สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

USING PL/M-51 COMPILER

โดย
นางสาวกึ่งแก้ว ปานน้อย
นายปิยะ พลายโถ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมไฟฟ้า
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2534

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

009607

บริญญนิพนธ์ปีการศึกษา 2534
เรื่อง USING PL/M-51 COMPILER
ผู้จัดทำ นางสาว กิ่งแก้ว ปานน้อย
นาย บิยะ พลายโถ

.....อาจารย์ที่ปรึกษา
(ผศ. พิพัฒน์ เลาหสงคราม)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

USING PL/M-51 COMPILER

นางสาว กิ่งแก้ว ปานน้อย

นาย ปิยะ พลายโถ

อ.พิพัฒน์ เลาหสงคราม

ปีการศึกษา 2534

บทคัดย่อ

ปริญญานิพนธ์ ฉบับนี้ เป็นการศึกษา เกี่ยวกับภาษา PL/M-51 เพื่อนำมาเขียนโปรแกรมไมโครคอนโทรลเลอร์ (MCS-51) มาใช้ในการควบคุมเครื่องปรับอากาศทำให้เครื่องปรับอากาศสามารถทำงานได้อย่างมีประสิทธิภาพมากขึ้นและประหยัดพลังงานไมโครคอนโทรลเลอร์จะทำการประมวล , ตัดสินใจในการควบคุมการทำงานของคอมเพรสเซอร์ (COMPRESSOR) และพัดลมภายใน (INDOOR FAN) ของเครื่องปรับอากาศตามความต้องการของผู้ใช้โดยผ่านทางรีโมทสวิทช์ (REMOTE SWITCH)

หลักการที่ใช้ควบคุมการทำงานของคอมเพรสเซอร์และพัดลมภายในก็คือใช้ความสัมพันธ์ระหว่างค่าความแตกต่างของอุณหภูมิห้องของผู้ใช้กำหนดกับระยะเวลาการทำงานของเครื่องปรับอากาศ จากหลักการดังกล่าวการทำงานของคอมเพรสเซอร์ก็จะถูกแบ่งออกเป็นโซน (ZONE) แต่ละโซนก็จะถูกควบคุมการ ON และการ OFF คอมเพรสเซอร์ด้วยช่วงเวลาที่ต่างกันเพื่อทำให้อุณหภูมิห้องขณะนั้นเป็นไปตามความต้องการของผู้ใช้ และประหยัดพลังงานมากที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

USING PL/M-51 COMPILER

KINGKAIL PANOIN

PIYA PLAITHO

PHIPAT LUAHASONGKRAM ADVISOR

GRADUATE YEAR 1991

ABSTRACT

This thesis is education PL/M-51 programming apply for Micro controller (MCS-51) to control air condition system. The air condition system can be operated with high efficiency and we can save energy too . The function of it is refer to processing decission in order to control the conpressor operation and indoor fan of air condition system. The operation of air condition system is depen on the user which can use remote switch for controling the system.

The principle that use to control the compressor and indoor fan is the relative of temperature difference between room temperature and setting temperature operating time.

The operating of the compressor is devided into zone ,each zone will control the state (ON /OFF) of the compressor by using interval time difference.

So that the. use can control room's temperature with high efficiency and saving more energy than the other.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทนำ

Project นี้เป็นการนำภาษา PL/M-51 ประยุกต์ใช้งานโดยใช้งานร่วมกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 เพื่อเขียนโปรแกรมควบคุมการทำงานของเครื่องปรับอากาศเมื่อมีการใช้งานกับรีโมทแบบมีสาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

โครงสร้างสถาปัตยกรรม MCS-51

ลักษณะทั่ว ๆ ไปของ MCS-51 จะประกอบด้วย

1. สร้างโดยใช้ HMOSและCHMOS/เทคโนโลยีและการทำงานด้วยแหล่งจ่ายไฟขนาด 5 V เพียงแหล่งเดียว
2. ซีพียูมีขนาด 8 บิต
3. มีวงจรออสซิเลเตอร์ และวงจรนาฬิกาบนชิป
4. ชุดแบงค์เรจิสเตอร์มี 4 ชุด และตัวรีจิเตอร์มี 8 ตัว ทำงานเช่นเดียวกับ MCS-48
5. มีตัวจับเวลา/ตัวนับ ขนาด 16 บิต 2 ชุดและสำหรับเบอร์ 8032/8052 มี 3 ชุด
6. มีพอร์ตไอโอแบบขนานสองทิศทางจำนวน 4 พอร์ต พอร์ตละ 8 บิตรวมทั้งหมดเป็น 32 เส้น แต่จะเหลือเพียง 16 เส้น สำหรับเบอร์ 8031 อีก 16 เส้นใช้ในการเข้าถึงทางแอดเดรสและข้อมูล
7. พอร์ตแบบอนุกรมสามารถที่จะโปรแกรมการรับส่งแบบ FULL DEPLEX ที่ความเร็วสูง
8. หนึ่งวัฏจักรคำสั่งจะใช้เวลา 1 ไมโครวินาที ด้วยการใช้คริสทัล 12 MHZ
9. แอดเดรสข้อมูลภายนอกได้ 64 กิโลไบต์
10. แอดเดรสโปรแกรมภายนอกได้ 64 กิโลไบต์
11. สามารถกำหนดเลขที่อยู่เลขข้อมูลขนาดไบต์หรือบิตได้โดยตรง
12. มีซอฟต์แวร์แฟลกสำหรับผู้ใช้ที่จะกำหนดเองได้ถึง 128 ตำแหน่งบิต
13. โครงสร้างอินเตอร์รัพท์ทำได้ 5 แหล่ง และ 6 แหล่งสำหรับ 8032/8052 พร้อมด้วยการจัดไพรโอริตี้ ได้ 2 ระดับ
14. ตัวโพรเซสเซอร์สามารถใช้ทำงานแบบบูลีนได้ สำหรับการใช้งานควบคุม
15. มีคำสั่งคูณ และหารทางฮาร์ดแวร์ทำได้ภายใน 4 ไมโครวินาที
16. ตัวเลขทางคณิตศาสตร์ ใช้ได้ทั้งแบบไบนารี่ และ เดซิมอล
17. การใช้พื้นที่สแต็กสำหรับโปรแกรมย่อยต่าง ๆ ทำได้กว้างขึ้น
18. ชุดคำสั่งของ MCS-51 จะมีมากกว่าชุดคำสั่งของ MCS-48

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตระกูล MCS-51 จะมีทั้งแบบ ROM ในตัว หรือไม่มี ROM หรือมี EPROM บนชิบเดียวกันและจะมีตำแหน่งขาที่เหมือนกัน รูปที่ 1 แสดงถึงตารางรายละเอียดของเบอร์ต่าง ๆ ในตระกูล MCS-51

เบอร์	หน่วยความจำภายใน		ตัวตั้งเวลา/ตัวนับจำนวน	อินเตอร์รัพต์
	โปรแกรม	ข้อมูล		
8052 AH	8K X 8K ROM	256 X 8 RAM	3 X 16 BIT	6
8051 AH	4K X 8K ROM	128 X 8 RAM	2 X 16 BIT	5
8051	4K X 8K ROM	128 X 8 RAM	2 X 16 BIT	5
8032 AH	ไม่มี ROM	256 X 8 RAM	3 X 16 BIT	6
8031 AH	ไม่มี ROM	128 X 8 RAM	2 X 16 BIT	5
8751 H-12	4K X 8 EPROM	128 X 8 RAM	2 X 16 BIT	5

รูปที่ 1 ตารางรายละเอียดของตระกูล MCS-51

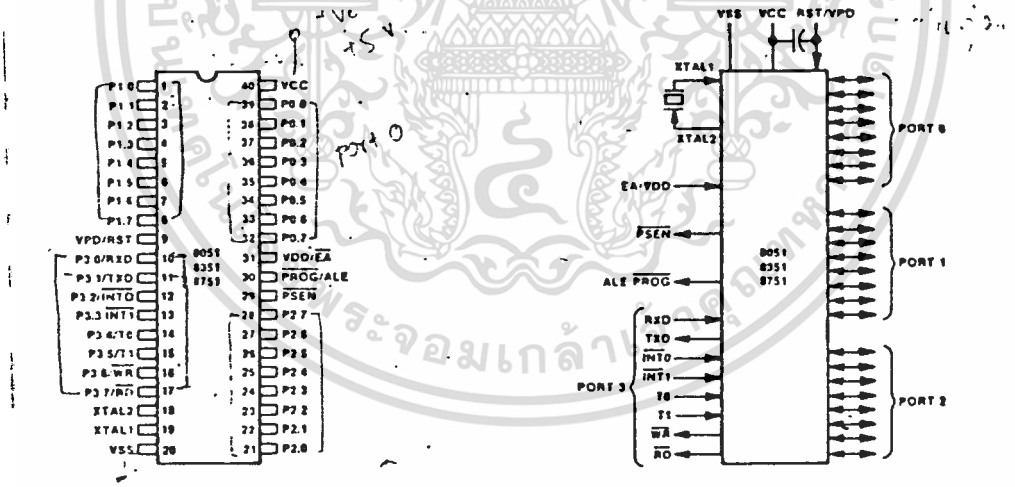
8751 H อยู่ในกลุ่มเดียวกันกับ 8051AH ที่เราสามารถโปรแกรมได้ด้วยระบบไฟฟ้าสามารถลบโปรแกรมออกด้วยแสงอุลตร้าไวโอเล็ต นอกเหนือจากไอซีที่แสดงในตารางข้างบนที่ใช้เทคโนโลยี CHMOS ที่ประหยัดพลังงานได้มากกว่า 4 เท่าของ HMOS ที่มีจำหน่ายในขณะนี้คือ 80C51,80C31 และ 87C51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจัดขาลักษณะภายนอกของ MCS-51

รูปที่ 2 แสดงการจัดขาตามลักษณะภายนอกของชิป MCS-51 ซึ่งมีรายละเอียดดังต่อไปนี้ คือ

- ขา Vss (ขา 20) เป็นขาสำหรับต่อลงดิน
- ขา Vcc (ขา 40) เป็นขาที่ต่อแรงดันไฟกระแสตรงขนาด 5 V และใช้สำหรับการโปรแกรม
- ขา PORT 0 (P0.0-P0.7/AD0-AD7) (ขา 32-39) เป็นพอร์ตไอโอ 8 บิต แบบ OPEN DRAIN BIDIRECTIONAL สามารถจะรับโหลด TTL ได้ถึง 8 ตัว การเขียนค่า '1' ไปที่พอร์ตนี้ จะเป็นการปล่อยลอย (FLOAT) ขาของพอร์ตนี้ ทำให้มันทำงานเป็นอินพุท มีสถานะอิมพีแดนซ์สูง ในการให้พอร์ตนี้บริการแก่ไอโอ พอร์ต 0 จะทำงานเป็นมัลติเพลกซ์ ด้วยสัญญาณแอดเดรสไบต์ต่ำกับบัสข้อมูล สำหรับการใช้งานด้านหน่วยความจำภายนอก ในการใช้งานแบบนี้จะใช้ลักษณะภายในเป็นตัวพูลอัพ พอร์ต 0 ยังใช้งานเป็นตัวส่งข้อมูลออกทางพร์อตนี้ เมื่อใช้บริการทางด้านตรวจสอบโปรแกรม ROM ภายใน และการโปรแกรมตัว EPROM ภายใน ถ้าใช้งานในลักษณะนี้การพูลอัพจากภานนอกจะต้องต่อด้วยค่า 10 กิโลโอห์ม



รูปที่ 2 A)ลักษณะขาภายนอกของ MCS-51 B)สัญญลักษณ์ทางตรรกของ MCS-51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ขา PORT 1 (P1.0 -P1.7) (ขา 1 - 8) เป็นพอร์ตไอโอ 8 บิตแบบ OPEN DRAIN BIDIRECTIONAL พร้อมด้วยการพูลอัพภายใน ถ้าเป็นพอร์ตเอาต์พุท บัฟเฟอร์สามารถขับโหลด TTL ตระกูล LSI ได้ 4 ตัว พอร์ต 1 เมื่อถูกเขียนค่า '1' ด้วยโปรแกรม มันจะมีสถานะสูงด้วยการพูลอัพภายใน การให้สถานะเช่นนี้ จะเป็นการ INITIAL ใช้งานพอร์ตนี้ให้เป็นอินพุต ขณะที่พอร์ต 1 เป็นอินพุท การให้สัญญาณลงต่ำจะเป็นการจ่ายกระแสออกเนื่องจากการพูลอัพภายใน

  ในเบอร์ 8052 ขา P1.0 และ P1.7 จะใช้งานเป็น T2 และ T2EX โดยขา T2 จะทำหน้าที่รับสัญญาณจากภายนอกให้ตั้งเวลา 2 ทำงาน และขา T2EX จะเป็นอินพุทผ่านเข้าตัวตั้งเวลา 2 ถูกกระตุ้นให้ทำงานแบบปกติตามโปรแกรมที่ตั้งไว้ หรือ เค็บเจอร์
- ขา PORT 2 (P2.0 - P2.7) (ขา 21 - 28) เป็นพอร์ตไอโอ 8 บิตแบบ OPEN DRAIN BIDIRECTIONAL ด้วยการพูลอัพภายใน พอร์ต 2 ที่ทำหน้าที่เป็นบัฟเฟอร์เอาท์พุตสามารถจ่ายโหลด TTL ตระกูล LSI ไได้ 4 ตัว พอร์ตจะถูกใช้งานเป็นตัวส่งแอดเดรสไบต์สูงด้วย เมื่อใช้งานร่วมกับหน่วยความจำภายนอกเพื่อให้แอดเดรสได้ถึง 16 บิต ด้วยการใช้งานแบบนี้มันจะมีพูลอัพภายในที่ช่วยให้การส่งค่า '1' ได้ระดับที่แน่นอนนอกจากนี้การใช้งายสำหรับแอดเดรสอันดับสูงยังใช้เป็นขาควบคุมในการใช้งานตรวจสอบ และการเขียนโปรมแกรมเบอร์ 8751 และตรวจสอบโปรแกรมภายใน 8051
- ขา PORT 3 (P3.0 - P3.7) (ขา 10 - 17)เป็นพอร์ตไอโอ 8 บิทแบบพูลอัพภายในนอกจากทำเป็นพอร์ตไอโอที่สามารถรับโหลด TTL ตระกูล LSI ได้ 4 ตัว แล้วยังใช้งานเป็นพิเศษสำหรับตระกูล MCS-51 ตามรายการข้างล่างนี้ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขาพอร์ต	ขา	การทำงานตามฟังก์ชั่นพิเศษ
P3.0	10	RxD พอร์ตอนุกรมอินพุท
P3.1	11	TxD พอร์ตอนุกรมเอาท์พุต
P3.2	12	INTO อินเตอร์รัพท์ภายนอกตัวที่ 1
P3.3	13	INT1 อินเตอร์รัพท์ภายนอกตัวที่ 2
P3.4	14	TO สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 0
P3.5	15	T1 สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 1
P3.6	16	WR สัญญาณควบคุมการเขียน
P3.7	17	RD สัญญาณควบคุมการอ่าน

การที่จะให้ทำงานตามฟังก์ชั่นข้างบน จะต้องเริ่มโปรแกรมด้วยการส่งค่า '1' ไปแลทช์ไว้ก่อนที่ให้ทำงานตามฟังก์ชั่นข้างบน

ขา RST (ขา 9) ต้องคงสถานะค่าสูงเป็นเวลาประมาณอย่างน้อยสองวัฏจักรระหว่างที่ออสซิเลเตอร์ทำงานขณะที่ต้องการรีเซ็ททั้งระบบทำงาน โดยจะต่อเรจิสเตอร์พูลดาวน์ (8.2กิโลโอห์ม) จากขา RST ไปต่อลงดิน และเพื่อให้ตัวชิบรีเซ็ทได้โดยอัตโนมัติ ขณะเปิดไฟให้ใช้คาบาซิเตอร์ (10 ไมโครฟารัด) ต่อคร่อมระหว่างขา RST กับ ขา Vcc

ขา ALE/PROG (ขา 30) เป็นขาแอดเดรสแลทช์อีนาเบิ้ลด้วยการส่งพลัลพ์ออกไปใช้สำหรับแลทช์ค่าแอดเดรสไบต์ต่ำจากพอร์ต 0 ในระหว่างการเข้าถึงข้อมูลจากหน่วยความจำภายใน ALE จะถูกส่งสัญญาณนาฬิกาออกมา ในอัตราความเร็วคงที่ที่ 1/8 ของความถี่ออสซิเลเตอร์ตลอดเวลา แม้ว่าจะไม่มีการเข้าถึงข้อมูลภายในดังนั้นจึงสามารถที่จะใช้สัญญาณจากขานี้เป็นตัวตั้งเวลาภายนอกหรือเป็นความถี่สัญญาณนาฬิกา แต่อย่างไรก็ตามความถี่สัญญาณนี้จะลดความถี่ช้าลงไปเท่าหนึ่งระหว่างการทำงานแบบเข้าถึงหน่วยความจำข้อมูลภายนอก ขานี้ยังเป็นสัญญาณพัลซ์เข้าสำหรับควบคุมโปรแกรม EPROM ภายในชิบ

ขา PSEN (ขา 29) PROGRAM STORAGE ENABLE เป็นสโตรบอ่านข้อมูลจากโปรแกรมหน่วยความจำภายนอก เมื่อชิบทำงานด้วยโปรแกรมภายนอก ขา PSEN จะสร้างสโตรบต่ำสองครั้งภายในแต่ละวัฏจักรแมชีน สัญญาณจะมีสถานะสูง หรือพัลซ์ต่ำทั้งสองลูกจะหายไป เมื่อทำงานในช่วงการอ่านหรือเขียนข้อมูลจากหน่วยความจำข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภายนอกและPSEN จะไม่มีพัลซ์ส่งออกถ้าชิบทำงานด้วยโปรแกรมหน่วยความจำภายใน

ขา EA/Vpp (ขา 31) มีสถานะสูง ตัว CPU ในชิปจะทำงานตามโปรแกรมที่อยู่ในหน่วยความจำภายใน (โดยที่โปรแกรมจะต้องไม่ยาวกว่า 4 กิโลไบต์ สำหรับเบอร์ 8051 AH และ 8 กิโลไบต์ สำหรับเบอร์ 8052 AH) การทำให้ EA มีสถานะต่ำจะเป็นการควบคุมให้ CPU ทำงานตามโปรแกรมหน่วยความจำภายนอก ซึ่งขยายโปรแกรมได้ยาวถึง 64 กิโลไบต์ ในตัว 8031 AH และ 8032 AH ขา EA จะต้องต่อลงดินเช่นกันแม้ว่าจะไม่มี ROM อยู่ภายในก็ตาม ในตัว 8751 H จะใช้ขานี้จ่ายแรงดันขนาด 21 V ขณะทำการเขียนโปรแกรมเข้า EPROM ของชิป 8751 H ตัวนี้

ขา XTAL1 ขา 19 ใช้เป็นตัวอินพุตเข้าสู่ออสซิเลเตอร์ขยายแบบ INVERT

ขา XTAL2 ขา 18 ใช้เป็นตัวเอาท์พุตเข้าสู่ออสซิเลเตอร์ขยายแบบ INVERT

ตามตาราง 1 MCS-51 ทั้งสามกลุ่ม คือ กลุ่มที่มี ROM ไม่มี ROM และพวก EPROM จะมีขาใช้งานเหมือนกันหมด ยกเว้นขา 1 จะใช้งานเป็น T2 และ ขา 2 เป็น T2EX ในเบอร์ 8032/8052 ตลอดถึงจังหวะเวลา (TIMING DIAGRAM) และคุณสมบัติทางไฟฟ้าทั้งสามจะแตกต่างกันเฉพาะการโปรแกรมบนชิป MCS-51 เท่านั้น ซึ่งแต่ละแบบจะจัดไปตามความต้องการของผู้ใช้ เช่น 8751 จะมี 4 กิโลไบท์ของ Ultraviolet-Erasable Programmable Read Only Memory (EPROM) เหมาะสำหรับการพัฒนาเครื่องต้นแบบ และการผลิตอุปกรณ์ที่มีจำนวนจำกัด เมื่อต้องการจะเขียนโปรแกรมเข้า EPROM จะมีตัวเขียนโปรแกรมพิเศษสำหรับเขียนโปรแกรมที่ผู้ออกแบบเขียนขึ้นมาได้ ถ้าโปรแกรมมีบักหรือส่วนผิดพลาดที่ต้องการจะแก้ไข ก็สามารถแก้ไขได้โดยการนำตัว 8751 นี้ไปล้างโปรแกรมเดิมออกด้วยแสงอุลตร้าไวโอเลต และอัดข้อมูลโปรแกรมที่ได้แก้ไขแล้วเข้าไปใหม่ ทำเช่นนี้จนกระทั่งได้โปรแกรมสมบูรณ์ และเมื่อต้องการผลิตจำนวนมากก็สามารถที่จะใช้ MCS-51 เบอร์ 8051 ที่มี 4 กิโลไบท์ของ ROM ซึ่งจะถูกอัดข้อมูลโปรแกรมตามความต้องการของผู้ออกแบบโดยโรงงานผู้ผลิตชิปเบอร์นี้ การผลิตลักษณะนี้จะถูกกว่าการใช้เบอร์ 8751 แต่โปรแกรมภายในไม่สามารถลบ และโปรแกรมใหม่ได้หลังการผลิตไปแล้ว

ส่วนเบอร์ 8031 จะไม่มีหน่วยความจำของโปรแกรมบนชิป แต่อาจจะต่อหน่วยความจำโปรแกรมจากภายนอกด้วย ROM EPROM หรือ PROM ได้ถึง 64 กิโลไบท์ ดังนั้น 8031 จึงเหมาะสำหรับการใช้งานที่โปรแกรมมีขนาดใหญ่กว่า 4 กิโลไบท์ และสำหรับผู้ออกแบบที่ต้องการแยกส่วนของโปรแกรมออกจากชิป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจัดการทางสถาปัตยกรรม

รูปที่ 4 เป็นบล็อกไดอแกรมที่แบ่งตามลักษณะงานทางสถาปัตยกรรมภายในของ MCS-51 โดยชิงเกิ่ลชิปแต่ละตัวของตระกูลนี้จะประกอบด้วยหน่วยศูนย์กลางประมวลผล หน่วยความจำสองชนิด คือ แบบ RAM กับ ROM หรือ EPROM พอร์ตเอาท์พุต อินพุต โหมดรีจิสเตอร์สถานะและข้อมูล ส่วนวงจรตรรกในการ RANDOM ที่จำเป็นสำหรับตัวแปรของฟังก์ชั่นการต่อพ่วงส่วนต่าง ๆ ที่กล่าวนี้จะติดต่อกันด้วยบัสข้อมูลขนาด 8 บิต และจะมีบัฟเฟอร์สำหรับการติดต่อข้อมูลกับภายนอกผ่านพอร์ตไอโอ เมื่อต้องการขยายหน่วยความจำหรือพอร์ตไอโอ

หน่วยศูนย์กลางประมวลผลหรือซีพียู

ซีพียูเป็นมันสมองของระบบไมโครคอมพิวเตอร์ การอ่านโปรแกรม และการทำงานตามคำสั่งโปรแกรมจะกระทำที่ส่วนนี้ โดยการใช้ส่วนคณิตศาสตร์ และตรรกศาสตร์ทำงานร่วมกับเรจิสเตอร์ A,B,PSW (Program Status Word) , SP (Stack Pointer) ตัวนับโปรแกรม (PC:Program Counter) ขนาด 16 บิต และตัวชี้ตำแหน่งข้อมูล (DPTR: DataPointer) ส่วนคณิตศาสตร์และตรรกศาสตร์ (ALU:Arithmetic Logic Unit) ALU นี้ทำงานในฟังก์ชั่นทางคณิตศาสตร์และตรรกศาสตร์ด้วยตัวแปรต่าง ๆ ขนาด 8 บิต ที่มีลักษณะการทำงานทางคณิตศาสตร์เป็น บวก ลบ คูณ หาร รวมทั้งทางตรรกศาสตร์ เช่น AND OR XOR รวมทั้งการเลื่อนข้อมูลและวนรอบบิต การเคลียร์ค่าและกลับค่า (Complement) เป็นต้น ALU ยังสามารถที่จะตัดสินใจในการให้กระโดดไปทำคำสั่งของโปรแกรมในส่วนอื่น ๆ ตามเงื่อนไขที่ตั้งขึ้นและยังแบ่งเรจิสเตอร์ชั่วคราวใช้สำหรับเป็นทางผ่านชั่วคราวของข้อมูลในการถ่ายเทภายในระบบคำสั่งอื่นที่มีการใช้ ALU ยังมีความสามารถที่จะเพิ่มค่าในเรจิสเตอร์ในลักษณะการบวกด้วยหนึ่ง (Increment)หรือคำนวณเลขที่อยู่ของข้อมูลที่จะนำไปเก็บหรือการลดค่าลงครั้งละหนึ่ง ในลักษณะการลบด้วยค่าหนึ่ง (Decrement) โดยอัตโนมัติ หรือใช้ในการเปรียบเทียบค่าของตัวแปรทั้งสอง

สิ่งสำคัญในการทำงานทางสถาปัตยกรรมของ MCS-51 คือ ความสามารถในการทำงานสำหรับข้อมูลขนาด 8 บิต และ 1 บิต การใช้งานในระดับบิตในการเซต เคลียร์ หรือ กลับค่า การเคลื่อนย้ายการทดสอบ และใช้ในการคำนวณทางตรรกขนาด 1 บิต ความสามารถเช่นนี้เหมาะสำหรับใช้ในงานควบคุมของสัญญาณเข้าและออกที่มีการคิดและออกแบบทางตรรกด้วยพีชคณิต Boolean ซึ่งโดยปกติทำได้ลำบากสำหรับไมโครโปรเซสเซอร์ทั่ว ๆไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

งานในลักษณะเช่นนี้จึงได้ชื่ออีกอย่างหนึ่งว่า [illegible] ผลบูลีน (Boolean Processor)

แอคคิวมูเลเตอร์ (Accumulator : ACC)

MCS-51 ก็เช่นเดียวกับ MCS-48 ที่ใช้ ACC ที่มีขนาด 8 บิต เป็นแอคคิวมูเลเตอร์หลักคำสั่งส่วนใหญ่จะอ้างถึงตัวเรจิสเตอร์นี้ โดยถือค่าภายในเป็นค่าตัวตั้งและรับค่าผลลัพธ์ที่ได้จากคำสั่งทางคณิตศาสตร์ เข้ามาเก็บไว้ ตัว ACC ยังสามารถใช้เป็นตัวแหล่งกระทำที่ถูกกระทำในการทำงานทางตรรก และใช้เป็นตัวกลางในการถ่ายโอนข้อมูลในการติดต่อกับอุปกรณ์ภายนอกไอโอ และหน่วยความจำภายนอก รวมถึงการตรวจสอบตารางข้อมูล

เรจิสเตอร์ B

เป็นเรจิสเตอร์พิเศษที่ใช้งานสำหรับคำสั่งของการคูณและหาร โดยใช้เป็นที่เก็บตัวคูณหรือตัวหารและเป็นที่เก็บผลลัพธ์ตัวที่สองหลังการคูณและเศษหลังการหาร

(MSB) (LSB)

CY	AC	FO	RS1	RSO	OV	-	P

สัญลักษณ์	ตำแหน่ง	ข้อกำหนดการทำงาน
CY	PSW7	แฟลกตัวทด จะเซต/เคลียร์ด้วยฮาร์ดแวร์หรือซอฟต์แวร์ระหว่างผลลัพธ์หลังการใช้คำสั่งทางคณิตศาสตร์ หรือตรรกศาสตร์ที่แน่นอน
AC	PSW6	แฟลกตัวทดของ Auxiliary จะเซต/เคลียร์ด้วยฮาร์ดแวร์ระหว่างการบวกลบ ที่แสดงผลจากการทดหรือยืมจากบิตที่ 3 ของ ACC
FO	PSW5	แฟลก 0 จะเซต/เคลียร์ด้วยซอฟต์แวร์ที่ผู้ใช้กำหนดสถานะแฟลกนี้เอง
RS1	PSW4	เรจิสเตอร์ตัวควบคุมการเลือกแบงค์ ด้วยค่า RS1 และ RSO

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

009607

RS0	PSW3	จะเซ็ต/เคลียร์ด้วยซอฟต์แวร์เพื่อเลือกกลุ่มเรจิสเตอร์ทำงานในแต่ละแบงค์ โดยปรับค่าใน RS1 และ RS0 ให้อีนาเบิ้ลคลุมลักษณะการเลือกแบงค์ต่อไปนี้

RS1	RS0	เลือกแบงค์	ค่าแอดเดรส
0	0	แบงค์ 0	00H - 07H
0	1	แบงค์ 1	08H - 0FH
1	0	แบงค์ 2	10H - 17H
1	1	แบงค์ 3	18H - 1FH

OV	PSW2	แฟลก Overflow จะเซต/เคลียร์ด้วยฮาร์ดแวร์ระหว่างการใช้คำสั่งที่แสดงผลถึงการเกิดลักษณะ Overflow ทางคณิตศาสตร์
-	PSW1	บิตสำรอง จะไม่สามารถเซต/เคลียร์ด้วยผู้ใช้ เพราะสำรองไว้สำหรับโรงงานผู้สร้าง
P	PSW0	แฟลกพาริตี้ จะเซต/เคลียร์ด้วยฮาร์ดแวร์ในแต่ละวัฏจักรคำสั่งแสดงถึงตัวเลขค่า '1' ในแต่ละบิตของแอคคิวมูเลเตอร์ เช่น '1' มี 6 ตัว จะเป็นพาริตี้คู่ P บิตจะเท่ากับ 0

รูปที่ 3 เรจิสเตอร์คำแสดงสถานะโปรแกรม PSW

หมายเหตุ ความหมายของฮาร์ดแวร์ และซอฟต์แวร์ในตารางต่าง ๆ ในแต่ละบิตของตัวเรจีสเตอร์ การที่บิตจะเซ็ตหรือเคลียร์นั้น ถ้าเกิดขึ้นจากฮาร์ดแวร์จะหมายถึงว่า ค่าบิตในเรจีสเตอร์จะเกิดเซ็ตตัวเองเนื่องจากผลของความหมายของการทำงานตามคำสั่งของบิตนั้น เช่น TI จะเซตตัวเองด้วยฮาร์ดแวร์ เมื่อการส่งข้อมูลได้สิ้นสุดถึง STOP บิตแล้ว ช่วยให้เราสามารถตรวจสอบได้ว่าการส่งข้อมูลครั้งละไบต์นั้นสิ้นสุดหรือยัง ถ้ายังจะได้รอต่อไปก่อน หรือมีการคำนวณแล้วผลลัพธ์เกิด OVERFLOW ใน PSW ก็จะเซ็ตตัวเองที่บิต OV ส่วนทางซอฟต์แวร์ หมายถึงว่าเราสามารถที่จะเซ็ต หรือเคลียร์ได้ด้วยการใช้คำสั่งต่าง ๆ ในการเซ็ตหรือเคลียร์ในบิตแต่ละบิตของเรจีสเตอร์เป็นลักษณะทางซอฟต์แวร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เรจีสเตอร์คำแสดงสถานะโปรแกรม (PROGRAM STATUS WORD : PSW)

เรจีสเตอร์ PSW เป็นเรจีสเตอร์ที่แสดงผลที่ได้หลังจากการใช้คำสั่งต่าง ๆ และใช้เป็นตัวเลือกกลุ่มการทำงานของเรจีสเตอร์กลุ่มต่าง ๆ ซึ่งมีรายละเอียดดังรูปที่ 3

ตัวชี้สแต็ก (STACK POINTER : SP)

MCS-51 จะรวมเอาสแต็กทางฮาร์ดแวร์ที่ใช้ RAM ภายในสำหรับการเชื่อมต่อระหว่างโปรแกรมหลัก สแต็กการผ่านพารามิเตอร์ระหว่างงานในแต่ละส่วนโปรแกรมและสแต็กเก็บตัวแปรข้อมูลชั่วคราว หรือสแต็กการเก็บสถานะระหว่างการบริการงานอินเตอร์รัพต์ไว้ภายในชิป โดยที่ SP จะมีขนาด 8 บิต จะเพิ่มค่าขึ้นโดยอัตโนมัติก่อนที่ข้อมูลจะนำมาเก็บในหน่วยความจำระหว่างการใช้คำสั่ง PUSH และ CALL และจะลดค่าของ SP ลงหลังจากที่ได้ถ่ายเทข้อมูลออกไปแล้วในคำสั่ง POP หรือ RETURN โดยทฤษฎีทางสถาปัตยกรรม MCS-51 สามารถใช้สแต็กให้มีเนื้อที่ถึง 128 ไบต์ แต่ในทางปฏิบัติ สำหรับโปรแกรมทั่วไปจะใช้น้อยกว่านี้ SP จะเริ่มที่ตำแหน่ง 07H ดังนั้น สแต็กจะเริ่มบรรจุข้อมูลที่ตำแหน่ง 08H MCS-51 สามารถเปลี่ยนแปลงค่าใน SP ได้ ซึ่งจะเป็นการเปลี่ยนแปลงตำแหน่งสแต็กไปยังที่ใด ๆ ของ RAM ภายในชิป

ตัวชี้ข้อมูล (DATA POINTER : DPTR)

DPTR เรจีสเตอร์ขนาด 16 บิตที่ประกอบด้วยไบต์สูง (DPH) และไบต์ต่ำ (DPL) ที่เราสามารถเลือกแบ่งออกเป็น เรจีสเตอร์ 8 บิตสองตัวที่ใช้ได้อย่างอิสระ หรือจะใช้รวมกันทั้ง 16 บิต ก็ได้ ในการ INCREMENT หรือ DECREMENT เพื่อประโยชน์ในการใช้เป็นฐานของเลขที่อยู่ในเรจีสเตอร์ในการกระโดดโดยทางอ้อมในการใช้คำสั่งเกี่ยวกับตารางข้อมูลและชี้แหน่งของหน่วยความจำภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พอร์ต 0 ถึง 3

เรจีสเตอร์ PO,P1,P2 และ P3 ของกลุ่มเรจีสเตอร์ฟังก์ชันพิเศษ (SPECIAL FUNCTION REGISTER : SFR) จะเป็นตัวเรจีสเตอร์ที่แลทช์ค่าของพอร์ต 0,1,2, และ 3 ตามลำดับ ในขณะใช้งาน

บัฟเฟอร์ข้อมูลอนุกรม

แบ่งออกเป็นเรจีสเตอร์สองตัว ตัวหนึ่งเป็นบัฟเฟอร์การส่งและอีกตัวเป็นบัฟเฟอร์การรับ เมื่อข้อมูลถ่ายเทเข้า SBUF มันจะถ่ายเข้าบัฟเฟอร์ซึ่งเป็นตัวจัดการส่งข้อมูลอนุกรม วิธีการเคลื่อนย้ายเข้า SBUF ขึ้นอยู่กับการเริ่มแรก (INITIAL)การส่งเมื่อข้อมูลย้ายออกจาก SBUF จะเป็นการรับข้อมูลจากบัฟเฟอร์ตัวรับ

เรจีสเตอร์ CAPTURE

ไอซีเบอร์8032/8052 จะมีคู่เรจีสเตอร์ (RCAP2H,RCAP2L) เพิ่มเติมเป็นเรจีสเตอร์เค็บเจอร์สำหรับตัวตั้งเวลาหมายเลข 2 ในโหมดการใช้งานของเรจีสเตอร์ตัวนี้จะรับการเปลี่ยนแปลงที่เข้ามาที่ขา T2EX ตัว TH2 และ TL2 จะลอกข้อมูลเข้าไปในเรจีสเตอร์คู่RCAP2H และ RCAP2L ด้วยการใช้ตัวตั้งเวลา จะมีโหมดการบรรจุอัตโนมัติขนาด 16 บิต สำหรับการใช้ตัวตั้งเวลา/ตัวนับ2 ซึ่งจะมีรายละเอียดในหัวข้อต่อไป

เรจีสเตอร์ควบคุม (CONTROL REGISTER)

กลุ่ม SFR ที่เป็น IP, IE, TMOD, TCON, T2CON, SCON และ PCON จะประกอบด้วยบิตที่ใช้ในการควบคุม และแสดงสถานะของการใช้งานในระบบอินเตอร์รัพต์ ตัวตั้งเวลา/ตัวนับ และพอร์ตอนุกรมซึ่งจะอธิบายโดยละเอียดในหัวข้อต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจัดการหน่วยความจำ

MCS-51 จะจัดแบ่งตำแหน่งสำหรับ SFR ให้ทำงานเป็นเรจิสเตอร์ต่าง ๆ ดังนี้

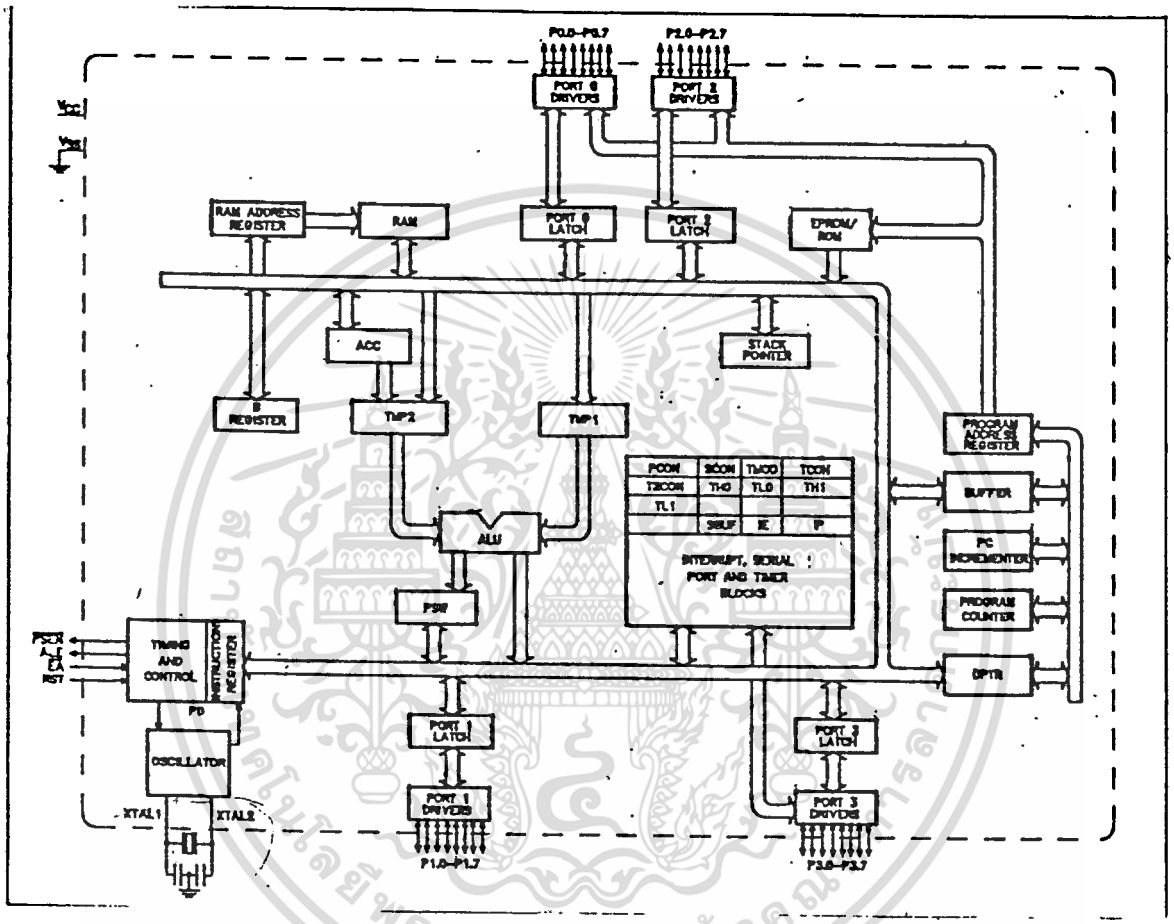
			ตำแหน่ง
*	ACC	Accumulator	0E0H
*	B	B เรจิสเตอร์	0F0H
*	PSW	Program Status Word	0D0H
	SP	Stack Pointer	081H
	DPTR	ตัวชี้ข้อมูล ประกอบด้วย DPL	083H
		และ DPL	082H
*	PO	พอร์ต 0	080H
*	P1	พอร์ต 1	090H
*	P2	พอร์ต 2	0A0H
*	P3	พอร์ต 3	0B0H
*	IP	ตัวควบคุมการอินเตอร์รัพท์ตามลำดับ	0B8H
*	IE	ตัวควบคุมการอินเตอร์รัพท์อีนาเบิ้ล	0A8H
	TMOD	ตัวควบคุมการเลือกโหมดตัวตั้งเวลา/ตัวนับ	089H
*	T2CON	ตัวควบคุมตัวตั้งเวลา/ตัวนับ 2	088H
	TCON	ตัวควบคุมตัวตั้งเวลา/ตัวนับ	0C8H
	THO	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 0 (ไบท์สูง)	08CH
	TLO	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 0 (ไบท์ต่ำ)	08AH
	TH1	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 1 (ไบท์สูง)	08DH
	TL1	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 1 (ไบท์ต่ำ)	08BH
+	TH2	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 2 (ไบท์สูง)	0CDH
+	TL2	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 2 (ไบท์ต่ำ)	0CCH
+	RLDH	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 2 ประจุใหม่อัตโนมัติ (ไบท์สูง)	0CBH
+	RLDL	เรจิสเตอร์ตัวตั้งเวลา/ตัวนับ 2 ประจุใหม่อัตโนมัติ (ไบท์ต่ำ)	0CAH
*	SCON	ควบคุมการส่งข้อมูลอนุกรม	098H
	SBUF	บัฟเฟอร์ข้อมูลการส่งอนุกรม	099H
	PCON	ควบคุมการใช้พลังงาน (Power)	097H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องหมาย * หน้าตัวเรจิสเตอร์แสดงว่า เรจิสเตอร์ตัวนั้นสามารถที่จะแอดเดรสข้อมูลได้ทั้งข้อมูลขนาดไบต์และบิต และเครื่องหมาย + นั้นแสดงว่าจะมีเฉพาะในเบอร์ 8032/8052 เท่านั้น

ตัว MCS-51 จะแยกแอดเดรสสำหรับหน่วยความจำของโปรแกรม และหน่วยความจำของข้อมูลออกจากกัน หน่วยความจำของโปรแกรมขยายได้ถึง 64 กิโลไบต์ และจำนวนไบต์ต่ำ 4 กิโลไบต์จะอยู่ใน 8051 หน่วยความจำของข้อมูลมี 128 ไบต์ (256 ไบต์สำหรับ 8032/8052) บนชิปและอีก 128 ไบต์ใช้สำหรับเรจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register : SFR) และหน่วยความจำข้อมูลภายนอกอีก 64 กิโลไบต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4 สถาปัตยกรรม MCS-51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โครงสร้างพอร์ตและการทำงาน

ใน MCS-51 มีพอร์ต 4 พอร์ต และทั้ง 4 พอร์ตเป็นแบบสองทิศทาง แต่ละพอร์ตจะประกอบด้วยแลทช์เป็น P0 ถึง P3 ของ SFR จะมีตัวขับเอาท์พุตและบัฟเฟอร์อินพุต ตัวขับเอาท์พุตของพอร์ต 0 และ 2 และขับบัฟเฟอร์อินพุตของพอร์ต 0 จะใช้งานสำหรับการเข้าถึงหน่วยความจำภายนอก ในการใช้งานในลักษณะนี้เอาท์พุตพอร์ต 0 จะทำหน้าที่เป็นตัวกำหนดไบต์ต่ำของแอดเดรสหน่วยความจำภายนอก โดยที่ค่าแอดเดรส และค่าข้อมูลจะถูกมัลติเพล็กซ์ด้วยช่วงจังหวะการเฟทช์และการอ่านหรือการเขียนข้อมูล ส่วนเอาท์พุตพอร์ต 2 จะทำหน้าที่เป็นตัวกำหนดส่งไบต์สูงของแอดเดรสในการเข้าถึงหน่วยความจำภายนอก

บางขาของตัวขับเอาท์พุตและบัฟเฟอร์อินพุทของขา 1.0,1.1 และ พอร์ต 3 ทั้งหมดสามารถนำไปใช้งานเป็นแบบหลายฟังก์ชั่น (MULTIFUNCTION) ได้ดังนี้

