ปีการศึกษา 2532

แผงวงจร MCS - 48 สามารถทำงานด้วยตัวเอง

โดย

นายยอเกียรติ ไชยเอียด

นายวรวิทย์ ชูพงศ์

นายนพรัตน์ ตะวันรอน

อาจารย์ที่ปรึกษา

รศ. พิพัฒน์ เลาหสงคราม

ปริญญานิพนธ์ปีการศึกษา 2532

เรื่อง แผงวงจร MCS-48 ทำงานด้วยตนเอง

ผู้จัดทำ

1. นายยอเกียรติ ไชยเอียด
2. นายวรวิทย์ ชูพงศ์
3. นายนพรัตน์ คะวันรอน

_____ อาจารย์ที่ปรึกษา

(ศิริชัย พานรงสง)

_____ อาจารย์ที่ปรึกษา

(_____)

_____ อาจารย์ที่ปรึกษา

(_____)

แผงวงจร MCS-48 สามารถทำงานด้วยตัวเองได้

นายยอเกียรติ ไชยเอียด
นายวรวิทย์ ชูพงศ์
นายนพรัตน์ ตะวันรอน
รศ. พิพัฒน์ เลาหสงคราม
อาจารย์ที่ปรึกษา

2532

บทคัดย่อ

เมื่อกล่าวถึงไมโครคอลโทรลเลอร์ ทุกคนคงรู้ว่าเป็น ซีพียู ประเภทหนึ่งที่นำมาใช้ในงานระบบควบคุมกระบวนการต่าง ๆ ซึ่ง ซีพียูประเภทนี้มีด้วยกันหลายตระกูล แต่ละตระกูลก็มีข้อดีข้อเสียที่ต่างกัน การที่จะนำ ซีพียู ตระกูลใดไปใช้งาน จะต้องพิจารณาถึงคุณสมบัติและความต้องการของงานเป็นหลักก่อนที่จะตัดสินใจเลือก ซีพียู ตระกูลใดไปใช้งาน

ปริญญานิพนธ์ฉบับนี้ประกอบด้วยวิธีการสร้างแผงวงจรเอ็มซีเอส 48 ที่สามารถทำงานด้วยตัวเอง และวิธีขยายหน่วยความจำของ ซีพียู ตระ เอ็มซีเอส 48 ตลอดจนการนำ แรมสองทางมาใช้ช่วยในการพัฒนา เพื่อเป็นประโยชน์แก่ผู้ที่จะนำแผงวงจร เอ็มซีเอส 48 ที่สามารถทำงานด้วยตัวเองไปใช้งานทางด้านการควบคุมกระบวนการต่าง ๆ ในตอนท้ายของปริญญานิพนธ์ได้แสดงการออกแบบและการพัฒนาซอฟท์แวร์ เพื่อนำมาใช้เป็นโปรแกรมมอนิเตอร์ โดยผู้ใช้หรือผู้ควบคุมสามารถกำหนดเงื่อนไขต่าง ๆ ได้ง่ายด้วยตนเอง

MCS-48 SINGLEBOARD STAND ALONE

YONGKIAT CHIYAEAD

VARAVIT CHOPOONG

NOPPARUST TAWANRONG

PHIPHAT LAOHASONGKRAM

ADVISOR

2532

ABSTRACT

When we talk about Micro controller,everyone knows that it is a kind of CPU which was brought to control various processing systems, CPU has many sorts and every sort has its strength and weakness, bringing what sorts of CPU to use must consider its quality and potential before decide to choose.

This thesis comprises how to make MCS-48 Single board stand alone, how to increase the memories of CPU of MCS-48 and how to put RAM-2 LINE to develope the work. It is useful for the one who use MCS-48 Single board stand alone to control the various processes. IN the end of the thesis has showed designing and developing the software, in order to make Program Monitor for user or controller. Can easily set many conditions by himself.

สารบัญ

บทนำ

เนื่องจากในปัจจุบันcomputer ได้เข้ามามีบทบาทอย่างมากในชีวิตประจำวันของเรา โดยเฉพาะในสาขาวิศวกรรมศาสตร์ได้มีการนำเอา computer มาช่วยงานในด้านต่าง ๆ มากมาย เช่นงานด้านประเมินผลภาพถ่ายทางดาวเทียม ได้นำ computer มาช่วยให้ได้ภาพที่มีข้อมูลถูกต้องแม่นยำขึ้น การใช้computer graphic มาใช้เป็นตัวสื่อความหมาย และอีกเรื่องหนึ่งที่จะไม่กล่าวถึงไม่ได้ คือ การนำ computerเป็นตัวควบคุมซึ่งมีความเกี่ยวข้องโดยตรงกับงานในด้านนี้ ในการนำ computer มาใช้งานในด้านควบคุมกรือเรียกกันติดปากว่า Micro controller ได้มีการใช้งานกันอย่างแพร่หลายทั้งในโรงงานอุตสาหกรรม และในงานที่ต้องการควบคุมเฉพาะอย่าง ซึ่งจากที่ผ่านมามีอุปสรรคมากมาย ในการทำงานด้านนี้ ผู้ทำหรือผู้สร้างต้องหาหนทางหรือวิธีแก้ปัญหาเอาเอง ซึ่งขึ้นอยู่กับความสามารถความชำนาญเฉพาะตัว ยากต่อการที่จะอธิบายให้คนอื่นเข้าใจได้ง่าย ทำให้เกิดปัญหาในการพัฒนาระบบนั้นต่อไปได้เพื่อลดปัญหาได้มีผู้คิดเครื่องมือเข้ามาช่วยงานในด้านนี้มากมาย เพื่อลดความยุ่งยากของขบวนการลง ทำให้สามารถเข้าใจได้ง่าย Emulator ก็เป็นเครื่องมือชนิดหนึ่งที่เป็นตัวช่วยให้การทำงานในด้านนี้เร็วขึ้น ซึ่งทางคณะผู้จัดทำก็ได้พัฒนา Emulator โดยใช้ Micro controller ตระกูล MCS-48 เป็นตัวหลัก โดยในระยะแรก คณะผู้จัดทำได้พัฒนาตัว Emulatormu ที่ว่านี้ให้สามารถที่จะรับการป้อนข้อมูลจากคีย์บอร์ด มาเก็บไว้ในRAMของตัว Emulatorซึ่งสามารถขยายได้ถึง8KBและนำมาประมวลผลแล้วนำผลที่ได้ส่งไปยังพอร์ตต่อไป นอกจากนั้นใน Emulatorมีการแสดงสถานะต่าง ๆด้วยการใช้ LED 7Segment ซึ่งในระยะต่อไปได้มีการเตรียมแผนที่จะพัฒนา Emulatorให้มีฟังก์ชันคีย์มาช่วยในการใช้งานได้สะดวกขึ้น พร้อมทั้งจัดทำคู่มือการใช้Emulatorตัวนี้เพื่อเพิ่มความสะดวกให้ผู้ใช้ที่ต้องการนำ Emulatorตัวนี้ไปใช้งาน

