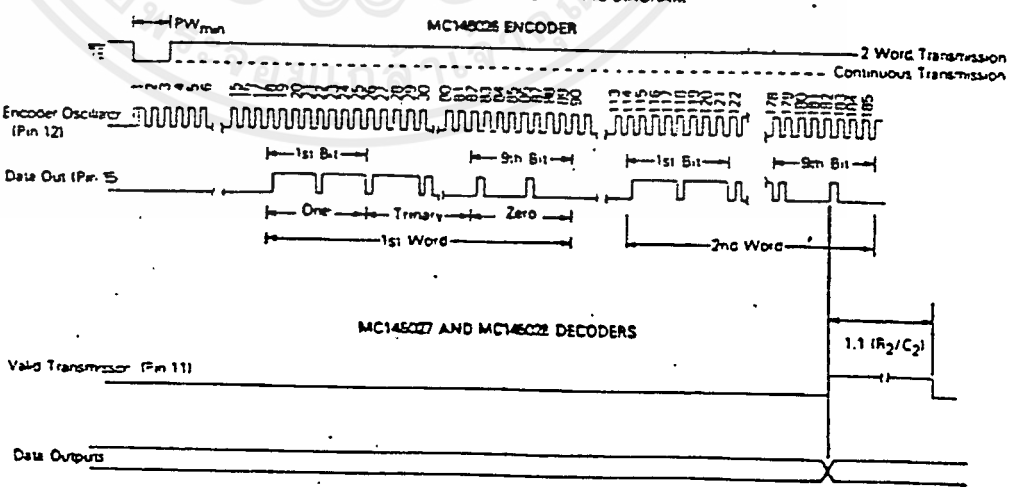


FIGURE 6 — ENCODER DATA WAVEFORMS (MC145026)