ขาพอร์ต	การใช้งานตามฟังก์ชั่น
P1.0	T2 (TIMER/COUNTER 2 สัญญาณอินพุทจากภายนอก)
P1.1	T2RST (TIMER/COUNTER 2 สัญญาณอินพุทการรีเซตภายนอก
P3.0	RxD (พอร์ตรับข้อมูลอนุกรม)
P3.1	TxD (พอร์ตส่งข้อมูลอนุกรม)
P3.2	INT0 (การใช้อินเตอร์รัพท์ภายนอกตัวที่ 1)
P3.3	INT1 (การใช้อินเตอร์รัพท์ภายนอกตัวที่ 2)
P3.4	T0 (TIMER/COUNTER 0 สัญญาณอินพุทภายนอก)
P3.5	T1 (TIMER/COUNTER 1 สัญญาณอินพุทภายนอก)
P3.6	WR (สโตรบการเขียนหน่วยความจำภายนอก)
P3.7	RD (สโตรบการอ่านหน่วยความจำภายนอก)

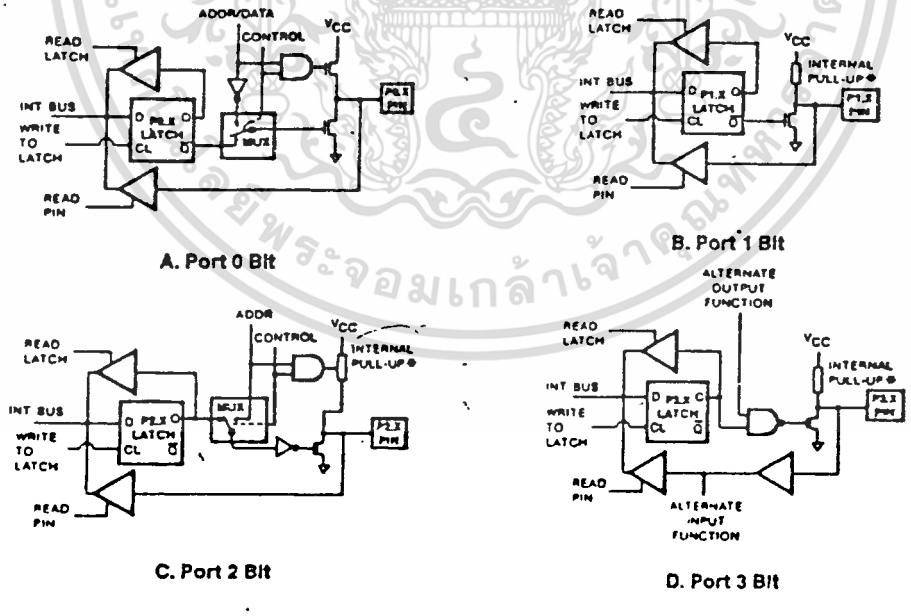
ตัวขับเอาท์พุตแลทช์ในการที่จะให้ทำงานตามตารางข้างบน จะต้องเริ่มโปรแกรมด้วยการเซ็ทค่า '1' เก็บค่าในแลทช์ก่อน เครื่องหมาย * แสดงถึงการใช้ตัว TIMER/COUNTER 2 ซึ่งมีเฉพาะในเบอร์ 8032/8052 เท่านั้น ทั้งโหมดตัวจับเวลาหรือตัวนับ การโหลดใหม่แบบอัตโนมัติของ TIMER/COUNTER 2 ที่เรจิสเตอร์ RLDH และ RLDL จะเกิดขึ้นถ้าการโหลดใหม่แบบอัตโนมัติถูกเลือกใช้งานด้วยการกำหนดในบิต CP/RL2 = 0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การกำหนดใช้งานไอโอ (I/O CONFIGURATION)

รูปที่ 5 เป็นพอร์ตวงจรการแลทช์และบัฟเฟอร์ แสดงรูปแบบของบิตแต่ละพอร์ตที่ พอร์ต 1.2 และ 3 จะมีพูลอัพภายใน พอร์ต 0 เอาท์พุตเป็น OPEN DRAIN แต่ละเส้นไอโอจะเป็นอิสระในการกำหนดเป็นอินพุตหรือเอาท์พุต พอร์ต 0 และ 2 อาจใช้เป็นไอโอทั่วไปไม่ได้ ถ้าถูกกำหนดให้ใช้เป็นบัสแอดเดรสและข้อมูลแล้ว กรณีการใช้พอร์ตเป็นอินพุทนั้นจะสามารถรับการเปลี่ยนแปลงระดับลอจิกจากระดับสูงสู่ระดับต่ำ เท่านั้น ดังนั้นการใช้ขาใดขาหนึ่งของพอร์ตเป็นอินพุตต้องทำการเซทขานั้นให้เป็นระดับสูงก่อน ซึ่งจะทำให้ตัวขับ FETS ทรานซิสเตอร์หยุดทำงานดังนั้นขาของพอร์ต 1,2 และ 3 ที่มีตัวต้านทานพูลอัพต่ออยู่จะยกระดับแรงดัน และทนกระแสได้เพียงพอกับระดับลอจิก TTL แต่สามารถจะลดระดับให้ต่ำด้วยแหล่งวงจรลอจิกภายนอกด้วยคุณสมบัติของขา 1,2,3 นี้ จะเรียกว่า QUASI-BIDIRECTIONAL

สำหรับพอร์ต 0 ค่า 1 ในพอร์ตที่แลทช์ไว้จะเป็นเหตุให้ขาภายนอกถูกปลดออกหรือลอยตัวปกติ พอร์ตทั้งหมดของ MCS-51 จะมีค่าเป็น '1' หลังการรีเซทแล้ว ถ้าค่า '0'
ถูกเขียนเข้าไปแลทช์ไว้ที่พอร์ต มันสามารถจะถูกกำหนดใหม่ให้เป็นอินพุตด้วยการเขียนค่า
'1'เข้าไปใหม่

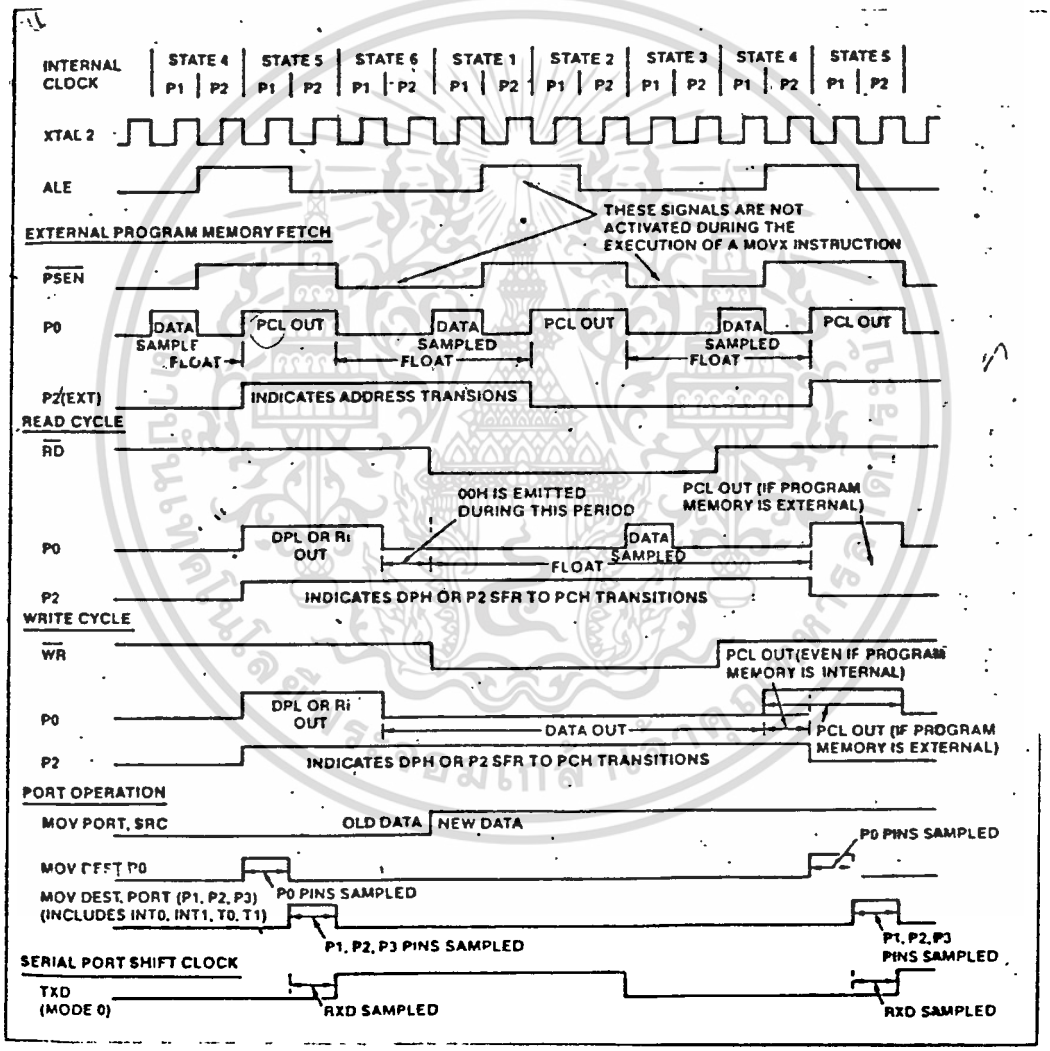


รูปที่ 5 เป็นพอร์ตการกำหนดแลทช์และบัฟเฟอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พอร์ต 0 มีความแตกต่างจากพอร์ตอื่นที่ไม่มีพูลอัพภายใน แต่มี FET ตัวบนมาต่อแทนเป็นตัวไดรเวอร์เอาต์พุต ดังรูป 5 A โดยถ้ามันให้วงจรทำงานจะเป็นการทำงานแบบบัสแอดเดรสและข้อมูล ด้วยการเข้าถึงข้อมูลภายนอกด้วยสัญญาณควบคุมภายใน แต่งานลักษณะนี้ปกติ FET จะไม่ทำงาน ในการใช้เรจีสเตอร์ P0 แลทช์ค่า "1"ไว้ตามวงจรเกทของ FET ตัวล่างจะต่อที่ Q ซึ่งเป็นลอจิกระดับต่ำ ทำให้ FET ทั้งสองไม่ทำงานทั้งคู่จะเป็นการปล่อยขาของพอร์ต 0 ให้ลอยตัว เป็นการให้อินพุตมีค่าอิมพีแดนซ์สูงนั่นเอง ขณะที่พอร์ต P2 ใน SFR จะมีค่าแอดเดรสที่ไม่เปลี่ยนแปลง

การเขียนไปยังพอร์ต

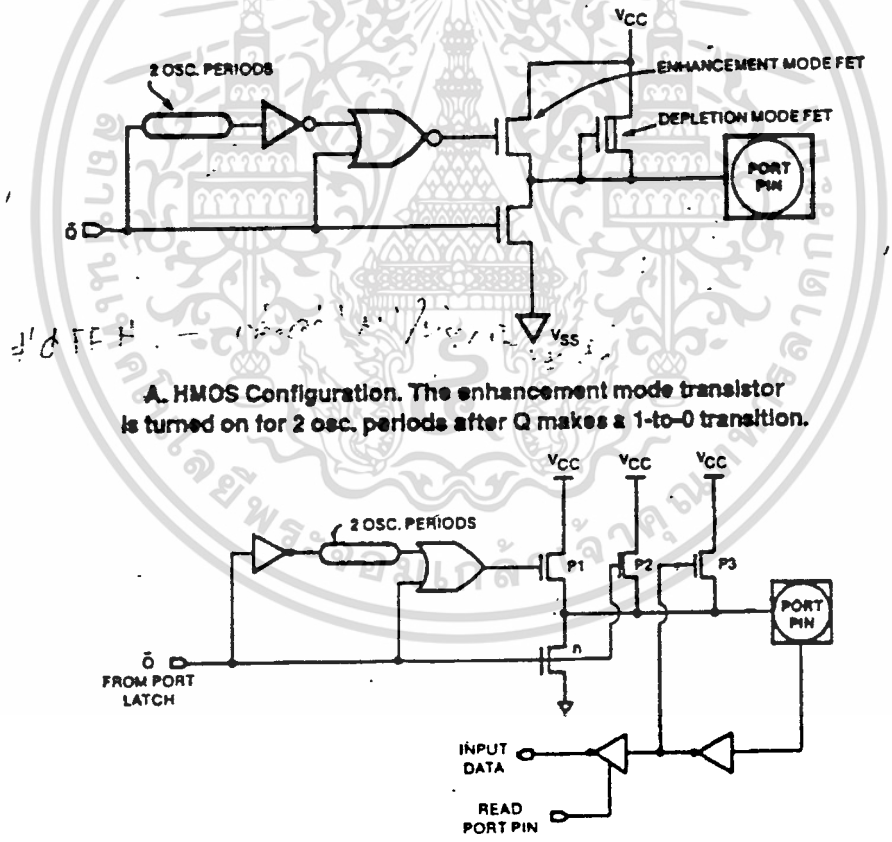


รูปที่ 6 วัฏจักรอ่านและเขียนไปยังพอร์ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานตามคำสั่งที่เปลี่ยนค่าในแลทช์ของแต่ละพอร์ต ค่าใหม่จะเข้ามาเก็บในช่วงระหว่าง S6P2 ของวัฏจักรสุดท้ายของคำสั่ง อย่างไรก็ตามพอร์ตจะเก็บค่าในแลทช์ เมื่อมีการใช้ส่งข้อมูลออกที่บัฟเฟอร์เอาต์พุตระหว่างเฟส 1 ของคาบเวลาใดๆ ของสัญญาณนาฬิกา (ส่วนระหว่างเฟส 2 บัฟเฟอร์เอาต์พุตจะยังคงเก็บค่าเริ่มแรกที่ปรากฏในเฟส 1 ก่อนหน้านั้น) โดยลำดับค่าใหม่ที่แลทช์ไว้ จะยังไม่ไปปรากฏที่ขาของพอร์ตจนกว่าจะถึงเฟส 1 ตัวใหม่ ซึ่งอยู่ในช่วง S1P1 ของวัฏจักรแมชีนตัวต่อมา ดังรูปที่ 6

ถ้าการเปลี่ยนจาก "0" เป็น "1" ของพอร์ต 1,2 หรือ 3 จะเป็นการให้พูลอัพทำงานในช่วงระหว่าง S1P1 และ S1P2 ของวัฏจักรเกิดการเปลี่ยนแปลงลักษณะงานเช่นนี้จะช่วยเพิ่มความเร็วของการเปลี่ยนแปลง การมีพูลอัพเพิ่มขึ้นสามารถที่จะจ่ายกระแสได้เพิ่มขึ้น 100 เท่า ตัวของการพูลอัพปกติ จะสังเกตได้ว่าการพูลอัพภายในเป็นเฟต ซึ่งไม่ใช่ตัวต้านทานเส้นตรง การจัดการพูลอัพมีแสดงในรูปที่ 7



รูปที่ 7 การจัดการพูลอัพภายในเริ่มแรกของพอร์ต 1,2 และ 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในพวก HMOS 8051 ส่วนที่คงที่ในการพูลอัพคือการใช้ทรานซิสเตอร์แบบ DEPLETIONMODE ด้วยการต่อเกตเข้าแหล่งกำเนิดแรงดัน ตัวทรานซิสเตอร์แบบนี้ที่ขาของพอร์ตจะสามารถจ่ายกระแสได้ประมาณ 0.25 มิลลิแอมป์ เมื่อเกิดลัดวงจรลงดิน ในการต่อขนานกับพูลอัพคงที่ของทรานซิสเตอร์แบบ ENHANCEMENT MODE ดังวงจรรูปที่ 7(A) ซึ่งจะแอ็กทีฟระหว่าง S1 เมื่อไรก็ตามที่บิตเกิดการเปลี่ยนแปลงจาก "0" เป็น "1" ระหว่างช่วงเวลานี้ ถ้าพอร์ตเกิดลัดวงจรลงดินจะทนกระแสได้ถึง 30 มิลลิแอมป์

ส่วนในพวก CHMOS พูลอัพจะประกอบด้วย pFETS สามตัว โดยที่ pFET จะเป็นพวก p-Channel ซึ่งจะทำงานเมื่อเกตของ FET มีค่าเป็น "0" และ pFET จะหยุดทำงาน เมื่อเกตมีค่าเป็น "1" ส่วน nFET ทำงานตรงข้ามกับ pFET

pFET ตัวที่ 1 ในรูปที่ 7(B) ทรานซิสเตอร์จะทำงานคลุมระยะ 2 คาบออสซิลเลเตอร์ หลังจากมีการเปลี่ยนแปลงค่าในพอร์ตแลทช์จาก "0" เป็น "1" ขณะที่ทำงาน มันจะทำให้ pFET3 ทำงานผ่านอินเวอร์เตอร์ อินเวอร์เตอร์ตัวนี้กับ pFET ที่ต่อจากพอร์ตแลทช์จะเก็บค่า "1" ไว้ จะสังเกตได้ว่า ขาที่มีค่าเป็น 1 ถ้ามี glitch ลงจากภายนอกเข้ามาก็สามารถที่จะทำให้ pFET3 หยุดทำงานได้ ทำให้ขาพอร์ตนี้อยู่ในสภาวะลอย pFET2 จะทำงาน ในขณะเดียวกันก็ทำให้ nFET หยุดทำงาน ดังนั้นการเก็บค่า 1 ใน pFET3 ก็จะอ่อนลงและจะสูญข้อมูลที่เก็บไว้ได้เมื่อมี glitch เข้าที่ขานี้

การบรรจุข้อมูลและการต่อพ่วงกับพอร์ต

เอาต์พุตของพอร์ต 1,2 และ 3 สามารถขับทีทีแอลสินพุตแบบแอสเอสได้ 4 ตัวพอร์ตเหล่านี้ของ HMOS สามารถขับได้ทั้งวงจรแบบทีทีแอล หรือ NMOS MCS-51 ทั้งแบบ HMOS และ CHMOS สามารถทำงานเป็นตัวขับลักษณะ Open-Collector เอาต์พุต Open-Drain โดยไม่ต้องมีพูลอัพจากภายนอก

พอร์ต 0 ตัวบัฟเฟอร์เอาต์พุตที่ขาสามารถขับทีทีแอลอินพุตได้ 8 ตัว อย่างไรก็ตามขาเหล่านี้ถ้าใช้ในการขับอินพุตแบบ nmos ต้องต่อพูลอัพจากภายนอก ยกเว้นการใช้งานเป็นบัสแอดเดรส/ข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบการอ่าน - เปลี่ยนค่า - เขียน (Read-Modify-Write)ผ่านพอร์ต