ปริญญานิพนธ์ฉบับนี้เป็นส่วนที่สำคัญในการศึกษา Single Board Stand Alone โดยใช้ MCS-48 เป็นตัวControlerภายในปริญญานิพนธ์จะกล่าวถึงโครงสร้างของEmulatorว่าประกอบด้วยอุปกรณ์อะไร การต่ออุปกรณ์ภายในEmulatorร่วมกับMCS-48 ต่ออย่างไร นอกจากนั้นปริญญานิพนธ์ฉบับนี้ยังกล่าวถึงวิธี

การสร้างEmulator และการทดลองไว้อย่างละเอียด รวมทั้งปัญหาที่เจอวิธีแก้ปัญหาที่นำมาใช้ ข้อดีข้อเสียของEmulator ตลอดจนข้อเสนอแนะที่ทางคณะผู้จัดทำเห็นว่าเป็นประโยชน์ในการพัฒนาและปรับปรุงตัว Emulatorต่อไป

บทที่ 2

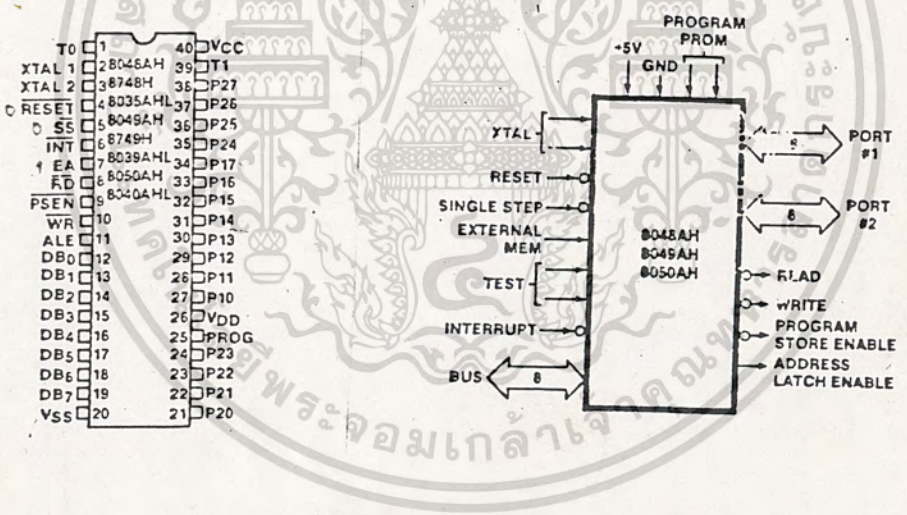
โครงสร้างสถาปัตยกรรมของ MCS - 48

2.1 บทนำ MCS - 48

ลักษณะโครงสร้างของ MCS - 48 แบ่งเป็น 2 ลักษณะ เช่นเดียวกับไมโครโพรเซสเซอร์ ทั่วไป คือ

1. โครงสร้างภายนอก
2. โครงสร้างภายใน

2.1.1 โครงสร้างภายนอก ดูจากรายละเอียดของขาต่าง ๆ ใน MCS -48 ขาทั้งหมดของ MCS - 48 ยกเว้นขาไฟเลี้ยงและขาสัญญาณนาฬิกาแล้ว ทุกขาทำหน้าที่เป็นสัญญาณอินพุตและเอาต์พุต โดยมีรายละเอียดของขาต่าง ๆ ดังนี้



รูปที่ 2.1 ลักษณะการจัดขาภายนอกของ MCS - 48

ขา 1 T0 สามารถทดสอบสัญญาณเข้า เพื่อใช้ในการถ่ายเทข้อมูลต่าง ๆ ตามข้อกำหนดที่ตั้งไว้ โดยการใช้คำสั่ง JT0 และ JNT0 และสามารถที่จะใช้T0 เป็น สัญญาณนาฬิกาส่งออก ได้ด้วยคำสั่ง ENT0 CLK T0 ยังถูกใช้เป็นขาควบคุมโหมดการโปรแกรมมิ่งและซิ้งได้ด้วย

ขา2 XTAL 1 สำหรับต่อขาข้างหนึ่งของคริสตอลภายนอก เพื่อ ใช้ออสซิลเลเตอร์ภายใน หรือ แหล่งกำเนิดสัญญาณจากภายนอกต่อเข้าขานี้

ขา 3 XTAL 2 สำหรับต่อขาอีกข้างหนึ่งของคริสตอล

ขา 4 RESET เป็นอินพุตสำหรับใช้ในการให้โพรเซสเซอร์เริ่มทำงานและยังเป็นสัญญาณ สำหรับ

แลทช์ ตำแหน่งของหน่วยความจำโปรแกรมภายในเพื่อประโยชน์ในการอ่านรหัสข้อมูลโปรแกรม

ขา 5 SS ขาซิงเกิ้ลแสตป จะใช้ร่วมกับ ALE เพื่อทดสอบคำสั่ง

ขา 6 INT เป็นขาอินพุตรับสัญญาณอินเตอร์รัพต์ภายนอก ถ้าต้องการจะใช้อินเตอร์รัพต์ได้ จะต้องสั่งด้วยซอฟต์แวร์ ให้เริ่ม

แรกด้วยคำสั่ง EN I สัญญาณอินเตอร์รัพต์ที่เข้ามาจะต้องอยู่ในสถานะต่ำอย่างน้อย 3 วัฏจักรแมชีน เพื่อ ให้แน่ใจว่าตัวโพรเซสเซอร์จะรับหรือแซมปิ้งอินเตอร์ รัพต์ได้ ปกติหลังจากรีเซตหรือเปิดเครื่องทุกครั้ง จะไม่สามารถใช้งานด้านอินเตอร์รัพต์ได้

ขา 7 EA เมื่อเป็นสถานะตรรกสูง จะเป็นการติดต่อรับรหัสข้อมูลจากภายนอกของชิงเกิ้ลชิป ตัวนี้ และควบคุมให้ตัวนับ

โปรแกรม (PROGRAM COUNTER) แฟทช์ รหัสจากโปรแกรมภายนอกทั้งหมด เพื่อใช้ประโยชน์ในการแก้ไข

และทำอีมูเลตหรือทดสอบกับเครื่องต้นแบบ

ขา 8 RD จะเป็นสัญญาณควบคุมการรับหรืออ่านข้อมูลจากภายนอกผ่านเข้าขาข้อมูลระหว่างการอ่าน และสัญญาณนี้จะสโตรป

ทุกครั้งที่มีการอ่านข้อมูลไม่ว่าจะอ่านรหัสจากภายนอกหรือภายใน

ขา 9 PSEN เป็นสัญญาณ PROGRAM STORE ENABLE สัญญาณควบคุมการอ่านรหัสคำสั่งทุกคำจากหน่วยความจำภายนอก และสัญญาณนี้จะเอ็กทิฟต่ำ เมื่อมีการแฟทช์ข้อมูลโปรแกรมจากภายนอกเท่านั้น