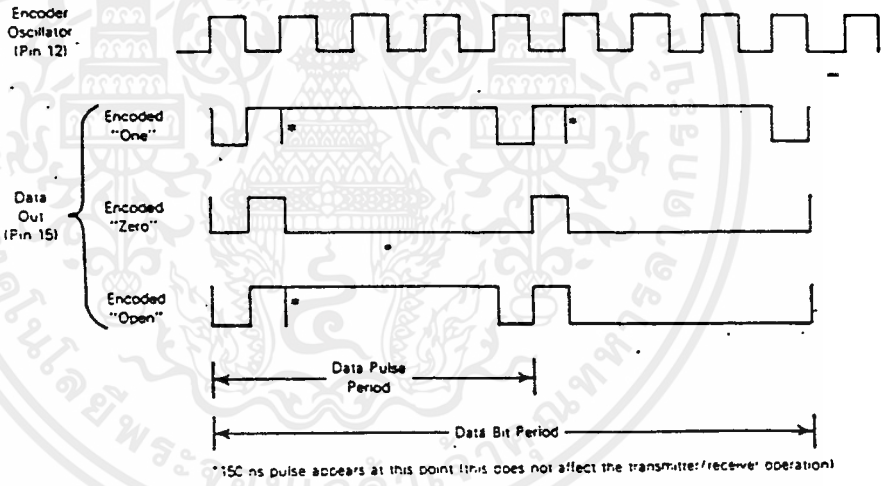


FIGURE 7 — MC145027 FLOWCHART

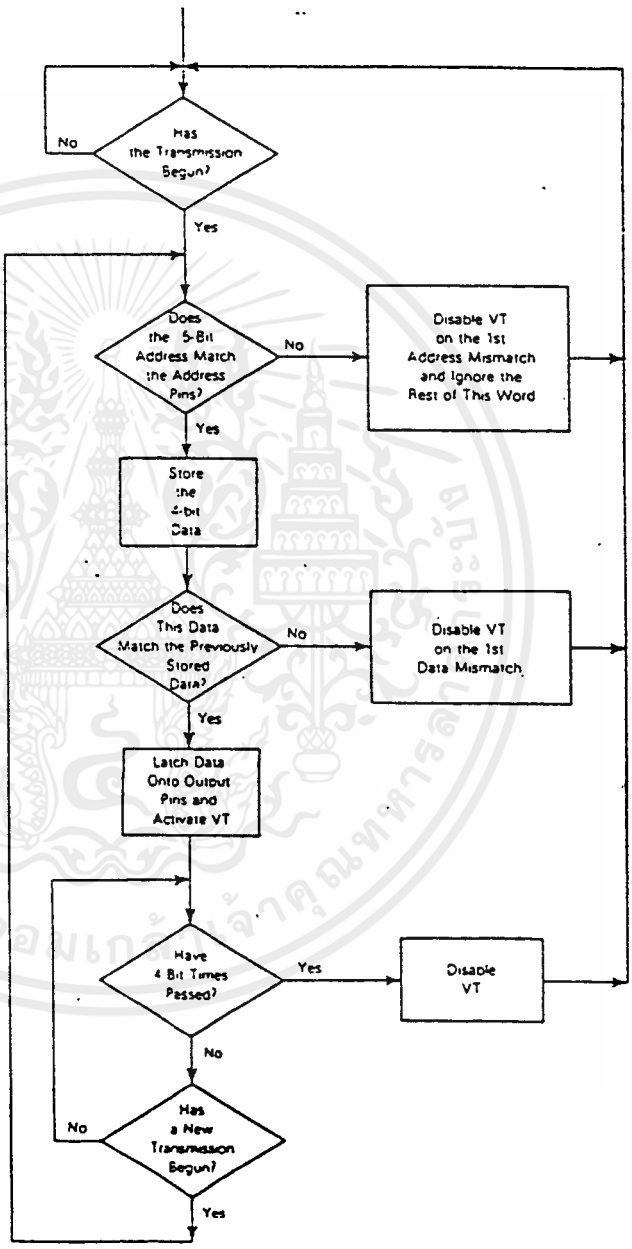