ด้วยลักษณะการใช้พอร์ตเป็นอินพุต ตามโครงสร้างรูป 7.12 บิตแลทช์แต่ละบิตพอร์ตของ SFR จะแทนด้วยวงจร D FLIP-LFOP ซึ่งจะถูกกระตุ้นให้เก็บค่าจากบัสภายในด้วยสัญญาณ "write to latch" ที่ถูกสร้างขึ้นด้วย CPU ค่า Q ที่เป็นเอาต์พุตของ FLIP-FLOP ก็จะจ่ายไปยังบัสภายในที่ถูกควบคุมด้วยสัญญาณ read latch จาก CPU ระดับสัญญาณที่ขาของพอร์ตจะไปบรากฏที่บัสภายในได้นั้น จะถูกควบคุมด้วยสัญญาณ read pin จาก CPU ดังนั้นบางคำสั่งในการอ่านค่าจากพอร์ตจะสร้างสัญญาณควบคุม read latch และบางคำสั่งจะสร้างสัญญาณ read pin ซึ่งควรจะแยกชนิดของคำสั่งทั้งสองนี้ได้ออกด้วยการทำความเข้าใจว่าถ้าเป็นชนิดแบบ read latch การทำงานของคำสั่งชนิดนี้จะมีลักษณะด้านการอ่านค่าผ่านเข้าพอร์ตแล้วทำการเปลี่ยนแปลงค่าที่อ่านเข้ามาแล้วทำการ latch ไว้ ซึ่งลักษณะงานเช่นนี้จะเป็นการอ่านค่า เปลี่ยนค่า และเขียนกลับไปที่แลทช์ใหม่เป็นกลุ่มคำสั่ง "Read-Modify-Write" ซึ่งเมื่อตัวโอเปอร์แรนด์ที่จะอ่านเข้ามาเป็นพอร์ตหนึ่งไบต์หรือบิตพอร์ต คำสั่ง Read-Modify-Write เหล่านี้จะแลทช์ข้อมูลมากกว่าที่จะส่งไปที่ขาโดยตรง ดังคำสั่งต่อไปนี้

```
ANL          (AND เช่น ANL P1, A)
ORL          (OR  เช่น ORL P2, A)
XRL          (EXOR เช่น XRL P3, A)
JBC          (JUMP ถ้าบิต = 1 และเคลียร์บิต เช่น
              JBC P1.1., LABEL)
CPL          (Complement บิต เช่น CPL P3.0)
INC          (Increment ข้อมูล เช่น INC P2)

DEC          (Decrement ข้อมูล เช่น กำแ p2)
DJNZ         (Decrement ข้อมูล และกระโดดถ้าข้อมูลไม่
              เป็นศูนย์ เช่น DJNZ P3, LABEL)
MOV  PX.Y,C  (ย้ายบิตทดไปบิต Y ของพอร์ต X)
CLR  PX.Y    (เคลียร์บิต Y ของพอร์ต X)
SET  PX.Y    (เซตบิต Y ของพอร์ต X)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามคำสั่งสุดท้ายไม่ค่อยแจ้งชัดว่าเป็นคำสั่งแบบ read-modify-write เพราะมันจะทำงานด้วยการอ่านทั้ง 8 บิตหรือไบต์จากพอร์ต แทนที่จะเป็นบิตเดียว ซึ่งทั้ง 8 บิตจะเซตค่าแอดเดรสบิตใหม่ ดังนั้น จึงเป็นการเขียนค่าตัวใหม่ทั้ง 8 บิตกลับเข้าที่แลทช์

เหตุผลหนึ่งที่คำสั่ง Read-Modify-Write เป็นการแลทช์ค่าพอร์ตโดยตรงมากกว่าการส่งออกที่ขา เพื่อเป็นการหลีกเลี่ยงความเป็นไปได้ที่อาจจะทำให้การการแสดงระดับสัญญาณที่ขาพอร์ตเกิดการผิดพลาดขึ้น ตัวอย่างเช่น พอร์ตหนึ่งบิตอาจใช้ขับตัวทรานซิสเตอร์ เมื่อค่า "1" ถูกส่งไปที่ขาบิตนั้นทรานซิสเตอร์จะทำงาน ถ้าซีพียูอ่านกลับในพอร์ตบิตเดียวกันที่ขามากกว่านี้ ดังนั้นการใช้วิธีอ่านค่าตัวแลกช์กลับไปก็จะได้ค่าที่ถูกต้อง คือ "1"

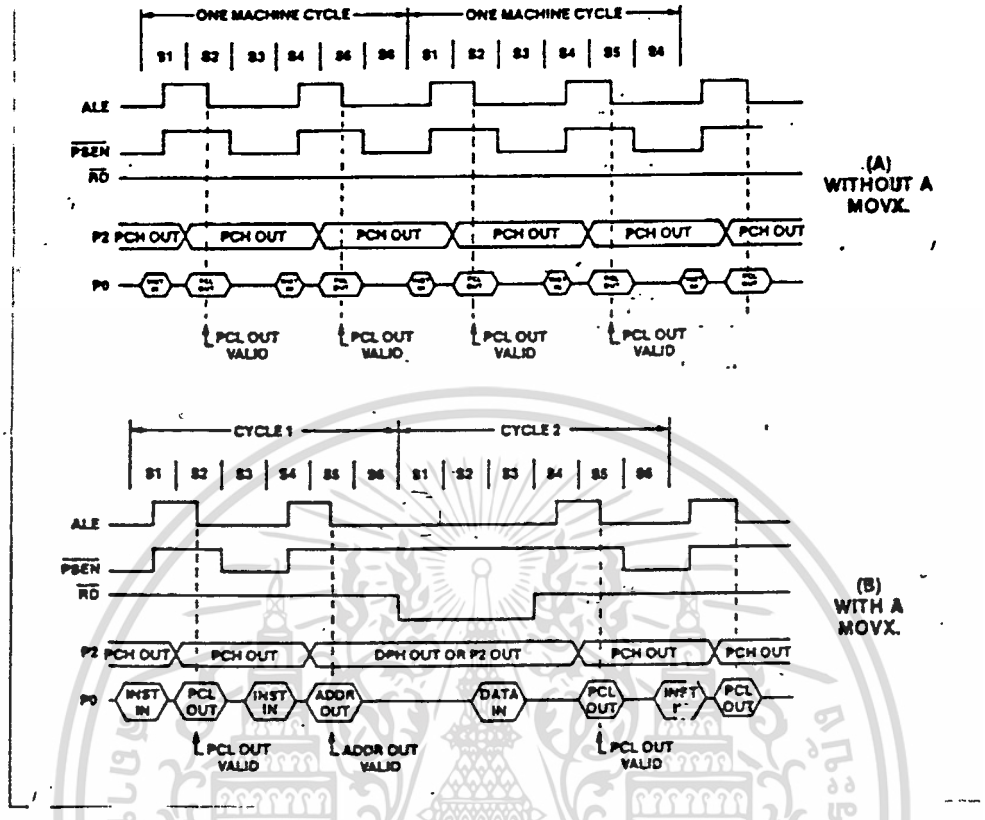
สัญญาณ PSEN

ใช้เป็นการควบคุมเฟทช์การอ่านโปรแกรมจากภายนอก PSEN จะไม่แอคตีฟถ้ามีการเฟทช์โปรแกรมภายใน เมื่อ CPU เข้าถึงการใช้โปรแกรมภายนอก PSEN จะแอคตีฟ 2 ครั้งในแต่ละวัฏจักรการเฟทช์ ยกเว้นคำสั่ง MOVX ช่วงเวลาของการที่ PSEN เกิดแอคทีฟจะไม่เหมือนกับช่วง RD แอคทีฟ ช่วงวัฏจักรการอ่านที่สมบูรณ์จะรวมเอาช่วงที่ ALE แอคทีฟและแอคทีฟซ้ำลูกที่สองกับสัญญาณการควบคุม RD ที่เกิดพัลช์ต่ำประกอบเข้าด้วยกัน ซึ่งจะใช้เวลา 12 สถานะคาบเวลา ส่วนช่วงเวลาของ PSEN ที่สมบูรณ์จะรวมเอาช่วงที่ ALE แอคทีฟและแอคทีฟซ้ำลูกที่สองกับสัญญาณควบคุม PSEN ประกอบเข้าด้วยกัน ซึ่งจะใช้เวลา 6 สถานะคาบเวลา ลักษณะการทำงานตามลำดับของวัฏจักรการอ่านทั้งสองแบบแสดงในรูปที่ 8

สัญญาณ ALE

ฟังก์ชั่นหลักของ ALE คือ การใช้งานในการให้จังหวะที่แน่นอนในการแลทช์เอาไบต์ต่ำของแอดเดรสจาก P0 ไปเก็บไว้ภายนอกเพื่อใช้ในการถอดรหัสแอดเดรสโปรแกรมภายนอกโดยจะให้ ALE ทำงานแอคทีฟสองครั้งในทุก ๆ วัฏจักรแมชีน สัญญาณนี้จะเกิดขึ้นตลอดแม้ว่าจะไม่ได้เฟ็ทช์จากภายนอก มีเพียงช่วงเวลาเดียวเท่านั้นที่ ALE ไม่เกิดพัลช์คือระหว่างการเข้าถึงหน่วยความจำภายนอก ตามรูปที่ 8(B) จะเห็นว่าพัลช์แรกของ ALE ในวัฏจักรที่สองของคำสั่ง MOVX ขาดหายไปหรือมีเพียงพัลช์เดียวในหนึ่งคำสั่ง ลักษณะพัลช์ที่เกิดขึ้นคงที่ในอัตรา 1/6 ของสัญญาณความถี่ออสซิเลเตอร์ และสามารถนำมาใช้เป็นสัญญาณนาฬิกาภายนอก หรือกำหนดเวลาได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8 จังหวะการเข้าถึงหน่วยความจำภายนอก

ตัวจับเวลา/ตัวนับ (TIMER/COUNTER)

MCS-51 มี 16 บิต ตัวจับเวลา/ตัวนับ 2 ตัว คือ TIMER/COUNTER 0 และ TIMER/COUNTER ส่วน 8032/8052 มีเพิ่มอีก 1 ชุด คือ TIMER/COUNTER 2 ขณะที่แต่ละ ตัวจับเวลา/ตัวนับ (TIMER/COUNTER) สามารถที่จะกำหนดให้ทำงานได้เป็นตัวจับเวลาหรือตัวนับ

ตัวจับเวลา/ตัวนับ0 และตัวจับเวลา/ตัวนับ 1

แต่ละตัวจะถูกกำหนดให้ทำงานเป็นตัวจับเวลาหรือเป็นตัวนับ ได้ด้วยการเซตหรือเคลียร์บิตที่ตัวควบคุมในเรจิสเตอร์ TMOD ในกลุ่ม SFR

ในฟังก์ชั่นตัวจับเวลา ตัวเรจิสเตอร์จะเพิ่มค่าทุก ๆ วัฏจักรแมชชีน ดังนั้น ตัวเลขในรีจิสเตอร์จะเป็นจำนวนของวัฏจักรแมชชีน เนื่องจาก แต่ละวัฏจักรแมชชีนประกอบด้วย 12 คาบของออสซิเลเตอร์ อัตราการนับแต่ละครั้ง จะกินเวลาเป็น 1/12 ของความถี่ออสซิเลเตอร์

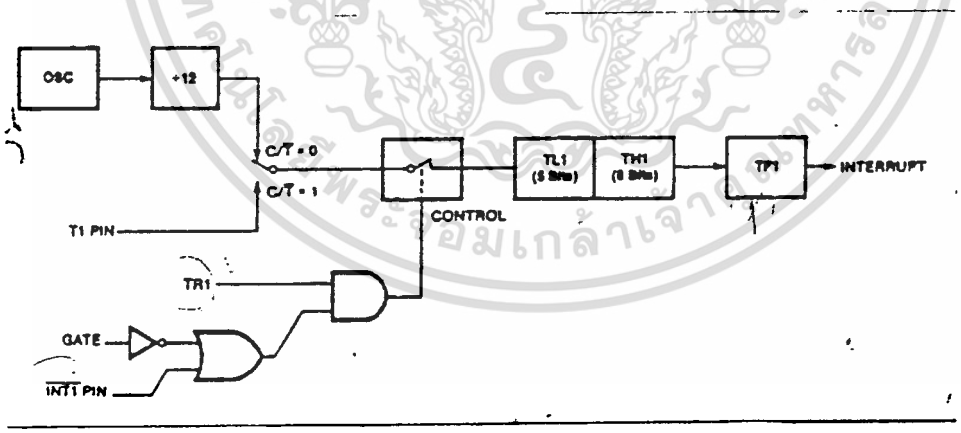
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในฟังก์ชั่นตัวนับเรจิสเตอร์จะเพิ่มค่าทุกครั้งที่มีการเปลี่ยนแปลงสถานะจาก '1' เป็น '0' ที่เข้ามาที่ขา T0 หรือ T1 ในฟังก์ชั่นนี้สัญญาณภายนอกที่เข้ามาจะถูกรับแซมปลิ้งระหว่างช่วง S5P2 ของทุกวัฏจักรแมชชีน โดยถ้าฉวมปลิ้งสัญญาณเข้าเป็นระดับสูงในวัฏจักรหนึ่ง ดังนั้น ถ้าในวัฏจักรหนึ่งต่อมาของสัญญาณเข้าเป็นระดับต่ำ เรจิสเตอร์จะนับเพิ่มขึ้น 1 ค่า โดยที่ค่าใหม่ของตัวนับ จะปรากฏที่เรจิสเตอร์ช่วง S3P1 ของวัฏจักร ซึ่งค่าหนึ่งที่รับเข้าไป จะใช้ช่วง 2 วัฏจักรแมชชีน (เท่ากับ 24 คาบ) ในการรับค่าช่วงการเปลี่ยน 1 เป็น 0 ดังนั้นค่าสูงสุดในการนับจะมีอัตรา 1/24 ของความถี่ออสซิเลเตอร์ และสัญญาณอินพุทที่นับนั้นจะไม่มีช่วงระยะห่างที่แน่นอนของ DUTY CYCLE แต่จะถูกนับ เมื่อระดับแรงดันที่ถูกแซมปลิ้งในแต่ละครั้งจะต้องมีช่วงคงที่อย่างน้อย 1 วัฏจักรแมชชีนก่อนที่จะเปลี่ยนค่าระดับแรงดันใหม่

ในการเลือกทำงานระหว่างตัวนับและตัวจับเวลา จะเลือกได้ 4 โหมด คือ โหมด 0 1 และ 2 เลือกได้ทั้งสองตัวของ TIMER/COUNTER ส่วนโหมด 3 จะทำงานแตกต่างออกไป ในที่นี้จะใช้การทำงานในโหมดที่ 2

โหมด 0

การใช้ตัวจับเวลา/ตัวนับ 0 หรือ 1 อยู่ในโหมด 0 จะทำงานคล้ายกับของ MCS-48 โดยตัวจับเวลาของ MCS-48 มีขนาด 8 บิต มีตัว Prescaler เป็นตัวหาร 12



รูปที่ 9 แสดงการทำงานในโหมด 0 ของตัวจับเวลา/ตัวหารนับ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9 แสดงการทำงานในโหมด 0 ของตัวจับเวลา/ตัวหารนับ 1 ขนาด 13 บิต

ในโหมดนี้ เรจิสเตอร์ตัวจับเวลาถูกกำหนดให้มี 13 บิต ด้วยการนับขึ้น เมื่อเป็น '1'หมดทุกบิตจะกลับมา '0' ทุกบิตใหม่ เมื่อกลับเป็น '0' ทุกบิตจะเป็นการเกิด Overflow ไปทดให้แฟลกอินเตอร์รัพท์ TF1 ปรับเป็น '1' การควบคุมให้เริ่มนับตัวอินพุทจะควบคุมด้วยการอีนาเบิ้ล TR1 = 1 , GATE = 0 และขา INT1 = 1 การปรับ GATE=1 เป็นการลงตัวนับให้ถูกควบคุมด้วยสัญญาณจากภายนอกเข้าขา INT1 TR1 จะเป็นบิตควบคุมในเรจิสเตอร์ TMOD ของ SFR

เรจิสเตอร์ตัวนับจะมี 13 บิต ประกอบด้วย TH1 8 บิต และ TL1 อีก 5 บิตอันดับต่ำ ส่วนอีก 3 บิต ที่เหลือในอันดับสูงของ TL1 จะไม่ใช้ การปรับแฟลก TR1 ให้ทำงานจะไม่เคลียร์ค่าในเรจิสเตอร์

การทำงานในโหมด 0 ในตัวจับเวลา/ตัวนับ 0 จะทำงานเหมือนกับตัวจับเวลา/ตัวนับ 1 โดยใช้ TR0 และ INT0 รวมกันควบคุมแทนสัญญาณต่าง ๆ ในรูปที่ 9 มีความแตกต่างในการควบคุมคือ บิตของGATE ทั้งสอง ตัวหนึ่งจะแทนตัวจับเวลา/ตัวนับ 1 (TMOD.7)และอีกตัวจะแทนตัวจับเวลา/ตัวนับ 0 (TMOD.3)

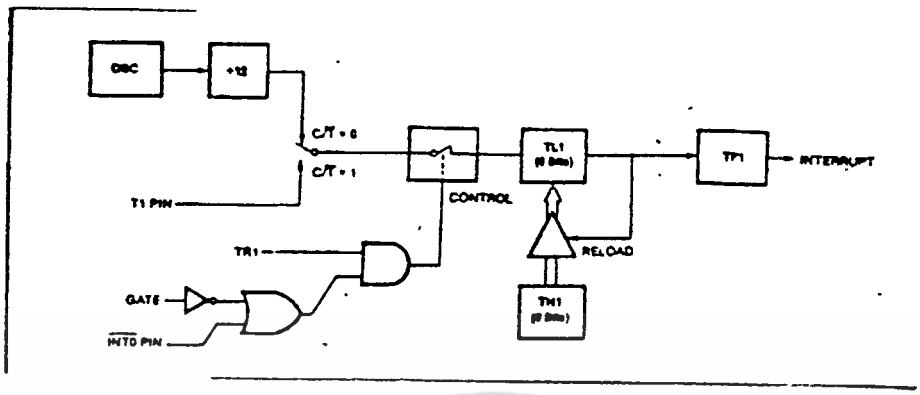
โหมด 1

โหมด 1 ทำงานเหมือนกับโหมด 0 ต่างกันแต่เฉพาะการใช้เรจิสเตอร์ตัวจับเวลา/ตัวนับ จะทำงานนับด้วยขนาด 16 บิต โดยไม่มี Prescaler คือความถี่ที่ออสซิเลเตอร์เป็นความถี่ที่เข้ามาถูกหารด้วยค่า 16 บิต ในเรจิสเตอร์ตัวนับ

โหมด 2

โหมด 2 มีการทำงานโดยการกำหนดให้ตัวนับ 8 บิต ของ TL1 และจะโหลดใหม่โดยอัตโนมัติทุกครั้ง เมื่อมีการ OVERFLOW จาก TL1 ดังรูปที่ 3 ไม่เพียงแต่ TF1 จะปรับเป็น '1' แต่ TL1 จะถูกโหลดโดยอัตโนมัติจากค่าที่ตั้งไว้ใน TH1 สามารถจะตั้ง TH0 และ TF0 จะเป็นตัวร่วมการทำงานในโหมดนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 10 ตัวจับเวลา/ตัวนับ1 ทำงานในโหมด 2 แบบโหลดใหม่ 8 บิต

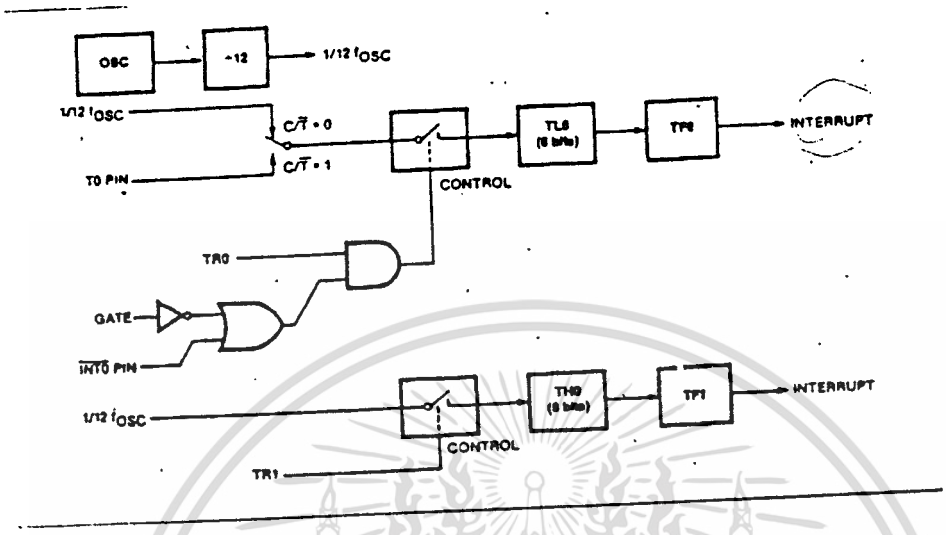
โหมด 3

ใช้ตัวจับเวลา/ตัวนับ 1 ในโหมด 3 มีการทำงานเป็นตัวนับ มีผลเช่นเดียวกับการตั้ง TR1 = 0 และใช้ตัวจับเวลา/ตัวนับ 0 ในโหมด 3 จัดการให้ TL0 และ TH0 เป็นตัวนับ 2 ตัวแรกที่แยกออกจากกัน วงจรตรรกควบคุมสำหรับโหมด 3 ที่ใช้ตัวจับเวลา/ตัวนับ 0 แสดงในรูปที่ 7.18 TL0 ใช้ตัวจับเวลา/ตัวนับ 0 เป็นบิตควบคุมของ C/T,GATE,TR0,INT0 และTF0 ตัว TH0 ถูกล็อกให้ทำงานในฟังก์ชั่น รูปที่ 11ใช้ตัวจับเวลา/ตัวนับ 1 ในโหมด 3 เป็นกลุ่มตัวนับขนาด 8 บิตสองตัว

ตัวจับเวลา (เป็นตัวนับวัฏจักรแมชีนได้) และจะใช้บิตที่ TR1 และ TF1 ของตัวจับเวลา 1 เป็นตัวควบคุม ดังนั้นจึงใช้ TH0 เป็นตัวจับเวลาเป็นการควบคุมอินเตอร์รัพต์ด้วยตัวจับเวลา 1

โหมด 3 จะใช้งานในความต้องการตัวจับเวลา/ตัวนับ ขนาด 8 บิต เพิ่มขึ้นด้วยการใช้ตัวจับเวลา 0 ในโหมด 3 ดังนั้น 8051 สามารถที่จะทำงานใช้ตัวจับเวลา/ตัวนับ ได้เป็น 3 ชุด ขณะที่ 8052 ก็จะใช้งานได้ 4 ชุด เมื่อใช้ตัวจับเวลา 0 อยู่ในโหมด 3 ตัวจับเวลา 1 สามารถที่จะ เปิด-ปิด ให้เข้าสู่หรือออกจากการทำงานของโหมด 3 หรือสามารถที่จะยังคงใช้ตัว GENERATOR ที่สร้างอัตราบ๊อดของการส่งข้อมูลอนุกรม หรือการใช้งานใด ๆ ที่ไม่ต้องการการอินเตอร์รัพต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 11 ใช้ตัวจับเวลา/ตัวนับ 1 ในโหมด 3 เป็นกลุ่มตัวนับขนาด 8 บิตสองตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

GATE	C/T	M1	M0	GATE	C/T	M1	M0

¦------Timer R1--------¦-------Timer R0-------¦

GATE : ควบคุมเกท เมื่อเซ็ทบิต '1' จะเป็นอินาเบิ้ล ตัวจับเวลา/ตัวนับเท่านั้น ขณะที่ขา INTx มีสถานะสูงและะขาควบคุม TRx ใน TCON จะถูกเซ็ทเป็น '1' เมื่อ ตัวนับภายในถูกเคลียร์ให้อีนาเบิ้ลเมื่อไรก็ตามที่บิตควบคุม TRx ถูกเซ็ทเป็น '1'

C/T : เลือกการทำงานแบบตัวจับเวลาหรือตัวนับ ถ้าเป็น '0' จะเลือกทำงานเป็นตัวจับเวลา (โดยใช้สัญญาณนาฬิกาภายในเป็นสัญญาณเข้าอ้างอิงถึง) ถ้าเป็น '1' จะเป็นการทำงานแบบตัวนับ และรับสัญญาณเข้าที่ขา Tx

M1	M0	การทำงาน
0	0	ทำงานแบบตัวจับเวลาของ MCS-48 ใช้ TLx เป็นตัวป้อนบิตอีก 5 บิต
0	1	การใช้ตัวจับเวลา/ตัวนับ ขนาด 16 บิต จะใช้ THx และ TLx เป็นตัวนับไม่มี Prescaler
1	0	การโหลดขนาด 8 บิต โดยอัตโนมัติที่ตัวนับและตัวจับเวลา โดยใช้ THx เก็บค่าที่ตั้งไว้และจะถ่ายเข้าไปที่ TLx ใหม่ทุกครั้งที่เกิด Overflow คือ TLx ถูกนับเป็น '0' หมด
1	1	ตัวจับเวลา 0 ทำงาน โดยให้ TL0 และ TH0 เป็นตัวนับแยกกัน

รูปที่ 12 TMOD : Timer/Counter Mode Control Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 TCON.7 ตัวจับเวลา 1 แฟล็กเป็น '1' เมื่อเกิด Overflow ถูกเซ็ทเป็นหนึ่งด้านฮาร์ดแวร์ทางสัญญาณ เมื่อตัวจับเวลา/ตัวนับ Overflow และจะเคลียร์ตัวเองเมื่ออินเตอร์รัพท์ไปแล้ว

TR1 TCON.6 ตัวจับเวลา 1 เป็นตัวควบคุมบิตให้เริ่มทำงาน จะเซ็ทหรือเคลียร์ด้วยซอฟต์แวร์ที่จะมาทำให้ตัวจับเวลา/ตัวนับ 1 เริ่มหรือหยุดการทำงาน

TF0 TCON.5 ตัวจับเวลา 0 แฟล็กเป็น '1' เมื่อเกิด Overflow ถูกเซ็ทเป็นหนึ่งด้านฮาร์ดแวร์ทางสัญญาณ เมื่อตัวจับเวลา/ตัวนับ Overflow และจะเคลียร์ตัวเองเมื่ออินเตอร์รัพท์ไปแล้ว

TR0 TCON.4 ตัวจับเวลา 0 เป็นตัวควบคุมบิตให้เริ่มทำงาน จะเซ็ทหรือเคลียร์ด้วยซอฟต์แวร์ที่จะมาทำให้ตัวจับเวลา/ตัวนับ 0 เริ่มหรือหยุดการทำงาน

IE1 TCON.3 อินเตอร์รัพต์ 1 เป็นแฟล็กขอบสัญญาณ เซ็ทด้วยฮาร์ดแวร์เมื่อสัญญาณขอบการอินเตอร์รัพต์ปรากฏเข้าที่ขา INT1 และเคลียร์เมื่อการทำงานอินเตอร์รัพต์สิ้นสุด

IT1 TCON.2 อินเตอร์รัพต์ 1 รูปแบบการควบคุมบิต จะเซ็ทหรือเคลียร์ได้ด้วยซอฟต์แวร์ที่จะเป็นตัวกำหนดให้การกระตุ้นอินเตอร์รัพต์จากภายนอกที่ขอบขาลงหรือระดับแรงดันต่ำโดยถ้า IT1 = 1 จะควบคุมอินเตอร์รัพต์ด้วยระดับแรงดันต่ำ

IE0 TCON.1 อินเตอร์รัพต์ 0 เป็นแฟล็กขอบสัญญาณ เซ็ทด้วยฮาร์ดแวร์เมื่อสัญญาณขอบการอินเตอร์รัพต์ปรากฏเข้าที่ขา INT0 และเคลียร์เมื่อการทำงานอินเตอร์รัพต์สิ้นสุด

IT0 TCON.0 อินเตอร์รัพต์ 0 รูปแบบการควบคุมบิต จะเซ็ทหรือเคลียร์ได้ด้วยซอฟต์แวร์ที่จะเป็นตัวกำหนดให้การกระตุ้นอินเตอร์รัพต์จากภายนอกที่ขอบขาลงหรือระดับแรงดันต่ำ

รูปที่ 13 TCON : Timer/Counter Control Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การอินเตอร์รัพท์

โดยทั่วไปความสามารถในการควบคุมการอินเตอร์รัพท์ เป็นการทำงานชนิดหนึ่งของ CPU ที่จะต้องศึกษาถึงความสามารถและเทคนิคการทำงาน การอินเตอร์รัพท์ของชิงเกิ้ลชิปมีความสัมพันธ์กันอย่างใกล้ชิดระหว่างอุปกรณ์ต่อพ่วงกับระบบ การทำงานของอุปกรณ์ต่อพ่วงเหล่านี้ มีระบบฮาร์ดแวร์ ที่ช่วยให้การส่งสัญญาณ REAL TIME กับ MONITOR ได้อย่างต่อเนื่อง โดยปราศจากการรบกวนต่อสัญญาณการทำงานของ CPU ตัวอย่างเช่น ขณะที่มีการรับสัญญาณอนุกรมจาก CRT ตัวหนึ่งก็จะมีการส่งสัญญาณไปยังอุปกรณ์ตัวอื่นและตัวจับเวลา/ตัวนับก็จะนับพัลซ์การเปลี่ยนแปลงที่เข้ามาอย่างรวดเร็วไปพร้อมกันได้ด้วย ในขณะที่ตัวจับเวลา/ตัวนับอีกตัวก็กำลังจัดความกว้างของพัลซ์ที่เข้ามา ซีพียูตัวนี้ จะรู้ได้อย่างไรว่า เมื่อไรถึงจะมีการรับและส่งสัญญาณอนุกรม CRT หรือให้ตัวจับเวลา/ตัวนับ มีการนับจำนวนและวัดความกว้างของพัลซ์ว่าจะสิ้นสุดลงเมื่อใด

ตัวโปรแกรม MCS-51 สามารถที่จะเลือกการโปรแกรมได้ 3 วิธีด้วยกันคือ พิจารณาการโปรแกรมตัวเรจิสเตอร์ TCON และ SCON ที่ประกอบด้วยสถานะบิตที่ถูกเซ็ททางฮาร์ดแวร์ เมื่อตัวจับเวลาตัวหนึ่งเกิด Overflow หรือเมื่อการรับส่งข้อมูลที่พอร์ตอนุกรมสิ้นสุดลง

เทคนิคการโปรแกรมวิธีแรกก็โดยการอ่านสถานะของเรจิสเตอร์ควบคุมเข้าไปยังแอคคิวมิวเลเตอร์ แล้วทดสอบสถานะบิตตามลักษณะการทำงานนั้น ๆ แล้วทำการกระโดดไปยังโปรมแกรมย่อยตามผลที่เกิดขึ้นชนิดนั้น ๆ ลักษณะการทดสอบร่วมกันครั้งละหลายลักษณะงานเช่นนี้ เปรียบเสมือนตัวโปรแกรมใช้ระบบไมโครโปรเซสเซอร์หลายตัวควบคุมชิปอุปกรณ์ต่อพ่วงต่าง ๆ ซึ่งผู้โปรแกรมจะต้องทำความเข้าใจอย่างลึกซึ้งถึงระบบ และจังหวะที่เกิดขึ้นในแต่ละงาน และการทดสอบแต่ละครั้งจะใช้คำสั่งไม่น้อยกว่า 3 คำสั่ง

วิธีที่สอง MCS-51 สามารถที่จะทำงานด้วยการกระโดดไปตามสถานะของการควบคุมหรือสถานะบิตของเรจิสเตอร์ควบคุมงานหรือการรับสัญญาณที่เข้ามาตามขาอินพุทแต่ละพุตด้วยการใช้คำสั่งเพียงคำสั่งเดียว ดังนั้นลักษณะงานสี่อย่างก็สามารถที่จะ
ใช้คำสั่งตรวจสอบได้ภายในสี่คำสั่ง ซึ่งจะใช้เวลาประมาณภายใน 8 ไมโครวินาที

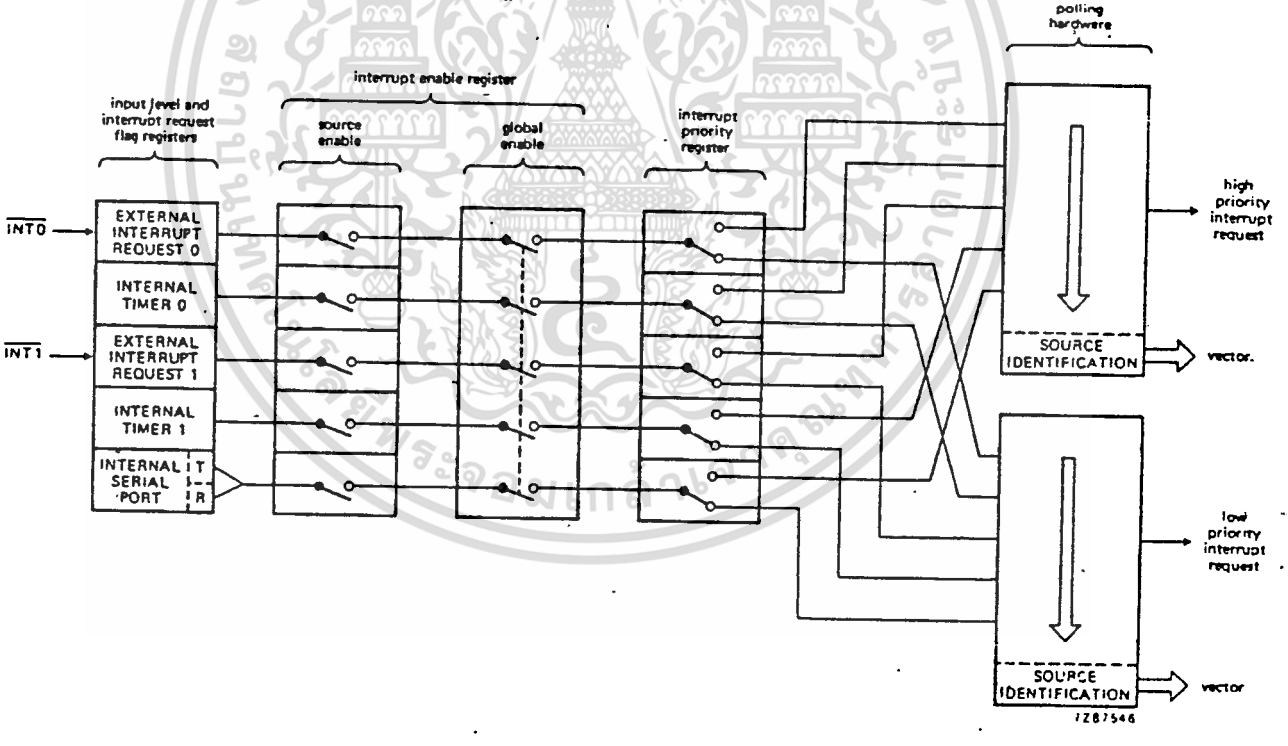
แต่วิธีทั้งสองที่กล่าวมาแล้ว จะต้องใช้ตัวซีพียูมาทำการตรวจสอบบิตสถานะต่าง ๆ อยู่ตลอดเวลา ลองเปรียบตัวซีพียู เหมือนกับตัวผู้จัดการของบริษัทซึ่งจะบริหารงานในบริษัทให้ก้าวหน้าได้อย่างดีนั้น จะต้องมีเวลาทำงานให้กับหน้าที่หลักของตัวเอง เช่นในการวางแผนต่าง ๆ ให้กับบริษัทได้อย่างต่อเนื่อง และใช้เวลาเพียงบางส่วนสำหรับพนักงานที่จะเข้ามาขัดจังหวะเพื่อขอปรึกษาแก้ไขปัญหาเพียงบางเวลาที่จำเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เท่านั้น เช่นเดียวกันแทนที่จะใช้ซีพียูทำงานในลักษณะที่ออกไปตรวจสอบสถานะการทำงานของอุปกรณ์ต่อพ่วงรอบข้างต่าง ๆ ที่ต้องการจะใช้บริการก็จะใช้อุปกรณ์ต่อพ่วงเป็นฝ่ายร้องขอการบริการเข้ามาที่ซีพียูแทนซึ่งเมื่อซีพียูถูกร้องขอเข้ามาก็จะปล่อยงานเดิม และเข้าสู่การบริการที่อุปกรณ์ต่อพ่วงชนิดนั้น ๆ ได้ร้องขอเข้ามาช่วงระยะเวลาหนึ่งแล้วจึงกลับเข้าทำงานหลักต่อไป เมื่อสิ้นสุดงานบริการนั้นแล้ว ทำให้รู้สึกว่าตัวไมโครโปรเซสเซอร์ทำงานพร้อมกันได้หลายงานในเวลาเดียวกัน

การใช้วิธีที่สามจะเป็นวิธีที่ดีที่สุด ในการใช้งานลักษณะนี้ด้วยการอินเตอร์รัพต์ทางฮาร์ดแวร์ ซึ่งในที่นี้เราก็จะวิธีนี้ด้วยในที่นี้เราจะใช้อุปกรณ์ต่อพ่วงเป็นฝ่ายร้องขอการบริการเข้ามาที่ CPU ซึ่งเมื่อ CPU ถูกร้องขอเข้ามาก็จะปล่อยงานเดิม และเข้าสู่การบริการที่อุปกรณ์ต่อพ่วงนั้น ๆ ได้ร้องขอเข้ามาช่วงระยะเวลาหนึ่งแล้วจึงกลับเข้าไปทำงานหลักต่อไป เมื่อสิ้นสุดการบริการนั้นแล้ว ทำให้รู้สึกว่าตัวไมโครโปรเซสเซอร์ทำงานได้พร้อมกันได้หลายงานในเวลาเดียวกัน

แผนภูมิทางฮาร์ดแวร์ของการอินเตอร์รัพท์ชนิดต่าง ๆ เป็นดังรูปที่ 14



รูปที่ 14 แผนภูมิทางฮาร์แวรของการอินเตอร์รัพท์ชนิดต่าง ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การอินเตอร์รัพท์ตัวจับเวลา0 และตัวจับเวลา1 จะทำงานได้ด้วยค่าของแฟลก TF0 และ TF1 ที่เซตเป็น 1 เมื่อค่าในเรจิสเตอร์ของตัวจับเวลา/ตัวนับถูกเพิ่มจาก 1 หมดทุกบิตกลายเป็น 0 หมดทุกบิท (การตั้งอินเตอร์รัพท์ของตัวจับเวลา 0 จะตั้งให้ทำงานอินเตอร์รัพท์ ไม่ได้ ถ้าให้ทำงานในโหมด 3 หลังจากการเกิดอินเตอร์รัพท์ ตัวจับเวลาแฟลกดังกล่าวจะถูกเคลียร์ให้เป็น 0 เมื่อฮาร์ดแวร์บนชิปเข้าทำงานในโปรแกรมบริการของการอินเตอร์รัพท์ตัวจับเวลาแล้ว

EA	X	ET2	ES	ET1	EX1	ET0	EX0

สัญญลักษณ์	ตำแหน่งบิต	ฟังก์ชั่น
EA	IE.7	จะดิสเอเบิ้ลการอินเตอร์รัพท์ทั้งหมด ถ้า EA = 0 จะไม่มีการอินเตอร์รัพท์ในการตอบรับ ถ้า EA = 1 สามารถที่จะอินเตอร์รัพท์ได้โดยแต่ละแหล่งอินเตอร์รัพท์จะมีอิสระในการเซ็ทหรือเคลียร์ให้อีนาเบิ้ลแต่ละบิตก่อนได้
-	IE.6	สำรอง
ET2	IE.5	จะอีนาเบิ้ลหรือดิสเอเบิ้ลอินเตอร์รัพท์ Overflow ของตัวจับเวลา 2 ถ้า ET2 = 0 การอินเตอร์รัพท์ตัวจับเวลา 2 จะดิสเอเบิ้ล
ES	IE.4	จะอีนาเบิ้ลหรือดิสเอเบิ้ลอินเตอร์รัพท์พอร์ตอนุกรมถ้า ES = 0 การอินเตอร์รัพท์พอร์ตอนุกรมจะดิสเอเบิ้ล
ET1	IE.3	จะอีนาเบิ้ลหรือดิสเอเบิ้ลอินเตอร์รัพท์ Overflow ของตัวจับเวลา 1 ถ้า ET1 = 0 การอินเตอร์รัพท์ตัวจับเวลา 1 จะดิสเอเบิ้ล
EX1	IE.2	จะอีนาเบิ้ลหรือดิสเอเบิ้ลอินเตอร์รัพท์จากภายนอก 1 ถ้า EX1 = 0 การอินเตอร์รัพท์จากภายนอก 1 จะดิสเอเบิ้ล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ET0	IE.1	จะอีนาเบิ้ลหรือดิสเอเบิ้ลอินเตอร์รัพต์ Overflow ของตัวจับเวลา 0 ถ้า ET0 = 0 การอินเตอร์รัพต์ตัวจับเวลา 0 จะดิสเอเบิ้ล
EX0	IE.0	จะอีนาเบิ้ลหรือดิสเอเบิ้ลอินเตอร์รัพต์จากภายนอก 0 ถ้า EX0 = 0 การอินเตอร์รัพต์จากภายนอก 0 (INT0) จะดิสเอเบิ้ล

รูปที่ 15 เรจิสเตอร์การอินเตอร์รัพต์อีนาเบิ้ล : IE

การอินเตอร์รัพท์ของการส่งข้อมูลอนุกรมจะเกิดขึ้นเมื่อ ORกันทางตรรกด้วยสัญญาณจากแฟลกของ RI กับ TI แฟลกทั้งสองจะไม่ถูกเคลียร์ด้วยฮาร์ดแวร์ เมื่อฮาร์ดแวร์บนชิปเข้าทำงานตามโปรแกรมบริการของการอินเตอร์รัพต์การส่งข้อมูลอนุกรม บิตในแฟล็กของ RI กับ TI จะถูกเคลียร์ด้วยซอฟต์แวร์เท่านั้น

เฉพาะตัว 8052 จะมีการอินเตอร์รัพต์ด้วยตัวจับเวลา 2 ด้วยการ OR กันทางตรรกด้วยสัญญาณจากแฟลกของ TF2 กับ EXF2 เช่นกันแฟลกทั้งสองจะไม่ถูกเคลียร์ด้วยฮาร์ดแวร์ เมื่อฮาร์ดแวร์บนชิปเข้าทำงานตามโปรแกรมบริการของตัวจับเวลา 2 โดยที่โปรแกรมบริการนี้จะต้องแยกให้ถูกว่าเป็นการบริการของ TF2 หรือ EXF2 ส่วนบิตในแฟลกของ TF2 และ EXF2 จะถูกเคลียร์ด้วยซอฟต์แวร์

บิตทุกบิตที่จะทำให้การอินเตอร์รัพต์ทำงานได้สามารถที่จะเซ็ทหรือเคลียร์ได้ด้วยซอฟต์แวร์ ซึ่งจะให้ผลเช่นเดียวกับการเซ็ทหรือเคลียร์ด้วยฮาร์ดแวร์ การอินเตอร์รัพต์สามารถที่จะถูกกำหนดให้ทำงานหรือยกเลิกได้ในขณะใดขณะหนึ่งของการทำงานด้วยการควบคุมกำหนดด้วยซอฟต์แวร์

แหล่งกำเนิดการอินเตอร์รัพต์แต่ละชนิดสามารถที่จะอีนาเบิ้ลหรือดิสเอเบิ้ลด้วยการเซ็ทหรือเคลียร์ค่าบิตภายในตัวเรจิสเตอร์ IE บิต EA ภายในเรจิสเตอร์ IE จะเป็นตัวควบคุมการอินเตอร์รัพต์ทุกชนิดให้เริ่มทำงานดังรูปที่ 15 ซึ่งแสดงรายละเอียดของเรจิสเตอร์ IE ในการเซ็ทค่าบิตต่าง ๆ เพื่อควบคุมการอินเตอร์รัพต์แต่ละแบบ ส่วนบิต IE.6 เป็นบิตสำรองผู้ใช้ไม่สามารถที่จะเซ็ทหรือเคลียร์บิตนี้ได้เช่นเดียวกันในตัว 8051 บิต IE.5 ก็จะเป็นบิตสำรองที่จะเซ็ทไม่ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การตอบสนองการอินเตอร์รัพต์โปรโตคอล (Protocol)

ตัวโปรเซสเซอร์จะตอบรับการร้องขอด้วยการเซ็ทแอคทีฟระดับความสำคัญของการอินเทอร์รพท์ FLIP-FLOP แล้วจึงจะทำงานทางฮาร์ดแวร์ในการเรียกโปรแกรมย่อยที่บริการการอินเตอร์รัพต์นั้น ๆ มันจะเคลียร์แฟลกการร้องขอการอินเตอร์รัพต์ (ยกเว้นมันจะไม่เคลียร์ INT0 หรือ INT1 เพราะมันไม่ได้ควบคุมแหล่งสัญญาณเหล่านี้ และมันจะไม่เคลียร์ TI , RI , TF2 หรือ EXF2) การเรียกโปรแกรมย่อยทางฮาร์ดแวร์จะทำการพุชค่าข้อมูลของตัวนับโปรแกรมเข้าที่บริเวณสแต็ค แต่จะไม่เก็บค่า PSW เข้าไปด้วย (และ PC จะถูกบรรจุใหม่ด้วยค่าแอดเดรสที่ขึ้นกับแหล่งชนิดการร้องขอของการอินเตอร์รัพต์) ซึ่งจะมีดังนี้

แหล่งการอินเตอร์รัพต์	ตำแหน่งเวคเตอร์แอดเดรส
External Interrupt 0 (IE0)	0003H
Timer 0 Overflow (TF0)	000BH
External Interrupt 1 (IE1)	0013H
Timer 1 Overflow (TF1)	001BH
Serial Port (RI+TI)	0023H
Timer 2 Overflow/Negative Transition on T2EX (ของเบอร์ 8032/8052 เท่านั้น)	002BH

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานจะเริ่มทำงานจาก เวคเตอร์แอดเดรสที่ให้มาจนกระทั่งถึงคำสั่ง RETI ซึ่งอยู่ในโปรแกรมย่อยการบริการอินเตอร์รัพต์ต่าง ๆ ดังกล่าวข้างบน คำสั่ง RETI จะเคลียร์แอคทีพระดับความสำคัญของการอินเตอร์รัพต์ของ FLIP-FLOP ที่ถูกเซ็ทเมื่อได้รับการร้องขอการอินเตอร์รัพต์ในตอนแรก ดังนั้นมันจะ POP เอายอดของสแตค 2 ไบต์มาบรรจุใหม่ใน PC เพื่อจะกลับเข้าทำงานโปรแกรมเดิมก่อนที่จะถูกอินเตอร์รัพต์ต่อไป

การโปรแกรมอินเตอร์รัพต์

ในการเลือกใช้อินเตอร์รัพต์ของ MCS-51 แบบใด ๆ จะมีวิธีเริ่มต้นโปรแกรมในการเลือกใช้อินเตอร์รัพต์อยู่ 3 ขั้นตอนด้วยกันคือ

1. เซ็ทค่าในบิต EA ของเรจิสเตอร์ IE เป็น '1'
2. เซ็ทค่าอินเตอร์รัพต์ในการเลือกใช้อินเตอร์รัพต์แต่ละแบบให้อีนาเบิ้ลเป็นหนึ่งในแต่ละบิตของเรจิสเตอร์ IE
3. เริ่มเข้าสู่งานบริการการอินเตอร์รัพต์ตามค่าแอดเดรส VECTOR ต่าง ๆ ตามตารางนี้

อินเตอร์รัพต์	VECTOR
IE_0	0003H
TF_0	000BH
IE_1	0013H
TF_1	001BH
R_I และ T_I	0023H
TF2 + EXF2	002BH

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

การใช้งาน 8255 PPI

ลักษณะทั่วไปของ 8255 PPI

8255 PPI (Programmable Peripheral Interface) เป็น LSI ขนาด 40 ขาทำหน้าที่อินเตอร์เฟสระหว่างไมโครโปรเซสเซอร์กับอุปกรณ์ภายนอก 8255 ถูกออกแบบมาใช้กับไมโครโปรเซสเซอร์เบอร์ 8080

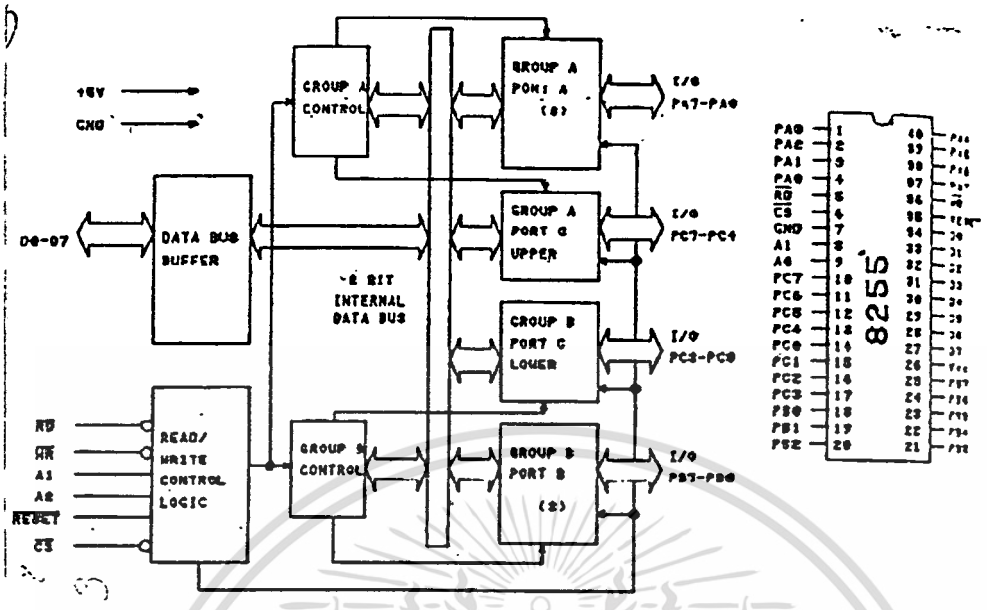
บล็อกไดอะแกรมของ 8255 แสดงได้ดังรูป 8.1 ซึ่งมีส่วนที่ติดต่อกับอุปกรณ์ภายนอก 4 กลุ่มคือ PA0-PA7,PB0-PB7,PC0-PC3 และ PC4-PC7 กลุ่มของสัญญาณควบคุมมี 2 กลุ่มคือ GROUP A CONTROL และ GROUP B CONTROL ซึ่งเป็นส่วนควบคุมการทำงานของทั้ง 3 พอร์ต DATA BUS BUFFER และ READ/WRITE CONTROL LOGIC ใช้สำหรับติดต่อกับไมโครโปรเซสเซอร์ทางบัสข้อมูลและสัญญาณควบคุมการอ่านและเขียนข้อมูลกับเรจิสเตอร์ที่อยู่ภายใน 8255

สัญญาณต่าง ๆ ของ 8255

หน้าที่ของสัญญาณต่าง ๆ ของ 8255 เป็นดังนี้

D0-D7	เป็นขาข้อมูลที่ใช้ต่อกับไมโครโปรเซสเซอร์
CS	(Chip Select Input) เมื่อขานี้มีค่าลอจิก 0 CPU สามารถติดต่อกับ 8255 ได้
RD	(Read Input) เมื่อขานี้มีค่าลอจิก 0 พร้อมกับ CS 8255 จะส่งข้อมูลออกมาทางบัสข้อมูล
WR	(Write Input) เมื่อขานี้มีค่าลอจิก 0 พร้อมกับ CS ข้อมูลที่อยู่บนบัสข้อมูลของระบบจะถูกเขียนลงไปใน 8255
A0-A1	(Address Input) ใช้สำหรับชี้ตำแหน่งของเรจิสเตอร์ภายใน 8255 ที่ cpu ต้องการติดต่อด้วย
RESET	เมื่อขานี้มีค่าลอจิก 1 8255 จะอยู่ในช่วงรีเซ็ท พอร์ตทุกพอร์ตจะอยู่ในโหมดของอินพุทพอร์ต
PA0-PA7	เป็นพอร์ตข้อมูลที่ใช้สำหรับต่อกับอุปกรณ์ภายนอก
PB0-PB7	เป็นพอร์ตข้อมูลที่ใช้สำหรับต่อกับอุปกรณ์ภายนอก
PC0-PC7	เป็นพอร์ตข้อมูลที่ใช้สำหรับต่อกับอุปกรณ์ภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 16 แสดงบล็อกไดอะแกรมของ 8255

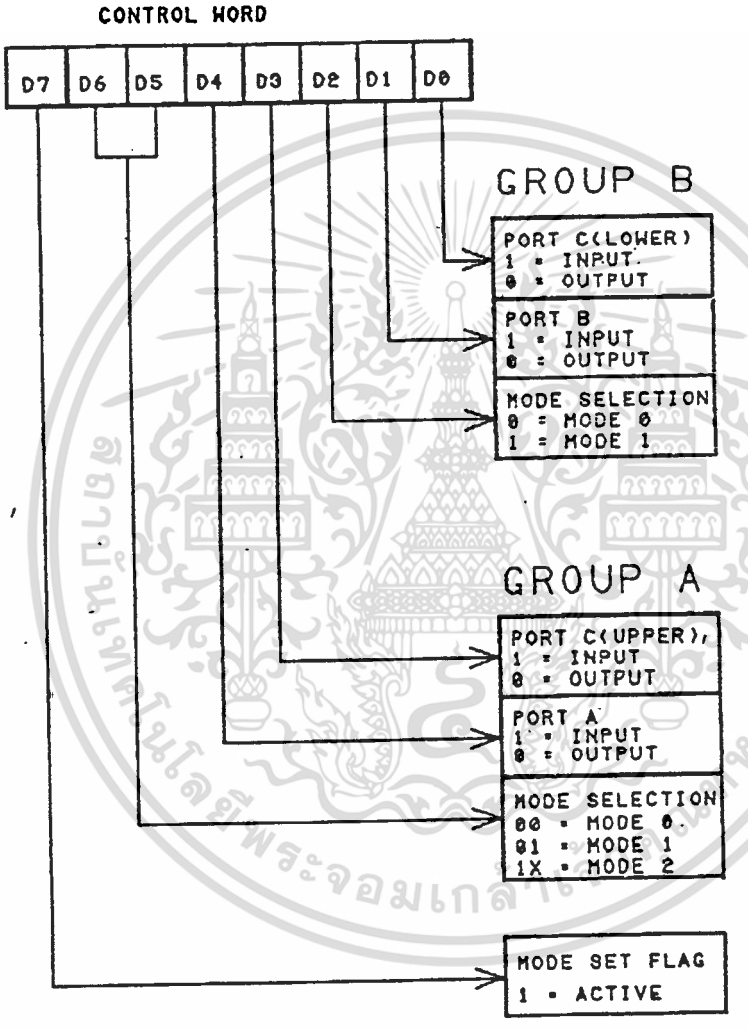
การติดต่อกับพอร์ตต่าง ๆ ของ 8255

ภายใน 8255 มีพอร์ตภายในอยู่ 4 พอร์ต ซึ่งเราสามารถติดต่อกับพอร์ตต่าง ๆ ได้ดังนี้

DEVICE PINS				PORT NAME
RD	WR	A1	A0	
1	0	0	0	Write PORT A data
0	1	0	0	Read PORT A data
1	0	0	1	Write PORT B data
0	1	0	1	Read PORT B data
1	0	1	0	Write PORT C data
0	1	1	0	Read PORT C data
1	0	1	1	Write Control Word
0	1	1	1	Illejal Read Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานของพอร์ต A,B,C จะกำหนดโดยข้อมูลที่ส่งไปยังพอร์ตควบคุมโดยแต่ละบิตจะมีความหมายดังแสดงในรูป 8.3 ซึ่งสามารถกำหนดการทำงานของ 8255 ได้ 3 โหมด



รูปที่ 17 แสดง CONTROL WORD ของ 8255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้งาน 8255 ในโหมด 0

ในที่นี้เราใช้งาน 8255 ในโหมด 0 ดังนั้นจึงจะอธิบายการใช้งาน 8255 ในโหมด 0 เท่านั้น

การทำงานของ 8255 ในโหมด 0 จะเป็นพอร์ตอินพุตหรือเอาท์พุตแบบธรรมดา เราสามารถกำหนดให้ 8255 ทำงานในโหมด 0 ได้โดยส่ง CONTROL WORD ไปยังพอร์ตควบคุม มีค่าต่อไปนี้

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0

จากคำสั่งควบคุมในรูป 8.3 เราสามารถอธิบายความหมายของบิตต่าง ๆ ได้ดังนี้

D7 = 1 กำหนดให้ข้อมูลนี้เป็น CONTROL WORD
D6,D5 = 0 กำหนดให้พอร์ต A ใน 8255 ทำงานในโหมด 0
D4 = 0 กำหนดให้พอร์ต A เป็นเอาท์พุต
D3 = 0 กำหนดสี่บิตบนของพอร์ต C เป็นเอาท์พุต
D2 = 0 กำหนดพอร์ต B ทำงานในโหมด 0
D1 = 0 กำหนดพอร์ต B เป็นเอาท์พุต
D0 = 0 กำหนดสี่บิตล่างของพอร์ต C เป็นเอาท์พุต

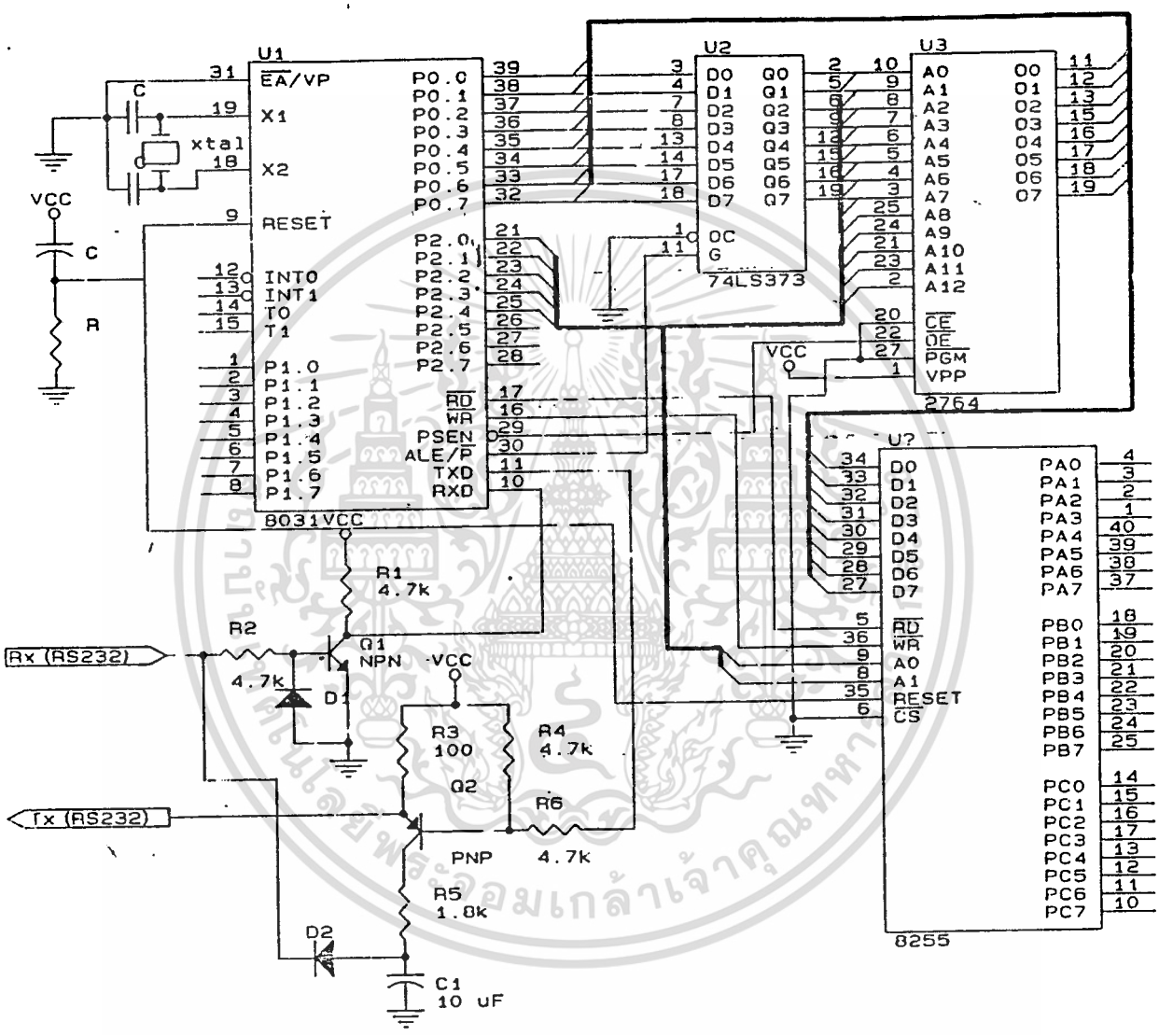
จาก CONTROL WORD ที่ส่งออกไปจะกำหนดให้พอร์ต A,B,C เป็นเอาท์พุตพอร์ตทั้งหมด ซึ่งเราสามารถต่ออุปกรณ์ภายนอกทั้งหมดได้ 24 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วงจรการขยายพอร์ตและวิธีการต่อ RS232

ชิปเบอร์ 8255 ชิปที่ทำงานเป็นพอร์ตได้ 3 พอร์ต โดยที่ไม่มี ROM หรือ RAM ในตัวชิปนี้ซึ่งเป็นชิปที่นิยมใช้กัน ได้แสดงวิธีการต่อพ่วงกับ MCS-48 ดังรูปที่ 5 เนื่องจากขาแอดเดรสของ 8255 คือ A0 และ A1 ไม่สามารถจัดการกับการมัลติเพล็กแอดเดรสกับข้อมูลเองได้ จึงต้องต่อกับขาเอาท์พุตของตัวแลทช์ 74LS373 ที่ใช้ในการแลทช์แอดเดรสข้อมูลโปรแกรม EPROM ภายนอกให้ได้ค่าการถอดรหัสแอดเดรสA0,A1 ได้ 4 ตำแหน่งเพื่อใช้ในการส่งข้อมูลออกและเข้าพอร์ตทั้งสามของ 8255และอีกทีหนึ่งแอดเดรสที่เหลือจะใช้ในการจัดการส่งค่าควบคุมลักษณะการใช้งานในการเริ่มต้นโปรแกรมที่เข้าชิป 8255 นี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 17 การต่อวงจรที่ใช้ชิปเบอร์8255 และการต่อ RS232

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

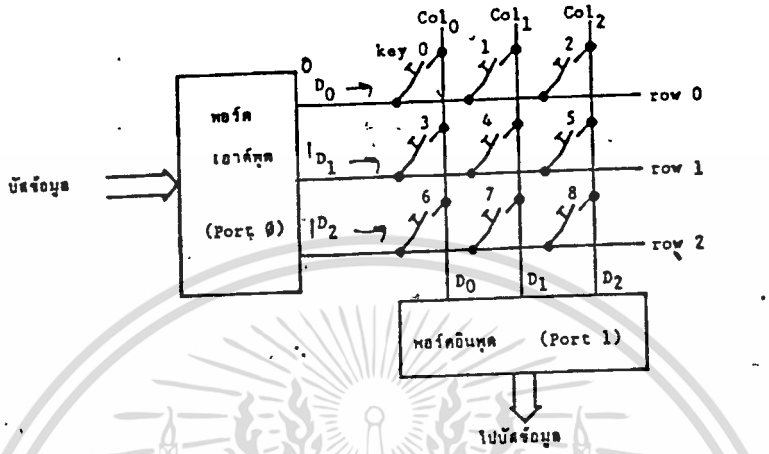
การอ่านข้อมูลจากสวิทช์จำนวนมาก

การต่อสวิทช์แบบปกติจะใช้ 1 บิตต่อสวิทช์1 ตัว แต่ในการใช้งานจริงของระบบไมโครคอมพิวเตอร์ต้องการใช้สวิทช์หรือคีย์เป็นจำนวนมาก ดังนั้นถ้าใช้ 1 สวิทช์ต่อ 1 บิตจะทำให้สิ้นเปลืองพอร์ตเป็นจำนวนมาก ในที่นี้จะใช้การต่อสวิทช์แบบเมตริกซ์ ซึ่งจะทำให้ลดจำนวนพอร์ตลง ซึ่งตัวอย่างของวิธีการต่อแบบนี้แสดงได้ดังรูป 18 จากรูปเห็นได้ว่าการต่อวงจรสวิทช์แบบนี้ทำให้ลดสายข้อมูลลงไปมาก เช่น ถ้าพิจารณาจากจำนวนของสวิทช์และจำนวนของสายเมื่อต่อแบบธรรมดาและจำนวนของสายเมื่อต่อแบบเมตริกซ์จะเป็นดังนี้

จำนวนสวิทช์	จำนวนสาย	
	ต่อแบบธรรมดา	ต่อแบบเมตริกซ์
1X1	1	1
2X2	4	4
3X3	9	6
4X4	16	8
5X5	25	10
8X8	64	16

จะเห็นได้ชัดว่า เมื่อจำนวนของสวิทช์มากขึ้นก็จะทำให้ประหยัดสายข้อมูลมากขึ้น ไปด้วยข้อยุ่งยากทางด้านฮาร์ดแวร์จะลดลง และการตรวจสอบสวิทช์ที่กดจะใช้วิธีการทางซอฟต์แวร์ช่วยจัดการ