ขา 10 WR เป็นสัญญาณควบคุมการส่ง หรือเขียนข้อมูลออกภายนอกผ่านบัสข้อมูล และสัญญาณนี้จะสโตรปออกทุกครั้งที่มีการเขียนข้อมูล

ขา 11 ALE เป็นสัญญาณ ADDRESS LATCUH ENABLE สัญญาณนี้จะเกิดทุกวัฏจักรของการแฟทช์ในแต่ละครั้ง และใช้ประโยชน์ในการส่งสัญญาณนาฬิกาเอาต์พุตด้วย

การใช้สัญญาณชอบขา ลงของสัญญาณนี้ เป็นการสโตรปแลทช์แอดเดรส ข้อมูลจากภายนอกในตัวหน่วยความจำโปรแกรม

ขา 12-19 D0-D7 เป็น BIDIRECTIONAL PORT ทำหน้าที่ต่าง ๆ ดังนี้

1. เป็นที่ส่งผ่านข้อมูลเข้าและออก ในขณะที่ซิงค์ด้วยสัญญาณ RD หรือ WR สโตรป และสามารถทำหน้าที่เป็นพอร์ต สำหรับแลทช์ข้อมูล

2. ใช้เป็นตัวส่งข้อมูลตัวนับโปรแกรมบิตอันดับต่ำ ระหว่างการแฟทช์โปรแกรมจากภายนอก และรับรหัสคำสั่งภายใต้การส่งสัญญาณควบคุม PSEN

3. ใช้เป็นตัวส่งข้อมูลทั้งแอดเดรสและข้อมูลในการทำงานกับแรมภายนอกด้วยคำสั่ง MOVX (STORE) โดยมีขาควบคุมALE RD และ WR

ขา 20 VSS เป็นสายดินของวงจร

ขา 21-24 P20-P28 สี่บิตแรกของ QUASI-BIDIRECTIONAL พอร์ต 2 ใช้เป็นตัวส่งค่า

ขา 35-38 P24-P27 อันดับสูงของแอดเดรสตัวนับโปรแกรม และใช้ในการติดต่อกับหน่วยความจำภายนอก และใช้เป็น 4 บิตไอโอบัสสำหรับการขยายออกสำหรับใช้กับพอร์ตไอโอของ 8243 และ อีกสี่บิตอันดับสูงใช้เป็นพอร์ต

ขา25 PROG ใช้เป็นสัญญาณเอาต์พุตสโตรบสำหรับควบคุมการใช้ร่วมกับการขยายพอร์ตไอโอกับแรม 8243 และใช้ป้อนโปรแกรมพัลส์ขนาด+18V เข้าขานี้สำหรับการเขียนโปรแกรมเข้า 8748/8749

ขา 26 VDD ปกติตัว MCS-48 นี้จะทำงานได้ด้วยการจ่ายแรงดัน 5 โวลต์ จ่ายเข้าที่ขานี้และถ้ามีสถานะต่ำเข้าที่ขานี้จะทำให้ MCS-48 เข้าโหมดการใช้แหล่งจ่ายไฟสำรองและการอัดโปรแกรมเข้า ชิงเกิ้ลชิป 8048/8749ด้วยการจ่ายแรงดันไฟ 21 โวลต์เข้าที่ขานี้

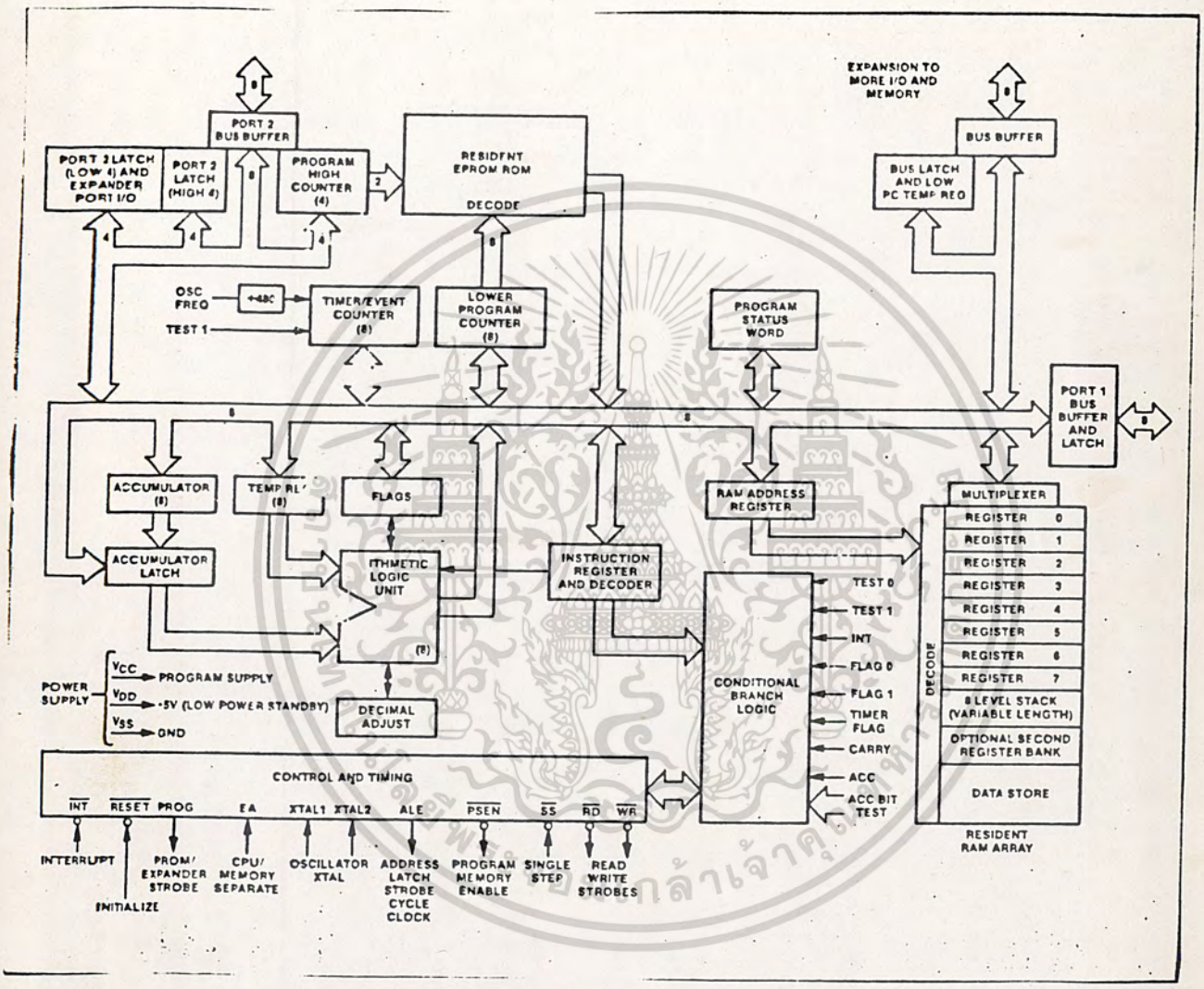
ขา 27-34 P10-P17 เป็นพอร์ตขนาด 8 บิตแบบ QUASI- BIDIRECTIONAL พอร์ต 2 โดยมี 50 K OHMSพูลอัพภายใน และขาเอาต์พุตจะเป็นแบบสามสถานะ