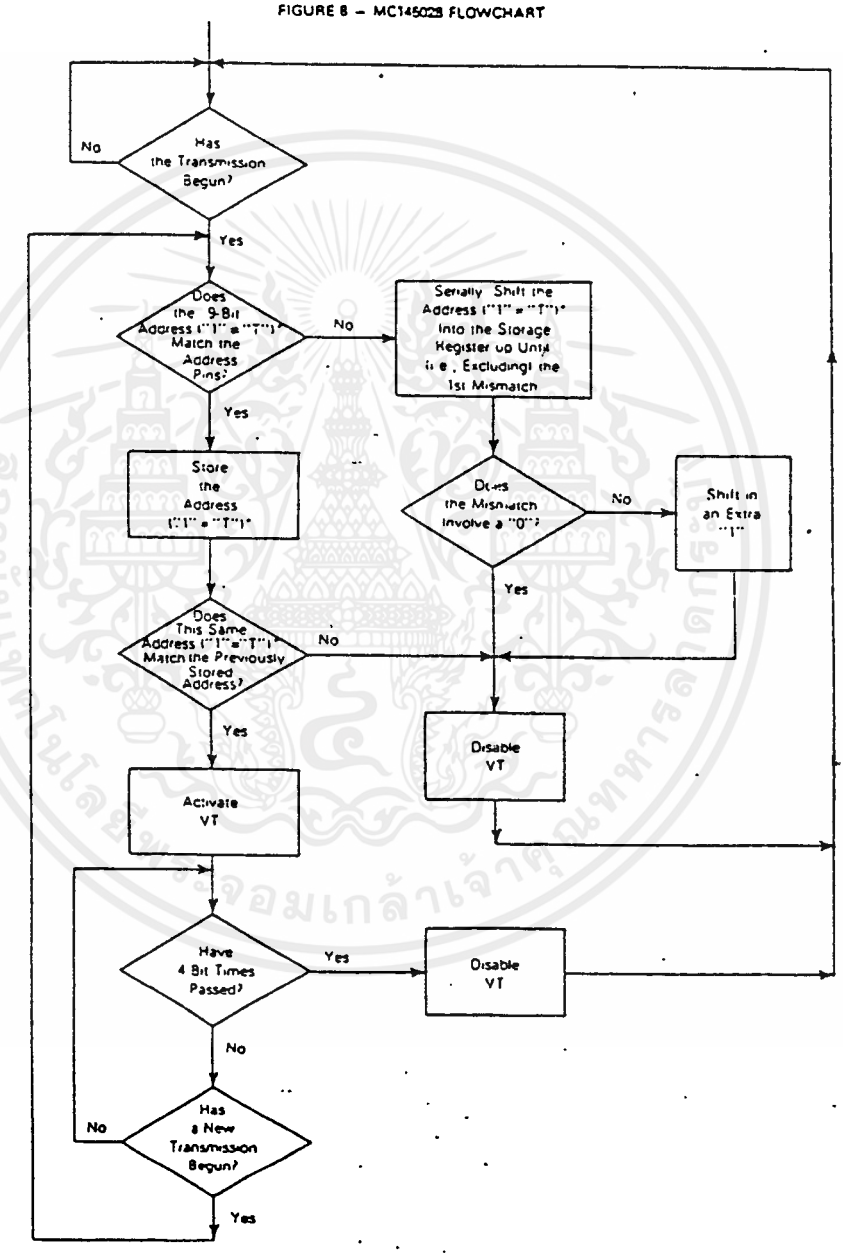
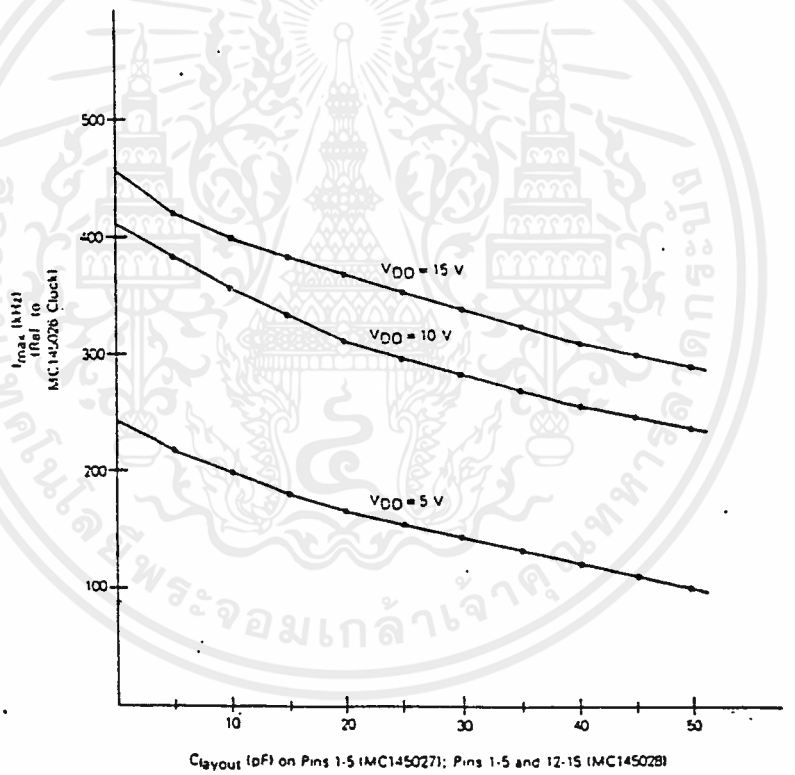