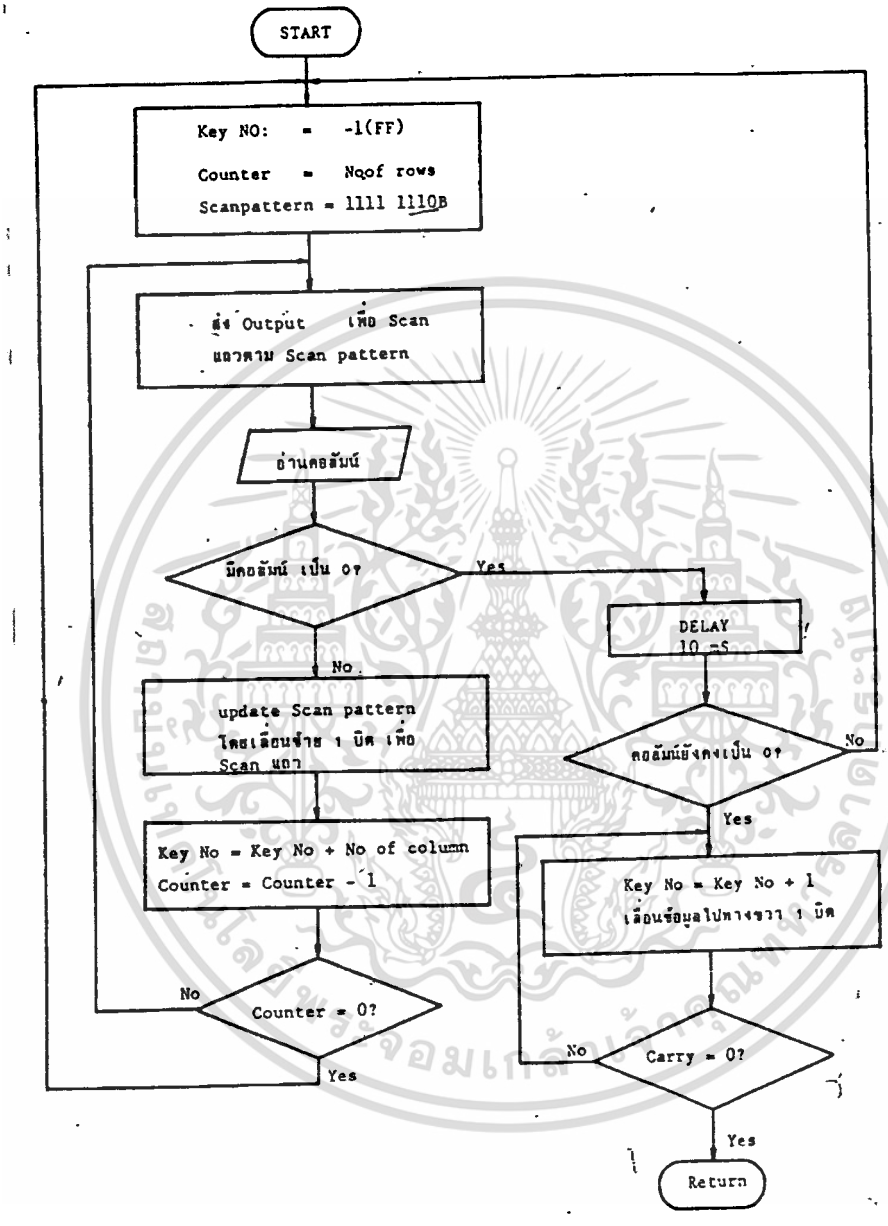
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 18 การต่อสวิทช์แบบเมตริกซ์แบบ 3X3

หลักการทำงานของโปรแกรมจะใช้วิธีกวาด (SCANNING) เพื่อหาว่าสวิทช์ใดที่ถูกกดโดยอาจเริ่มต้นที่แถวที่ 0 โดยกำหนดให้แถวที่ 0 มีระดับเป็น 0 ส่วนแถวอื่น ๆ มีระดับเป็น 1 แล้วทำการตรวจสอบทีละคอลัมน์ โดยเริ่มจากคอลัมน์ที่ 0 ตรวจสอบว่ามีระดับเป็น 0 หรือไม่ ถ้ามีระดับเป็น 0 แสดงว่าสวิทช์ที่ตำแหน่งนี้ถูกกด ถ้าไม่มีก็เลื่อนไปคอลัมน์ถัดไป และจะทำการเลื่อนและตรวจสอบทั้งทางด้านคอลัมน์และแถวต่อ ๆ ไปจนกว่าจะหมดหรือกระทั่งพบสวิทช์ที่ถูกกด และเมื่อพบว่าสวิทช์ตัวใดถูกกดแล้วต้องทราบว่าสวิทช์นั้นคือสวิทช์ตำแหน่งใดเพื่อที่จะได้ตีความหมาย หรือสร้างรหัสขึ้นมาตามสวิทช์นั้น ๆ ตัวอย่างการตรวจสอบเพื่อหาตำแหน่งของสวิทช์ที่ถูกกดสามารถทำได้ดังผังงานรูปที่ 19

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 19 ผังงานของการอ่านคีย์ที่ต่อแบบเมทริกซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากฟังงานแสดงการอ่านสวิทช์ที่ต่อแบบเมทริกซ์ดังรูปที่ 19 โดยจะตรวจสอบว่า เมื่อมีการกดสวิทช์ สวิทช์นั้นคือสวิทช์อะไร เพื่อที่จะได้สร้างรหัสของสวิทช์ที่กดนั้น การตรวจสอบสามารถอธิบายได้เป็นลำดับดังนี้

1. ตั้งหมายเลขสวิทช์ไว้ที่ -1 ตัวนับเท่ากับจำนวนของแถว ตั้งแพทเทอร์นของการกวาดโดยให้ทุกบิตเป็น 1 ยกเว้นบิตของแถว 0 ให้เป็นระดับลอจิก 0
2. ส่งแพทเทอร์นที่ตั้งไว้ไปยังพอร์ตเอาท์พุตของคีย์บอร์ด
3. อ่านข้อมูลจากพอร์ตอินพุท ตรวจสอบว่ามีคอลัมน์ใดเป็น 0 หรือไม่ ถ้ามีให้ไปลำดับ 7
4. เปลี่ยนค่าของแพทเทอร์นที่ใช้ในการกวาดโดยการเลื่อนไปทางซ้าย 1 บิต
5. เพิ่มค่าหมายเลขสวิทช์โดยการบวกด้วยจำนวนคอลัมน์
6. ลดค่าของตัวนับแถวไป 1 ถ้าตัวนับไม่เท่ากับ 0 ให้กลับไปลำดับที่ 2 และถ้าเท่ากับ 0 แล้วแสดงว่าไม่มีการกดสวิทช์ ให้กลับไปเริ่มต้นใหม่
7. เมื่อพบการกดสวิทช์ ให้หน่วงเวลาประมาณ 10 mS เพื่อการ ดีบาวนซ์ หลังจากนั้นอ่านข้อมูลที่พอร์ตอินพุทใหม่ เพื่อตรวจสอบว่าสวิทช์ยังคงถูกกดอยู่หรือไม่ ถ้ามีการเปลี่ยนแปลงให้กลับไปลำดับที่ 1 ใหม่
8. เพิ่มค่าหมายเลขสวิทช์ขึ้นอีก 1 เลื่อนข้อมูลที่อ่านได้ทางคอลัมน์ไปทางขวา 1 บิต
9. ทำการทดสอบแฟลกตัวทด ถ้าเท่ากับ 1 กลับไปลำดับที่ 8
10. เก็บค่าหมายเลขสวิทช์ที่ได้ แล้วกลับสู่โปรแกรมหลัก เพื่อนำค่าหมายเลขสวิทช์ที่ได้ไปตีความเพื่อทำงานต่อไป

ถ้าการกดสวิทช์ใช้เวลานาน CPU อาจตรวจพบว่ามีการกดสวิทช์อีก และจะทำงานในหน้าที่นั้นซ้ำไปซึ่งไม่ควรจะเป็นเช่นนี้ ดังนั้นจึงต้องมีการแก้ไข คือ ก่อนที่จะอ่านสวิทช์ใหม่จะต้องตรวจสอบก่อนว่าสวิทช์ทุกสวิทช์ได้ถูกปล่อย (RELEASE) หมดแล้วหรือไม่ ถ้าไม่ก็จะรอจนกว่าจะปล่อยให้หมดเสียก่อนแล้วจึงเข้าไปทำการอ่านการกดสวิทช์ ดังนั้นควรจะต้องมีโปรแกรมการตรวจสอบการปล่อยสวิทช์ก่อนที่จะมีการอ่าน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

โปรแกรมควบคุมการทำงานของระบบ

บทนำ

สำหรับการเขียนโปรแกรมควบคุมการทำงานของเครื่องปรับอากาศในตอนที่ 1 นี้ จะเป็นการเขียนโปรแกรมควบคุมการทำงานตามวงจรรูปที่ 21 โดยจะมีการรับค่าจากสวิทช์ 5 ตัวเข้าไปทำการประมวลผล เพื่อให้มีการทำงานตามสวิทช์ที่ถูกกด

การใช้งานและขอบเขตของงาน

วิธีการใช้รีโมท

1 เมื่อกดสวิทช์ POWER LED "POWER-ON","AUTO" และ "24 C" จะติด เครื่องปรับอากาศจะทำงานตามอุณหภูมิที่ตั้งไว้ คือ 24 องศาเซลเซียส และพัดลมจะทำงานแบบ AUTO คือ ทำงานที่ high speed, medium speed และ low speed ตามลำดับ ซึ่งขึ้นอยู่กับอุณหภูมิห้อง และอุณหภูมิที่ตั้งไว้

2 ถ้าไม่ต้องการให้พัดลมทำงานแบบ AUTO แต่ต้องการเลือกให้พัดลมทำงานที่ high speed, medium speed หรือ low speed อย่างใดอย่างหนึ่งเราก็สามารถทำได้ โดยกดสวิทช์ FAN ซ้ำ ๆ กัน

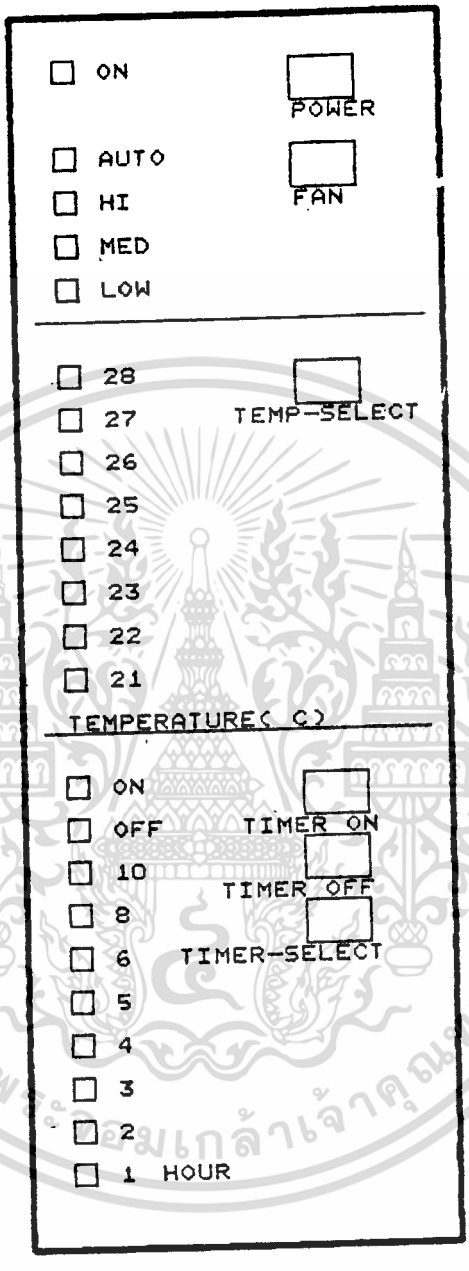
3 เราสามารถเลือกอุณหภูมิที่ต้องการให้เครื่องปรับอากาศทำงาน ตามอุณหภูมิที่เราตั้งไว้ได้โดยกดสวิทช์ TEMP-SELECT ซ้ำ ๆ กัน ซึ่งอุณหภมินี้เราสามารถเลือกได้ตั้งแต่ 21-28 องศาเซลเซียส

4 สำหรับการตั้งเวลาปิดล่วงหน้านั้น LED"POWER-ON" จะต้องติดอยู่ (เครื่องปรับอากาศทำงานอยู่) จากนั้นเราก็สามารถตั้งเวลาปิดล่วงหน้าได้ โดยการกดสวิทช์ TIMER-OFF ซึ่งจะทำให้ LED "OFF" และ "1 HOUR" ติดสว่างอยู่ และถ้าต้องการยกเลิกการตั้งเวลานี้ก็ทำได้โดย การกดสวิทช์ TIMER-OFF ซ้ำอีกครั้ง

5 สวิทช์ TIMER-SELECT จะใช้สำหรับเลือกจำนวนชั่วโมงที่ต้องการตั้งเวลาปิดหรือเปิดล่วงหน้า ซึ่งสามารถเลือกได้ตั้งแต่ 1-10 ชั่วโมง

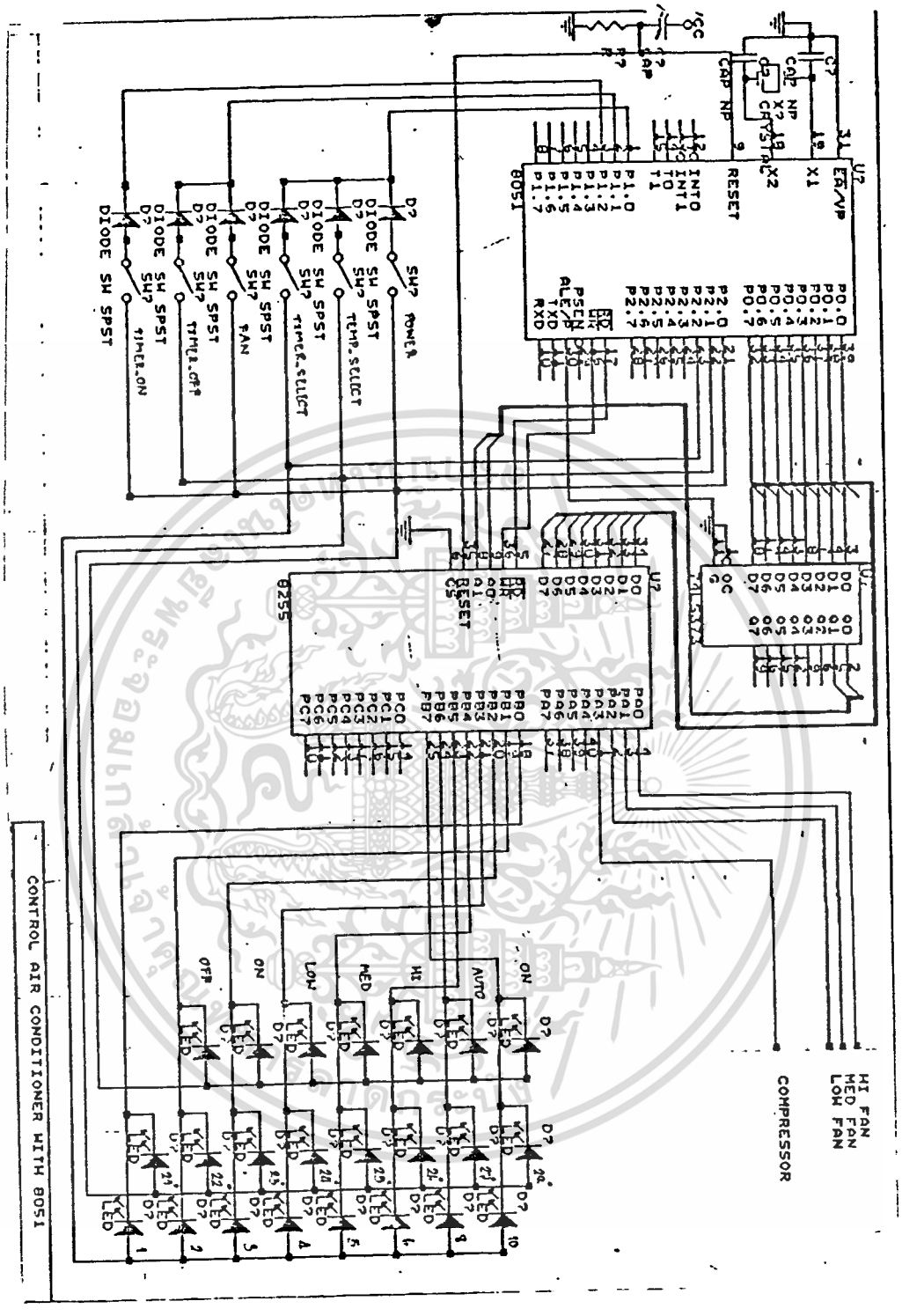
6 การตั้งเวลาเปิดล่วงหน้า ทำได้โดยการกดสวิทช์ TIMER-ON มีผลทำให้ LED "ON","AUTO" และ "24 C" ติด สำหรับการตั้งเวลาเปิดล่วงหน้านี้ จะต้องให้เครื่องปรับอากาศอยู่ในสภาวะที่ไม่ทำงานเท่านั้น (สังเกตได้จาก LED "POWER-ON" จะดับ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 20 REMOTE CONTROL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 21 วงจรแสดงระบบ

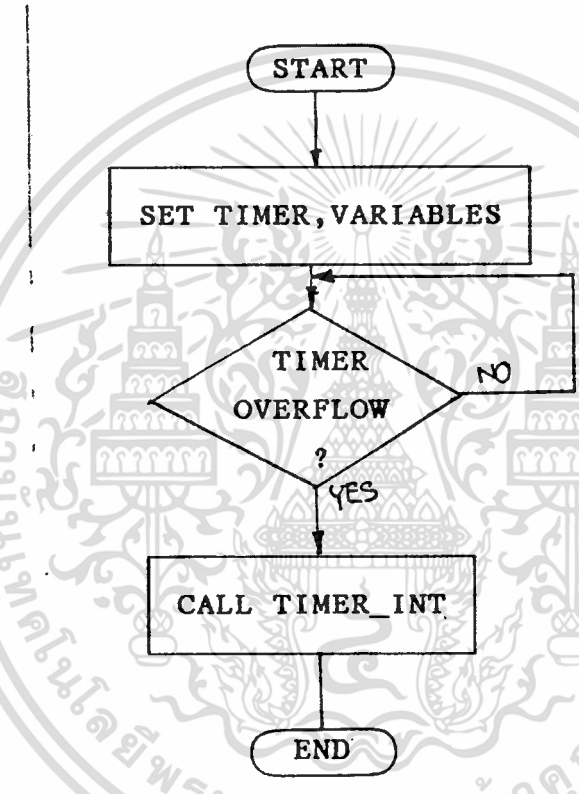
โปรแกรมควบคุมการทำงานตามผังงานรูปที่ 22

ในตอนแรกจะเป็นการเซ็ทให้ตัวจับเวลา 0 (TIMER 0)เริ่มทำงานโดยเราจะเซ็ทให้ตัวจับเวลา 0 ทำงานในโหมด 2 ซึ่งจะมีการโหลดขนาด 8 บิต โดยอัตโนมัติที่ตัวนับและตัวจับเวลา โดยใช้ TH0 เก็บค่าที่ตั้งไว้ และจะถ่ายเข้าไปที่ TL0 ใหม่ทุกครั้งที่เกิด Overflow คือ TL0 ถูกนับเป็น '0' หมด เมื่อเกิด Overflow ก็จะไปทดให้แฟลกอินเตอร์รัพต์ TF0 ปรับเป็น '1'

การที่จะให้ตัวจับเวลา 0 ทำงานในโหมด 2 นั้น สามารถทำได้โดยเซ็ทให้ TMOD=F2H นอกจากนี้ ยังต้องมีการเซ็ทค่าให้กับ TCON ด้วย เพื่อควบคุมให้บิตเริ่มทำงาน หรือควบคุมให้ตัวจับเวลาเริ่มทำงานนั่นเอง โดยเราจะเซ็ทให้ TCON=10H

ในที่นี้เราต้องการให้เกิดการอินเตอร์รัพต์ทุก ๆ 200 MS(micro second) ดังนั้นจึงตั้งค่า TH0=38H และในตอนเริ่มต้นเราจะเซ็ทให้ TL0=38H ด้วย เนื่องจากว่า เมื่อ TL0 ถูกนับเป็น "0" ทุกบิตแล้วเกิด Overflow ไปทดให้แฟลกอินเตอร์รัพต์ TF0 เป็น '1' จะเป็นเวลา 200 MS พอดี จากนั้นจะเข้าสู่โปรแกรมบริการการอินเตอร์รัพต์ ที่ชื่อว่า TIMER_INT ทันที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 22 ผังงานของ MAIN PROGRAM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานของฟังงานตามรูปที่ 23

ส่วนที่ 1 : เมื่อเกิด Overflow จาก TL0 TF0 จะปรับเป็น '1' และ TL0 จะถูกโหลดโดยอัตโนมัติจากค่าที่ตั้งไว้ใน TH0 เมื่อ TF0 เป็น '1' ก็จะเข้าสู่ TIMER_INT PROCEDURE ทันที ซึ่งจะมีการนับค่าเวลาเก็บไว้ คือ Overflow 1 ครั้ง เวลาจะเพิ่มขึ้น 200 MS เราจะบวกค่าเวลาที่เพิ่มขึ้นนี้ไปเรื่อย ๆ จนกระทั่งค่าเวลาครบ 20 mS ก็จะไปทำการสแกนคีย์บอร์ด หรือสวิทช์ของรีโมทนั่นเอง ซึ่งใช้หลักการของการต่อสวิทช์แบบเมทริกซ์ เมื่อทำการสแกนคีย์เสร็จ ก็จะเป็นการเชคว่าจากสวิทช์ที่ถูกกดนั้น มีการเลือกใช้การตั้งเวลาปิด/เปิดล่วงหน้า หรือไม่

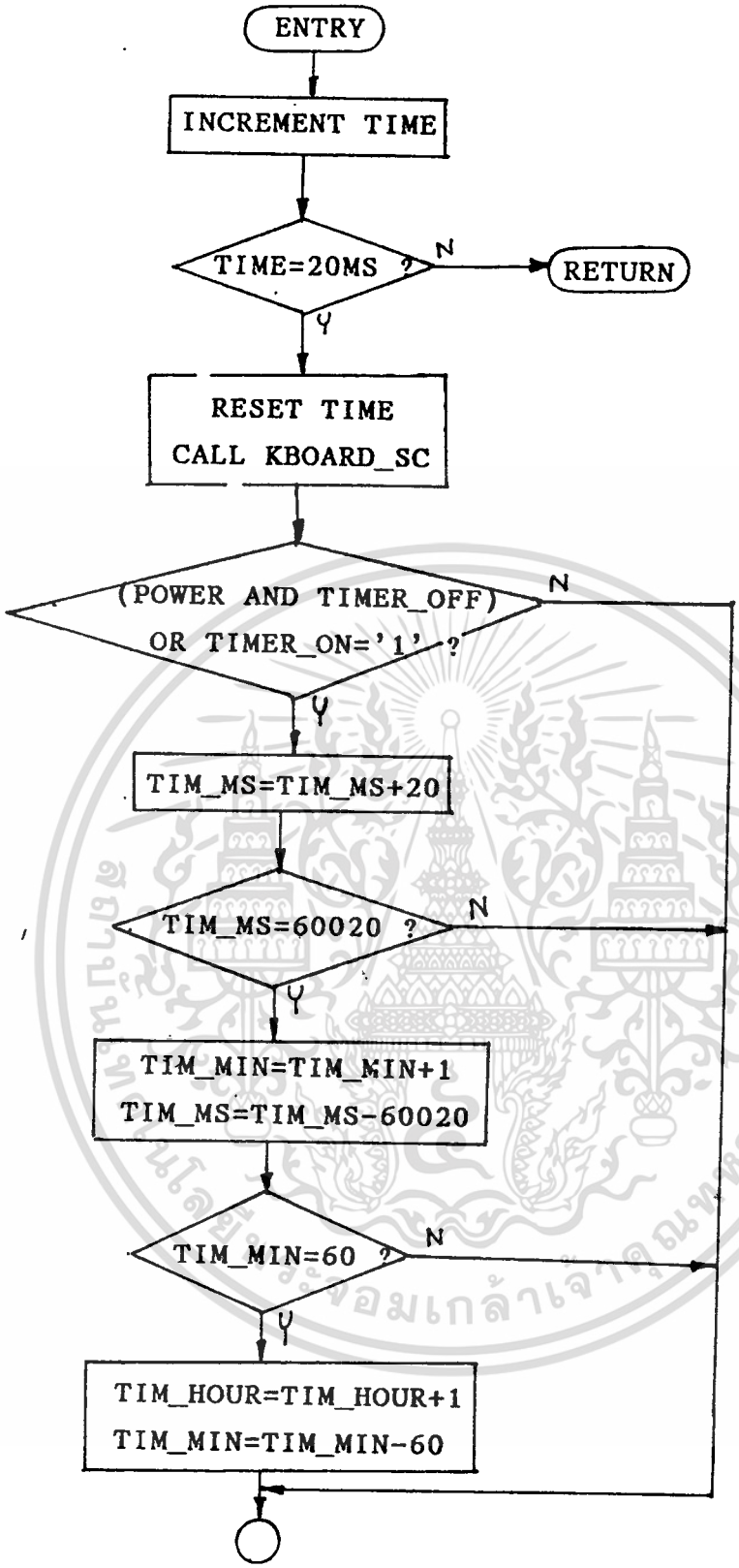
-ถ้ามี จะทำการนับค่าเวลาเก็บไว้ โดยจะบวกเพิ่มขึ้นครั้งละ 20 mS จนครบ 1 นาที เมื่อครบ 1 นาทีก็จะนำค่าเวลานี้ไปเก็บไว้ที่อีกตัวแปรหนึ่ง(TIM_MIN) และตัวแปรนี้จะเพิ่มค่าขึ้นครั้งละ 1 ทุกครั้งที่เวลาครบ 1 นาที และเมื่อตัวแปรนี้มีค่าเท่ากับ 60

(1 ชั่วโมง) ก็จะไปทำให้ตัวแปรชั่วโมงมีค่าเพิ่มขึ้น 1

-แต่ถ้าไม่มีการเลือกใช้การตั้งเวลาปิด/เปิดล่วงหน้าแล้ว โปรแกรมก็จะไม่ต้องทำงานในส่วนดังกล่าวข้างบนคือ จะมาทำในส่วนของการควบคุมเอาท์พุทเลย (ฟังงานของ TIMER_INT PROCEDURE:SECTION 2)

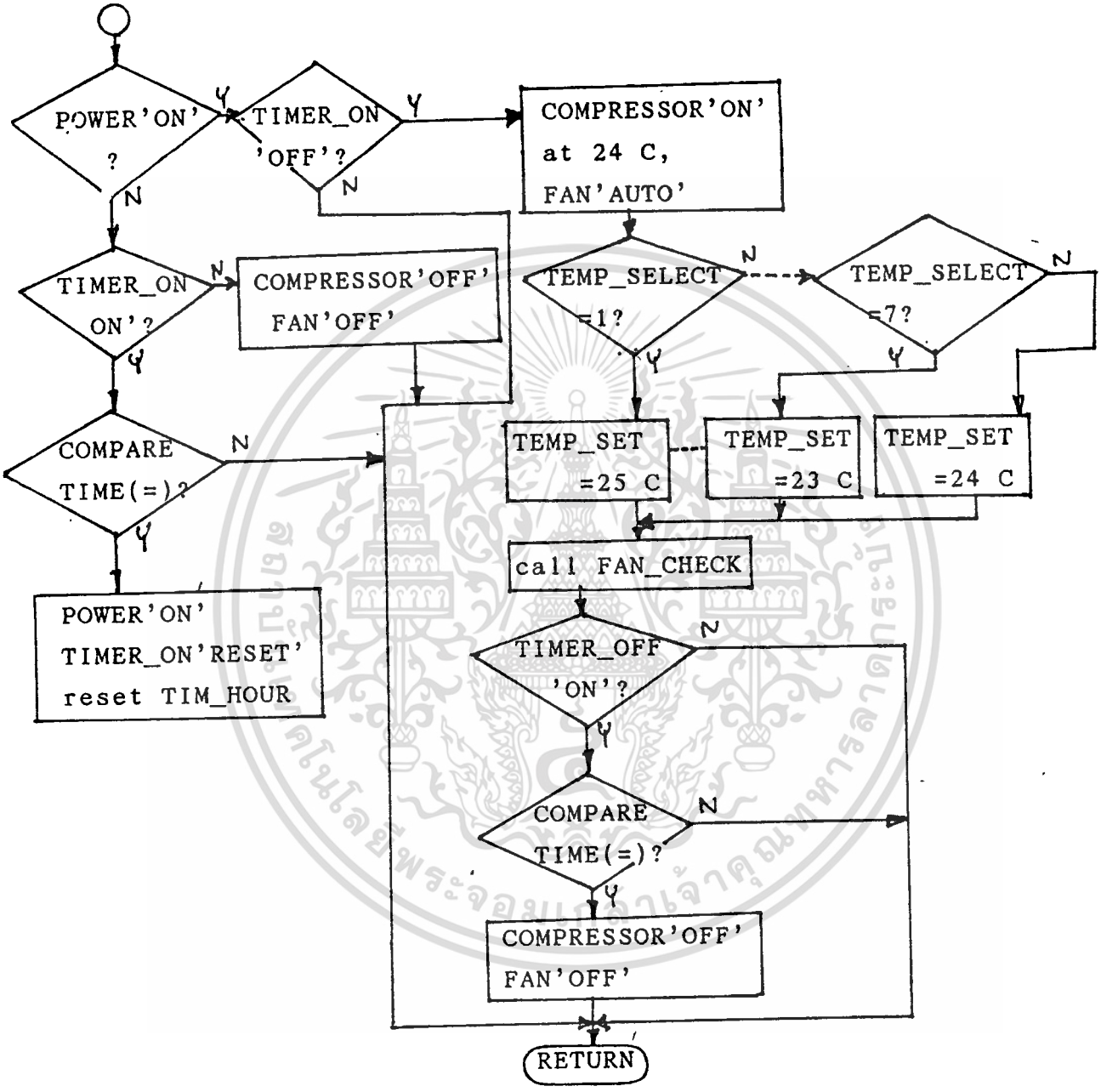
ส่วนที่ 2 : การทำงานในส่วนนี้ จะเป็นการเชคสวิทช์ที่ถูกกดและค่าของสวิทช์ต่าง ๆ เพื่อนำมาใช้ควบคุมการทำงานของ คอมเพรสเซอร์และพัดลม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 23 ผังงานของ TIMER_INT PROCEDURE : SECTION 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 23 (ต่อ) ผังงานของ TIMER_INT PROCEDURE : SECTION 2

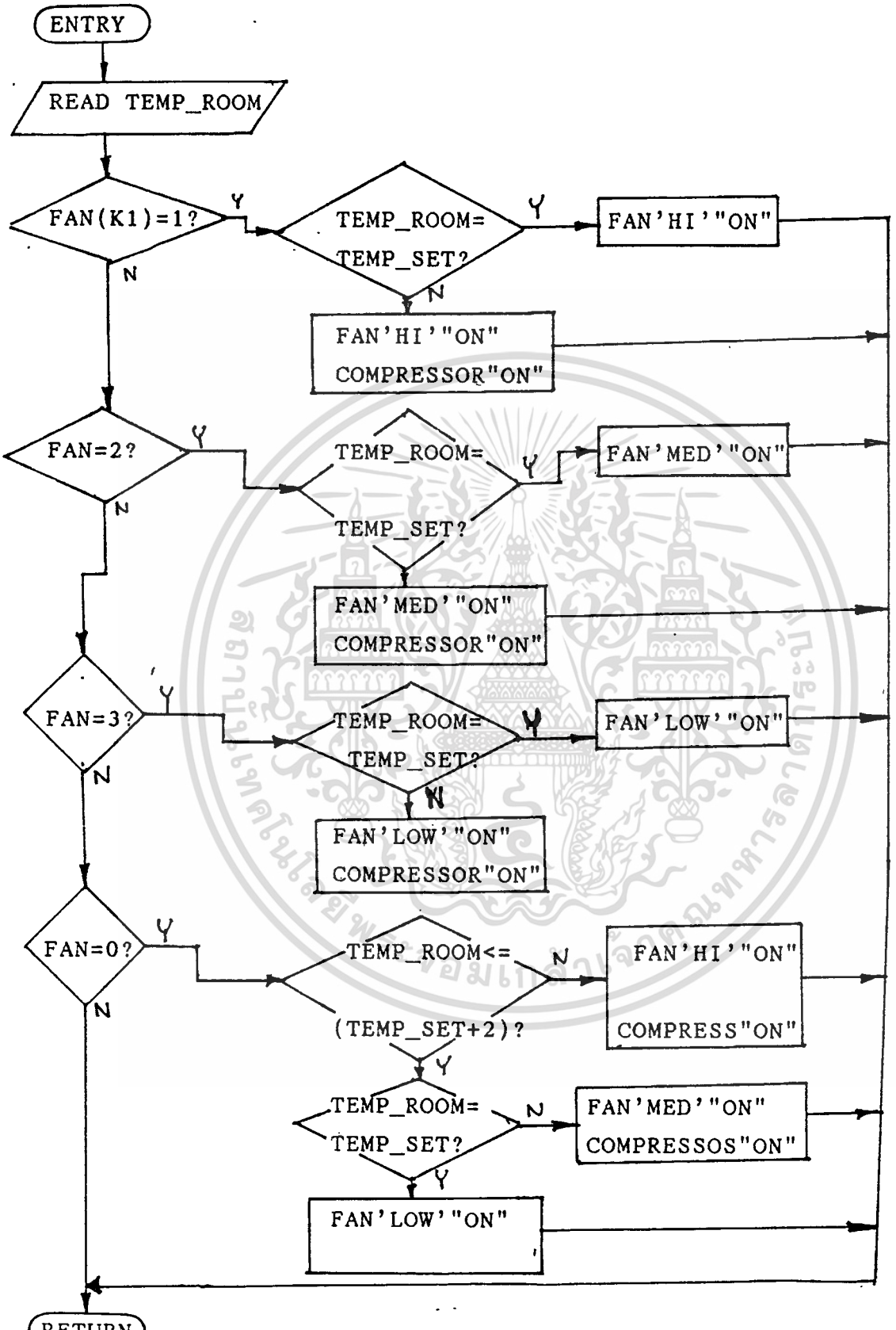
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานตามผังงานรูปที่ 24

สำหรับการทำงานในส่วนนี้ จะเป็นการรับค่าอุณหภูมิที่ตั้งไว้ (TEMP_SET) โดยผู้ใช้ และค่าอุณหภูมิห้องเข้ามา แล้วทำการเชคสวิทช์ FAN ว่าถูกกดกี่ครั้ง ถ้าเชคได้ว่ามีการกดสวิทช์ FAN 1 ครั้ง, 2 ครั้ง หรือ 3 ครั้งแล้ว จากนั้นก็จะเป็นการเชค ว่า อุณหภูมิห้องมีค่าเท่ากับอุณหภูมิที่ตั้งไว้ หรือไม่ ถ้าใช่ พัดลมจะทำงานตาม speed ที่ตั้งไว้ ส่วนคอมเพรสเซอร์จะหยุดทำงาน แต่ถ้าอุณหภูมิห้องไม่เท่ากับอุณหภูมิที่ตั้งไว้ พัดลมจะทำงานตาม speed ที่ตั้ง และคอมเพรสเซอร์จะทำงาน

ถ้าสวิทช์ FAN ถูกกด 4 ครั้งหรือไม่ถูกกดเลย จะทำให้พัดลมทำงานแบบ AUTO คือ ทำงานที่ high speed ก่อนเมื่ออุณหภูมิห้องมีค่ามากกว่าอุณหภูมิที่ตั้งไว้ 2 องศาเซลเซียสพัดลมก็จะทำงานที่ medium speed และเมื่ออุณหภูมิห้องมีค่าเท่ากับอุณหภูมิที่ตั้งไว้ พัดลมจะทำงานที่ low speed สำหรับคอมเพรสเซอร์นั้น จะทำงานเมื่ออุณหภูมิห้องไม่เท่ากับอุณหภูมิที่ตั้งไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 24 ผังงานของ FAN_CHECK PROCEDURE

การทำงานตามผังงานรูปที่ 25

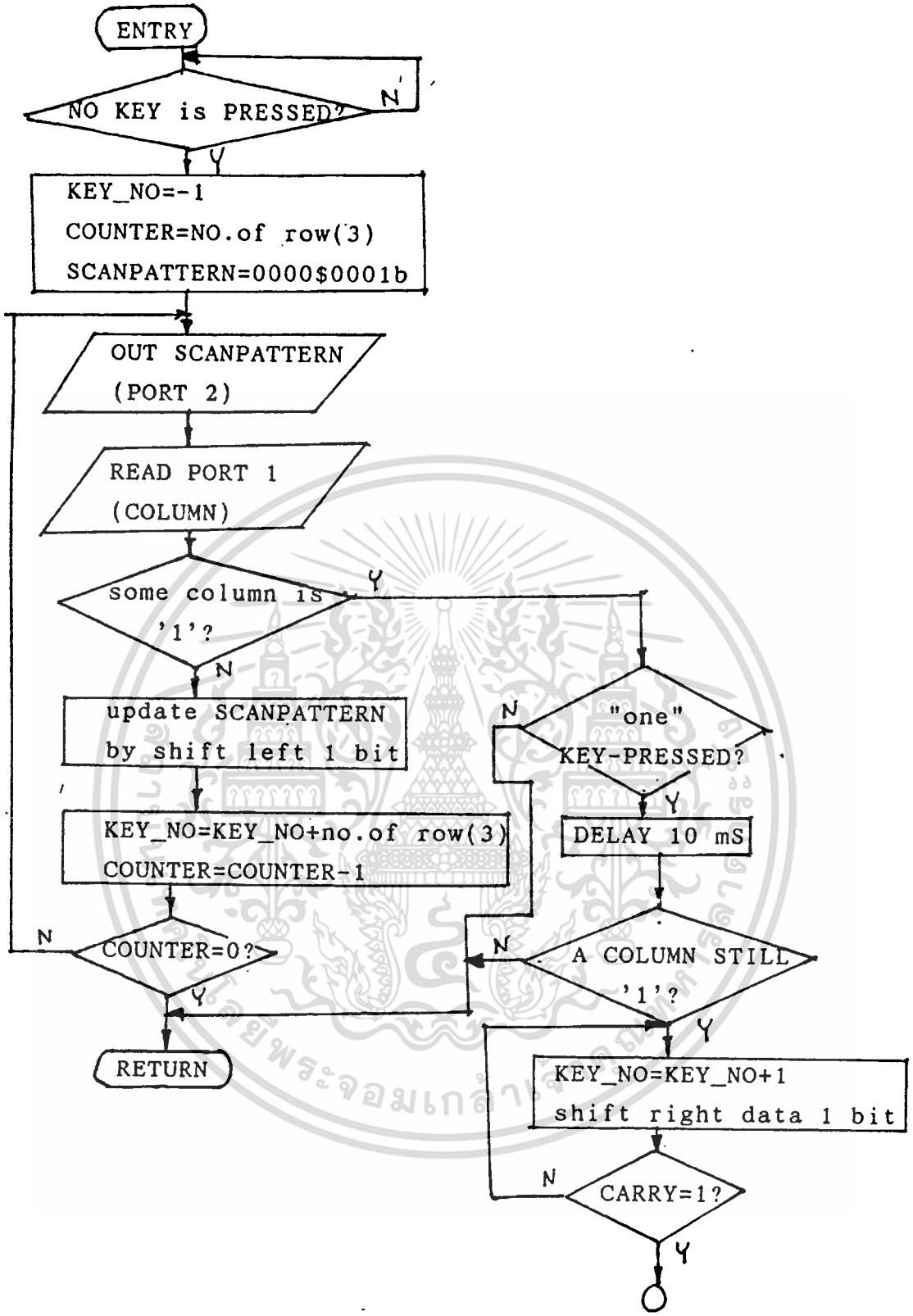
ส่วนที่ 1 : การทำงานในส่วนนี้ เป็นการสแกนหาสวิทช์ที่ถูกกด โดยจะมีการเชคก่อนว่าสวิทช์ทุกสวิทช์ได้ถูกปล่อยหมดแล้วหรือไม่ ถ้าไม่ก็จะรอจนกว่าจะปล่อยให้หมดเสียก่อน แล้วจึงเข้าไปทำการอ่านการกดสวิทช์ สำหรับการอ่านการกดสวิทช์นั้น เราจะให้ PORT 2 มีค่าเท่ากับ SCANPATTERN คือ มีค่าเป็น 0000 0001B, 0000 0010B และ 0000 0100B ตามลำดับ การอ่านการกดสวิทช์จะใช้ PORT 1 เป็นพอร์ตอินพุท เมื่อเราเชคได้ว่ามีการกดสวิทช์แล้ว ก็จะทำการเชคต่อไปอีกว่าสวิทช์ที่ถูกกดนั้นมีเพียง 1 สวิทช์หรือไม่ ถ้าไม่ใช่จะ return กลับไปยังจุดที่เรียกมา แต่ถ้าใช่ จะทำการหน่วงเวลาไว้ 10 mS เพื่อการดีบาวนซ์ หลังจากนั้นอ่านข้อมูลที่พอร์ตอินพุท (PORT 1) ใหม่ เพื่อตรวจสอบว่าสวิทช์ยังคงถูกกดอยู่หรือไม่ ถ้ามีการเปลี่ยนแปลง ให้ return กลับไป แต่ถ้าสวิทช์ยังคงถูกกดอยู่ ก็ทำการหาหมายเลขสวิทช์ที่ถูกกดแล้วเก็บค่าหมายเลขนี้ไว้ในตัวแปร KEY_NO

หมายเหตุ
- คีย์ 0 (KEY_NO=0 : K0) หมายถึง สวิทช์ POWER
- คีย์ 1 (KEY_NO=1 : K1) หมายถึง สวิทช์ FAN
- คีย์ 2 (KEY_NO=2 : K2) หมายถึง สวิทช์ TIMER-ON
- คีย์ 3 (KEY_NO=3 : K3) หมายถึง สวิทช์ TEMP-SELECT
- คีย์ 4 (KEY_NO=4 : K4) หมายถึง สวิทช์ TIMER-OFF
- คีย์ 6 (KEY_NO=6 : K6) หมายถึง สวิทช์ TIMER-SELECT

ส่วนที่ 2 : ในส่วนนี้จะเป็นการเชคหาสวิทช์ที่ถูกกด และเก็บค่าจำนวนครั้งของการกดสวิทช์นั้น ๆ เพื่อนำค่าเหล่านี้ไปควบคุม LED ดวงต่าง ๆ ที่รีโมทให้ ติดหรือดับ (SECTION 3) และควบคุมการทำงานของคอมเพรสเซอร์และพัดลมด้วย

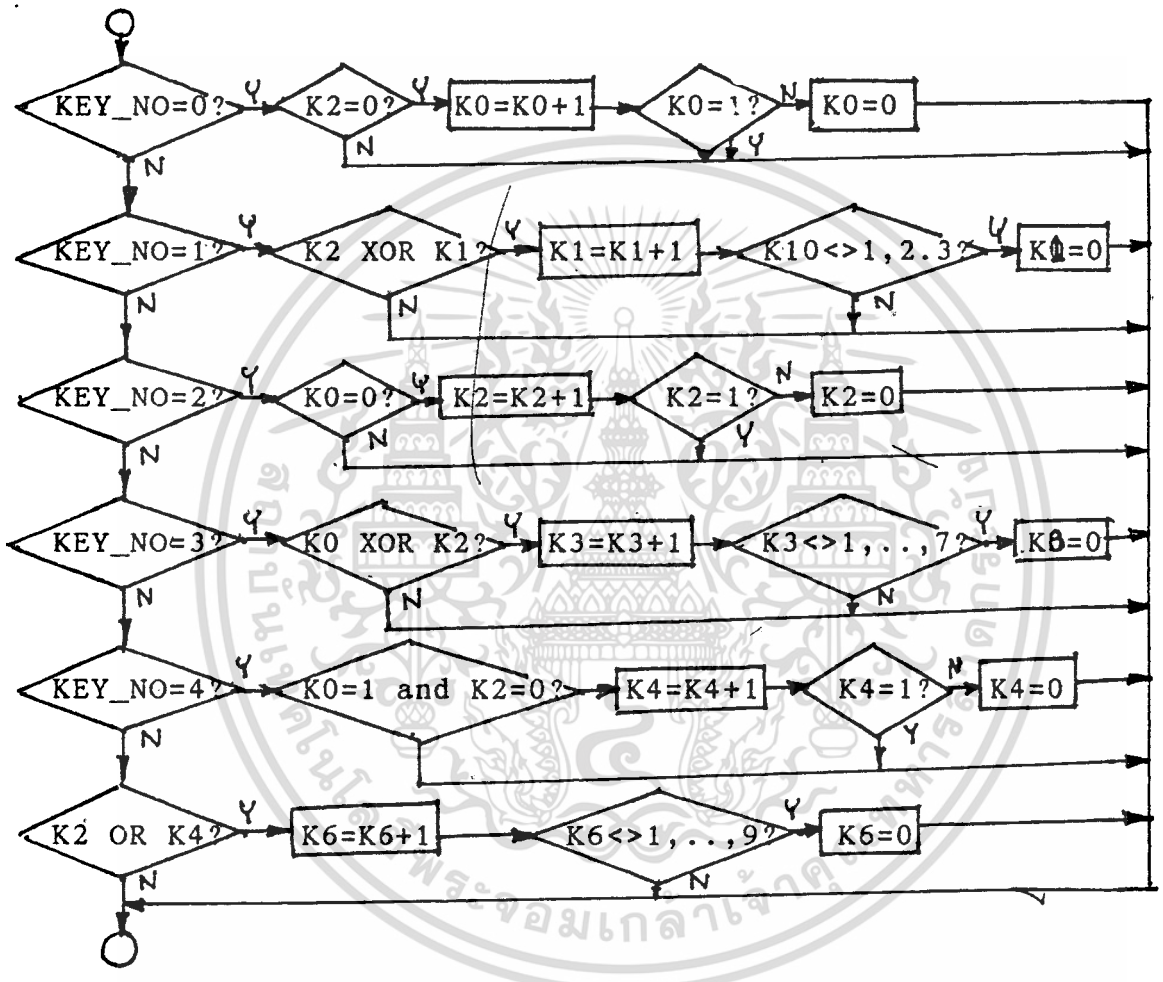
ส่วนที่ 3 : การทำงานในส่วนที่ 3 นี้จะเป็นการควบคุมว่าจะให้ LED ดวงไหนติดและดวงไหนไม่ติด ซึ่งขึ้นอยู่กับสวิทช์ที่ถูกกดด้วย โดย LED ในแต่ละแถว (ดูวงจรในรูปที่ 21 ประกอบ) จะสลับกันติดและดับ เช่น 20 mS แรกให้แถวที่ 1 ติด, 20 mS ต่อมาให้แถวที่ 2 ติด และอีก 20 mS ต่อมาให้แถวที่ 3 ติด (ในขณะที่LED แถวใดแถวหนึ่งติดอยู่นั้น แถวอื่น ๆ จะดับหมด) และวนรอบเช่นนี้ไปเรื่อย ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



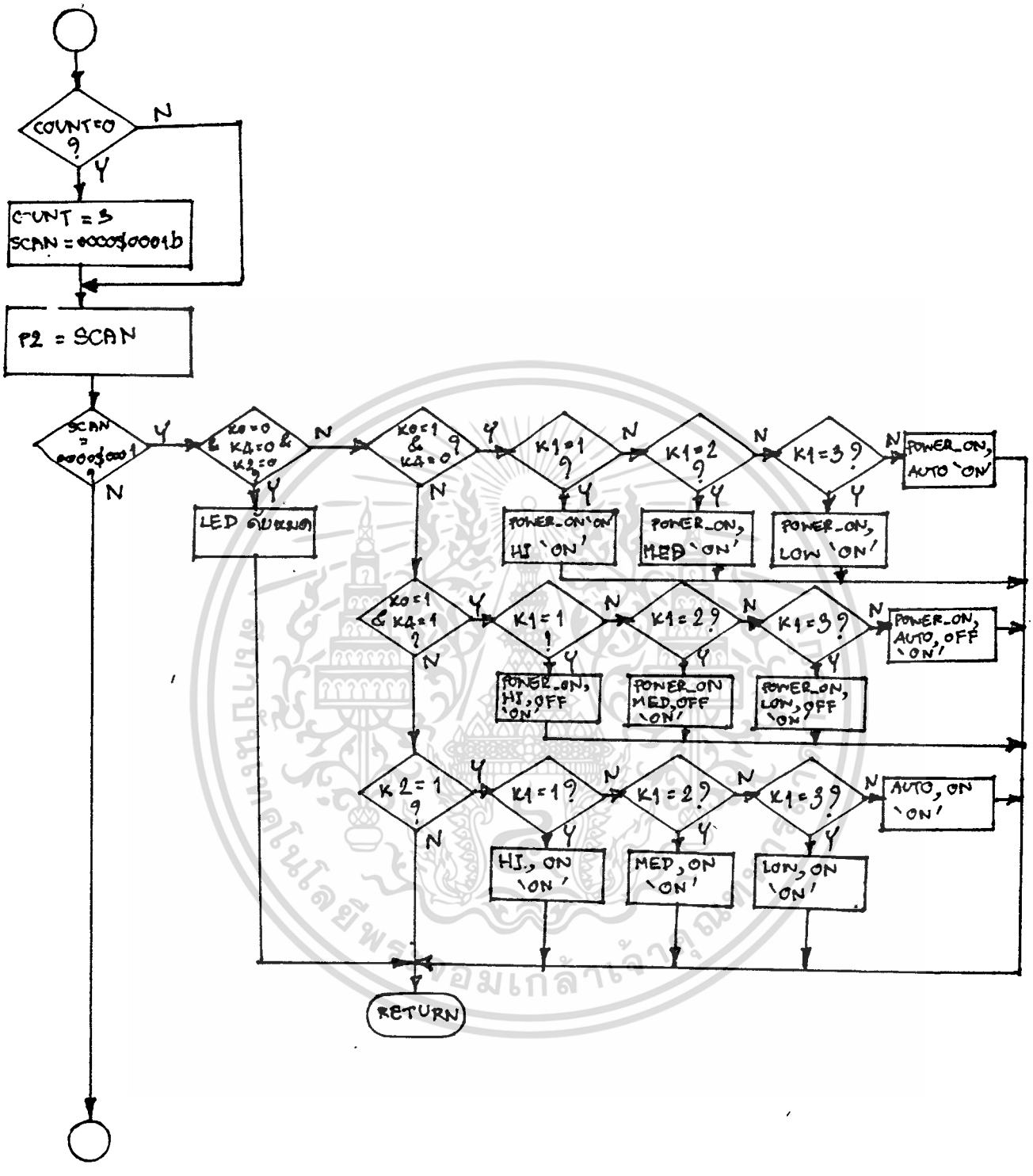
รูปที่ 25 ผังงานของ KBOARD_SC PROCEDURE : SECTION 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



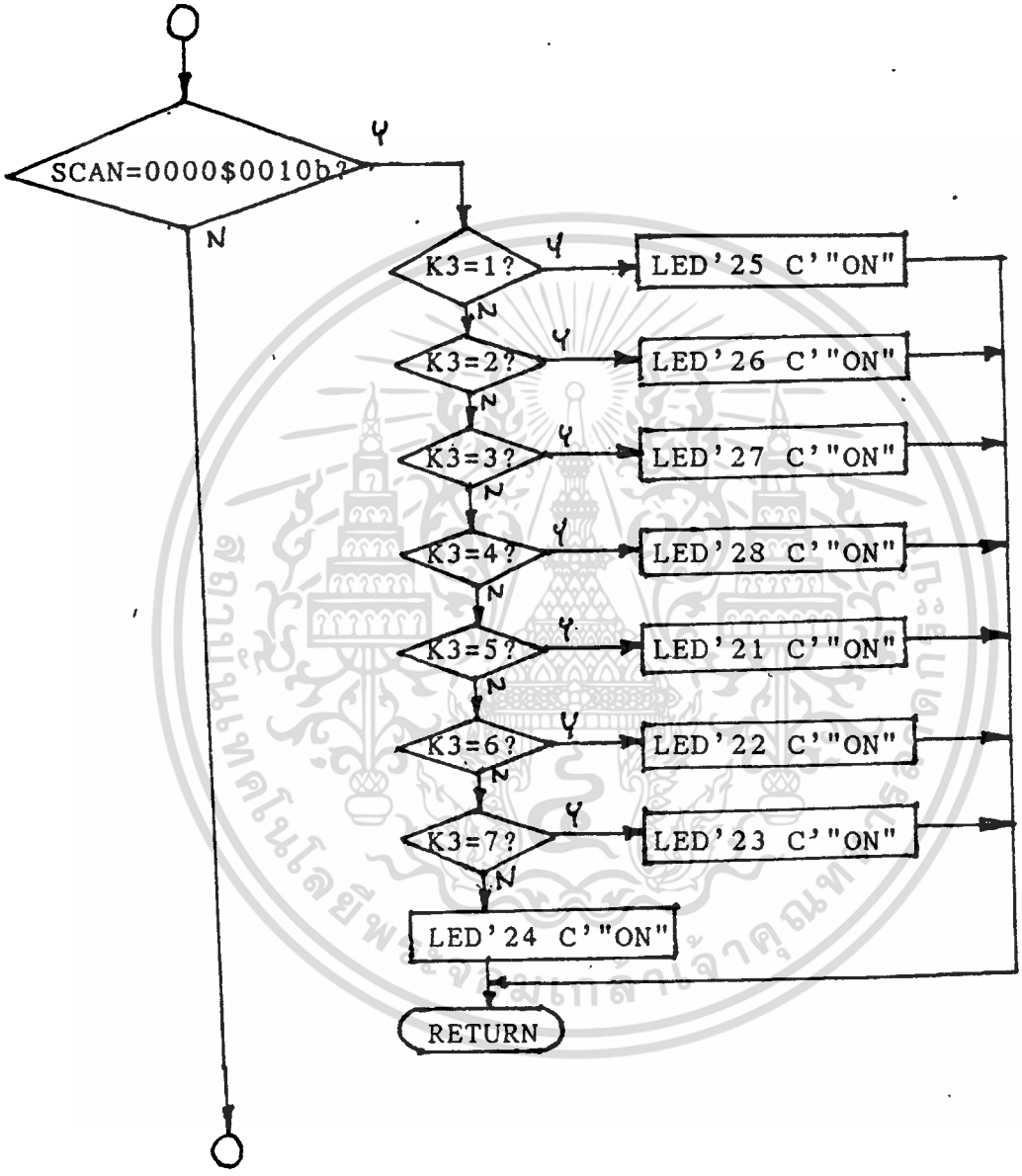
รูปที่ 25 (ต่อ) ผังงานของ KBOARD_SC PROCEDURE : SECTION 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



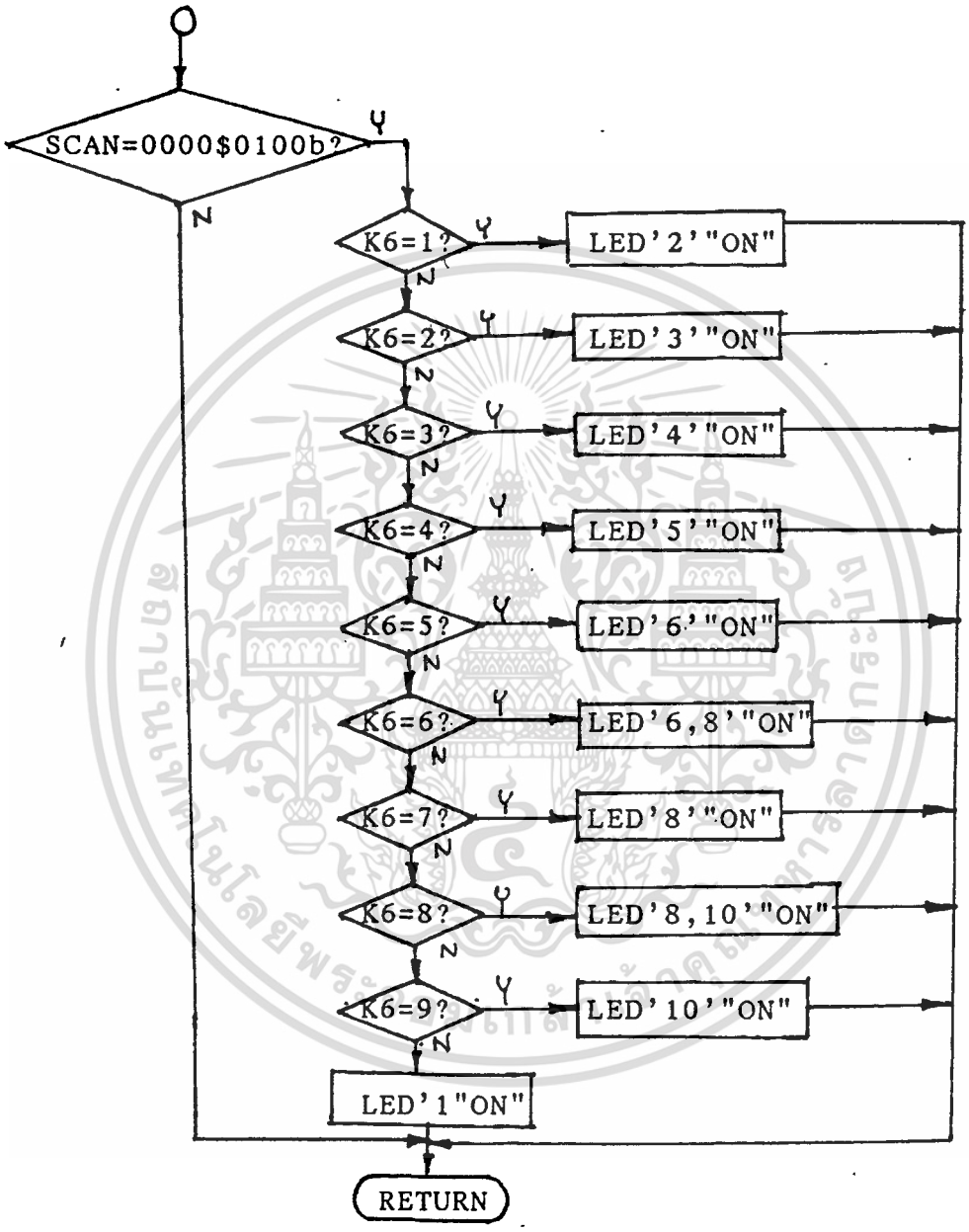
รูปที่ 25 (ต่อ) ผังงานของ KBOARD_SC PROCEDURE : SECTION 3.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 25 (ต่อ) ผังงานของ KBOARD_SC PROCEDURE : SECTION 3.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 25 (ต่อ) ผังงานของ KBOARD_SC PROCEDURE : SECTION 3.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมควบคุมการทำงานของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* main program */
MAIN : do;
$include (reg51.dcl)
declare   TIM   byte,
          TIM_MS  word,
          TIM_MIN  byte,
          TIM_HOUR  byte;
declare   KEY_NO  byte,
          (K0,K1,K2,K3,K4,K6) byte,
          COUNT byte,
          SCAN byte;
declare   PA_PORT_ADDRESS byte CONSTANT (00H),
          PB_PORT_ADDRESS byte CONSTANT (01H),
          CONTROL_WORD_ADDRESS byte CONSTANT (03H),
          PA byte at(PA_PORT_ADDRESS) reg,
          PB byte at(PB_PORT_ADDRESS) reg;
declare   ENABLE_TO_INTERRUPT literally 'ET0=1';
declare   TEMP_SET byte;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SET_UP : TMOD = F2H;
         TCON = 10H;
         TL0  = 38H;
         TH0  = 38H;

         TIM  = 0;
         TIM_MS = 0;
         TIM_MIN= 0;
         TIM_HOUR=0;

         COUNT = 3;
         SCAN  = 0000$0001b;
         (K0,K1,K2,K3,K4,K6) = 0;
         CONTROL_WORD_ADDRESS = 1000$0000b;
         P1 = 1111$1111b;
         ENABLE_TO_INTERRUPT;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
FAN_CHECK : PROCEDURE (TEMP_SET) using TO_REG_BANK;
declare   TEMP_ROOM byte,
          TEMP_SET  byte;
read   TEMP_ROOM;
if (K1=1) then
do;
   if (TEMP_ROOM=TEMP_SET) then
   PA_PORT_ADDRESS=0000$0001b;
   else PA_PORT_ADDRESS=0000$1001b;
end;
if (K1=2) then
do;
   if (TEMP_ROOM=TEMP_SET) then
   PA_PORT_ADDRESS=0000$0010b;
   else PA_PORT_ADDRESS=0000$1010b;
end;
if (K1=3) then
do;
   if (TEMP_ROOM=TEMP_SET) then
   PA_PORT_ADDRESS=0000$0100b;
   else PA_PORT_ADDRESS=0000$1100b;
end;
if ((K1<>1) AND (K1<>2) AND (K1<>3)) then
do;
   if (TEMP_ROOM<=(TEMP_SET+2)) then
   do;
      if (TEMP_ROOM<=(TEMP_SET+2)) then

      PA_PORT_ADDRESS=0000$0100b;
      else PA_PORT_ADDRESS=0000$1010b;
   end;
   else PA_PORT_ADDRESS=0000$1001b;
end;
return;
end FAN_CHECK;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
KBOARD_SC : PROCEDURE using TO_REG_BANK;
declare  COUNTER byte,
         SCANPATTERN byte,
         B  byte,
         H  byte;

     P2 = 0000$0111b;
R1 : B = P1;
     if (B<>1111$1000b) then
     goto R1;
R2 : KEY_NO = -1;
     COUNTER = 3;
     SCANPATTERN = 0000$0001b;
R3 : P2 = SCANPATTERN;
     B  = P1;
     H  = 0000$0000b;
     if (B=H) then
     do;
        SHL(SCANPATTERN,1);
        KEY_NO=KEY_NO+3;
        COUNTER=COUNTER-1;
        if (COUNTER=0) then goto DISPLAY;
        else goto R3;
     end;
     else
M : do while ((B=0000$0001b) or (B=0000$0010b)
    or (B=0000$0100b));
    call TIME(100);
    B = P1;
    if (B=H) then return;
R4 : KEY_NO = KEY_NO+1;
     SCR(B,1);
     if (CARRY<>1) then goto R4;
end M;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if (KEY_NO=0) then
do;
    if (K2=0) then
    do;
        K0=K0+1;
        if (K0<>1) then K0=0;
    end;
    else return;
end;
if (KEY_NO=1) then
do;
    if (K0 xor K2) then
    do;
        K1=K1+1;
        if ((K1<>1) and (K1<>2) and (K1<>3)) then
        K1=0;
    end;
    else return;
end;

if (KEY_NO=2) then
do;
    if (K0=0) then
    do;
        K2=K2+1;
        if (K2<>1) then K2=0;
    end;
    else return;
end;

if (KEY_NO=3) then
do;
    if (K0 xor K2) then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      do;
          K3=K3+1;
          if ((K3<>1) and (K3<>2) and (K3<>3) and (K3<>4)
          and (K3<>5) and (K3<>6) and (K3<>7)) then
          K3=0;
      end;
      else return;
 end;

 if (KEY_NO=4) then
 do;
      if ((K0=1) and (K2=0)) then
      do;
           K4=K4+1;
           if (K4<>1) then K4=0;
      end;
      else return;
 end;

 if ((KEY_NO<>0) and (KEY_NO<>1) and (KEY_NO<>2)
    and (KEY_NO<>3) and (KEY_NO<>4)) then
 do;
      if (K2 or K4) then
      do;
          K6=K6+1;
          if ((K6<>1) and (K6<>2) and (K6<>3) and (K6<>4)
          and (K6<>5) and (K6<>6) and (K6<>7) and (K6<>8)
          and (K6<>9)) then
          K6 = 0;
      end;
 else return;
 end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    /* display by LED */
DISPLAY : if (COUNT=0) then
          do;
              COUNT = 3;
              SCAN = 0000$0001b;
          end;
          P2 = SCAN;
          if (SCAN=0000$0001b) then
    D1 : do;
              if ((K0=0) and (K4=0) and (K2=0)) then
                 PB_PROT_ADDRESS = 1111$1111b;
              if ((K0=1) and (K4=0) then
              do;
                  if (K1=1) then PB_PORT_ADDRESS= 0101$1111b;
                  if (K1=2) then PB_PORT_ADDRESS= 0110$1111b;
                  if (K1=3) then PB_PORT_ADDRESS= 0111$0111b;
                  if (K1=0) then PB_PORT_ADDRESS= 0011$1111b;
              end;
              if ((K0=1) and (K4=1)) then
              do;
                  if (K1=1) then PB_PORT_ADDRESS= 0101$1101b;
                  if (K1=2) then PB_PORT_ADDRESS= 0110$1101b;
                  if (K1=3) then PB_PORT_ADDRESS= 0111$0101b;
                  if (K1=0) then PB_PORT_ADDRESS= 0011$1101b;
              end;
              if (K2=1) then
              do;
                  if (K1=1) then PB_PORT_ADDRESS= 1101$1011b;
                  if (K1=2) then PB_PORT_ADDRESS= 1110$1011b;
                  if (K1=3) then PB_PORT_ADDRESS= 1111$0001b;
                  if (K1=0) then PB_PORT_ADDRESS= 1011$1101b;

              end;
    end D1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    if (SCAN=0000$0010b) then
D2: do;
        if (K3=1) then PB_PROT_ADDRESS = 1110$1111b;
        if (K3=2) then PB_PORT_ADDRESS = 1101$1111b;
        if (K3=3) then PB_PORT_ADDRESS = 1011$1111b;
        if (K3=4) then PB_PORT_ADDRESS = 0111$1111b;
        if (K3=5) then PB_PORT_ADDRESS = 1111$1110b;
        if (K3=6) then PB_PORT_ADDRESS = 1111$1101b;
        if (K3=7) then PB_PORT_ADDRESS = 1111$1011b;
        if (K3=0) then PB_PORT_ADDRESS = 1111$0111b;
end D2;

if (scan=0000$0100b) then
D3: do;
    if (K6=1) then PB_PORT_ADDRESS=1111$1101b;
    if (K6=2) then PB_PORT_ADDRESS=1111$1011b;
    if (K6=3) then PB_PORT_ADDRESS=1111$0111b;
    if (K6=4) then PB_PORT_ADDRESS=1110$1111b;
    if (K6=5) then PB_PORT_ADDRESS=1101$1111b;
    if (K6=6) then PB_PORT_ADDRESS=1001$1111b;
    if (K6=7) then PB_PORT_ADDRESS=1011$1111b;
    if (K6=8) then PB_PORT_ADDRESS=0011$1111b;
    if (K6=9) then PB_PORT_ADDRESS=0111$1111b;
    if (K6=0) then PB_PORT_ADDRESS=1111$1110b;
end D3;
SHL(SCAN,1);
COUNT=COUNT-1;
return;
end KBOARD_SC;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
TIMER_INT: PROCEDURE interrupt TO_VECT using TO_REG_BANK;

TIM=TIM+1;
if (TIM=100) then
X:do;
     TIM=TIM-100;
     call KBOARD_SC;
     if (K2 or (K0 and K4)) then
     Y:do;
          TIM_MS=TIM_MS+20;
          if (TIM_MS=60020) then
          do;
             TIM_MIN=TIM_MIN+1;
             TIM_MS =TIM_MS-60020;
          end;
          if (TIM_MIN=60) then
          do;
             TIM_HOUR=TIM_HOUR+1;
             TIM_MIN =TIM_MIN-60;
          end;
     end Y;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if (K0=0) then
P1: do;
        if (K2=1) then
        P2: do;
                if ((K6+1)=TIM_HOUR) then
                P3: do;
                        K0=1;
                        K2=0;
                        TIM_HOUR=0;
                        goto P4;
                end P3;
                else goto P5;
        end P2;
        else
        do;
           PA_PORT_ADDRESS=0000$0000b;
           goto P5;
        end;
end P1;
else
P4: do;
        if (K2=0) then
        do;
           TEMP_SET=24;
           call FAN_CHECK(24);
        end;
end P4;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if (K3=1) then TEMP_SET=25;
if (K3=2) then TEMP_SET=26;
if (K3=3) then TEMP_SET=27;
if (K3=4) then TEMP_SET=28;
if (K3=5) then TEMP_SET=21;
if (K3=6) then TEMP_SET=22;
if (K3=7) then TEMP_SET=23;
if (K3=0) then TEMP_SET=24;

call FAN_CHECK(TEMP_SET);

if (K4=1) then
if ((K6+1)=TIM_HOUR) then
do;
   PA_PORT_ADDRESS=0000$0000b;
   (K0,K1,K2,K3,K4,K6)=0;
end;

end X;
P5:end TIMER_INT;
end MAIN;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

การนำ Micropeocessor มาควบคุมเครื่องปรับอากาศ

Fcaturea หรือ ฟังก์ชั่น (Frnetion) ในการควบคุมเครื่องปรับอากาศ

1. ตั้งเวลาให้เครื่องปรับอากาศ ทำงานหรือหยุดทำงานได้ ภายในเวลา 1–2 ชม.
2. ตั้งอุณหภูมิห้องได้ 18–32°C
3. ควบคุมความเร็วของพัดลมภายในห้องได้ (Indoor Fan)
4. ตั้งโหมด (Mode) ได้ ซึ่งแบ่งเป็น
   - 4.1 โหมดการทำงาน (OPERATION Mode)
     - นอนหลับ (SLEEP) → ปกติ (NORMAL)
   - 4.2 ฟังก์ชั่นโหมด (Function Mode)
     - Cooling, Day, Auto
5. เช็ค (Check) เวลาทำงานคือถ้าทำงานติดต่อกันเกิน 12 ชม. แล้ว เครื่องปรับอากาศจะต้องหยุดการทำงานของตัวมันเอง

การทำงานทั้งหมดจะผ่านทาง Remote Control สำหรับรายละเอียดของแต่ละฟังก์ชั่น มีดังนี้

การตั้งเวลาให้เครื่องปรับอากาศเริ่มทำงาน

– เมื่อส่วนควบคุมเช็คที่ Remote มีการตั้งเวลาเริ่มทำงานของเครื่องปรับอากาศ โดยตั้งได้ตั้งแต่ 1 ชม. ถึง 12 ชม. แล้วส่วนควบคุมจะคอยตรวจสอบเวลานี้ทุก ๆ 1 วินาที พร้อมทั้งตรวจสอบว่ามีการเปลี่ยนแปลงที่ Remote หรือเปล่า เช่น อาจเปลี่ยนให้เครื่อง Remote ทำงานแล้วเช็คเวลาหยุดลักษณะนี้ส่วนควบคุมที่จะสั่งให้เครื่องปรับอากาศทำงานเลยแล้วคอยตรวจสอบเวลาหยุด

สำหรับช่วงที่ตั้งเวลาให้เครื่องปรับอากาศ เริ่มทำงานนี้ ส่วนควบคุมที่จะมีหน้าที่เพียงแค่ตรวจสอบเวลาเท่านั้นว่าครบกำหนดหรือยัง เมื่อใดที่ถึงกำหนดเวลาที่ตั้งไว้แล้วนั้น ส่วนควบคุมก็จะไม่สนใจเวลาที่ตั้งไว้แล้ว จะสั่งงานตามฟังก์ชั่นต่าง ๆ ทันที

การตั้งเวลาให้เครื่องปรับอากาศหยุดทำงาน

ลักษณะนี้จะตรงข้ามกับข้างต้น เพราะเมื่อจ่ายไฟฟ้า (Power Supply) เข้าเครื่องแล้วมันจะทำงานตามฟังก์ชั่นอื่น ๆ ที่ตั้งไว้ ส่วนควบคุมจะตรวจสอบรีโมท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และเวลาการทำงานของเครื่องเมื่อใดก็ตามที่ถึงเวลาปิด (OFF) เครื่องปรับอากาศแล้วส่วนควบคุมที่จะตัด Supply ของตัวเอง

การตั้งอุณหภูมิห้อง

– ที่ Remote จะมีสวิทย์ตั้งอุณหภูมิได้ตั้งแต่ 18 – 32 °C ส่วนควบคุมจะทำการอ่านอุณหภูมิห้องขณะนั้น แล้วนำมาเปรียบเทียบกับค่าที่ผู้ใช้ตั้งไว้ถ้าอุณหภูมิห้องต่ำกว่าอุณหภูมิที่ผู้ใช้ตั้งไว้ ส่วนควบคุมจะคอยตรวจสอบอุณหภูมิอยู่ตลอดเวลา เมื่อใดก็ตามที่อุณหภูมิห้องสูงกว่าอุณหภูมิที่ตั้งไว้ ส่วนควบคุมจะทำการควบคุมการ ON OFF ของ คอมเพรสเซอร์เพื่อรักษาหรือปรับให้อุณหภูมิห้องเท่ากับอุณหภูมิที่ผู้ใช้ตั้งไว้

การควบคุมความเร็วของพัดลม

– ในการควบคุมความเร็วของพัดลมนั้น จะเป็นไปตามความเร็วที่ผู้ใช้ต้องการโดยส่วนควบคุมจะทำการอ่านค่ามาจาก Remote ในกรณีที่ผู้ใช้ตั้งไว้ที่อัตโนมัติ (Auto) ส่วนควบคุมจะอ่านค่าอุณหภูมิห้องขณะนั้น และอ่านค่าอุณหภูมิที่ผู้ใช้ตั้งนำมาเปรียบเทียบกัน และตัดสินใจว่าจะควบคุมความเร็วพัดลมไว้ที่ตำแหน่งใด

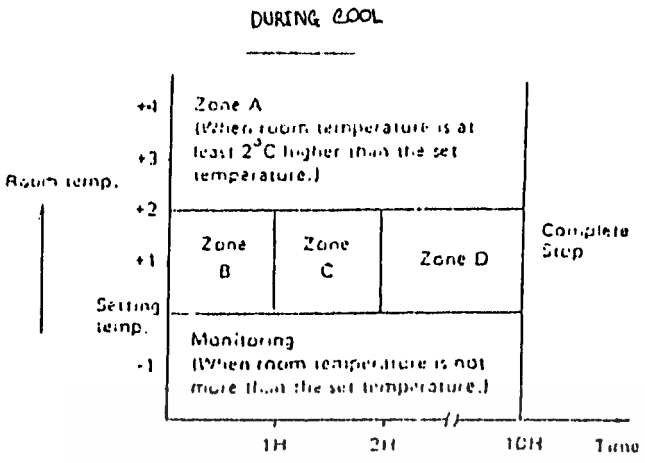
โหมดการทำงาน (Operetion Mode)

– การควบคุมเครื่องปรับอากาศผู้ใช้สามารถตั้งโหมดการทำงานได้ 2 โหมด คือ ปกติ (Normal Mode) และ นอนหลับ (Sleep Mode) ส่วนควบคุมจะทำงานตามที่ผู้ใช้ตั้งไว้ สำหรับ Sleep Mode ส่วนควบคุมจะอ่านค่าอุณหภูมิห้องขณะนั้น และอุณหภูมิที่ผู้ใช้ตั้งไว้ และนำมาควบคุมการ ON-OFF ของคอมเพรสเซอร์ให้เหมาะสมกับเวลาที่ผ่านไป และส่วนควบคุมจะตรวจสอบสวิทย์ Operation ว่าตั้งอยู่ที่ตำแหน่งใด ซึ่งมีความสัมพันธ์กับการทำงานของคอมเพรสเซอร์ด้วย ดังรายละเอียดข้างล่าง

1. เมื่อสวิทย์ตั้งที่ Sleep Mode

– Cool เมื่อสวิทย์อยู่ที่ตำแหน่ง Cool การทำงานจะเป็นดังตารางที่ 2 จากตารางที่ 1 จะเห็นว่าส่วนควบคุมจะแบ่งการทำงานของคอมเพรสเซอร์เป็นโซนกำหนดระยะเวลา ON OFF คอมเพรสเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Zone	Lapse from Start	Interval for Compressor and Fan ON-OFF	Indoor Fan Speed
A	0 to 10 hours	18–3 minutes	AUTO position: Low speed (Constant)
B	0 to 1 hour	3–4 minutes	
C	1 to 2 hours	3–7 minutes	Other position: Desired speed (Constant)
D	2 to 10 hours	3–10 minutes	

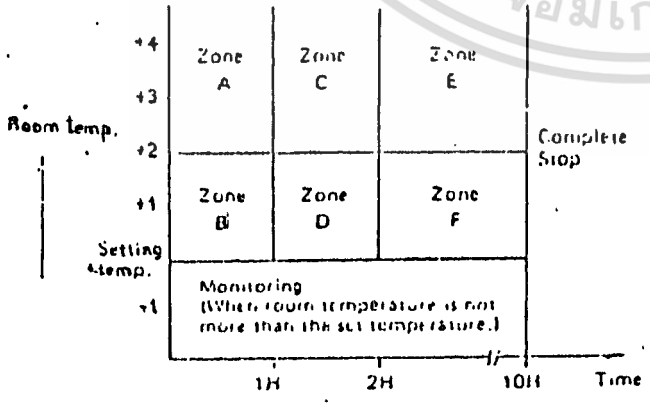
ตารางที่6-1

แสดงการทำงานของการควบคุม คอมเพรสเซอร์

เช่น Zone A จะ ON คอมเพรสเซอร์ 18 นาที และOFF คอมเพรสเซอร์ 3 นาที ส่วนควบคุมจะ คอยตรวจสอบการทำงานของระบบด้วย ถ้าระบบทำงานติดต่อกันเกิน 10 ชม. ส่วนควบคุมจะหยุดการทำงานทั้งระบบ การควบคุมพัดลมภายในห้อง ถ้าสวิทย์ควบคุมตั้งอยู่ที่ AUTO พัดลมจะทำงานที่ LOW ถ้าตั้งตำแหน่งอื่นพัดลมจะทำงานตามตำแหน่งนั้น

– Dry

เมื่อ Switch Operation ตั้งไว้ที่ตำแหน่ง (Dry) ส่วนควบคุมจะแบ่งการทำงานของเครื่องปรับอากาศเป็น 6 โซน ดังตารางที่6-2



Zone	Lapse from start	Interval for Compressor and Fan ON-OFF	Indoor Fan Speed
A	0 to 1 hour	5–4 minutes	Low speed (Constant)
B		3–4 minutes	
C	1 to 2 hours	5–7 minutes	
D		3–7 minutes	
E	2 to 10 hours	5–10 minutes	
F		3–10 minutes	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่6-2 การทำงานของการควบคุม คอมเพรสเซอร์ ในโหมด Dry

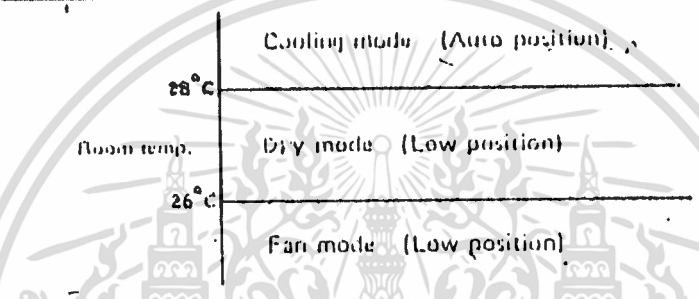
– Automatic Operation

เมื่อสวิทย์ตั้งไว้ที่ Auto การทำงานของคอมเพรสเซอร์ จะขึ้นกับอุณหภูมิห้องเพียงอย่างเดียวโดย ส่วนควบคุม จะไม่สนใจค่าอุณหภูมิที่ตั้งไว้ ส่วนควบคุมจะเข้าไปทำงานในโหมดต่าง ๆ เอง ตารางที่ 6-3 อุณหภูมิที่ส่วนควบคุม กำหนดมีดังนี้

เมื่ออุณหภูมิห้องสูงกว่า 28 °C ส่วนควบคุมจะเข้าไปทำงานใน COOL MODE

เมื่ออุณหภูมิห้องอยู่ที่ 26 °C — 28 °C ส่วนควบคุมจะทำงานใน DRY MODE

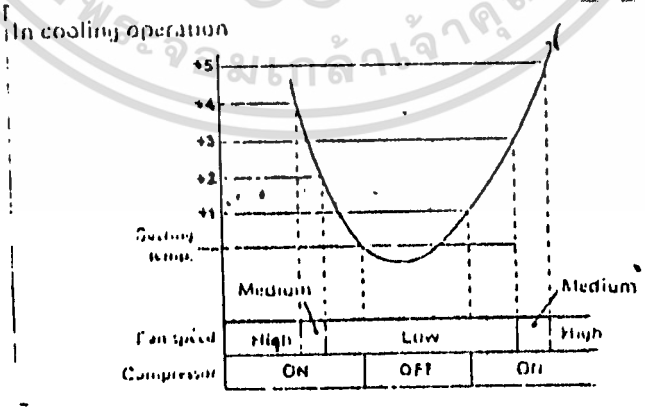
เมื่ออุณหภูมิห้องอยู่ต่ำกว่า 26 °C ส่วนควบคุมจะทำงานใน FAN MODE



ตารางที่ 6-3

– Automatic Fan Speed Control

ในโหมดนี้ ส่วนควบคุมคอมเพรสเซอร์และพัดลมภายใน



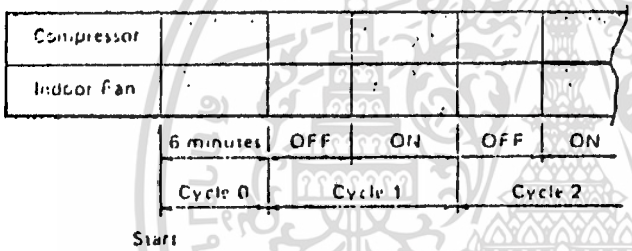
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6-4 แสดงการทำงานของเครื่องปรับอากาศใน Automatic Fan Spped