ขา 39 T1 เป็นสัญญาณทดสอบเข้าในไมโครโพรเซสเซอร์ ด้วยคำสั่ง JT1,JNT1 สามารถใช้ขา T1 ให้เป็นตัวนับเหตุการณ์ที่เกิดขึ้นจากภายนอก ส่งเข้ามาด้วยการใช้คำสั่ง STRT CNT (START COUNTER)

ขา 40 Vcc จ่ายแรงดันไฟ 5 โวลต์เข้าที่ขานี้ ในขณะที่ทำงานปกติ และขณะที่อัดโปรแกรมเข้า 8748/8749

4.1.2 โครงสร้างสถาปัตยกรรมภายในของ MCS-48

โครงสร้างภายในของ MCS-48 เหมือนกับโครงสร้างของไมโครคอมพิวเตอร์ทั่วไป จะประกอบด้วยบัสที่ทำหน้าที่เชื่อมต่อกับอุปกรณ์อื่น ๆ โดยมีส่วนประกอบภายในต่าง ๆ ดังรูปที่ 2.2 และรายละเอียดของแต่ละส่วนจะเป็นดังนี้



รูปที่ 2.2 โครงสร้างสถาปัตยกรรมภายในของ MCS-48

2.1.2.1 <u>หน่วยคำนวณคณิตศาสตร์</u> (ALU) ALU จะรับหรือทำงานร่วมกับข้อมูลจากแหล่งข้อมูล 1 กับ 2 แหล่ง เพื่อไปทำการประมวลผลทางคณิตศาสตร์ภายใต้การควบคุมของตัวถอดรหัสคำสั่ง ตัว ALU มีความสามารถที่จะทำงานตามหน้าที่ต่าง ๆ ดังต่อไปนี้

1. บวกด้วยตัวทดหรือปราศจากตัวทด (ADD WITH OR WITHOUT CARRY)

2. ทำงานทางตรรก (AND,OR,EXCLUSIVE OR)

3. การเพิ่มหรือลดค่าหนึ่งค่า (INCREMENT/DECREMENT)

4. การแปลงกลับค่าบิต (BIT COMPONENT)

5. วนบิตทางซ้ายหรือขวา (LEFT OR RIGHT ROTATE)

6. การสลับค่านิบเบิ้ล (SWAP NIBBLE)

7. การปรับค่าเป็น BCD DECIMAL (BCD DECIMAL ADJUST)

ถ้าการทำงานของ ALU ให้ผลลัพธ์มากกว่า 8 บิต จะเกิดตัวทดจากบิตหลักสูงสุด (MOST SIGNIFICANT BIT MSB) ตัวแฟลกทด (CARRY FLAG) ภายใน PSW (PROGRAM STATUS WORD) จะถูกเซ็ต

2.1.2.2 <u>แอกคิวมิวเลเตอร์</u> จะเป็นตัวเรจิสเตอร์ที่สำคัญที่สุดของ MCS-48 ในตัวโพรเซสเซอร์ จะทำหน้าที่เป็นตัวรับข้อมูลจากแหล่งข้อมูลที่ส่งเข้ามา และเป็นตัวส่งผลลัพธ์ให้กับเรจิสเตอร์ตัวรับ (DESTINATION REGISTER) ในการทำงานของ ALU การติดต่อกับอุปกรณ์ภายนอกและติดต่อกับหน่วยความจำ จะต้องผ่านแอกคิวมิวเลเตอร์เสมอ

2.1.2.3 <u>แฟลกตัวทด</u> (CARRY FIAG) เป็นตัวบอกสถานะการทำงานของ MCS-48 ว่าการทำงานเกินจำนวนบิตที่มีอยู่หรือไม่ ถ้าเกินแฟลกตัวทดนี้จะถูกเซ็ต

2.1.2.4 <u>ตัวถอดรหัส</u> (INSTRUCTION REGISTER AND DECODER) รหัสการทำงาน (OPERATION CODE = OP CODE) ของแต่ละคำสั่งจะเก็บอยู่ในตัวถอดรหัสคำสั่งนี้ จากนั้นจะถูกเปลี่ยนเป็นสัญญาณในการควบคุมตามหน้าที่ของคำสั่งนั้นต่อไป

2.1.3 หน่วยความจำ (MEMORY)

หน่วยความจำของ MCS-48 นั้นแบ่งการทำงานออกได้เป็น 2 ชนิด คือ

2.1.3.1 หน่วยความจำโปรแกรม (PROGRAM MEMORY)

หน่วยความจำนี้จะใช้สำหรับเก็บชุดคำสั่งการทำงานของ MCS-48 ที่เขียนขึ้น ขนาดหน่วยความจำสูงสุดของส่วนนี้ คือ 4 กิโลไบต์ โดยส่วนหนึ่งจะอยู่ภายในชิปของไอซี และอีกส่วนหนึ่งเป็นการต่อเพิ่มภายนอก ในกรณีที่หน่วยความจำโปรแกรมภายในชิป จะมีขนาดความจุและชนิดต่าง ๆ กัน แล้วแต่เบอร์ในตระกูล MCS-48 ดังตารางที่ 2.1

เบอร์	หน่วยความจำภายใน		RAM สำรอง
	โปรแกรม ROM	ข้อมูล RAM	
8050 AH	4K x 8	256 x 8	มี
8049 AH	2K x 8	128 x 8	มี
8048 AH	1K x 8	64 x 8	มี
8040 AHL	ไม่มี	256 x 8	มี
8039 AHL	ไม่มี	128 x 8	มี
8035 AHL	ไม่มี	64 x 8	มี
8749 H	2K x 8 (EPROM)	128 x 8	ไม่มี
8748 H	1K x 8 (EPROM)	64 x 8	ไม่มี

ตารางที่ 2.1 เป็นรายละเอียดต่าง ๆ ของตระกูล MCS-48

ตำแหน่งที่สำคัญของหน่วยความจำส่วนนี้ คือ

เลขที่อยู่ 0 (ADDERSS 000) เมื่อถูกรีเซต MCS-48 จะทำการอ่านหรือเฟทช์คำสั่งตัวแรกที่ตำแหน่งนี้

เลขที่อยู่ 3 (ADDRESS 003) MCS-48 จะอ่านคำสั่งแรกที่ตำแหน่งนี้เมื่อมีสัญญาณอินเตอร์รัพต์เข้ามา ถ้าอินเตอร์รัพต์อีนาเบิ้ล จะเป็นเหตุให้โปรแกรมบริการอินเตอร์รัพต์ที่ตำแหน่งนี้

เลขที่อยู่ 7 (ADDRESS 007) MCS-48 จะอ่านคำสั่งแรกที่ตำแหน่งนี้ เมื่อมี

การอินเตอร์รัพต์ทำงานแบบจับเวลา/ตัวนับ มีผลจากการที่ตัวจับเวลา/ตัวนับเวลาเกิด Overflow