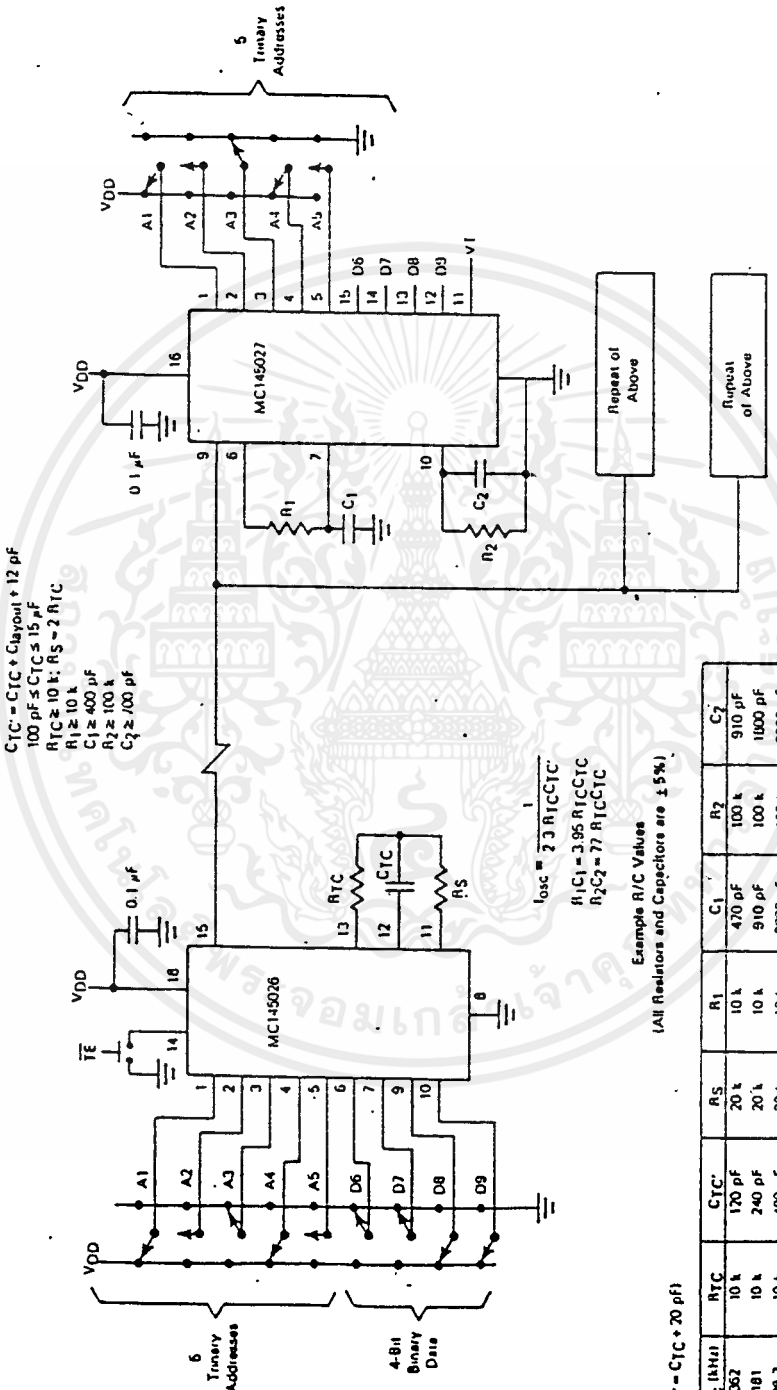
FIGURE 8 – MC145028 FLOWCHART



*For shift register comparisons, a "T" is stored as a "1"

FIGURE 9 – MC145027/MC145028
f_{max} vs C_{layout}





fosc (kHz)	RTC	CTC*	RS	R1	C1	R2	C2
362	10 k	120 pF	20 k	10 k	470 pF	100 k	910 pF
181	10 k	240 pF	20 k	10 k	910 pF	100 k	1800 pF
80.7	10 k	490 pF	20 k	10 k	2000 pF	100 k	3900 pF
42.0	10 k	1020 pF	20 k	10 k	3900 pF	100 k	7500 pF
21.6	10 k	2020 pF	20 k	10 k	8200 pF	100 k	0.015 μF
8.63	10 k	5100 pF	20 k	10 k	0.02 μF	200 k	0.02 μF
1.71	50 k	5100 pF	100 k	50 k	0.02 μF	200 k	0.01 μF

*CTC = CTC + 20 pF)

FIGURE 10 – TYPICAL APPLICATION

กิตติกรรมประกาศ

คณะผู้จัดทำขอขอบคุณผู้ที่ให้คำปรึกษาแนะนำทุกท่าน โดยเฉพาะอาจารย์ที่ปรึกษา คือ ผศ.พิพัฒน์ เลาหสงคราม ที่กรุณาให้คำปรึกษาและแนะนำในการทำปริญญานิพนธ์นี้ จนสำเร็จลุล่วงไปด้วยดี.

หากคุณความดีของปริญญานิพนธ์นี้มีอยู่บ้าง ผู้เขียนขอมอบให้กับผู้ที่มีพระคุณต่อผู้เขียนทุกท่าน

นาย ยิ่งยง ไชยฟู
นาย สถาพร จิตหัตถะ
นาย ปรีดา แซ่อ้อ

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

บรรณานุกรม

1. ไพบูลย์ อังคะตะลาพร,มานพ ปิ่นเมณีนพรัตน์,วิชัย ชินสกุลเจริญ
" ปริญญานิพนธ์ บ้านอัตโนมัติ (HOME AUTOMATIC V.2) ปีการศึกษา 2534
ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง "
2. Kenneth J. Ayala," The 8051 Microcontroller ARCHITECTURE ,
PROGRAMMING , and APPLICATIONS ",Western Carolina University
,WEST PUBLISHING COMPANY
3. Nationnal Semiconductor Corporation
" Special Purpose Linear Devices Databook "
1989 Edition
- LM 1893 / LM 2893 Carrier Current Transceivers
4. คู่มือ/เทียบเบอร์ ไอซี TTL บริษัทซีเอ็ดยูเคชั่น จำกัด
5. คู่มือ ไอซี CMOS 4000 SERIES บริษัทซีเอ็ดยูเคชั่น จำกัด
6. คู่มือ ไอซี CMOS 54C/74C SERIES บริษัทซีเอ็ดยูเคชั่น จำกัด
7. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 86 มิถุนายน 2531
8. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 89 พฤศจิกายน-ธันวาคม 2531
9. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 93 กรกฎาคม 2532
10. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 96 ตุลาคม-พฤศจิกายน 2532
11. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 99 เมษายน-พฤษภาคม 2533
12. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 114 กุมภาพันธ์ 2535
13. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 115 มีนาคม-เมษายน 2535
14. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 116 พฤษภาคม 2535
15. วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ ฉบับที่ 117 มิถุนายน 2535