2. เมื่อตั้งสวิทย์ที่ Normal Mode

– ตั้ง Operation ไว้ที่ Dry Mode

การทำงานในโหมดนี้ ส่วนควบคุม จะควบคุมภายในห้องมีความชื้นต่ำในขณะที่ยังคงรักษาอุณหภูมิห้องให้คงที่ เมื่อส่วนควบคุม ON คอมเพรสเซอร์ พัดลมภายในก็จะทำงานเหมือนกับคอมเพรสเซอร์ ช่วงเวลาของการ ปิด – เปิด คอมเพรสเซอร์ จะขึ้นกับค่าอุณหภูมิห้องและอุณหภูมิที่ตั้งไว้ ซึ่งแสดงความสัมพันธ์คือตารางที่ 6-5 ตัวอย่างเช่น เมื่ออุณหภูมิห้อง (TA) ต่ำกว่า อุณหภูมิที่ผู้ใช้ตั้งไว้ ส่วนควบคุมจะทำการ on คอมเพรสเซอร์และพัดลมภายในเป็นเวลา 3 นาที แล้วจะ OFF เป็นเวลา 6 นาที



Relation between room temperature (TA) and set temperature (TS)	Compressor and Indoor Fan	
	OFF	ON
TA > TS+2	4 minutes	6 minutes
TS+2 > TA > TS	5 minutes	5 minutes
TS > TA	6 minutes	3 minutes

ตารางที่ 6-5 แสดงการทำงานของ Dry Mode

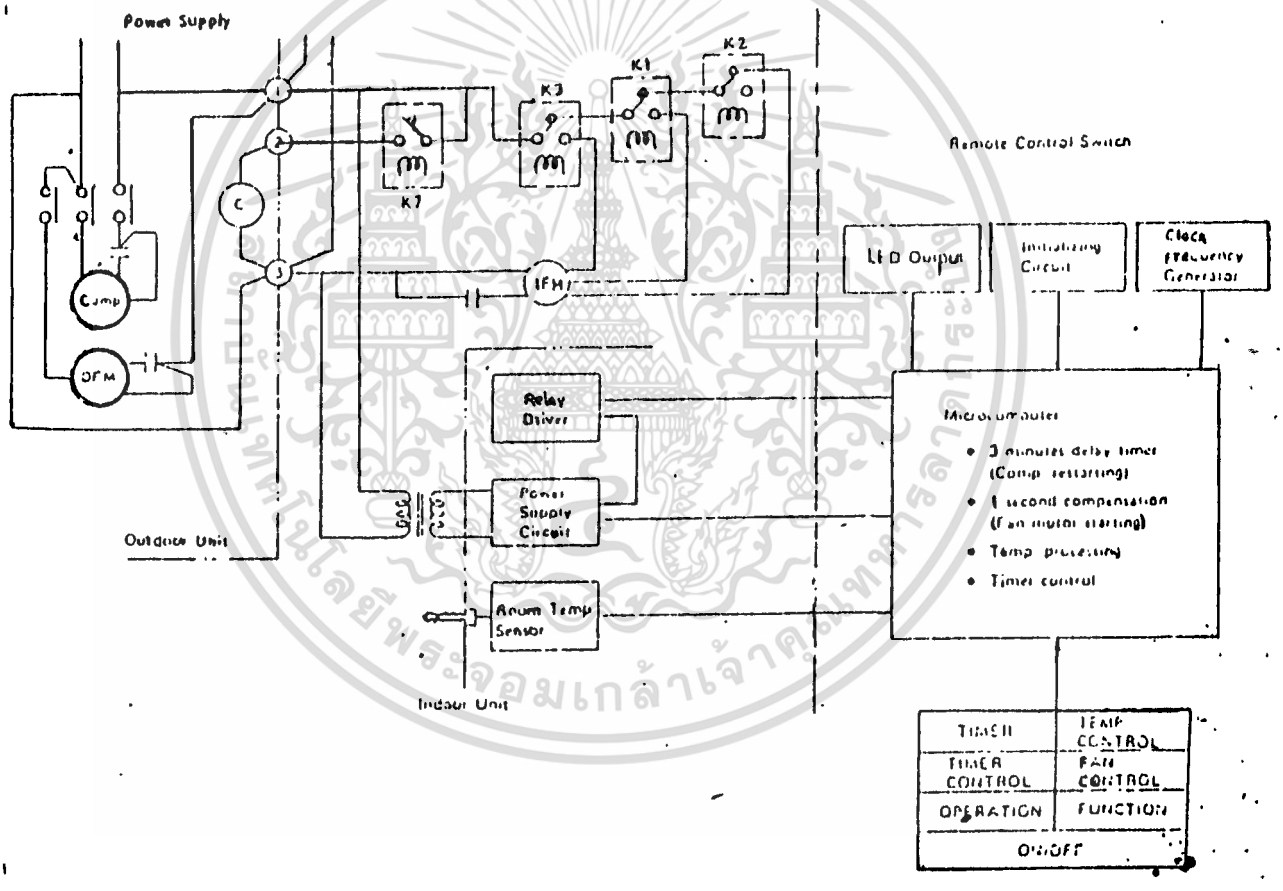
สำหรับความเร็วของพัดลมภายในนั้น จะถูกกำหนดให้ทำงานในสปีด LOW ซึ่งไม่ขึ้นกับสวิทย์ที่ตั้งไว้

ถ้าอุณหภูมิห้อง TA ลดต่ำกว่า 15°C แล้วส่วนควบคุมจะ OFF คอมเพรสเซอร์แต่สำหรับพัดลม ภายในนั้น จะทำงานที่ ความเร็วสูง (High Speed) เพื่อป้องกันไม่ให้อุณหภูมิลดต่ำมากเกินไป และเมื่ออุณหภูมิห้องเพิ่มขึ้นถึง 18°C แล้ว การทำงานของคอมเพรสเซอร์ และพัดลมภายใน จะกลับมาอยู่ในไซเคิล ศูนย์ (Cycle 0)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

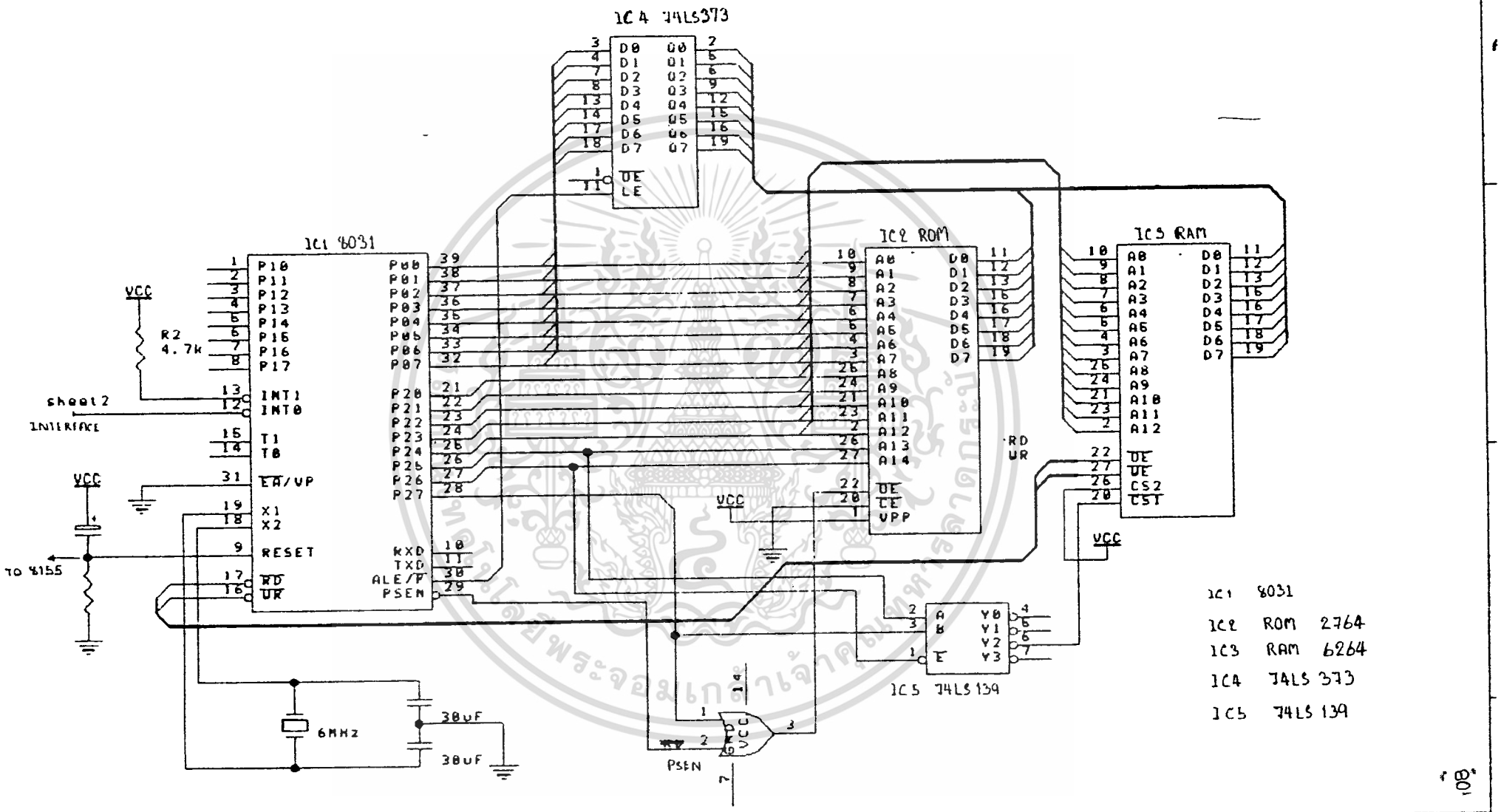
โปรแกรมควบคุม

การควบคุมการทำงานในส่วนต่าง ๆ นั้น จำเป็นต้องใช้โปรแกรมการควบคุม ซึ่งโปรแกรมต่างๆ นั้น จะถูกบรรจุภายใน ROM ส่วนควบคุมจะทำงานตามคำสั่งนั้น ดังนั้นโปรแกรมจึงเป็นส่วนสำคัญอย่างมากที่ต้องออกแบบและวางแผนงานในการเขียนโปรแกรมเพื่อให้ระบบทำงานได้อย่างมีประสิทธิภาพ โปรแกรมจะเป็นตัวกำหนดว่าส่วนควบคุมจะทำการควบคุมอุปกรณ์ใดบ้างจากรูปแสดง บล็อกไดอะแกรม ที่ส่วนควบคุมต้องทำการติดต่อด้วย

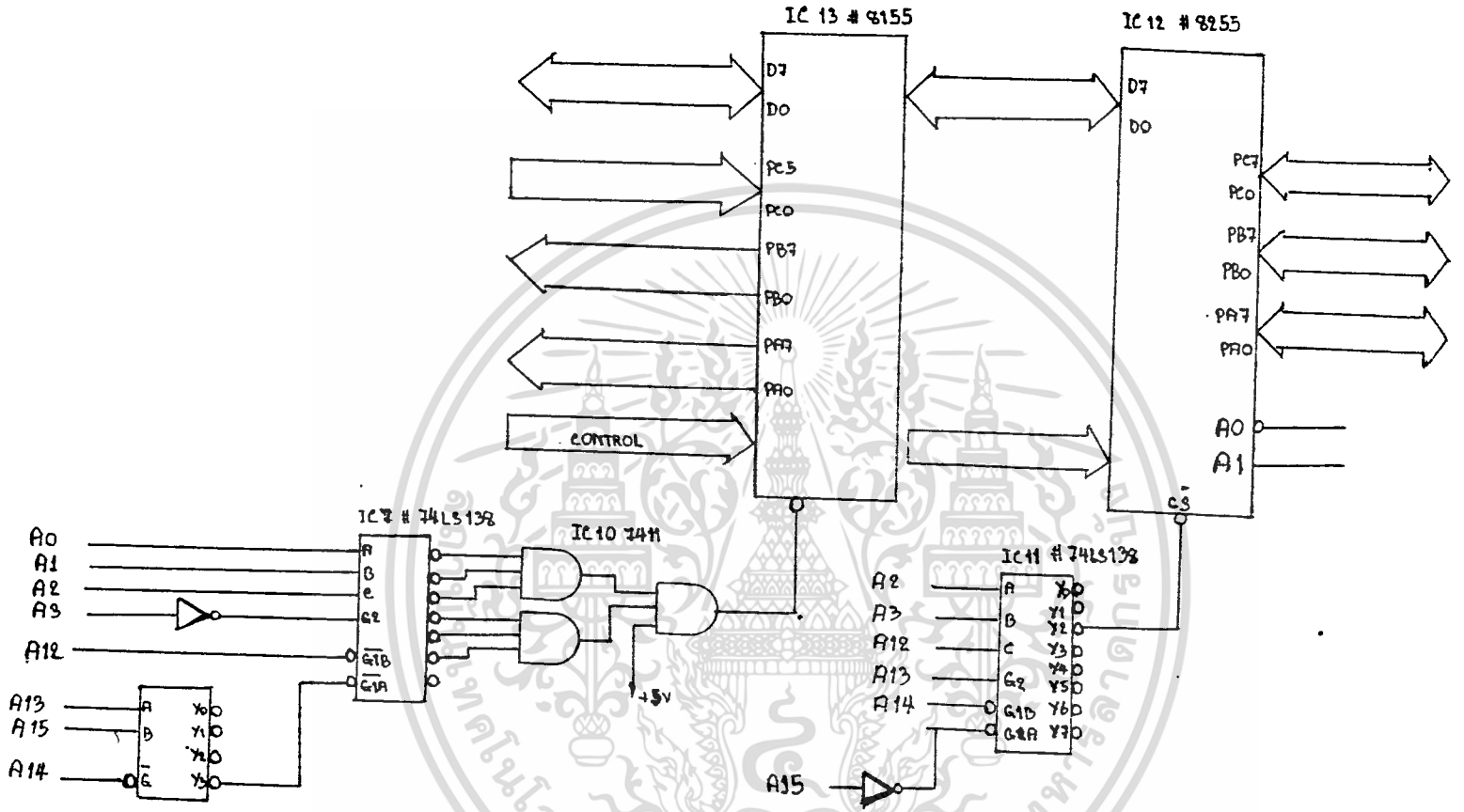


แสดงบล็อกไดอะแกรมของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

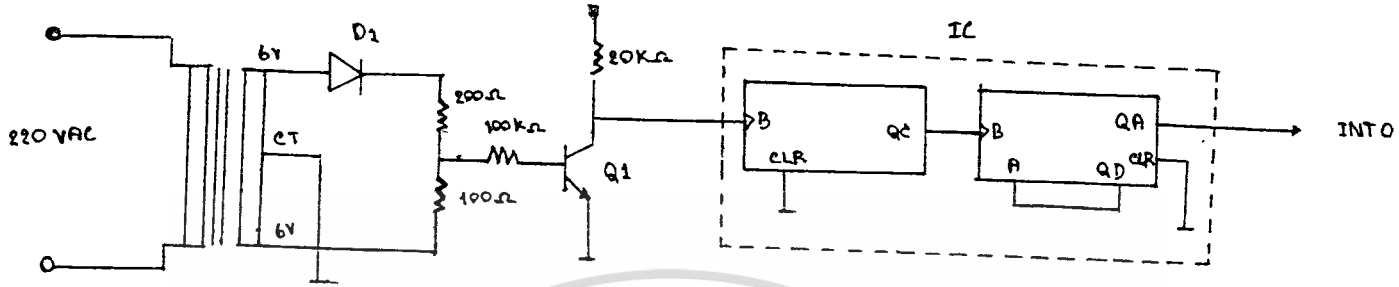


รูป 6-6 วงจรของสัญญาณควบคุม 8031

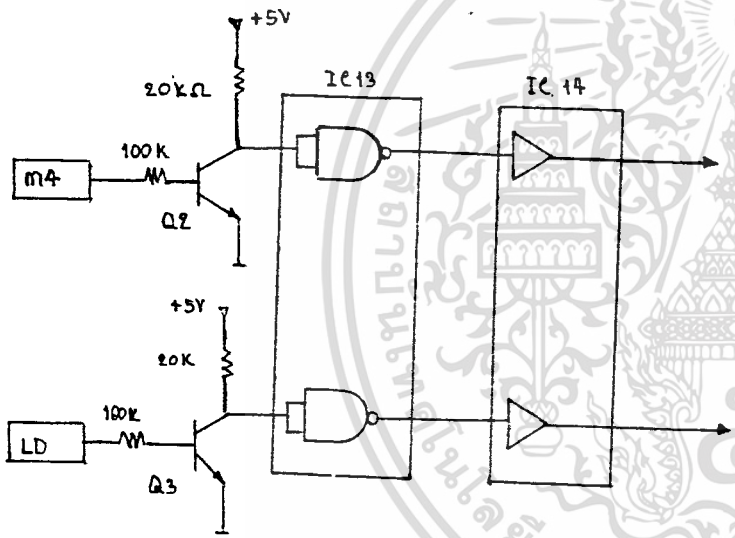


รูปที่ 3 วงจรส่วนของสัญญาณ INPUT / OUT PUT

IC 7	#	74LS138
IC 10		7411
IC 11		74LS138
IC 12		8255
IC 13		8155



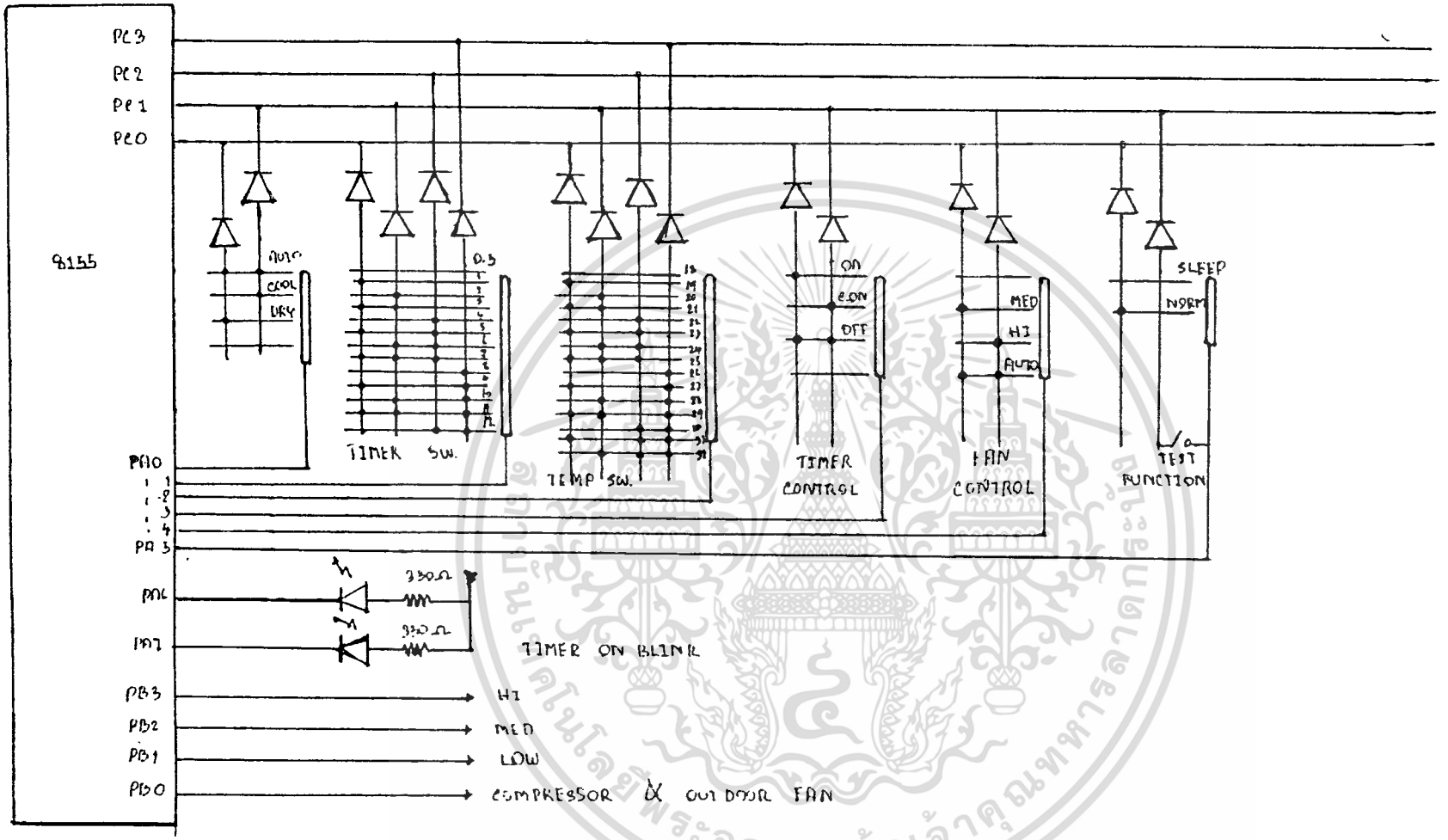
รูป 6-วงจรสร้างสัญญาณ INTERRUPT



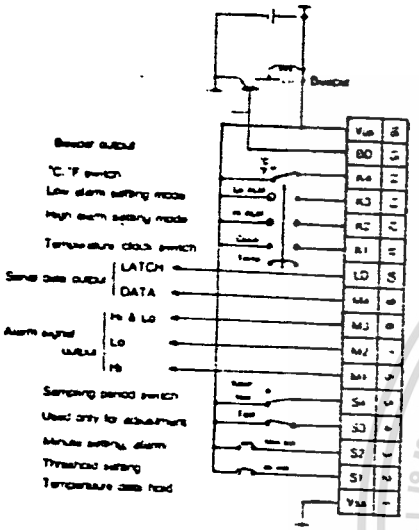
รูป 6-วงจรสร้างชุดสัญญาณขัดจังหวะ

Q_1, Q_2, Q_3	#	BC 548
D1		1N4001
IC 13		7400
IC 14		74LS 224
IC		74390

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



รูป 6-10 วงจร Remote SWITCH



รูปที่ 6.11

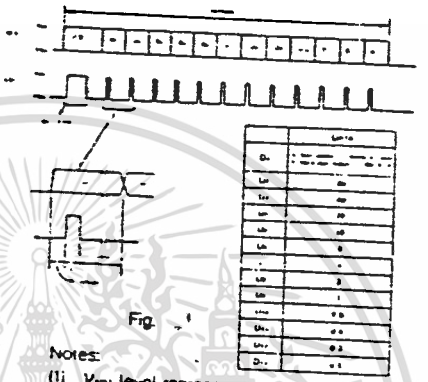
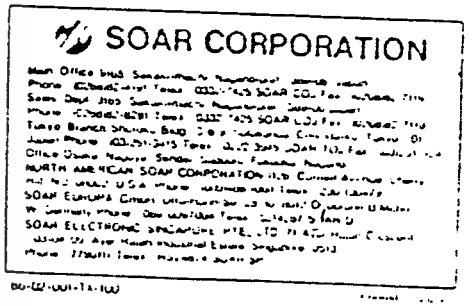


Fig.

Notes:
(1) V_{SH} level represents 1. V_{SS} level represents 0.
(2) A9 data are 1 when the temperature is outside the measurable range

รูปที่ 6.12



รูปที่ 6.13 แสดงข้อมูลอุณหภูมิ ที่ TX-100 ส่งออกมา

รูปที่ 6.14 แสดงการตั้งโหมด การทำงานของ TX-100

วงจรที่ได้ออกแบบเพื่อใช้ในการควบคุมเครื่องปรับอากาศซึ่งแบ่งออกเป็น 2 ส่วนคือ ส่วน ซีพียู และส่วนอินพุทเอาท์พุท

ส่วนซีพียู ประกอบด้วย วงจรหน่วยความจำ (Memory) ซึ่งมีทั้งแรม(Ram) และรอม(Rom) วงจร ดีโคด (Decode) หน่วยความจำ , วงจรดีโคดพอร์ท , วงจรดีโคดสัญญาณควบคุม

ส่วนอินพุท เอาท์พุท ประกอบด้วย ซีพีพีพอร์ท เบอร์ 8155, วงจรสวิทย์รีโมท, วงจรกำเนิดสัญญาณ อินเตอร์รีพท์ และวงจรที่ทำหน้าที่อินเตอร์เฟสกับชุดอ่านอุณหภูมิ TX-100

รูปที่ 6-6 แสดงวงจรของ ส่วน ซีพียู ซึ่งสามารถแยกอธิบายได้ดังนี้

Ic 1 คือ CPU 8031 ทำงานที่ความถี่ 6MHZ

IC 3 คือ RAM สามารถ เลือกใช้ได้ 2 เบอร์ คือ 6116(2KB) และ 6264 (8KB) ทำหน้าที่ในเก็บข้อมูลไว้ชั่วคราว เช่น ผลการคำนวณ, การเก็บข้อมูลเกี่ยวกับเวลา และสถานะของเครื่องปรับอากาศ

TC ทำหน้าที่เป็นดีมัลติเพลก (Demultiplex) โดยจะแสดงค่าแอดเดรส-ไบท์ต่ำๆไว้เมื่อมีสัญญาณ ALE (Addresslatchenable) ออกมาจาก ซีพียู ซึ่งใช้ไอซี เบอร์ 74LS373

IC 5 ทำหน้าที่ ดีโคดสัญญาณ เลือก เมมโมรี่ (Select Memory) ใช้ไอซี เบอร์ 74139 (Dual 1 of 4 Decoder) โดยได้กำหนดตำแหน่งของเมมโมรี่ไว้ดังนี้

ROM =	2716	เริ่มจาก	0000	ถึง 03FFH	2KB
	2732	เริ่มจาก	0000	ถึง 07FFH	4KB
	2764	เริ่มจาก	0000	ถึง 0FFFH	8KB
	27128	เริ่มจาก	0000	ถึง 1FFFH	16KB
ROM	27256	เริ่มจาก	0000	ถึง 7FFFH	(32KB)
RAM =	6116	เริ่มจาก	8000H	ถึง 83FFH	(2KB)
	6264	เริ่มจาก	8000H	ถึง 9FFFH	(8KB)

IC 7 ทำหน้าที่ ดีโคด สัญญาณควบคุมพอร์ทใช้ไอซีเบอร์ 74LS138 (1 of 8 Decoder Multiplex)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อไปจะกล่าวถึงวงจรที่เป็นส่วนอินพุท เอาท์พุท ของวงจรที่สมบูรณ์ดังแสดง

ในรูปที่ 6-5 ซึ่งไอซีแต่ละตัวจะทำหน้าที่ ดังต่อไปนี้

IC 10 ทำหน้าที่ ช่วยในการดีโคด 8155 เพราะพอร์ท 8155 ใช้นั้นมีจำนวนถึง 6 พอร์ท ซึ่งไอซี 10 ใช้เบอร์ 7411 (Triple 3 Input And Gate)

IC 11 ทำหน้าที่ ดีโคดพอร์ทเช่นกัน ออกแบบไว้เพื่อการขยายระบบซึ่งเตรียมไว้กับ ชิพ ซัพพอร์ท เบอร์ 8255 ใช้ไอซี เบอร์ 74138

และสุดท้ายคือ IC 12 เป็น ชิพ ซับพอร์ท 8255 ออกแบบเผื่อไว้โดยใช้เบอร์ทั้งหมดถึงตารางที่ 6-14

PORT	หน้าที่
00	CONTROL PORT 8155
01	PORT A
02	PORT B
03	PORT C
04	8 LSB TIMER COUNTER
05	6 MSB TIMER COUNTER

ตารางที่ 6-13 แสดงเบอร์พอร์ทที่ใช้ในการควบคุม 8155

PORT	หน้าที่
10	PORT A
11	PORT B
12	PORT C
13	CONTROL PORT 8255

ตารางที่ 6-14 แสดงเบอร์พอร์ทใช้กับ 8255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อไปจะกล่าวถึง ส่วนที่กำเนิดสัญญาณอินเตอร์รัพท์ ซึ่งวงจรสมบูรณ์ ดังรูปที่ 6-8 การทำงานก็คือ นำสัญญาณไฟฟ้า กระแสสลับมาเร็คติไฟร์ (Rectifier) แบบฮาร์ฟเวฟ (Half wave) แล้วจัดสัญญาณให้อยู่ในรูปของสัญญาณดิจิตอลจากนั้นผ่านวงจรนับ 50 (counter) เพื่อกำเนิดสัญญาณอินเตอร์รัพท์ทุก ๆ 1 วินาที

วงจรนับ 50 นั้นใช้ไอซี 74390 (Dual Decodc Counter) โดยตัวแรกจัดให้นับ 5 และ เอาท์พุทของตัวแรก ถูกนำมาเปรียบเทียบให้ตัวต่อไปนับ 10 อีกครั้ง

ส่วนที่ติดต่อกับชุดอ่านอุณหภูมิ TX-100 คือ วงจรรูปที่ 4 Q2 และ Q3 (Bc548) ทำหน้าที่ช่วยขยายสัญญาณ ที่ TX-100 ส่งออกมาเพื่อให้เหมาะสมกับระดับดิจิตอล ส่วนหนึ่ง ของ IC 13 ใช้เบอร์ 7400 (Dual Nand Gate) ต่อวงจรให้เป็น Not Gate เมื่อ Inverse สัญญาณ G2 และ G3 ให้กลับมาเหมือนเดิม เอาท์พุทจากไอซี 13 จะถูกส่งผ่านไอซี 14 เบอร์ 74244 (Octal Buffer/Line Driver,3 State outptut) เพื่อเป็นบัฟเฟอร์ (Buffer) ก่อนที่จะต่อกับพอร์ท C ของ 8155

สำหรับชุดอ่านอุณหภูมิ TX-100 ผลิตโดยบริษัท Sola จะมีเทอร์มินัล (Terminal) ออกมา 16 ขา เราสามารถใช้สัญญาณ M4 และ LD คือ ค่าอุณหภูมิขณะนั้นโดยจะส่งสัญญาณอนุกรม (Serial) ขนาด 13 บิท เป็นเลข บีซีดี (BCD) ดังแสดงในรูปที่6-9ช่วงเวลาของการส่งข้อมูลออกมาสามารถเลือกได้ 2 ช่วงคือ ทุกๆ 1 วินาที และทุก ๆ 10 วินาที โดยนำขาที่ 5 ต่อกับไฟบวก 1.5 v หรือขา 16

TX-100 ไม่เพียงแต่จะเป็นเครื่องอ่านอุณหภูมิได้เท่านั้น ยังสามารถตั้งอุณหภูมิต่ำสุดหรืออุณหภูมิสูงได้ และเมื่ออุณหภูมิขณะนั้นเท่ากับอุณหภูมิที่ตั้งไว้แล้วTX-100 ที่จะส่งสัญญาณเตือน (Alarm Signal) ออกมาและ TX-100 ยังสามารถวัดค่าของอุณหภูมิออกมาเป็นหน่วยย่อยของ องศาเซลเซียสหรือองศาฟาเรนไฮท์โดยทำการตั้งที่ ขา 14 คือแสดงในรูปที่6-11ซึ่งแสดงการต่อเพื่อตั้งโหมดการทำงานของ TX-100

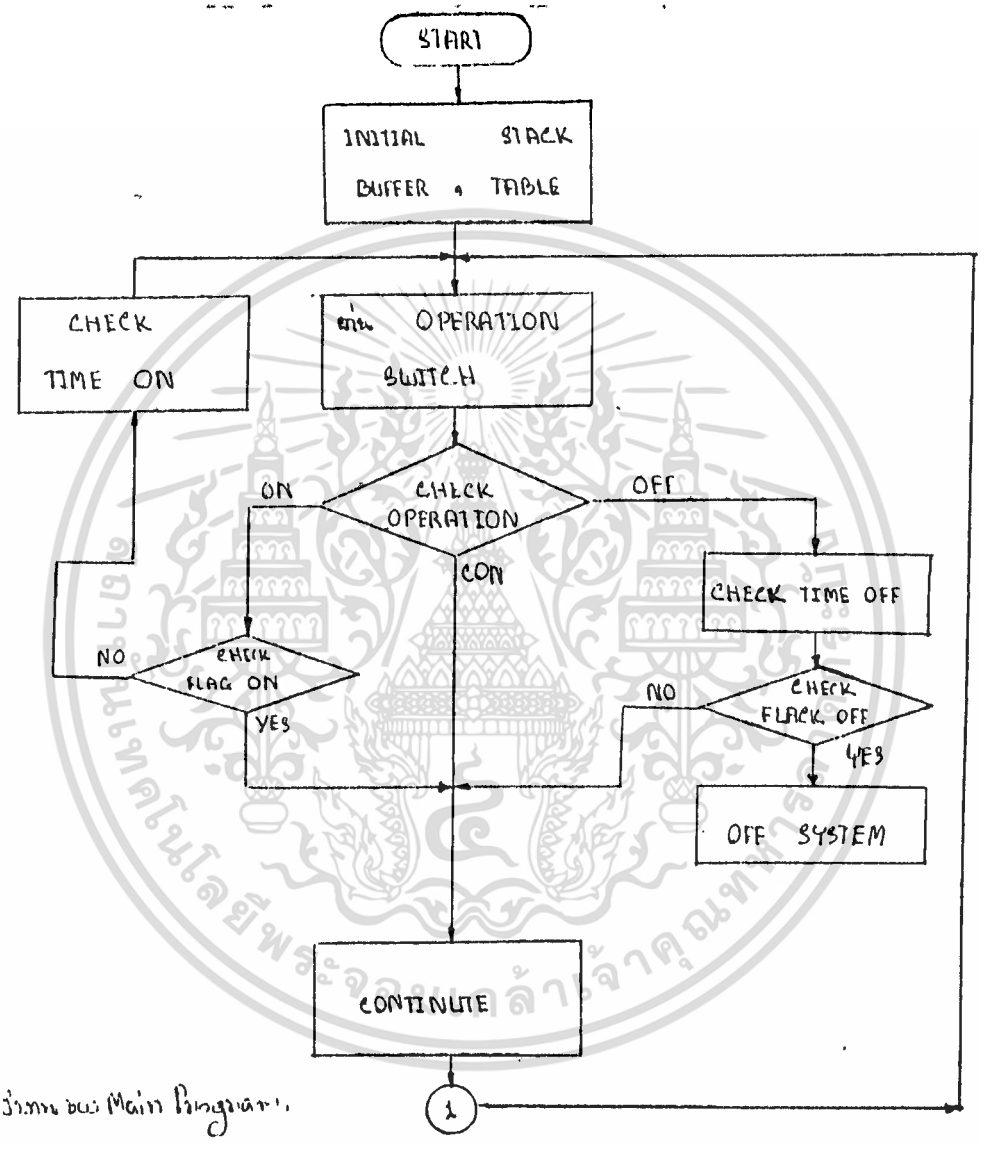
สำหรับที่ในการควบคุมเครื่องปรับอากาศนี้เราตั้งให้ TX-100 ส่งข้อมูลของอุณหภูมิออกมาทุก 1 วินาที และใช้หน่วยเป็นองศา เซลเซียส

วงจรรูปที่6-10เป็นวงจรสมบูรณ์ของรีโมท สวิทย์จะเห็นว่ามีไดโอด(Diode) ทำหน้าที่เอ็นโคด(Encode) พอร์ท A ส่งผ่าน สวิทย์จากนั้นไดโอด จะทำหน้าที่เอ็นโคดเพื่อเป็น อินพุทของพอร์ท C ดังนั้นในการโปรแกรม 8155 นั้นจะกำหนดไว้ว่าให้พอร์ท A และ พอร์ท B เป็นเอาท์พุทพอร์ทส่วนพอร์ท เป็นอินพุทพอร์ท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผังงานของโปรแกรม Flow Chart

1. **เมนโปรแกรม**(Main Proprам)

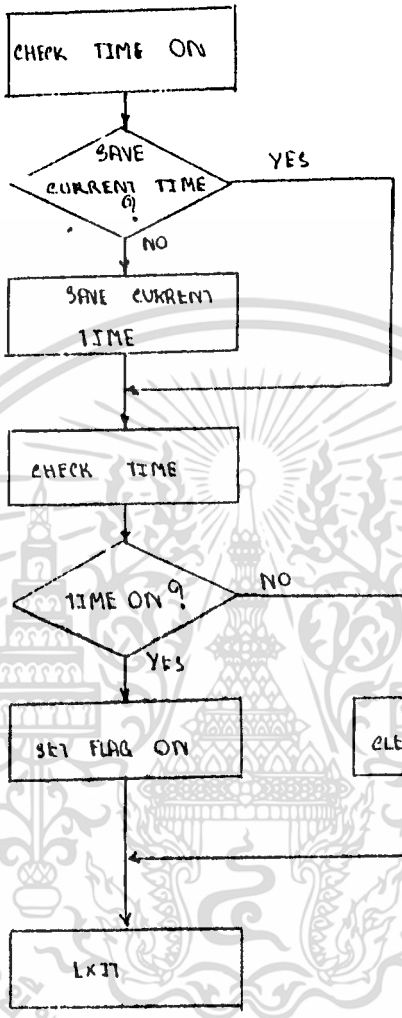


รูปแสดงผังงานของ Main Program

โปรแกรมส่วนนี้จะทำงานเป็นซีเควน (Sequential) จนกว่าจะมีการหยุดระบบ (OFF System) สำหรับส่วนที่ใช้ควบคุมการทำงานจริงของเครื่องปรับอากาศคือส่วนโปรแกรมคอนดินิวท์สวิทย์จะเป็นตัวกำหนดการทำงานในขั้นต้นจากนั้นโปรแกรมคอนดินิวท์จะทำการตรวจสอบตำแหน่งต่างๆ ของสวิทย์อื่นต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. โปรแกรมตั้งเวลาเริ่มทำงาน

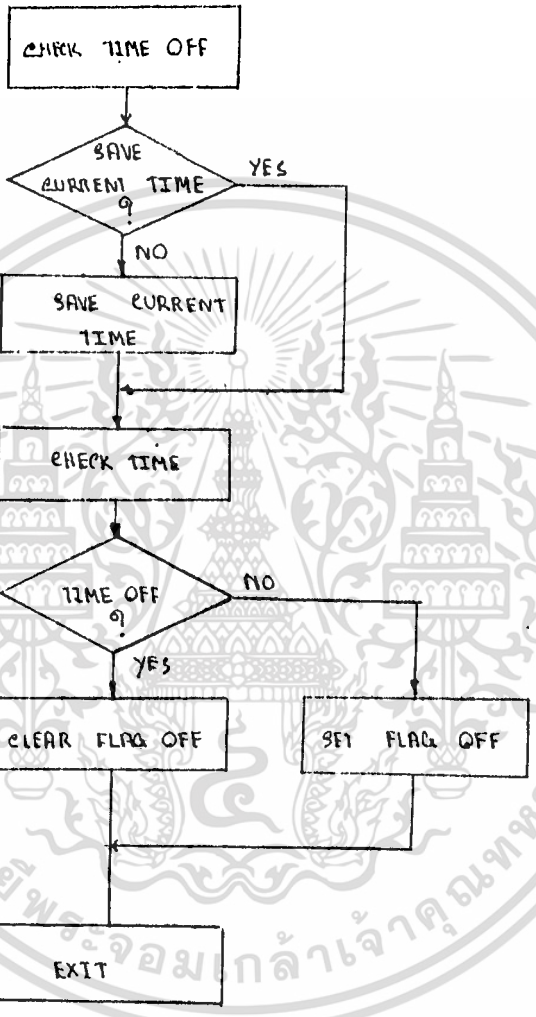


รูปแสดงผังงานของโปรแกรมตั้งเวลาเริ่มทำงาน

เริ่มต้นของโปรแกรมนี้คือการเก็บเลอร์เร็นไทม์ (Current Time) ลงในบัฟเฟอร์และเซ็ตแฟลก ขึ้นมาตัวหนึ่งเพื่อบอกว่ามีการเก็บค่าเวลาไว้แล้วจากนั้นจะเข้าขบวนการเปรียบเทียบว่าผ่านไปกับค่าเวลาที่ผู้ใช้ตั้งไว้กับรีโมทสวิทย์ว่าถึงเวลานั้นหรือยังถ้าถึงเวลาแล้วก็จะเซ็ตแฟลกอีกตัวหนึ่งเพื่อบอกให้เมนโปรแกรมทราบว่าถึงเวลาที่ตั้งไว้แล้วให้เริ่ม ON เครื่องปรับอากาศได้แต่ถ้าไม่ถึงแฟลกตัวนี้ก็จะไม่ถูกเซ็ตแล้วจะกลับเข้าเมนโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

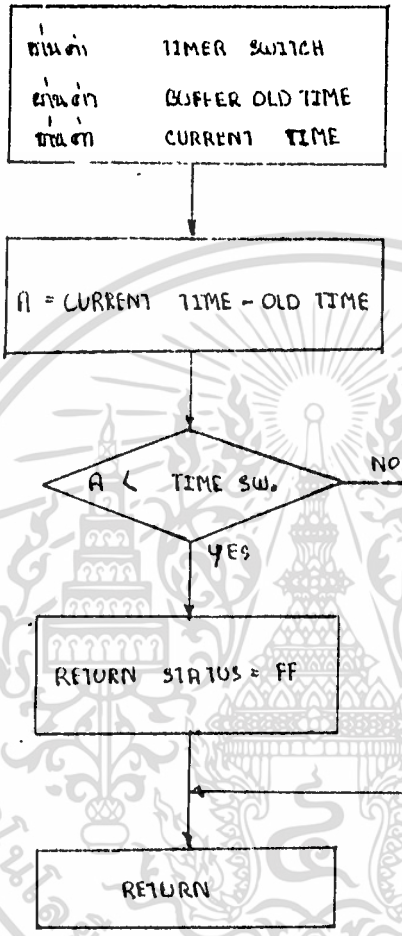
3. <u>โปรแกรมตั้งเวลาหยุดทำงาน</u>



โปรแกรมส่วนนี้คล้ายกับโปรแกรมตั้งเวลาเริ่มทำงานเพียงแต่การกำหนดแฟลก
คนละตัวกันกับโปรแกรมตั้งเวลาเริ่มทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. โปรแกรมตรวจสอบเวลา (CHECR TIME)

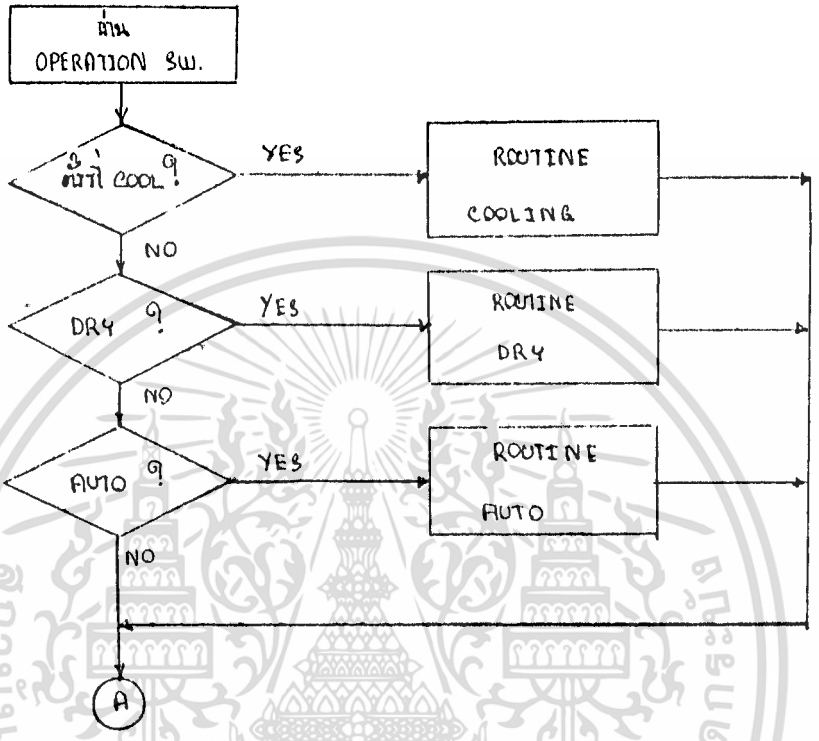


โปรแกรมตรวจสอบเวลานี้จะนำค่าช่วงเวลาที่เปลี่ยนไป (เวลาปัจจุบัน – เวลาเก่า) มาเปรียบเทียบกับช่วงเวลาที่ผู้ใช้ตั้งจากรีโมทสวิทย์ถ้าช่วงเวลาที่ผู้ใช้ตั้งไว้มากกว่าช่วงเวลาที่เปลี่ยนไปโปรแกรมส่วนนี้จะส่งค่า FF กลับให้โปรแกรมที่เรียกแต่ถ้าช่วงเวลาที่เปลี่ยนไปมากกว่าช่วงเวลาที่ผู้ใช้ตั้งโปรแกรมจะส่งค่า 00 กลับ

การส่งค่ากลับในลักษณะนี้จะช่วยให้โปรแกรมที่เรียกใช้โปรแกรมส่วนนี้สะดวกต่อการตรวจสอบสถานะของเวลาได้ดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. โปรแกรม คอนทินิวท์ (Continue)



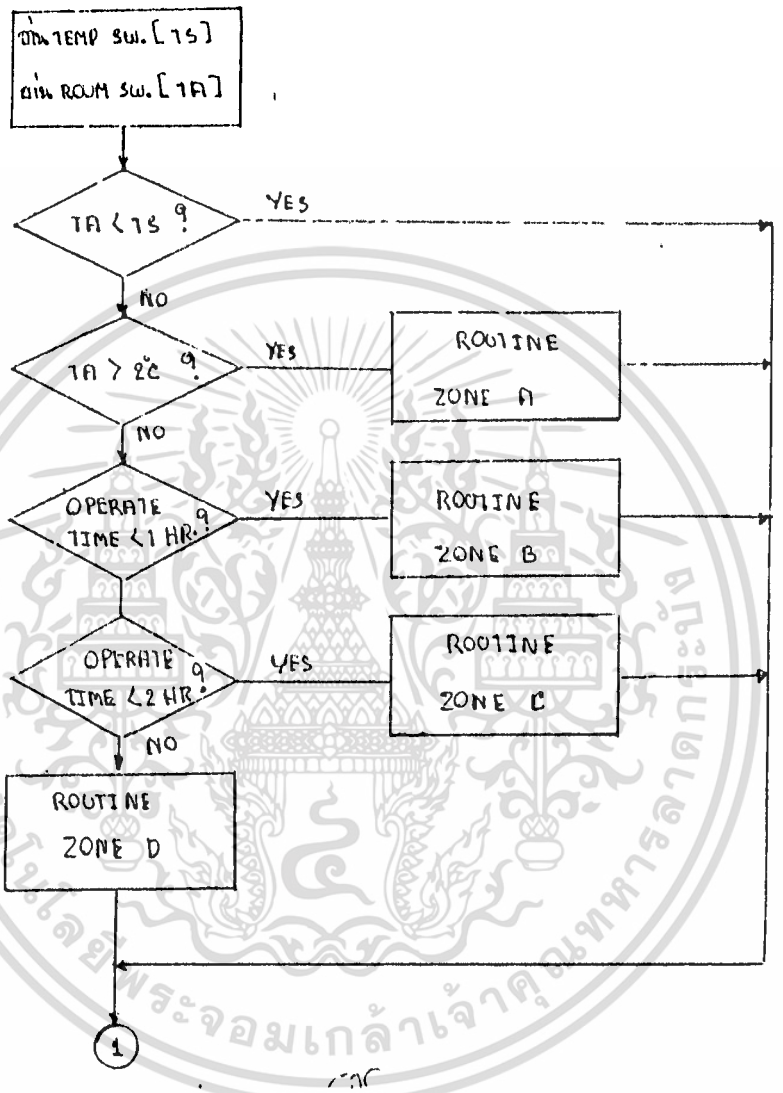
จากรูป แสดงการตรวจสอบ Operation Switch ของ Continue

จากผังงานเมื่อโปรแกรมตรวจสอบโอเปอร์เรชั่นสวิทย์แล้วก็จะกระโดดไปทำซับรูทีน (Subroutine นั้นๆ เมื่อสิ้นสุดการทำงานนี้แล้วก็จะไปเริ่มต้นที่เมนโปรแกรม คือกระโดดไปที่จุดที่ 1 เริ่มตรวจสอบสวิทย์ต่างๆ ว่ามีการเปลี่ยนตำแหน่งหรือเปล่าถ้าไม่มีการเปลี่ยนตำแหน่งแล้ว โปรแกรมต่างๆ ก็จะทำงานตามที่มันเคยทำอยู่เป็นวงรอบจนกระทั่งครบเวลาต่างๆ เช่น ตอนแรกเริ่มมีการ ON ให้คอมเพรสเซอร์ทำงานเมื่อไม่มีการเปลี่ยนสวิทย์ต่างๆ ที่เกี่ยวกับการทำงานของคอมเพรสเซอร์แล้วคอมเพรสเซอร์ก็จะทำงานอยู่ตลอดเวลาจนกระทั่งครบเวลาตามที่ได้กำหนดไว้จึงจะหยุดการทำงานของคอมเพรสเซอร์

การควบคุม คอมเพรสเซอร์นั้นจะเป็นไปตามการที่ผู้ใช้ตงสวิทย์ไว้และพิจารณาความสัมพันธ์ระหว่างอุณหภูมิห้องกับระยะเวลาการทำงานของเครื่องต่อไปจะกล่าวถึงส่วนย่อยๆ ลงไปอีก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. โปรแกรม COOLING

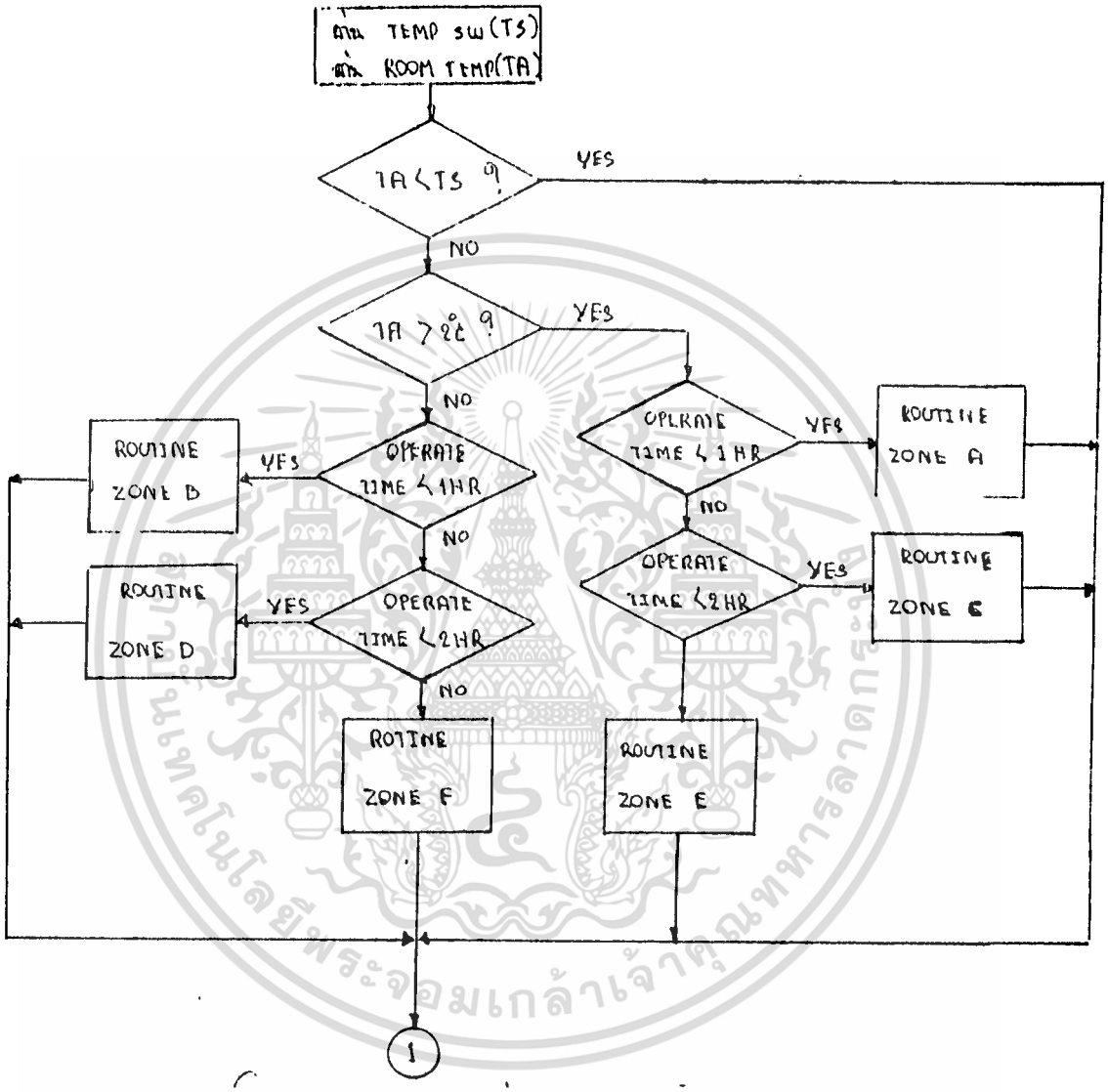


รูปแสดงการแบ่ง ZONE ของเครื่องปรับอากาศในโหมด COOLING

จากผังงานจะเห็นว่ามีการทำงานตามตารางที่ 1 ซึ่งนำความสัมพันธ์ระหว่างเวลาการทำงานของเครื่องและความแตกต่างของอุณหภูมิห้องกับอุณหภูมิผู้ใช้กำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. โปรแกรม DRY

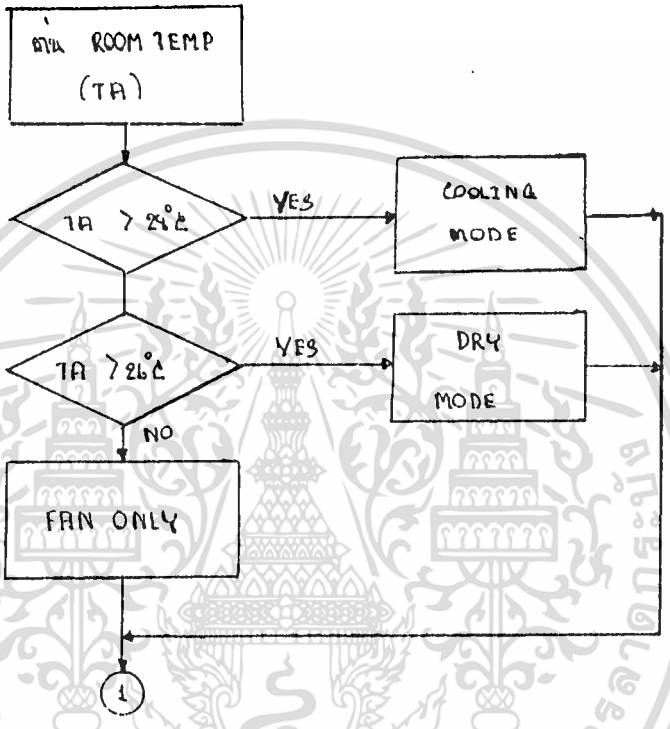


จากรูปแสดงการแบ่งโซนการทำงานของเครื่องปรับอากาศในโหมด DRY

การทำงานตามผังงานนี้จะเป็นไปตามตารางที่ 2 การควบคุมพัดลมภายในจะอยู่ที่ตำแหน่ง LOW คงที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8. โปรแกรม AUTO

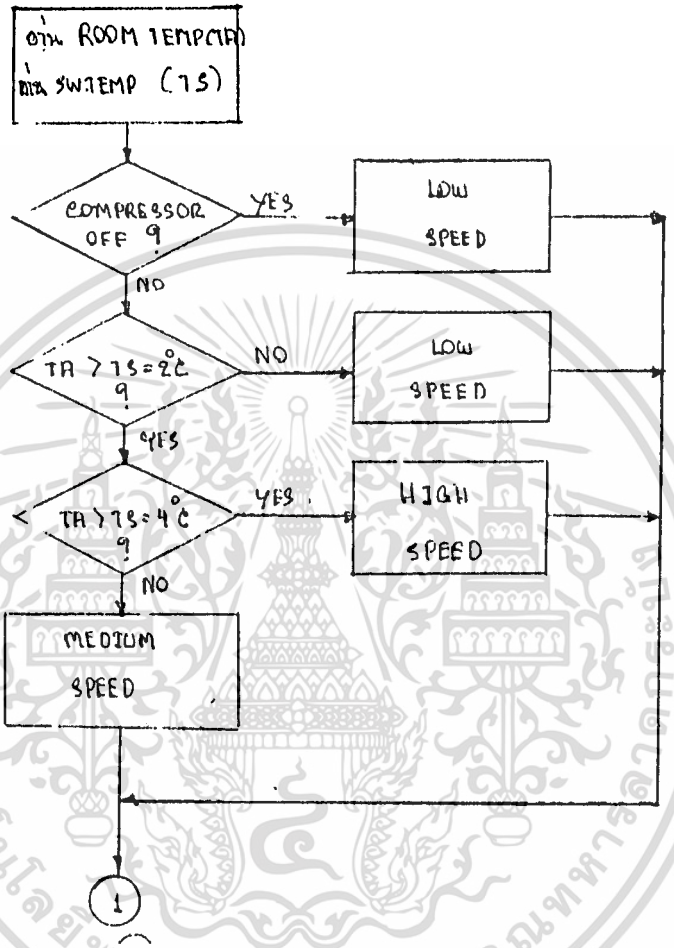


จากรูปแสดงการทำงานในโหมด AUTO

จากผังงานจะเห็นว่าโปรแกรมนั้นจะพิจารณาถึงอุณหภูมิห้องขณะนั้นไม่สนใจว่าค่าอุณหภูมิที่ผู้ใช้ตั้งไว้เมื่ออุณหภูมิห้องอยู่ในย่านอุณหภูมิที่กำหนดไว้โปรแกรมก็จะถึงการทำงานในโหมดต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9. โปรแกรม Automatie Fan Speed Control

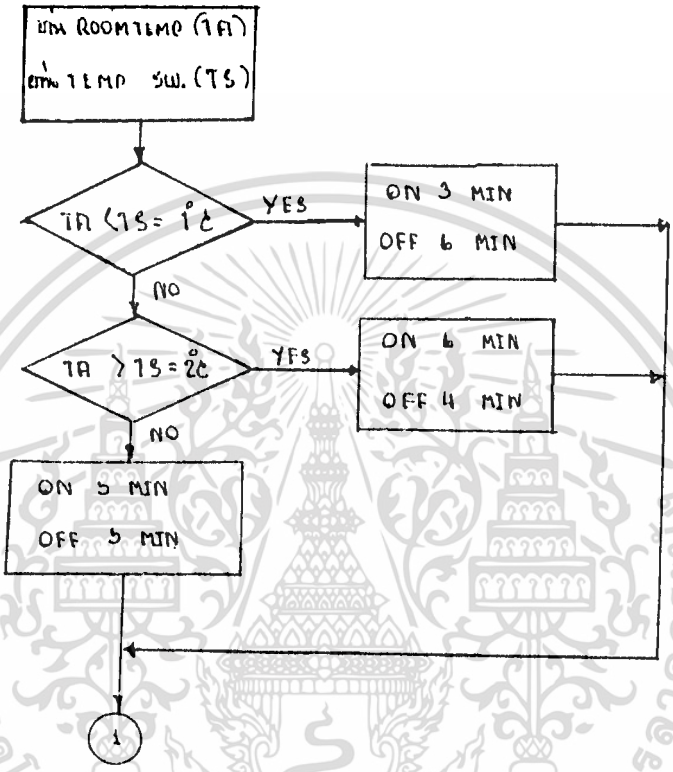


จากรูป – แสดงการควบคุมพัดลมภายในห้อง

จากผังงานจะเห็นว่าเมื่อคอมเพรสเซอร์หยุดทำงานพัดลมภายในจะทำงานที่ LOW SPEED เสมอแต่เมื่อคอมเพรสเซอร์ทำงานความเร็วของพัดลมภายในจะขึ้นกับความแตกต่างของอุณหภูมิห้องและอุณหภูมิที่ผู้ใช้ตั้งไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น ทั้งนี้ต้องขออนุญาตจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังก่อนเท่านั้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

10. โปรแกรม Normal ใน Dry Mode



รูป แสดงการควบคุมเครื่องปรับอากาศในโหมด DRY แบบ NORMAL

จากผังงานการทำงานเครื่องปรับอากาศจะเป็นไปตามตารางที่ 5 ในบทที่ 3 การทำงานเป็น Cycle คือทั้งคอมเพรสเซอร์และพัดลมภายในจะมีการทำงานพร้อมกันแต่ละ Cycle จะมีการ ON - OFF คอมเพรสเซอร์ด้วยช่วงเวลาที่แตกต่างกันขึ้นกับความแตกต่างของอุณหภูมิห้องและอุณหภูมิของผู้ใช้ตั้งไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
MAIN_PROGRAM;

DO;

$INCLUDE (REG51.DCL);

DECLARE

        /*      DECLARE  LOCATION PORT      */

        OUT_DATA            BYTE AT (A001H)  AUXILIARY,

        (ROOM_SW,OPERATION_SW,TIMER_SW,TEMP_SW,TIMER_CONTROL,FAN_CONTROL,

        TEST_FUNCTION)

                            BYTE  AT (A002H)  AUXILIARY,


        MULTIPLEX_PORT      BYTE  AT (A000H)  AUXILIARY,


        P8255_PORT          BYTE  AT (4000H)  AUXILIARY,


        TIME_STATUS         BYTE    EXTERNAL   AUXILIARY,

        OPERATE_TIME        BYTE    EXTERNAL   AUXILIARY,

        OPERATE_SW          BYTE    EXTERNAL   AUXILIARY,

        TA                  BYTE    EXTERNAL   AUXILIARY,

        TS                  BYTE    EXTERNAL   AUXILIARY,

        OPERATE_CHECK       BYTE    EXTERNAL   AUXILIARY,

        OUT_DATA            BYTE    EXTERNAL   AUXILIARY,


        CURRENT_TIME        WORD    EXTERNAL   AUXILIARY,

        BUFFER_OLD_TIME     WORD    EXTERNAL   AUXILIARY,

        MINUTE_TIME         WORD    EXTERNAL   AUXILIARY,


        ON_FLAG             BIT     AT(20H)    REGISTER,

        OFF_FLAG            BIT     AT(21H)    REGISTER,


        (QUIT,GOWORK)       LABEL;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* ROOM_TEMP IS TEMPERATURE READ FROM SENSOR */



/*
************************************************
*                                              *
*              ON TIME CONTROL                 *
*                 PROCEDURE                    *
*                                              *
************************************************
*/
ON_TIME_CONTROL : PROCEDURE   PUBLIC;
BUFFER_TIME = CURRENT_TIME;
CALL  CHECK_TIME;
IF   TIME_STATUS = 00H  THEN
     ON_FLAG = 1;
ELSE ON_FLAG = 0;
RETURN  ON_FLAG;
END  ON_TIME_CHECK;




************************************************
*                                              *
*              OFF TIME CONTROL                *
*                 PROCEDURE                    *
*                                              *
************************************************
*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
OFF_TIME_CONTROL : PROCEDURE  PUBLIC;
BUFFER_TIME = CURRENT_TIME;
CALL  CHECK_TIME;
IF    TIME_STATUS = 00H  THEN
      OFF_FLAG = 0;
ELSE OFF_FLAG = 1;
RETURN  OFF_FLAG;
END  ON_TIME_CHECK;

/*
************************************************
*                                              *
*                CHECK  TIME                   *
*                 PROCEDURE                    *
*                                              *
************************************************
*/

CHECK_TIME : PROCEDURE    PUBLIC;
      DECLARE   A     BYTE  AUXILIARY;
A = CURRENT_TIME - BUFFER_OLD_TIME;
IF    A <   TIMER_SET   THEN
      TIME_STATUS = FFH;

ELSE TIME_STATUS = 00H;
RETURN    TIME_STATUS;
END  CHECK_TIME;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*
**************************************************
*                                                *
*                  CONTINUE                      *
*                 PROCEDURE                      *
*                                                *
**************************************************
*/
CONTINUTE : PROCEDURE;
     DECLARE   (AUTO,COOL,DRY,C) BYTE,

AUTO = 03H;
COOL = 02H;
DRY  = 01H;
C = OPERATE_SW  AND  00000011B - 1;
DO  CASE  C;
     CALL   COOL_ROUTINE;              /*  CASE 0  */
     CALL   DRY_ROUTINE;               /*  CASE 1  */
     CALL   AUTO_ROUTINE;              /*  CASE 2  */
END;
END  CONTINUTE;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*
**************************************************
*                                                *
*                  COOL CONTROL                  *
*                   PROCEDURE                    *
*                                                *
**************************************************
*/
COOL_ROUTINE : PROCEDURE;
TS = TEMP_SW;
SENSOR_CH0 = 0;
TA= ROOM_TEMP;
IF    TA <> TS  THEN
      DO;
        IF TA > 2 THEN   CALL  ZONE1_A_ROUTINE;
        IF OPERATE_TIME < 1 THEN  CALL  ZONE1_B_ROUTINE;
        IF OPERATE_TIME < 2 THEN  CALL  ZONE1_C_ROUTINE;
        CALL  ZONE1_D_ROUTINE;
      END;
END  COOL_ROUTINE;

/* ZONE A -> ZONE D OF COOL MODE CONTROL FAN ON LOW SPEED. */

ZONE1_A : PROCEDURE :
IF CURRENT_TIME <= 10 THEN
   DO;
     IF MINUTE_TIME <= 18 THEN
        OUT_DATA = 03H;  /* ON  FAN & COMPRESSOR 18 MINUTES */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      IF (MINUTE_TIME >18) AND (MINUTE_TIME <=21)    THEN
         OUT_DATA = 00H;   /* OFF FAN & COMPRESSOR 3 MINUTES */
      OUT_PORT = OUT_DATA;
   END;
ELSE
   DO;
      OUT_PORT = 00H
   END;
END  ZONE1_A;


ZONE1_B : PROCEDURE :
IF CURRENT_TIME <= 1 THEN
   DO;
      IF MINUTE_TIME <= 3 THEN
         OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 3 MINUTES */
      IF (MINUTE_TIME >3) AND (MINUTE_TIME <=7)    THEN
         OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 4 MINUTES */
      OUT_PORT = OUT_DATA;
   END;
ELSE
   DO;
      OUT_PORT = 00H
   END;
END  ZONE1_B;


ZONE1_C : PROCEDURE :
IF (CURRENT_TIME <= 1) AND (CURRENT_TIME <= 2) THEN
   DO;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      IF MINUTE_TIME <= 3 THEN
         OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 3 MINUTES */
      IF (MINUTE_TIME >3) AND (MINUTE_TIME <=10)   THEN
         OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 7 MINUTES */
      OUT_PORT = OUT_DATA;
   END;
ELSE
   DO;
     OUT_PORT = 00H
   END;
END  ZONE1_C;

ZONE1_D : PROCEDURE ;
IF (CURRENT_TIME <= 2) AND (CURRENT_TIME <= 10)  THEN
   DO;
     IF MINUTE_TIME <= 1 THEN
        OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 3 MINUTES */
     IF (MINUTE_TIME >3) AND (MINUTE_TIME <=13)   THEN
        OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 10 MINUTES */
     OUT_PORT = OUT_DATA;
   END;
ELSE
   DO;
     OUT_PORT = 00H
   END;
END  ZONE1_D;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*
*************************************************
*                                               *
*                  DRY  CONTROL                 *
*                   PROCEDURE                   *
*                                               *
*************************************************
*/
DRY_ROUTINE : PROCEDURE;
TS = TEMP_SW;
TA = ROOM_TEMP;
IF   TA > 2 THEN
     DO;
       IF  OPERATE_TIME < 1 THEN
           CALL ZONE2_C_ROUTINE;

       ELSE
          DO;
            IF  OPERATE < 2 THEN  CALL  ZONE2_A_ROUTINE;
             ELSE  CALL  ZONE2_E_ROUTINE;
          END;
ELSE
   DO;
     IF  OPERATE_TIME < 1 THEN  CALL  ZONE2_B_ROUTINE;
     IF  OPERATE_TIME < 2 THEN  CALL  ZONE2_C_ROUTINE;
     ELSE  CALL  ZONE2_F_ROUTINE;
END  DRY_ROUTINE;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* ZONE A -> ZONE F OF DRY MODE CONTROL FAN ON LOW SPEED. */


ZONE2_A : PROCEDURE :
IF CURRENT_TIME <= 1 THEN
   DO;
     IF MINUTE_TIME <= 5 THEN
        OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 5 MINUTES */
     IF (MINUTE_TIME >5) AND (MINUTE_TIME <=9)   THEN
        OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 4 MINUTES */
     OUT_PORT = OUT_DATA;
   END;
ELSE
   DO;
     OUT_PORT = 00H
   END;
END  ZONE2_A;


ZONE2_B : PROCEDURE :
IF CURRENT_TIME <= 1  THEN
   DO;
     IF MINUTE_TIME <= 1 THEN
        OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 3 MINUTES */
     IF (MINUTE_TIME >3) AND (MINUTE_TIME <=7)   THEN
        OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 4 MINUTES */
     OUT_PORT = OUT_DATA;
   END;
ELSE
   DO;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      OUT_PORT = 00H
   END;
END  ZONE2_B;

ZONE2_C : PROCEDURE :
IF (CURRENT_TIME <= 1) AND (CURRENT_TIME <= 2) THEN
   DO;
      IF MINUTE_TIME <= 5 THEN
         OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 5 MINUTES */
      IF (MINUTE_TIME >5) AND (MINUTE_TIME <=12)   THEN
         OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 7 MINUTES */
      OUT_PORT = OUT_DATA;
   END;
ELSE
   DO;
      OUT_PORT = 00H
   END;
END  ZONE2_C;

ZONE1_D : PROCEDURE :
IF (CURRENT_TIME <= 2) AND (CURRENT_TIME <= 10)  THEN
   DO;
      IF MINUTE_TIME <= 3 THEN
         OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 3 MINUTES */
      IF (MINUTE_TIME >3) AND (MINUTE_TIME <=10)   THEN
         OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 7 MINUTES */
      OUT_PORT = OUT_DATA;
   END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ELSE
```

```
    DO;
      OUT_PORT = 00H
    END;
END  ZONE2_D;

ZONE1_E : PROCEDURE :
IF (CURRENT_TIME <= 2) AND (CURRENT_TIME <= 10)  THEN
    DO;
      IF MINUTE_TIME <= 5 THEN
         OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 5 MINUTES */
      IF (MINUTE_TIME >5) AND (MINUTE_TIME <=15)   THEN
         OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 10 MINUTES */
      OUT_PORT = OUT_DATA;
    END;
ELSE
    DO;
      OUT_PORT = 00H
    END;
END  ZONE1_E;

ZONE2_F : PROCEDURE :
IF (CURRENT_TIME <= 2) AND (CURRENT_TIME <= 10)  THEN
    DO;
      IF MINUTE_TIME <= 3 THEN
         OUT_DATA = 03H;  /* ON FAN & COMPRESSOR 3 MINUTES */
      IF (MINUTE_TIME >3) AND (MINUTE_TIME <=13)   THEN
         OUT_DATA = 00H;  /* OFF FAN & COMPRESSOR 10 MINUTES */
      OUT_PORT = OUT_DATA;
    END;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ELSE
   DO;
     OUT_PORT = 00H
   END;
END  ZONE1_F;



/*
************************************************
*                                              *
*                AUTO  CONTROL                 *
*                  PROCEDURE                   *
*                                              *
************************************************
*/

AUTO_ROUTINE : PROCEDURE;
SENSOR_CH0 = 0;
TA = ROOM_TEMP;
IF   TA > 28 THEN   CALL  COOL_ROUTINE;
IF   TA > 26 THEN   CALL  DRY_ROUTINE;
OUT_PORT = 02H
END  AUTO_ROUTINE;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*
*************************************************
*                                               *
*              AUTO FAN CONTROL                 *
*                PROCEDURE                      *
*                                               *
*************************************************

*/

AUTO_FAN_CONTROL : PROCEDURE;
TS = TEMP_SW;
SENSOR_CH0 = 0;
TA = ROOM_TEMP;
IF   COMPRESSOR = 00H   THEN   OUT_PORT = 02H;
IF   TA > TS  = 4       THEN   OUT_PORT = 08H;
ELSE  OUT_PORT = 04H;
END  AUTO_FAN_CONTROL;

/*
*************************************************
*                                               *
*              DRY  CONTROL                     *
*                PROCEDURE                      *
*                                               *
*************************************************

*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DRY_ROUTINE : PROCEDURE;
TA = ROOM_TEMP;
TS = TEMP_SW;
IF   TA < TS = 1  THEN
     DO;
       CALL  ON_3MIN;
       CALL  OFF_6MIN;
     END;
IF   TA > TS = 2  THEN
     DO;
       CALL  ON_6MIN;
       CALL  OFF_4MIN;
     END;
ELSE
     DO;
       CALL  ON_5MIN;
       CALL  OFF_5MIN;
     END;
END  DRY_ROUTINE;

/*
**************************************************
*                                                *
*               RUNNNING  PROGRAM                *
*                                                *
**************************************************
*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
RUNNING_PROGRAM :

DO;


GOWORK:  OPERATE_CHECK = TIMER_CONTROL  AND  000000011B;

IF   OPERATION_CHECK = 1   THEN

       IF FLAG_ON = 0  THEN  GOTO  GOWORK;


IF   OPERATION_CHECK = 3   THEN

       IF FLAG_OFF = 1  THEN  CALL  OFF_SYSTEM;

     END;


IF   OPERATE_CHECK = 2   THEN

       IF FLAG_OFF = 1  THEN  CALL OFF_SYSTEM;


QUIT: CALL  CONTINUTE;

END  RUNNING_PROGRAM;

END  MAIN_PROGRAM;
```