ตำแหน่งแรกของทั้ง 3 ตำแหน่งดังกล่าวโดยปกติแล้วจะเขียนด้วยคำสั่ง JUMP ไปยังโปรแกรมบริการต่าง ๆ ที่เขียนไว้ ดังนั้น คำสั่งกระโดดตัวแรกจะถูกใช้งานหลังจากที่โปรแกรมเริ่มทำงาน (Initialize) ถูกเก็บเข้าที่เลขที่อยู่ 0 ในการทำงานเริ่มต้น คำสั่งของการกระโดดไปใช้บริการอินเตอร์รัพต์จากภายนอก เริ่มแรกจะถูกเก็บเข้าที่เลขที่อยู่ 3 และคำสั่งของการกระโดดไปใช้บริการโปรแกรมการใช้อินเตอร์รัพต์ตัวจับเวลา/ตัวนับเริ่มแรกจะถูกเก็บเข้าที่เลขที่อยู่ 7

หน่วยความจำโปรแกรมสามารถที่จะใช้เป็นตัวเก็บข้อมูลคงที่ และโปรดแกรมคำสั่งด้วยการใช้คำสั่ง MOVP และ MOVP3 ช่วยให้การดึง ตารางข้อมูลคงที่ทำได้ง่ายขึ้น

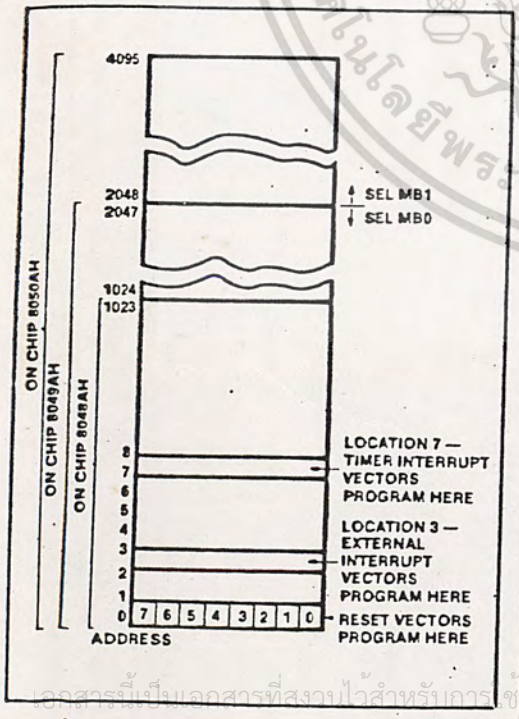
2.1.3.2 <u>หน่วยความจำข้อมูล</u>

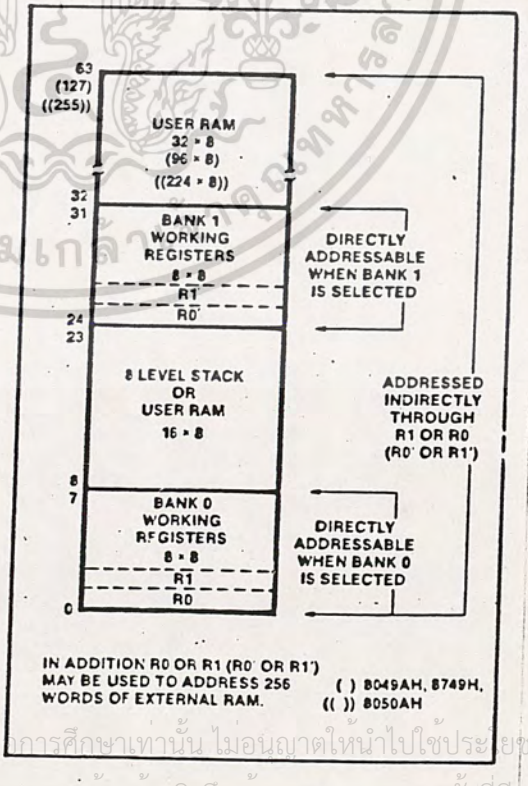
หน่วยความจำส่วนนี้จะเป็นชนิดแรม ที่อยู่ภายในชิปของ MCS-48 โดยมีขนาด 64,128 และ 256 ไบต์ ขึ้นอยู่กับเบอร์ของชิปตระกูล MCS-48 ดังตารางที่ 4.1 ภายในส่วนของหน่วยความจำส่วนนี้จะทำหน้าที่เป็นเจริสเตอร์ใช้งาน 16 ตัว โดยแบ่งเป็น 2 ชุด ๆ ละ 8 ตัว (หรือแบงค์ 1 ในขณะที่ MCS-48 ทำงานอยู่นั้น เรจิสเตอร์จะถูกใช้งานเพียงชุดเดียว โดยสามารถจะเลือกชุดของเรจิสเตอร์จากคำสั่ง SEL RB (Register Bank Switch) เป็นลักษณะที่สามารถจะเลือกเลขที่อยู่โดยตรง (Direct Address) ได้ จากเลขที่อยู่ 0 - 7 ในแบงค์ 0 และ แบงค์ 1 อาจใช้งานเป็นเรจิสเตอร์ขยายจาก แบงค์แรก หรือใช้เป็นเรจิสเตอร์สำรองสำหรับการใช้งานระหว่างการใช้งานโปรแกรมย่อย และในโปรแกรมหลักยังคงใช้ แบงค์ แรกในการเก็บข้อมูลโดยทันทีด้วยแบงค์สวิตช์ ขณะเดียวกันถ้าแบงค์ 1 ไม่ถูกนำมาใช้ เลขที่อยู่ 24-31 ยังคงใช้เป็นแอดเดรสด้วยการเก็บข้อมูลแบบแรมได้

นอกจากทำหน้าที่เป็นเรจิสเตอร์แล้วที่หน่วยความจำข้อมูลเลขที่อยู่ 8 - 23 ก็จะเป็นเนื้อที่สแต็กด้วยการเก็บตัวนับโปรแกรมเป็นคู่ โดยใช้งานครั้งละ 2 ตำแหน่งหน่วย

ความจำตำแหน่งเหล่านี้จะถูกแอดเดรสด้วยตัวชี้แรมของ R0 และ R1 การใช้พื้นที่นี้เหมาะสำหรับการใช้โปรแกรมย่อยหลายโปรแกรมเชื่อมโยงกัน แต่ต้องต่ำกว่า 8 โปรแกรมย่อย เพราะเก็บได้เพียง 8 คู่ ทำนองเดียวกัน พื้นที่ของสแต็กนี้ ถ้าไม่ถูกเป็นเรจิสเตอร์สแต็ก ก็สามารถที่จะใช้เป็นที่เก็บข้อมูล แรมทั่วไปได้ แต่ถ้าใช้เป็นสแต็กสำหรับโปรแกรมย่อยแล้ว ก็ไม่สามารถที่จะใช้เป็นที่เก็บข้อมูลแบบ แรมในเวลาเดียวกัน

ในกรณีที่ผู้ใช้ต้องการใช้เนื้อที่แรมมากกว่าที่มีอยู่ในตัว MCS-48 ก็สามารถเพิ่มแรม ภายนอกได้ โดยใช้คำสั่ง MOVX R,A เมื่อใช้ R เป็นเรจิสเตอร์ 0 หรือ เรจิสเตอร์ 1 ทำหน้าที่เป็นตัวชี้ในการติดต่อกับแรมที่เพิ่มขึ้นภายใน 256 ไบต์ หน่วยความจำส่วนนี้โดยปกติจะใช้สำหรับเก็บข้อมูลชั่วคราว เมื่อมีการอ่านข้อมูลเข้าสู่ใน MCS-48 จะมองข้อมูลส่วนนี้ผิดกับข้อมูลที่อยู่ในส่วนของหน่วยความจำโปรแกรมที่กล่าวมาแล้ว ขณะเดียวกัน การใช้เรจิสเตอร์ตัวชี้ R0 และ R1 ของแรม สามารถที่จะเพิ่มเรจิสเตอร์ตัวชี้ได้อย่างง่าย ๆ ด้วยการใช้คำสั่ง BANK SWITCH เมื่อต้องการใช้ตำแหน่งข้อมูลกับแรมถึง 4 ตำแหน่งในการเข้าถึงข้อมูลแต่ละครั้ง



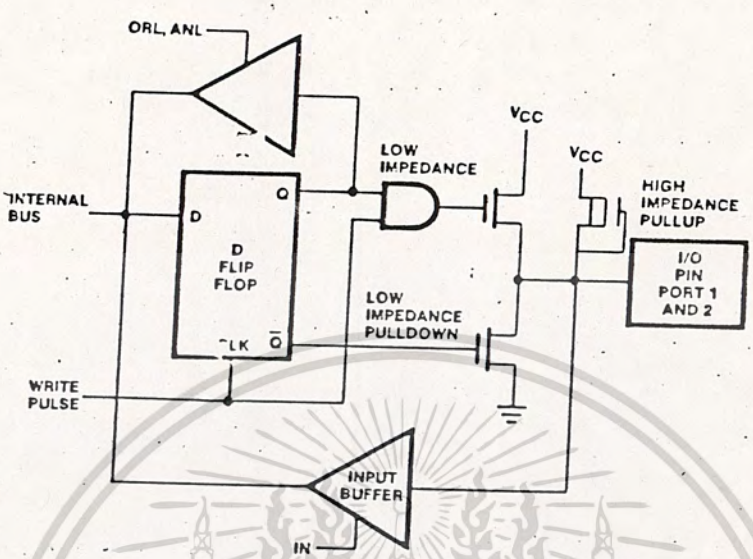


รูปที่ 2.3 PROGRAM MEMORY MAP รูปที่ 2.4 DATA MEMORY MAY

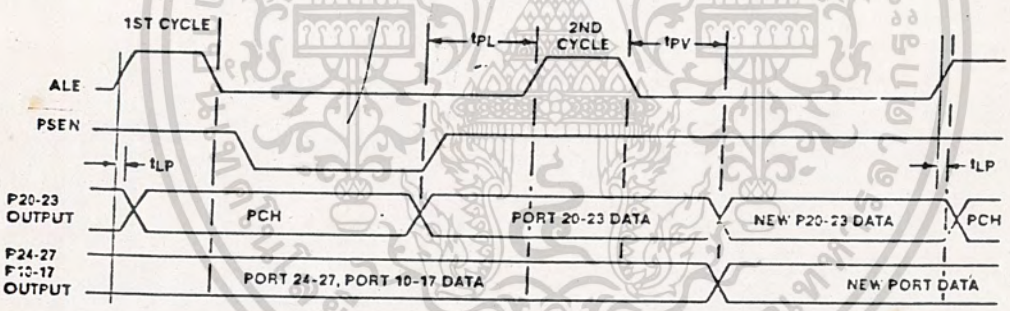
2.1.4 ส่วนติดต่ออุปกรณ์ภายนอก อินพุต/เอาต์พุต

ในส่วนการติดต่ออุปกรณ์ภายนอกของ MCS-48 สามารถแบ่งออกได้ 2 ลักษณะ คือ ทำหน้าที่เป็นพอร์ต กับ บัส โดยจะแยกกล่าวดังรายละเอียดต่อไปนี้

2.1.4.1 พอร์ต (Port) ภายในตัว MCS-48 จะมีพอร์ตคล้ายกัน 2 พอร์ต คือพอร์ต 1 และ พอร์ต 2 ที่มีขนาด 8 บิต พอร์ตทั้งสองเป็นพอร์ตชนิด QUASI - BIDIRECTIONAL เนื่องจากโครงสร้างวงจรของพอร์ตแต่ละเส้นจะทำงานเป็นอินพุต และเอาต์พุต ด้วยลักษณะข้อมูลเอาต์พุตจะเป็นสเตติกส์แลทช์ (STATIC LATCH) และข้อมูลจะยังคงอยู่จนกว่าจะมีการเขียนข้อมูลพอร์ตใหม่ จากรูปที่ 2.5 แสดงวงจรภายในของขาแต่ละขาของพอร์ตทั้ง 2 จะเห็นได้ว่า ตัวพูลอัพ High Impedence ที่ต่อกับ Vcc จะทำหน้าที่ยกระดับแรงดันและกระแสให้เพียงพอกับระดับลอจิกของทีทีแอลที่จะมาต่อพ่วง จากวงจรจะเห็นว่าใน กรณีของการอินพุตนั้น สามารถจะรับการเปลี่ยนระดับลอจิกจากระดับสูงสู่ ระดับต่ำเท่านั้น ดังนั้น ในการใช้ขาใดขาหนึ่งของพอร์ตเป็นอินพุตสัญญาณนั้นต้องทำการเซตให้ขานั้นอยู่ในระดับลอจิก 1 เสมอ ในการรีเซตเริ่มแรกที่สำคัญในการอ่านหรือเขียนจากพอร์ต จะใช้คำสั่ง ORL และ ANL เมื่อทำ งานตัวไมโครจะอ่านข้อมูลจากพอร์ตด้วยคำสั่งอ่านข้อมูลและตามด้วยการ เขียนข้อมูลกลับไปที่พอร์ต การเขียนข้อมูลมีความจำเป็นต้องมีเพื่อให้อินพุตบัฟเฟอร์อิมพีแดนซ์ต่ำเกิดขึ้นชั่วขณะหนึ่ง แม้ข้อมูลจะเปลี่ยนแปลงจาก 1 ก็ตามการกำหนดทำเช่นนี้ก็เพื่อที่จะกำหนด (Config) ใช้พอร์ตเป็นทั้งอินพุตและ เอาต์พุตในพอร์ตเดียวกันได้