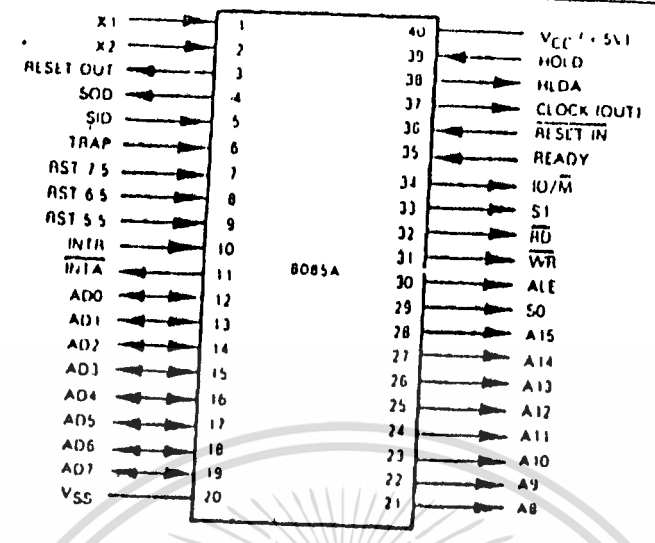
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
บทวิจารณ์และสรุป

โปรเจคนี้เป็น Software Programming เป็นการศึกษาเกี่ยวกับโครงสร้างของภาษา PL/M-51 เพื่อนำไปควบคุมการทำงานของเครื่องปรับอากาศโดยทั่วไปตามความต้องการของผู้ใช้ผ่านทาง Remote Switch ปัญหาที่พบก็คือ ต้องใช้เวลาในการศึกษาการทำงานของเครื่องปรับอากาศเป็นอย่างมาก เนื่องจากมีความรู้เรื่องนี้ไม่มากนักประกอบกับภาษา PL/M-51 เป็นภาษาใหม่สำหรับผู้จัดทำ