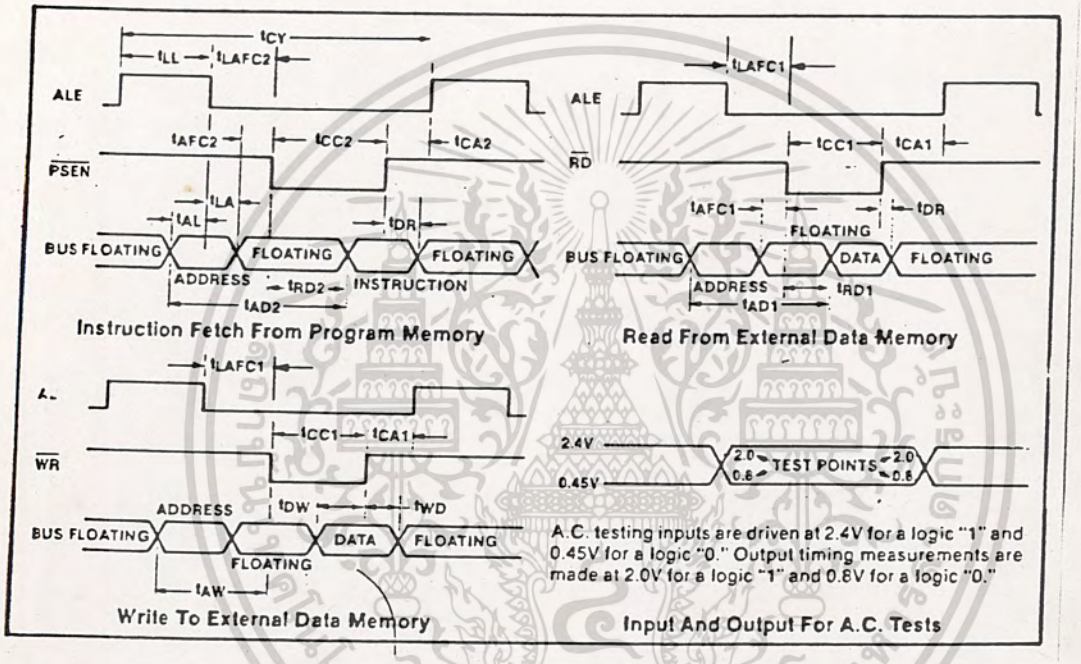
รูปที่ 4.5 โครงสร้างของพอร์ต QUASI-BIDIRECTIONAL



รูปที่ 2.6 แผนภูมิจังหวะเวลาของการใช้พอร์ต 1 และ 2 เป็นพอร์ตเอาต์พุต

ในกรณีที่ใช้หน่วยความจำโปรแกรมภายนอก 4 บิตแรกของพอร์ต 2 จะทำหน้าที่ในการกำหนดเลขที่ 4 บิตอันดับสูงของหน่วยความจำโปรแกรมเพื่อทำการอ่านคำสั่งการทำงาน ในกรณีนี้ถ้าต้องการใช้ 4 บิตแรกของพอร์ต 2 ทำหน้าที่สำหรับติดต่ออุปกรณ์ภายนอกอีก จะเกิดปัญหาที่ว่า MCS-48 ใช้พอร์ต 2 เป็นตัวชี้เลขที่อยู่ 4 บิตสุดท้ายอยู่แล้วสามารถแก้ปัญหานี้ได้โดยการแลทช์เข้าช่วย เพราะสัญญาณที่ออกมาที่ 4 บิตแรกของพอร์ต 2 จะเป็นลักษณะมัลติเพลกซ์ระหว่างข้อมูลที่ติดต่อกับอุปกรณ์ภายนอกกับเลขที่

อยู่แอดเดรสที่ส่งออกมา โดยรูปแบบของการติดต่ออุปกรณ์ภายนอกจะออกมาในขณะช่วงขึ้นของสัญญาณ ALE แต่รูปแบบของแอดเดรสจะปรากฏเมื่อช่วงลงของสัญญาณ ALE ดังนั้น ข้อมูลสำหรับติดต่ออุปกรณ์ภายนอกกับแอดเดรสจึงแยกออกมาได้ ดังรูปที่ 2.6 เป็นแผนภูมิจังหวะเวลาของการใช้พอร์ต 1 และ 2 เป็นพอร์ตเอาต์พุต ร่วมกับการส่งบิตแอดเดรสอันดับสูงของตัวนับโปรแกรม (PCH)



รูปที่ 2.6 แผนภูมิจังหวะเวลาของการฟฟท์โปรแกรม การอ่านและเขียนข้อมูลจากภายนอก

2.1.4.2 บัส (BUS) ในส่วนนี้พอร์ต 0 จะทำหน้าที่ 2 อย่างขึ้นอยู่กับกรณีที่ใช้คือ ในกรณีที่ใช้หน่วยความจำโปรแกรมภายใน บัสจะทำหน้าที่เช่นเดียวกับพอร์ตทั่วไปที่เป็นพอร์ตชนิด Bidirectional Port ซึ่งจะต่างกับพอร์ต 1 และ พอร์ต 2 ที่กล่าวมาแล้ว โดยจะทำหน้าที่เป็นพอร์ตเอาต์ที่แลทช์ข้อมูลไว้ด้วยคำสั่ง OUTL และยังเป็นอินพุตพอร์ตที่ไปแลทช์ข้อมูลไว้ที่พอร์ตด้วยคำสั่ง INS และทั้งสองคำสั่งนี้ จะสร้างพัลส์สโตรบที่ขา RD หรือ WR และที่จังหวะขอบขาลงของ RD หรือ WR จะเป็นช่วงจังหวะข้อมูลที่ปรากฏที่พอร์ตนี้ แต่พอร์ตนี้จะทำหน้าที่ผสมเป็นทั้งอินพุตและเอาต์พุตพร้อมกันไม่ได้ แต่ในกรณีที่ใช้หน่วยความจำโปรแกรมภายนอกแล้ว พอร์ตนี้จะทำหน้าที่เป็นบัสใน

การติดต่อกับหน่วยความจำโปรแกรมภายนอก ในขณะที่ทำการอ่านคำสั่ง จะให้แอดเดรส และอ่านคำสั่งเข้าที่บัสนี้ด้วยวิธีมัลติเพลกซ์ เช่นเดียวกับพอร์ต 2 ที่กล่าวมาแล้ว ในการมัลติเพลกซ์นั้น MCS-48 จะส่งสัญญาณแอดเดรสออกมาที่พอร์ตนี้ก่อน ก็จะถูกแลทช์ไว้ที่อุปกรณ์การแลทช์แอดเดรสภายนอก แล้วจึงทำการอ่านคำสั่งที่ได้จากหน่วยความจำโปรแกรมภายนอก ในขณะที่ไม่มีการอ่านและเขียนบัสจะอยู่ในสถานะอิมพีแดนซ์สูง ดังรูปที่ 2.7 เป็นการแสดงช่วงจังหวะของการใช้พอร์ตบัส ทำงานทั้งแบบการแฟทช์โปรแกรม การอ่านและเขียนข้อมูลจากภายนอก ซึ่งจะต้องส่งค่าแอดเดรสออกมาก่อนแล้วจึงตามด้วยรหัสคำสั่งหรือข้อมูล โดยใช้ขา ALE, PSEN, WR และ RD เป็นตัวควบคุมบอกจังหวะของข้อมูลในช่วงจังหวะใด ๆ ที่เกิดขึ้น