สำหรับส่วนอื่น ๆ ที่ทำหน้าที่ควบคุมเครื่องปรับอากาศจริง ๆ ได้แก่ วงจรรีเลย์ (Relay) ที่ใช้ ควบคุมการทำงานของคอมเพรสเซอร์และพัดลมภายในนั้นไม่ได้ออกแบบแต่ได้แสดงตำแหน่ง ของ คอนแทค ของ รีเลย์ที่จำเป็นต้องต่อกับเครื่องปรับอากาศไว้แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



PIN NAME	DESCRIPTION	TYPE
AD0 - AD7	Address/Data Bus	Bidirectional tristate
A8 - A15	Address Bus	Output tristate
ALE	Address Latch Enable	Output
$\overline{RD}$	Read Control	Output tristate
$\overline{WR}$	Write Control	Output, tristate
IO/$\overline{M}$	I/O or Memory Indicator	Output, tristate
S0 S1	Bus State Indicators	Output
READY	Wait State Request	Input
SID	Serial Data Input	Input
SOD	Serial Data Output	Output
HOLD	Hold Request	Input
HLDA	Hold Acknowledge	Output
INTR	Interrupt Request	Input
TRAP	Non-maskable Interrupt Request	Input
RST 5.5 RST 6.5 RST 7.5	Hardware vectored interrupt requests	Input Input Input
$\overline{INTA}$	Interrupt Acknowledge	Output
$\overline{RESET\ IN}$	System Reset	Input
RESET OUT	Peripherals Reset	Output
X1, X2	Crystal or RC Connections	Input
CLK	Clock Signal	Output
V_{CC} V_{SS}	Power Ground	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 3.5 A Summary Of Instruction Object Codes And Execution Cycles

Instruction	Object Code	Bytes	Cycles 8085	Cycles 8080A
ACI data	CE yy	2	7	7
ADC reg	10001sss	1	4	4
ADC M	8E	1	7	7
ADD reg	10000sss	1	4	4
ADD M	86	1	7	7
ADI data	C6 yy	2	7	7
ANA reg	10100sss	1	4	4
ANA M	A6	1	7	7
ANI data	E6 yy	2	7	7
CALL label	CD pppp	3	18	17
CC label	DC pppp	3	9/18	11/17
CM label	FC pppp	3	9/18	11/17
CMA	2F	1	4	4
CMC	3F	1	4	4
CMP reg	10111sss	1	4	4
CMP M	BE	1	7	7
CNC label	D4 pppp	3	9/18	11/17
CNZ label	C4 pppp	3	9/18	11/17
CP label	F4 p...	3	9/18	11/17
CPE label	EC p...	3	9/18	11/17
CPI data	FE yy	2	7	7
CPO label	E4 pppp	3	9/18	11/17
CZ label	CC pppp	3	9/18	11/17
DAA	27	1	4	4
DAD rp	00rp1001	1	10	10
DCR reg	00sss101	1	4	5
DCR M	35	1	10	10
DCX rp	00rp1011	1	6	5
DI	F3	1	4	4
EI	FB	1	4	4
HLT	76	1	5	7
IN port	DB yy	2	10	10
INR reg	00sss100	1	4	5
INR M	34	1	10	10
INX rp	00rp0011	1	6	5
JC label	DA pppp	3	7/10	10
JM label	FA pppp	3	7/10	10
JMP label	C3 pppp	3	10	10
JNC label	D2 pppp	3	7/10	10
JNZ label	C2 pppp	3	7/10	10
JP label	F2 pppp	3	7/10	10
JPE label	EA pppp	3	7/10	10
JPO label	E2 pp...	3	7/10	10
JZ label	CA pppp	3	7/10	10
LDA addr	3A pppp	3	13	13
LDAX pr	000x1010	1	7	7
LHLD addr	2A pppp	3	16	16

Instruction	Object Code	Bytes	Cycles
LXI rp data16	00rp0001 yyyy	3	10
MOV s d	01dddsss	1	5
MOV M reg	01110sss	1	7
MOV reg M	01ddd110	1	7
MVI reg data	00ddd110 yy	2	7
MVI M data	36 yy	2	10
NOP	00	1	4
ORA reg	10110sss	1	4
ORA M	B6	1	7
ORI data	F6 yy	2	7
OUT port	D3 yy	2	10
PCHL	E9	1	5
POP rp	11rp0001	1	10
PUSH rp	11rp0101	1	11
RAL	17	1	4
RAR	1F	1	4
RC	DB	1	5/11
RET	C9	1	10
RIM	20	1	4*
RLC	07	1	4
RM	F8	1	5/11
RNC	D0	1	5/11
RNZ	C0	1	5/11
RP	F0	1	5/11
RPE	E8	1	5/11
RPO	E0	1	5/11
RRC	0F	1	4
RST n	11nnn111	1	11
RZ	C8	1	5/11
SBB reg	10011sss	1	4
SBB M	9E	1	7
SBI data	DE yy	2	7
SHLD addr	22 pppp	3	16
SIM	30	1	4*
SPHL	F9	1	5
STA addr	32 pppp	3	13
STAX pr	000x0010	1	7
STC	37	1	4
SUB reg	10010sss	1	4
SUB M	96	1	7
SUI data	D6 yy	2	7
XCHG	EB	1	4
XRA reg	10101sss	1	4
XRA M	AE	1	7
XRI data	EE yy	2	7
XTHL	E3	1	18

Object Code

ddd Destination register — same coding as sss

nnn Restart number 000 to 111

pppp A 16-bit memory address

sss Source register — same coding as ddd

x Register pair 0 = BC
1 = DE

xx Register pair 00 = BC
01 = DE
10 = HL
11 = SP or if PUSH POP: PSW

sss Register 111 = A
000 = B
001 = C
010 = D
011 = E
100 = H
101 = L

yy An 8 bit binary data unit

yyyy A 16-bit binary data unit

*8085 instructions

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 3-7 Correspondence between 8080A and Z80 Mnemonics

8080A Mnemonic		Z80 Mnemonic	
ACI	data	ADC	A,data
ADC	reg or M	ADC	A,reg or (HL)
ADD	reg or M	ADD	A,reg or (HL)
ADI	data	ADD	A,data
ANA	reg or M	AND	reg or (HL)
ANI	data	AND	data
CALL	addr	CALL	addr
CC	addr	CALL	C,addr
CM	addr	CALL	M,addr
CMA		CPL	
CMC		CCF	
CMP	reg or M	CP	reg or (HL)
CNC	addr	CALL	NC,addr
CNZ	addr	CALL	NZ,addr
CP	addr	CALL	P,addr
CPE	addr	CALL	PE,addr
CPI	data	CP	data
CPO	addr	CALL	PO,addr
CZ	addr	CALL	Z,addr
DAA		DAA	
DAD	rp	ADD	HL,rp
DCR	reg or M	DEC	reg or (HL)
DCX	rp	DEC	rp
DI		DI	
EI		EI	
HLT		HALT	
IN	port	IN	A,(port)
INR	reg or M	INC	reg or (HL)
INX	rp	INC	rp
JC	addr	JP	C,addr
JM	addr	JP	M,addr
JMP	addr	JP	addr
JNC	addr	JP	NC,addr
JP	addr	JP	P,addr
JNZ	addr	JP	NZ,addr
JPE	addr	JP	PE,addr
JPO	addr	JP	PO,addr
JZ	addr	JP	Z,addr
LDA	addr	LD	A,(addr)
LDAX	B or D	LD	A,(BC) or (DE)

8080A Mnemonic		Z80 Mnemonic	
LHLD	addr	LD	HL,(addr)
LXI	rp,data16	LD	rp,data16
MOV	reg,reg or M	LD	reg,reg or (HL)
MOV	reg or M,reg	LD	reg or (HL),reg
MVI	reg or M,data	LD	reg or (HL),data
NOP		NOP	
ORA	reg or M	OR	reg or (HL)
ORI	data	OR	data
OUT	port	OUT	(port),A
PCHL		JP	(HL)
POP	rp	POP	rp
PUSH	rp	PUSH	rp
RAL		RLA	
RAR		RRA	
RC		RET	C
RET		RET	
RLC		RLCA	
RM		RET	M
RNC		RET	NC
RNZ		RET	NZ
RP		RET	P
RPE		RET	PE
RPO		RET	PO
RRC		RRCA	
RST	n	RST	n
RZ		RET	Z
SBB	reg or M	SBC	A,reg or (HL)
SBI	data	SBC	A,data
SHLD	addr	LD	(addr),HL
SPHL		LD	SP,HL
STA	addr	LD	(addr),A
STAX	B or D	LD	(BC) or (DE),A
STC		SCF	
SUB	reg or M	SUB	reg or (HL)
SUI	data	SUB	data
XCHG		EX	DE,HL
XRA	reg or M	XOR	reg or (HL)
XRI	data	XOR	data
XTHL		EX	(SP),HL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 3.8 Correspondence between Z80 and 8080A Mnemonics

Z80 Mnemonic	8080A Mnemonic
ADC A,data	ACI data
ADC A,(HL)	ADC M
ADC A,reg	ADC reg
ADC A,(ey + displ)	—
ADC HL,rp	—
ADD A,data	ADI data
ADD A,(HL)	ADD M
ADD A,reg	ADD reg
ADD A,(ey + displ)	—
ADD HL,rp	DAD rp
ADD IX,pp	—
ADD IY,rr	—
AND data	ANI data
AND (HL)	ANA M
AND reg	ANA reg
AND (ey + displ)	—
BIT b,(HL)	—
BIT b,reg	—
BIT b,(ey + displ)	—
CALL addr	CALL addr
CALL C,addr	CC addr
CALL M,addr	CM addr
CALL NC,addr	CNC addr
CALL NZ,addr	CNZ addr
CALL P,addr	CP addr
CALL PE,addr	CPE addr
CALL PO,addr	CPO addr
CALL Z,addr	CZ addr
CCF	CMC
CP data	CPI data
CP (HL)	CMP M
CP reg	CMP reg
CP (ey + displ)	—
CPD	—
CPDR	—
CPI	—
CPIR	—
CPL	CMA
DAA	DAA
DEC (HL)	DCR M
DEC reg	DCR reg
DEC rp	DCX rp
DEC ey	—
DEC (ey + displ)	—
DI	DI
DJNZ disp	—
EI	EI
EX AF,AF'	—
EX DE,HL	XCHG
EX (SP),HL	XTHL
EX (SP),ey	—
EXX	—
HALT	HLT
IM m	—
IN A,(port)	IN port
IN reg,(C)	—
INC (HL)	INR M
INC reg	INR reg

Z80 Mnemonic	8080A Mnemonic
INC rp	INX rp
INC ey	—
INC (ey + displ)	—
IND	—
INDR	—
INI	—
INIR	—
JP addr	JMP addr
JP C,addr	JC addr
JP (HL)	PCHL
JP M,addr	JM addr
JP NC,addr	JNC addr
JP NZ,addr	JNZ addr
JP P,addr	JP addr
JP PE,addr	JPE addr
JP PO,addr	JPO addr
JP Z,addr	JZ addr
JP ey	—
JR C,disp	—
JR disp	—
JR NC,disp	—
JR NZ,disp	—
JR Z,disp	—
LD A,(addr)	LDA addr
LD A,(BC) or (DE)	LDAX B or D
LD A,I	—
LD A,R	—
LD (addr),A	STA addr
LD (addr),BC or DE	—
LD (addr),HL	SHLD addr
LD (addr),SP	—
LD (addr),ey	—
LD (BC) or (DE),A	STAX B or D
LD BC or DE,(addr)	—
LD HL,(addr)	LHLD addr
LD (HL),data	MVI M,data
LD (HL),reg	MOV M,reg
LD I,A	—
LD R,A	—
LD reg,data	MVI reg,data
LD reg,(HL)	MOV reg,M
LD reg,reg	MOV reg,reg
LD reg,(ey + displ)	—
LD rp,data16	LXI rp,data16
LD SP,(addr)	—
LD SP,HL	SPHL
LD SP,ey	—
LD ey,data16	—
LD ey,(addr)	—
LD (ey + displ),data	—
LD (ey + displ),reg	—
LDD	—
LDDR	—
LDI	—
LDIR	—
NEG	—
NOP	NOP
OR data	ORI data

— indicates that there is no corresponding instruction

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 3.8 Correspondence between Z80 and 8080A Mnemonics (Continued)

Z80 Mnemonic		8080A Mnemonic	
OR	(HL)	ORA	M
OR	reg	ORA	reg
OR	(ix + displ)	—	
OTDR		—	
OTIR		—	
OUT	(C),reg	—	
OUT	(port),A	OUT	port
OUTD		—	
OUTI		—	
POP	qq	POP	qq
POP	ix	—	
PUSH	qq	PUSH	qq
PUSH	ix	—	
RES	b,(HL)	—	
RES	b,reg	—	
RES	b,(ix + displ)	—	
RET		RET	
RET	C	RC	
RET	M	RM	
RET	NC	RNC	
RET	NZ	RNZ	
RET	P	RP	
RET	PE	RPE	
RET	PO	RPO	
RET	Z	RZ	
RETI		—	
RETN		—	
RL	(HL)	—	
RL	reg	—	
RL	(ix + displ)	—	
RLA		RAL	
RLC	(HL)	—	
RLC	reg	—	
RLC	(ix + displ)	—	
RLCA		RLC	
RLD		—	

Z80 Mnemonic		8080A Mnemonic	
RR	(HL)	—	
RR	reg	—	
RR	(ix + displ)	—	
RRA		RAR	
RRC	(HL)	—	
RRC	reg	—	
RRC	(ix + displ)	—	
RRCA		RRC	
RRD		—	
RST	n	RST	n
SBC	A,data	SBI	data
SBC	A,(HL)	SBB	M
SBC	A,reg	SBB	reg
SBC	A,(ix + displ)	—	
SBC	HL,ss	—	
SCF		STC	
SET	b,(HL)	—	
SET	b,reg	—	
SET	b,(ix + displ)	—	
SLA	(HL)	—	
SLA	reg	—	
SLA	(ix + displ)	—	
SRA	(HL)	—	
SRA	reg	—	
SRA	(ix + displ)	—	
SRL	(HL)	—	
SRL	reg	—	
SRL	(ix + displ)	—	
SUB	data	SUI	data
SUB	(HL)	SUB	M
SUB	reg	SUB	reg
SUB	(ix + displ)	—	
XOR	data	XRI	data
XOR	(HL)	XRA	M
XOR	reg	XRA	reg
XOR	(ix + displ)	—	

— indicates that there is no corresponding instruction

Table 3.9 Unused 8080A Operation Codes and Their Z80 Meanings

8080A Operation Code	Z80 Use
08	EX AF,AF'
10	DJNZ dsp
18	JR dsp
20 (RIM on 8085)	JR NZ dsp
28	JR Z dsp
30 (SIM on 8085)	JR NC dsp
38	JR C dsp
CB	BIT RES RL RLC RR RRC SET SLA SRA SRL
D9	EXX
DD	All instructions involving Register IX
ED	ADC HL,ss / LD A,I / NEG / CPD / LD A,R / OTDR / CPDR / LD (nn),ss / OTIR / CPI / LD I,A / OUT (C),reg / CPIR / LD R,A / OUTD / IM m / LD ss,(nn) / OUTI / IN reg,(C) / LDD / RETI / IND / LDDR / RETN / INDR / LDI / RLD / INI / LDIR / RRD / INIR / SBC HL,ss
FD	All instructions involving Register IY

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

intel

8155H/8156H/8155H-2/8156H-2
2048-BIT STATIC HMOS RAM WITH I/O PORTS AND TIMER

- Single +5V Power Supply with 10% Voltage Margins
- 30% Lower Power Consumption than the 8155 and 8156
- 100% Compatible with 8155 and 8156
- 256 Word x 8 Bits
- Completely Static Operation
- Internal Address Latch
- 2 Programmable 8-Bit I/O Ports
- 1 Programmable 6-Bit I/O Port
- Programmable 14-Bit Binary Counter/ Timer
- Compatible with 8085AH, 8085A and 8088 CPU
- Multiplexed Address and Data Bus
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® 8155H and 8156H are RAM and I/O chips implemented in N-Channel, depletion load, silicon gate technology (HMOS), to be used in the 8085AH and 8088 microprocessor systems. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085AH CPU. The 8156H-2 and 8156H-2 have maximum access times of 330 ns for use with the 8085AH-2 and the 5 MHz 8088 CPU.

The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.

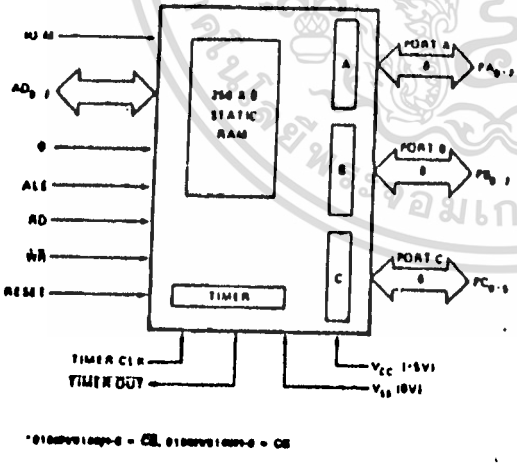


Figure 1. Block Diagram

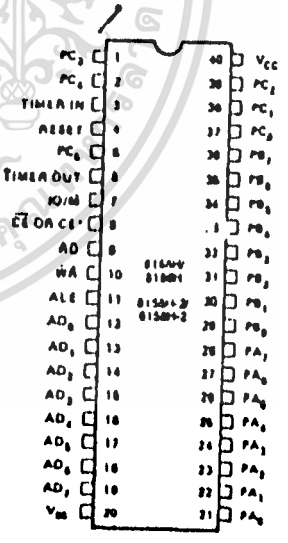


Figure 2. Pin Configuration

Intel Corporation Assumes No Responsibility for the Use of Any Circuitry Other Than Circuitry Embodied in an Intel Product. No Other Circuit Patent Licenses are Implied.
© INTEL CORPORATION, 1981

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 1. Pin Description

Symbol	Type	Name and Function
RESET	I	Reset: Pulse provided by the 8085AH to initialize the system (connect to 8085AH RESET OUT) Input high on this line resets the chip and initializes the three I/O ports to input mode. The width of RESET pulse should typically be two 8085AH clock cycle times.
AD_{0-7}	I/O	Address/Data: 3-state Address/Data lines that interface with the CPU lower 8-bit Address/Data Bus The 8-bit address is latched into the address latch inside the 8155H/56H on the falling edge of ALE The address can be either for the memory section or the I/O section depending on the IO/$\overline{M}$ input The 8 bit data is either written into the chip or read from the chip, depending on the $\overline{WR}$ or $\overline{RD}$ input signal
CE or $\overline{CE}$	I	Chip Enable: On the 8155H, this pin is $\overline{CE}$ and is ACTIVE LOW On the 8156H, this pin is CE and is ACTIVE HIGH
$\overline{RD}$	I	Read Control: Input low on this line with the Chip Enable active enables and AD_{0-7} buffers If IO/$\overline{M}$ pin is low, the RAM content will be read out to the AD bus Otherwise the content of the selected I/O port or command/status registers will be read to the AD bus
$\overline{WR}$	I	Write Control: Input low on this line with the Chip Enable active causes the data on the Address/Data bus to be written to the RAM or I/O ports and command/status register depending on IO/$\overline{M}$
ALE	I	Address Latch Enable: This control signal latches both the address on the AD_{0-7} lines and the state of the Chip Enable and IO/$\overline{M}$ into the chip at the falling edge of ALE
IO/$\overline{M}$	I	I/O Memory: Selects memory if low and I/O and command/status registers if high.
PA_{0-7}(8)	I/O	Port A: These 8 pins are general purpose I/O pins The in/out direction is selected by programming the command register
PB_{0-7}(8)	I/O	Port B: These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register
PC_{0-5}(6)	I/O	Port C: These 6 pins can function as either input port, output port, or as control signals for PA and PB Programming is done through the command register. When PC_{0-5} are used as control signals, they will provide the following: PC_0 — A INTR (Port A Interrupt) PC_1 — ABF (Port A Buffer Full) PC_2 — $\overline{A\ STB}$ (Port A Strobe) PC_3 — B INTR (Port B Interrupt) PC_4 — B IBF (Port B Buffer Full) PC_5 — $\overline{B\ STB}$ (Port B Strobe)
TIMER IN	I	Timer Input: Input to the counter-timer
$\overline{TIMER\ OUT}$	O	Timer Output: This output can be either a square wave or a pulse, depending on the timer mode.
V_{CC}		Voltage: +5 volt supply
V_{SS}		Ground: Ground reference.

FUNCTIONAL DESCRIPTION

The 8155H/8156H contains the following:

- 2k Bit Static RAM organized as 256 x 8
- Two 8-bit I/O ports (PA & PB) and one 6-bit I/O port (PC)
- 14-bit timer-counter

The IO/$\overline{M}$ (IO/Memory Select) pin selects either the five registers (Command, Status, PA_{0-7}, PB_{0-7}, PC_{0-5}) or the memory (RAM) portion.

The 8-bit address on the Address/Data lines, Chip Enable input CE or $\overline{CE}$, and IO/$\overline{M}$ are all latched on-chip at the falling edge of ALE.

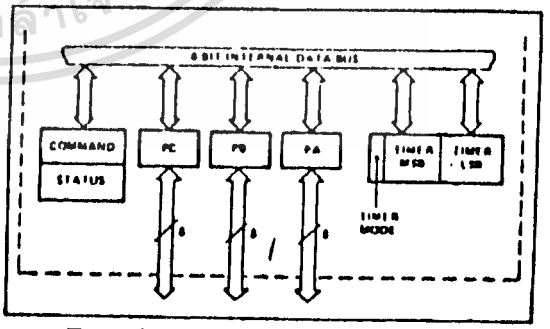


Figure 3. 8155H/8156H Internal Registers

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

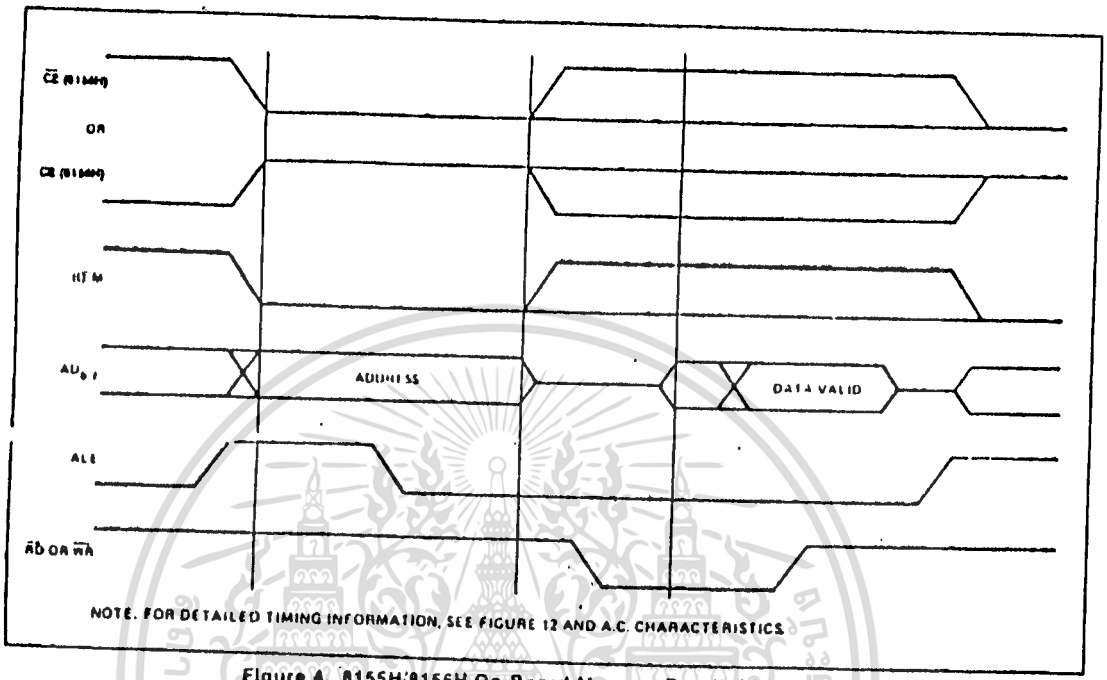


Figure 4. 8155H/8156H On-Board Memory Read/Write Cycle

PROGRAMMING OF THE COMMAND REGISTER

The command register consists of eight latches. Four bits (0-3) define the mode of the ports, two bits (4-5) enable or disable the interrupt from port C when it acts as control port, and the last two bits (6-7) are for the timer.

The command register contents can be altered at any time by using the I/O address XXXXX000 during a WRITE operation with the Chip Enable active and IO/$\overline{M}$ = 1. The meaning of each bit of the command byte is defined in Figure 5. The contents of the command register may never be read.

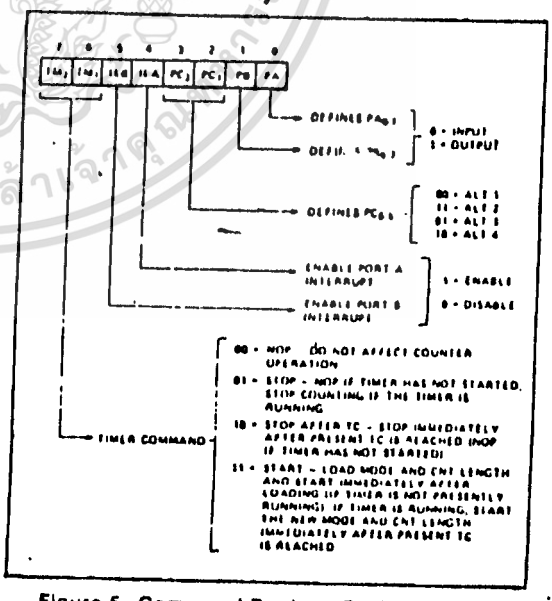


Figure 5. Command Register Bit Assignment

READING THE STATUS REGISTER

The status register consists of seven latches, one for each bit, six (0-5) for the status of the ports and one (6) for the status of the timer.

The status of the timer and the I/O section can be polled by reading the Status Register (Address XXXXX000). Status word format is shown in Figure 6. Note that you may never write to the status register since the command register shares the same I/O address and the command register is selected when a write to that address is issued.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

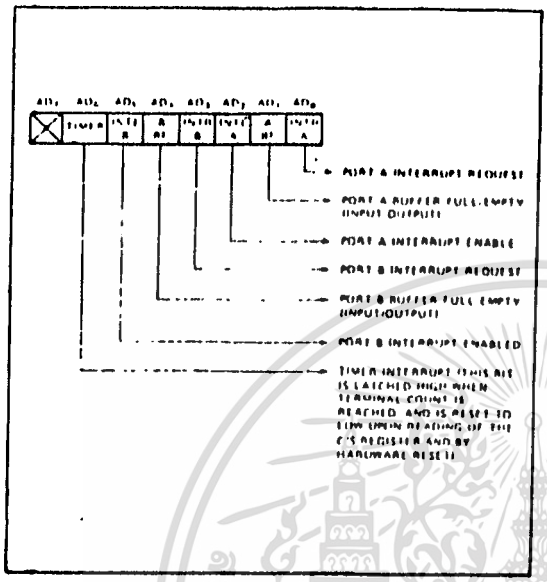


Figure 6. Status Register Bit Assignment

INPUT/OUTPUT SECTION

The I/O section of the 8155H/8156H consists of five registers: (See Figure 7.)

- **Command/Status Register (C/S)** — Both registers are assigned the address XXXXX000. The C/S address serves the dual purpose.

  When the C/S registers are selected during WRITE operation, a command is written into the command register. The contents of this register are *not* accessible through the pins.

  When the C/S (XXXXX000) is selected during a READ operation, the status information of the I/O ports and the timer becomes available on the AD_{0-7} lines.

- **PA Register** — This register can be programmed to be either input or output ports depending on the status of the contents of the C/S Register. Also depending on the command, this port can operate in either the basic mode or the strobed mode (See timing diagram). The I/O pins assigned in relation to this register are PA_{0-7}. The address of this register is XXXXX001.

- **PB Register** — This register functions the same as PA Register. The I/O pins assigned are PB_{0-7}. The address of this register is XXXXX010.

- **PC Register** — This register has the address XXXXX011 and contains only 6 bits. The 6 bits can be programmed to be either input ports, output ports or as control signals for PA and PB by properly programming the AD_7 and AD_3 bits of the C/S register.

  When PC_{0-5} is used as a control port, 3 bits are assigned for Port A and 3 for Port B. The first bit is an interrupt that the 8155H sends out. The second is an output signal indicating whether the buffer is full or empty, and the third is an input pin to accept a strobe for the strobed input mode. (See Table 2.)

When the 'C' port is programmed to either ALT3 or ALT4, the control signals for PA and PB are initialized as follows:

CONTROL	INPUT MODE	OUTPUT MODE
BF	Low	Low
INTR	Low	High
STB	Input Control	Input Control

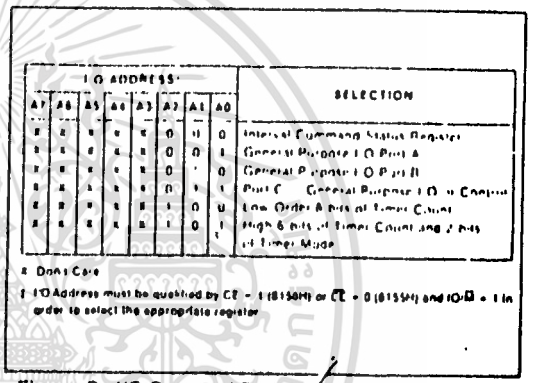


Figure 7. I/O Port and Timer Addressing Scheme

Figure 8 shows how I/O PORTS A and B are structured within the 8155H and 8156H:

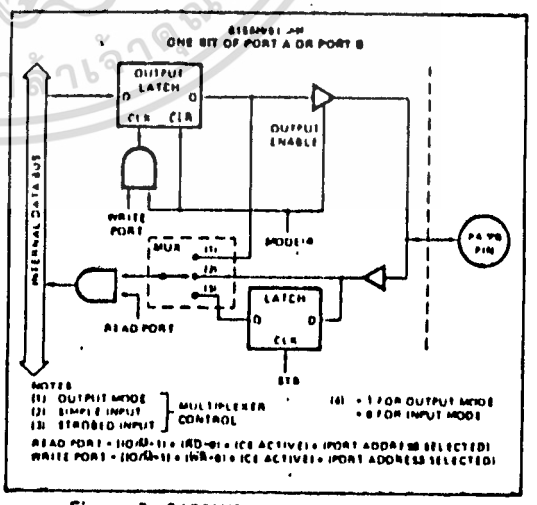


Figure 8. 8155H/8156H Port Functions

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 2. Port Control Assignment

Pin	ALT 1	ALT 2	ALT 3	ALT 4
PC0	Input Port	Output Port	A INTR (Port A Interrupt)	A INTR (Port A Interrupt)
PC1	Input Port	Output Port	A BF (Port A Buffer Full)	A BF (Port A Buffer Full)
PC2	Input Port	Output Port	A $\overline{STB}$ (Port A Strobe)	A $\overline{STB}$ (Port A Strobe)
PC3	Input Port	Output Port	Output Port	B INTR (Port B Interrupt)
PC4	Input Port	Output Port	Output Port	B BF (Port B Buffer Full)
PC5	Input Port	Output Port	Output Port	B $\overline{STB}$ (Port B Strobe)

Note in the diagram that when the I/O ports are programmed to be output ports, the contents of the output ports can still be read by a READ operation when appropriately addressed.

The outputs of the 8155H/8156H are "glitch-free" meaning that you can write a "1" to a bit position that was previously "1" and the level at the output pin will not change.

Note also that the output latch is cleared when the port enters the input mode. The output latch cannot be loaded by writing to the port if the port is in the input mode. The result is that each time a port mode is changed from input to output, the output pins will go low. When the 8155H/56H is RESET, the output latches are all cleared and all 3 ports enter the input mode.

When in the ALT 1 or ALT 2 modes, the bits of PORT C are structured like the diagram above in the simple input or output mode, respectively.

Reading from an input port with nothing connected to the pins will provide unpredictable results.

Figure 9 shows how the 8155H/8156H I/O ports might be configured in a typical MCS-85 system.

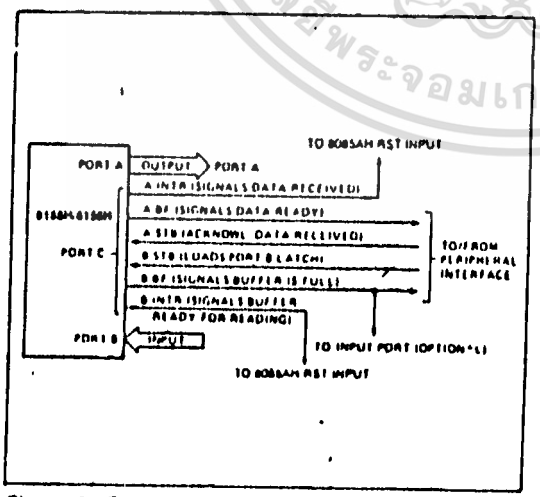


Figure 9. Example: Command Register = 00111001

TIMER SECTION

The timer is a 14-bit down-counter that counts the TIMER IN pulses and provides either a square wave or pulse when terminal count (TC) is reached.

The timer has the I/O address XXXXX100 for the low order byte of the register and the I/O address XXXXX101 for the high order byte of the register. (See Figure 7.)

To program the timer, the COUNT LENGTH REG is loaded first, one byte at a time, by selecting the timer addresses. Bits 0-13 of the high order count register will specify the length of the next count and bits 14-15 of the high order register will specify the timer output mode (see Figure 10). The value loaded into the count length register can have any value from 2H through 3FFFH in Bits 0-13.

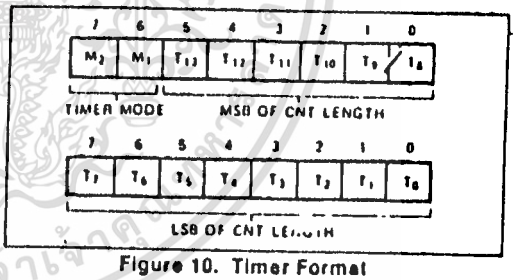


Figure 10. Timer Format

There are four modes to choose from. M2 and M1 define the timer mode, as shown in Figure 11.

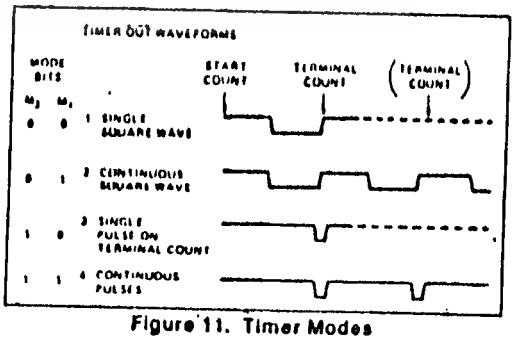


Figure 11. Timer Modes

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Bits 6-7 (TM2 and TM1) of command register contents are used to start and stop the counter. There are four commands to choose from:

TM_2	TM_1	
0	0	NOP — Do not affect counter operation.
0	1	STOP — NOP if timer has not started; stop counting if the timer is running
1	0	STOP AFTER TC — Stop immediately after present TC is reached. NOP if timer has not started.
1	1	START — Load mode and CNT length and start immediately after loading. (If timer is not presently running). If timer is running, start the new mode and CNT length immediately after present TC is reached.

Note that while the counter is counting, you may load a new count and mode into the count length registers. Before the new count and mode will be used by the counter, you must issue a START command to the counter. This applies even though you may only want to change the count and use the previous mode.

In case of an odd-numbered count, the first half-cycle of the squarewave output, which is high, is one count longer than the second (low) half-cycle, as shown in Figure 12.

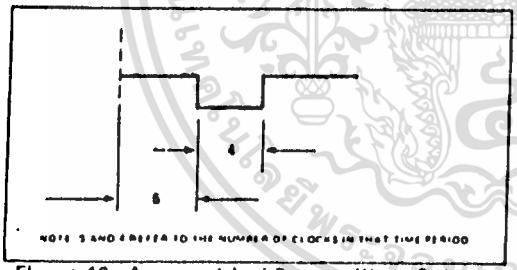


Figure 12. Asymmetrical Square-Wave Output Resulting from Count of 9

The counter in the 8155H is not initialized to any particular mode or count when hardware RESET occurs, but RESET does *stop* the counting. Therefore, counting cannot begin following RESET until a START command is issued via the C/S register.

Please note that the timer circuit on the 8155H/8156H chip is designed to be a square-wave timer, not an event counter. To achieve this, it counts down by twos (twice in completing one cycle). Thus, its registers do not contain values directly representing the number of TIMER IN pulses received. You cannot load an initial value of 1 into the count register and cause the timer to operate, as its terminal count value is 10 (binary) or 2 (decimal). (For the detection of single pulses, it is suggested that one of the hardware interrupt pins on the 8085AH be used.) After the timer has started counting down, the values residing in the count registers can be used to calculate the actual number of TIMER IN pulses required to complete the timer cycle if desired. To obtain the remaining count, perform the following operations in order:

1. Stop the count.
2. Read in the 16-bit value from the count length registers.
3. Reset the upper two mode bits.
4. Reset the carry and rotate right one position all 16 bits through carry.
5. If carry is set, add 1/2 of the full original count (1/2 full count — 1 if full count is odd).

Note: If you started with an odd count and you read the count length register before the third count pulse occurs, you will not be able to discern whether one or two counts has occurred. Regardless of this, the 8155H/56H always counts out the right number of pulses in generating the TIMER $\overline{OUT}$ waveforms.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8085A MINIMUM SYSTEM CONFIGURATION

Figure 13a shows a minimum system using three chips, containing:

- 256 Bytes RAM
- 2K Bytes EPROM
- 38 I/O Pins
- 1 Interval Timer
- 4 Interrupt Levels

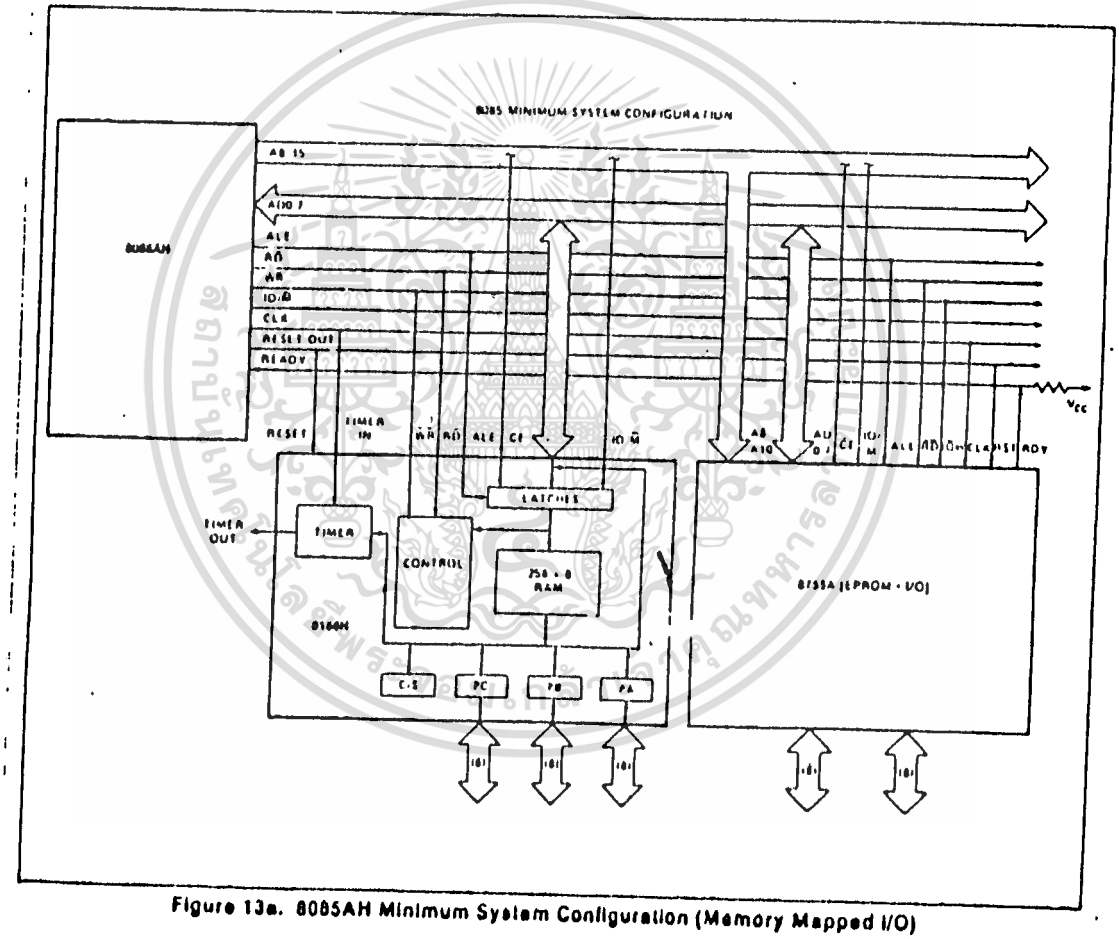


Figure 13a. 8085AH Minimum System Configuration (Memory Mapped I/O)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8088 FIVE CHIP SYSTEM

Figure 13b shows a five chip system containing

- 1.25K Bytes RAM
- 2K Bytes EPROM
- 38 I/O Pins
- 1 Interval Timer
- 2 Interrupt Levels

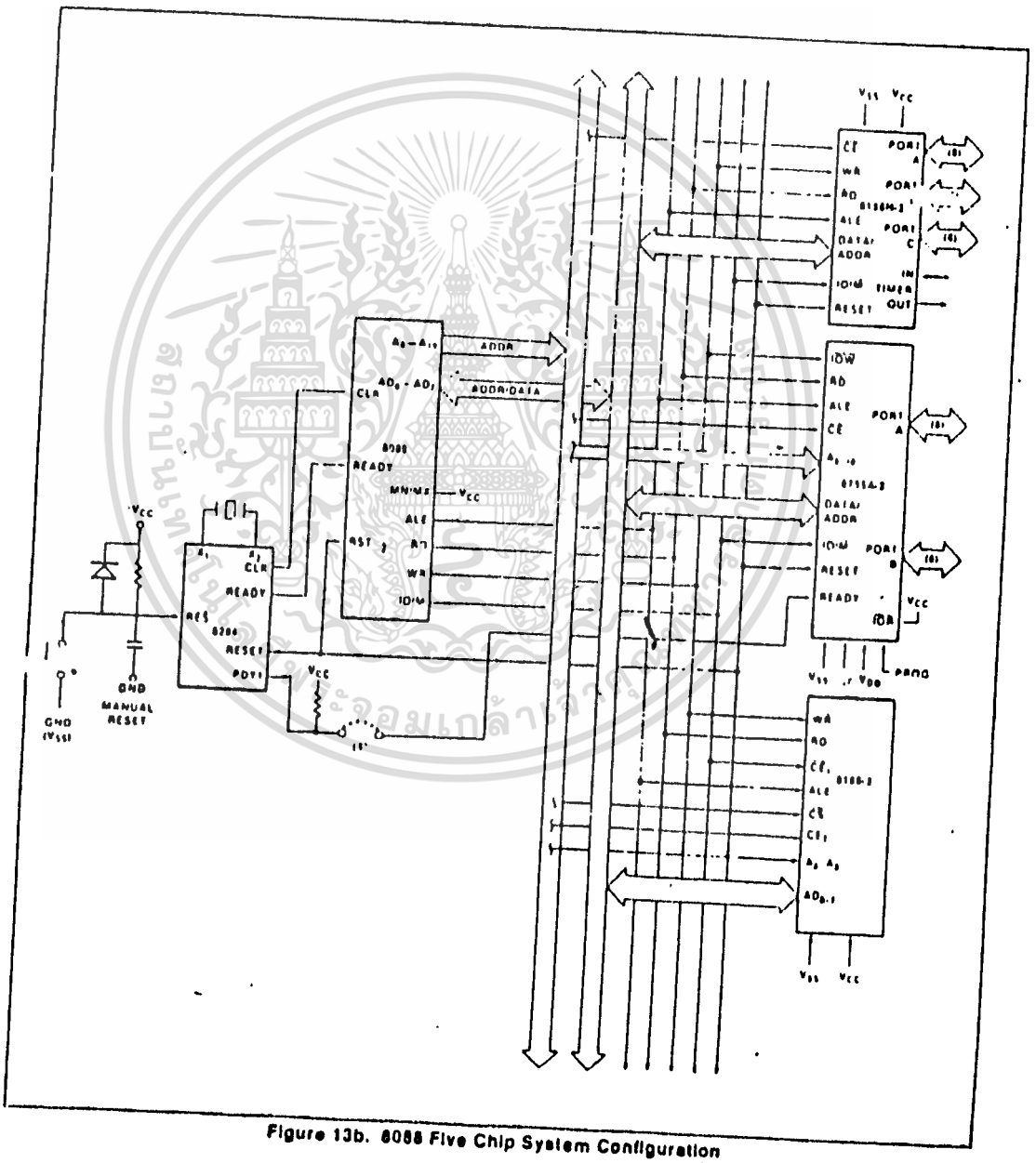


Figure 13b. 8088 Five Chip System Configuration

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin With Respect to Ground	-0.5V to +7V
Power Dissipation	1.5W

*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5V ± 10%)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA
I_{IL}	Input Leakage		±10	μA	$0V \leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current		± 10	μA	$0.45V \leq V_{OUT} \leq V_{CC}$
I_{CC}	V_{CC} Supply Current		125	mA	
I_{IL}(CE)	Chip Enable Leakage 8155H 8156H		 +100 -100	 μA μA	$0V \leq V_{IN} \leq V_{CC}$

A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5V ± 10%)

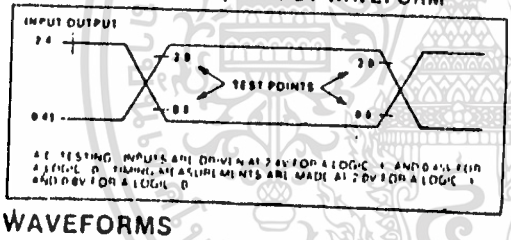
		8155H/8156H		8155H-2/8156H-2		
Symbol	Parameter	Min.	Max.	Min.	Max.	Units
t_{AL}	Address to Latch Set Up Time	50		30		ns
t_{LA}	Address Hold Time after Latch	80		30		ns
t_{LC}	Latch to READ/WRITE Control	100		40		ns
t_{RD}	Valid Data Out Delay from READ Control		170		140	ns
t_{LD}	Latch to Data Out Valid		350		270	ns
t_{AD}	Address Stable to Data Out Valid		400		330	ns
t_{LL}	Latch Enable Width	100		70		ns
t_{RDF}	Data Bus Float After READ	0	100	0	80	ns
t_{CL}	READ/WRITE Control to Latch Enable	20		10		ns
t_{CC}	READ/WRITE Control Width	250		200		ns
t_{DW}	Data In to WRITE Set Up Time	150		100		ns
t_{WD}	Data In Hold Time After WRITE	25		25		ns
t_{RV}	Recovery Time Between Controls	300		200		ns
t_{WP}	WRITE to Port Output		400		300	ns
t_{PR}	Port Input Setup Time	70		50		ns
t_{RP}	Port Input Hold Time	50		10		ns
t_{SBF}	Strobe to Buffer Full		400		300	ns
t_{SS}	Strobe Width	200		150		ns
t_{RBF}	READ to Buffer Empty		400		300	ns
t_{SI}	Strobe to INTR On		400		300	ns

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

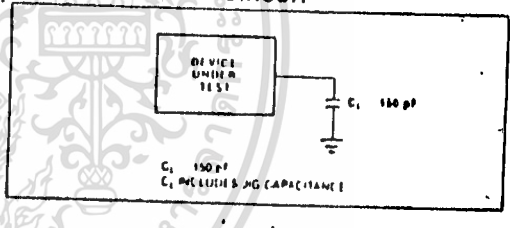
A.C. CHARACTERISTICS (Continued) (T_A = 0°C to 70°C, V_{CC} = 5V ± 10%)

Symbol	Parameter	8155H/8156H		8155H-2/8156H-2		Units
		Min.	Max.	Min.	Max.	
t_{NDI}	READ to INTR Off		400		300	ns
t_{PSS}	Port Setup Time to Strobe	50		0		ns
t_{PHS}	Port Hold Time After Strobe	120		100		ns
t_{SBI}	Strobe to Buffer Empty		400		300	ns
t_{WBI}	WRITE to Buffer Full		400		300	ns
t_{WI}	WRITE to INTR Off		400		300	ns
t_{TL}	TIMER-IN to TIMER-OUT Low		400		300	ns
t_{TH}	TIMER IN to TIMER OUT High		400		300	ns
t_{RDE}	Data Bus Enable from READ Control	10		10		ns
t_1	TIMER-IN Low Time	80		40		ns
t_2	TIMER-IN High Time	120		70		ns
t_{WT}	WRITE to TIMER-IN (for writes which start counting)	360		200		ns

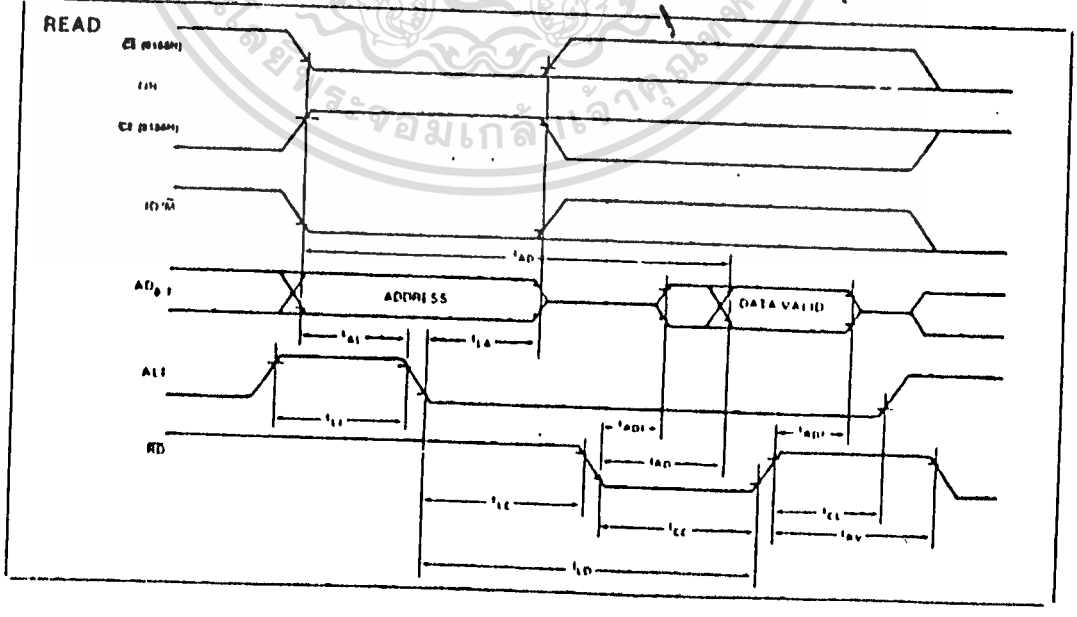
A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



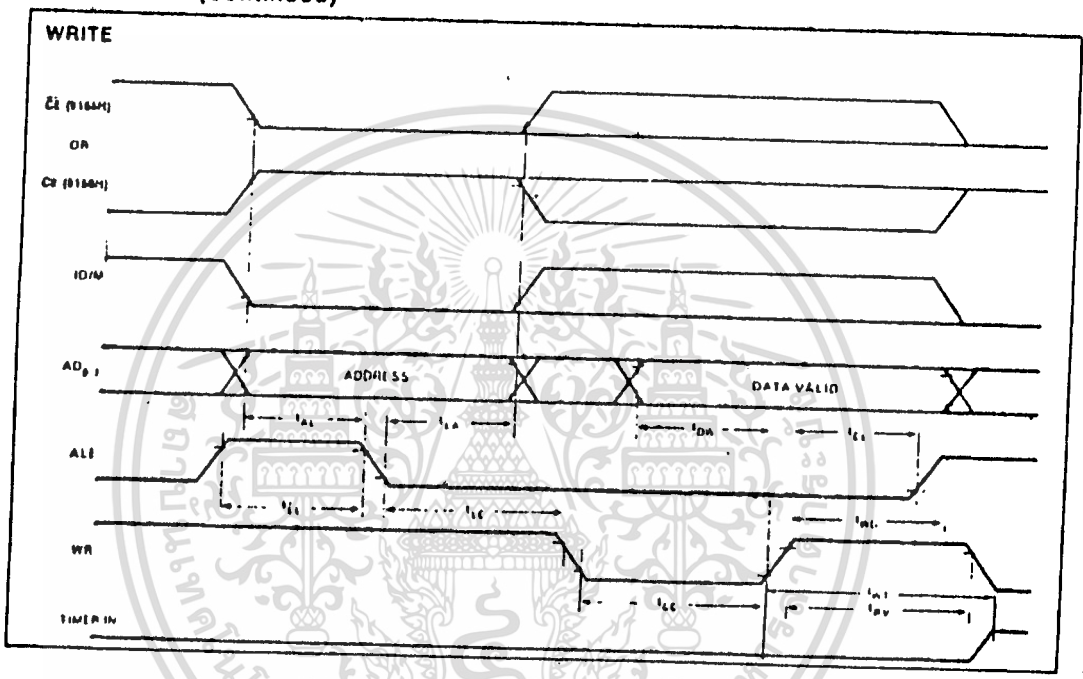
WAVEFORMS



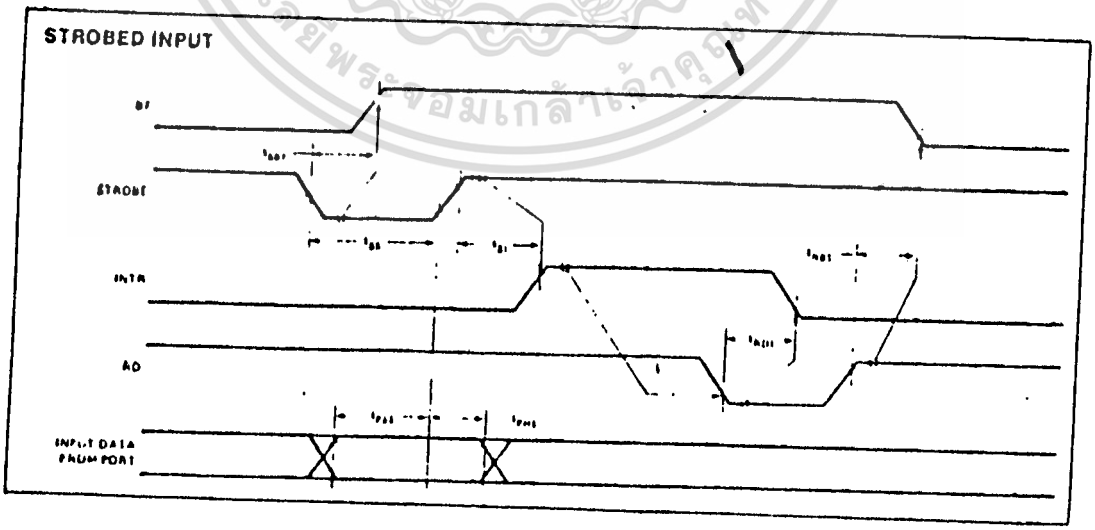
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

WAVEFORMS (Continued)

WRITE

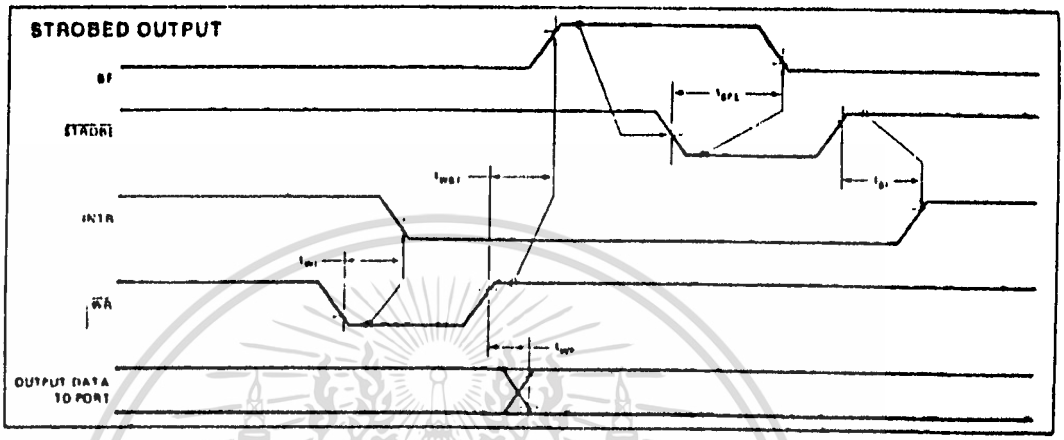


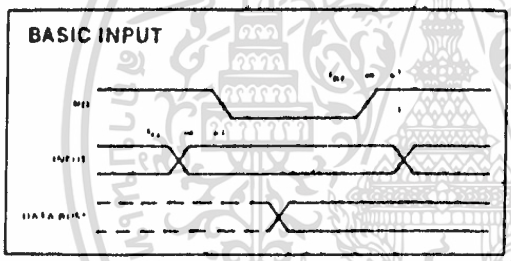
STROBED INPUT

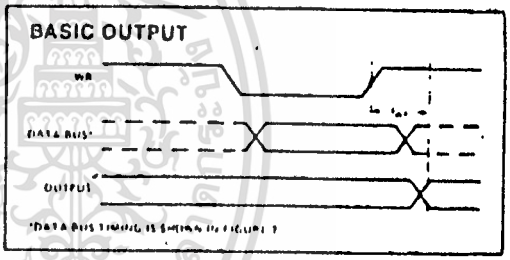


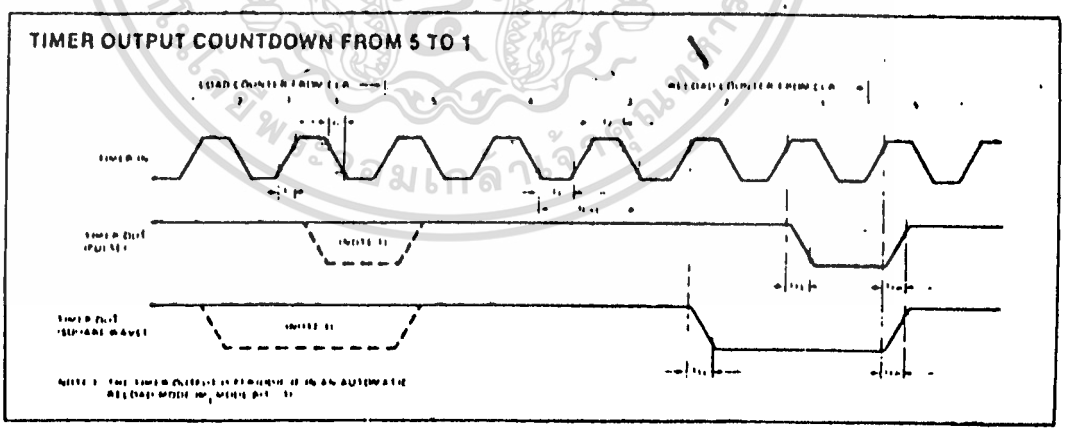
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

WAVEFORMS (Continued)









เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

THERMOMETER MODULE
MODEL TX-100
INSTRUCTION MANUAL

We would like to express our appreciation for your purchase of Model TX-100 Thermometer Module. You will find it to be a meticulously engineered, highly reliable instrument. To ensure that you get the most out of your new TX-100, we recommend that you read this manual carefully before use.

1. GENERAL

The TX-100 is multifunction module that measures temperature, clock, and provides temperature alarms.

It uses a thermistor sensor, and provides digital readout of temperature from -19.9°C to +69.8°C with 0.1°C resolution. Its ultra-small size and battery-holder case enable it to be installing in a wide range of equipment.

2. SPECIFICATIONS

Battery	: One AA cell (1.5V)
Temperature	: -19.9°C to +69.8°C, or 0°F to 159.6°F
Resolution	: 0.1°C (from 1°C to 29°C) : 0.2°C max. (outside the above range)
Temperature measurement accuracy	: ±1°C (0°C to 40°C, with 1.5V power)
Clock accuracy	: ±0.5s/day
Display	: °C, °F, time (hour : minute, 12-hour cycle), TEMP, HI, LO
Operating condition	: 0°C to 60°C, 60% RH Max. (No condensation)
Battery life	: One year (Carbon-zinc cell, clock mode or 10s temperature sampling)
Dimensions	: 65(H) × 35(W) × 26.5(D)mm
Weight	: Approx. 45g (incl. battery)
Accessories	: One AA battery

3. INSTRUCTIONS

3-1 Preparations and Precautions

(1) Place one AA battery in the battery case. Note the polarity marks.
(2) Avoid use in locations subject to high temperature, high humidity, or vibration.
(3) Avoid locations where there are abrupt temperature changes.
(4) If used near equipment that emits electrical noise, the TX-100 may give a less accurate or unstable reading.
(5) If the sensor attached to the case is used, the temperature measurement range should be restricted to the operating temperature range 0°C to 60°C.

3-2 Names of Parts

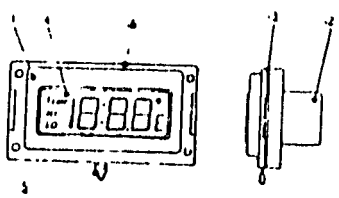


Fig 1

(1) Front case
(2) Rear case ····Also holds the battery
(3) Main board
(4) LCD
(5) Sensor ·····R_T = 40kΩ (TYP)
B = 3550K ±2%
(6) Output pins ···Function selection and alarm output signals

3-3 Pin Numbers and Names

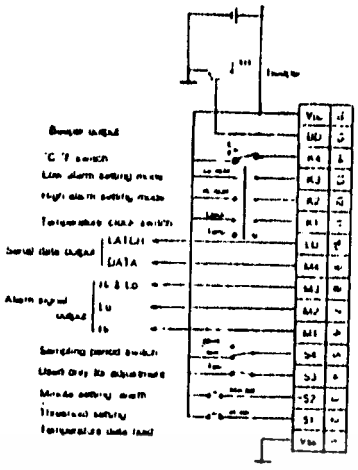


Fig 2

3-4 Thermometer Mode

When K1, K2 and K3 are all open, the TX-100 operates as a thermometer and displays the measured temperature on the LCD. The readout can be switched to °F units by providing Vss-level input to K4. The sampling period is 10s if S4 is open, or 1s if S4 is connected to Vss. When the TX-100 is not used in the thermometer mode, S4 should always be open, otherwise the time and temperature alarm thresholds cannot be set correctly. Temperatures from -19.9°C to +69.8°C can be measured, but if the sensor attached to the case is used, the measurement range should be restricted to the operating temperature range (0°C to 40°C). Temperatures below -19.9°C or above 69.8°C are displayed as "- -". This display may also indicate a broken sensor wire.

3-5 Temperature Alarm

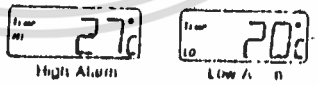
3-5-1 Temperature Alarm Setting

(a) Mode Selection

To set the high alarm threshold, connect K2 to Vss. To set the low alarm threshold, connect K3 to Vss. The temperature alarm function operates in the thermometer mode and alarm setting mode, but not in the clock mode.

(b) Set/Reset

In the high and low alarm threshold setting modes, the TX-100 switches between set and reset functions each time Vss-level input is applied to S1. When the set function is operative, the characters HI or LO are displayed on the LCD as in Fig. 3.



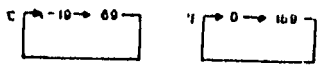
High Alarm Low Alarm

Fig 3

(c) Alarm Temperature Setting

In the alarm setting mode, the alarm threshold can be set by applying Vss-level input to S2. The alarm threshold value increases in 1° steps as shown below.

If S2 is held at the Vss-level for 2s or more, the value changes continuously. Apply input to S2 automatically places the TX-100 in the set mode.



(d) Alarm Beeper

When the temperature goes outside one of the alarm thresholds, a beeper signal is output at the BO pin as shown in Fig. 4. The beeper sounds for 5s. It can be stopped during this interval by applying Vss-level input to S1 or S2.

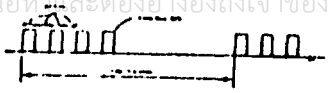


Fig 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3-5-2 Temperature Alarm Operation

(a) 10s Sampling Period

When the sampling period is 10s, the temperature alarm signals M1, M2 and M3 and the beeper signal BD hold their status for 1 minute, as indicated in Fig 5. After an alarm threshold is set by S7, the alarm function is inhibited for about 10s to 20s.

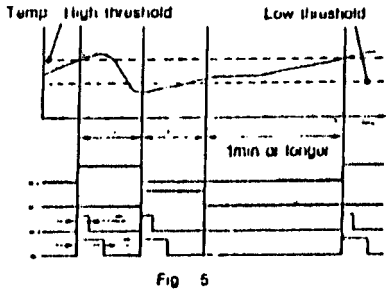


Fig 5

(b) 1s Sampling Period

The 1min. hold function described above does not operate.

3-6 Sensor

The TX-100 employs a thermistor temperature sensor. If a thermistor other than the one provided is used, it should have the following constants.

R = 40kΩ (TYP)

B = 3550K ± 2%

When using optional sensor (TP-100,170), Adjust the variable resistor.

The optimum resistance can be determined by placing the sensor in a constant-temperature bath, comparing the indicated temperature with the bath temperature, and choosing a resistance that makes the two agree.

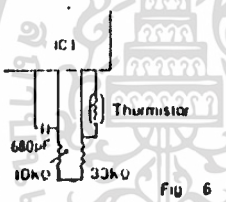


Fig 6

3-7 Clock Mode

When K1 is connected to Vss, the TX-100 operates in the 12-hour clock mode, indicating the hour and minute.

The time can be set by applying Vss-level input to S1 for the hour and S2 for the minute. The hour and minute setting are independent. The hour does not change when the minute passes 59.

To set the time, first apply Vss-level input to S1 or S2 for at least 2s. The colon will stop blinking. Thereafter, the hour or minute value increments by 1 each time Vss-level input is applied to S1 or S2. If the input is held for 2s or more, the value increments continuously.

The second value is reset each time the minute value is set. The colon resumes blinking 10s to 20s after the setting is completed.

3-8 Serial Data Output

The temperature measurement is provided as serial output on the M4 and LD pins. Figure 7 shows the timing of the output signals.

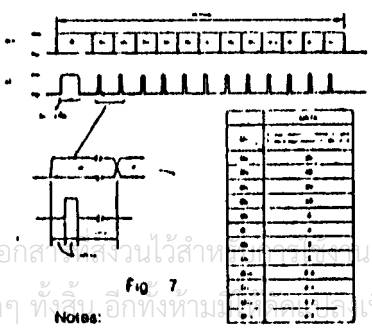


Fig 7

Notes:

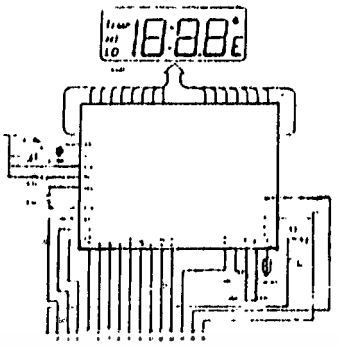
(1) Vss level represents 1, Vdd-level represents 0.

(2) All data are 1 when the temperature is outside the measurable range.

3-9 Test Mode

It is possible to display all segments of the LCD by providing Vss-level input to S1 and S3.

4. PIN-OUT DIAGRAM



SOAR CORPORATION

[illegible]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ทำซ้ำ ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของลิขสิทธิ์ทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โปรเจคนี้สำเร็จได้ด้วยความร่วมมือจากหลาย ๆ ฝ่ายขอขอบคุณอาจารย์พิพัฒน์ เลาหสงคราม อาจารย์ที่ปรึกษา ที่ให้ความรู้เกี่ยวกับการทำงานของเครื่องปรับอากาศและต้องขอขอบคุณอาจารย์ภายในภาควัดคุมฯ ทุกท่านที่ได้ให้คำปรึกษารวมทั้งเพื่อน ๆ ภายในห้องที่ได้ให้ข้อมูลที่เป็นประโยชน์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. วิบูลย์ ชื่นแขก ; "ไมโครโปรเซสเซอร์"
2. อุดม จีนประดับ/วรพงษ์ รัตนโภคา ; "การทดลองไมโครโปรเซสเซอร์ 2"
3. สมศักดิ์ สุโมตยกุล ; "เครื่องทำความเย็นและเครื่องปรับอากาศ"
4. ผศ.พิพัฒน์ เลาหสงคราม ; "โครงสร้างสถาปัตยกรรม MCS-51"
5. Intel ; "Intel PL/M-51 User's Guide for DOS Systems"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้