2.1.5 สัญญาณการตรวจสอบและสัญญาณอินเตอร์รัพต์ (TEST And INT INPUT)

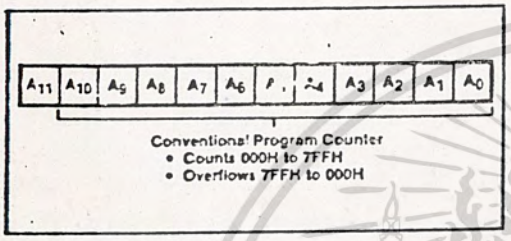
ในตระกูล MCS - 48 ได้จัดขาไว้ 3 ขาสำหรับเป็นอินพุต และไว้ใช้ทดสอบเงื่อนไขเพื่อทำงานร่วมกับคำสั่ง JUMP ซึ่งได้แก่ T0,T1,INT หลังจากทำการทดสอบเงื่อนไขของสัญญาณทั้งสามแล้ว การทำงานของMCS - 48 สามารถข้ามไปทำงานตามคำสั่งในโปรแกรมย่อย ที่เขียนไว้เพื่อทำงานตามเงื่อนไขที่ถูกต้อง โดยไม่จำเป็นต้องผ่านแอกคิวมิวเลเตอร์ นอกจากนี้แล้ว สัญญาณทั้งสามนี้ ยังสามารถทำหน้าที่อย่างอื่นได้อีก ซึ่งจะอธิบายในหัวข้อต่อไป

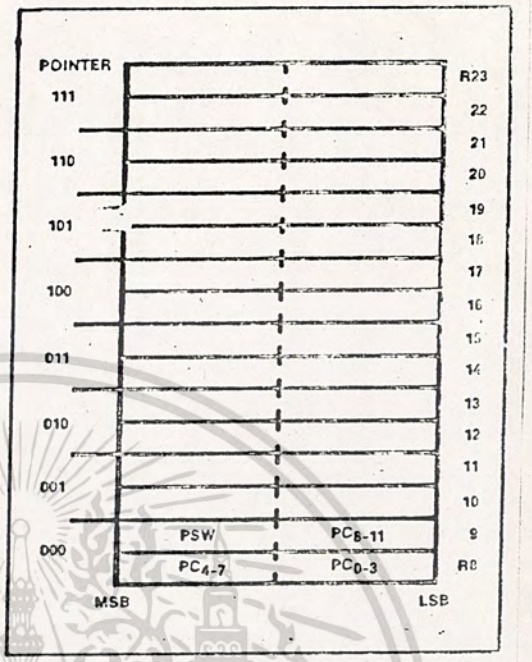
2.1.6 (PROGRAM COUNTER AND STACK)

ตัวนับโปรแกรมเป็นตัวนับอิสระขนาด 12บิต ดังรูปที่ 2.8 ในการทำงานปกติ จะเพิ่มที่ละหนึ่งเพื่อใช้สำหรับการชี้แอดเดรสของหน่วยความจำโปรแกรมเพื่ออ่านคำสั่งในการทำงาน ด้วยขนาด 10,11,12 บิตของตัวนับโปรแกรมนี้ จะใช้เป็นตัวชี้เลขที่อยู่ขนาด 1024,2048 และ 4096 ไบต์ของหน่วยความจำโปรแกรมบนชิปเบอร์ 8048 AH,8049 AH และ 8050 AHตามลำดับ สำหรับบิต 11 ของตัวนับโปรแกรมจะไม่เปลี่ยนไปตามการเพิ่มของตัวนับโปรแกรม สำหรับบิตนี้สามารถเปลี่ยนได้โดยใช้คำสั่ง

SELECT MEMORY BANK เป็นการเลือกชุดของหน่วยความจำโปรแกรม 2 กิโลไบต์แรกหรือ 2 กิโลไบต์หลัง ในกรณีที่ใช้หน่วยความจำโปรแกรมภายนอก บิต 0 - 7 ของตัวนับโปรแกรมนี้จะส่งออกมาปรากฏที่บัสเมื่อสัญญาณ ALE เริ่มตกลง ส่วนบิต 8 - 11 ของตัวนับโปรแกรมจะถูกส่งออกทาง 4 บิตอันดับต่ำของพอร์ต 2 เพื่อใช้สำหรับเป็นตัวชี้แอดเดรสของหน่วยความจำโปรแกรม

ในการอินเตอร์รัพต์หรือการใช้คำสั่ง CALLเข้าสู่โปรแกรมย่อย จะเป็นเหตุให้ค่าตัวนับโปรแกรมถูกย้ายไปเก็บไว้ที่บริเวณสแต็ก 8 คู่เรจิสเตอร์เป็น PROGRAM COUNTER STACK ดังรูปที่ 2.9 ตัวสแต็กจะมีค่า 3 บิต ที่อยู่เป็นส่วนหนึ่งของ PSW จะเป็นตัวชี้ 8 คู่เรจิสเตอร์ในบริเวณสแต็ก โดยที่ข้อมูลในแรมของตำแหน่ง8 ถึง 23 จะใช้เป็นเรจิสเตอร์สแต็ก 8 คู่สำหรับเก็บค่าตัวชี้โปรแกรม 12 บิต และค่านิบเบิ้ลสูงของเรจิสเตอร์ที่ทำหน้าที่เป็นตำแสดงสถานะของโปรแกรมดังรูปที่ 2.9 ตัวชี้สแต็กเมื่อถูกInitial เริ่มแรกจะชี้ที่ 000 ซึ่งคลุมตำแหน่งแรมที่ R8 และ R9 เป็นคู่เรจิสเตอร์สแต็ก ดังนั้นเมื่อเกิดมีการอินเตอร์รัพต์ หรือมีการเรียกโปรแกรมย่อยจะมีการเก็บข้อมูลของตัวชี้โปรแกรมไว้ที่ตำแหน่งR8และ R9 และตัวชี้สแต็กจะเพิ่มขึ้นอีกหนึ่ง คือ 001ชี้ที่คู่ตำแหน่ง R10 และ R11 ดังนั้นสามารถใช้โครงข่ายการเรียกโปรแกรมย่อยซ้อนภายในโปรแกรมย่อย จะเรียกได้ 8 ครั้ง โดยไม่เกิด Overflow ของสแต็ก ถ้าเกิด overflowและจะทับค่าเดิม เนื่องจากการเกิด overflow จาก 000 มาหา 111 การสิ้นสุดโปรแกรมย่อยด้วยการใช้คำสั่ง RET หรือ RETR จะเป็นการย้ายข้อมูลในคู่สแต็กเรจิสเตอร์มาเข้าสู่ตัวชี้โปรแกรม





รูปที่ 2.8 ตัวนับโปรแกรม รูปที่ 2.9 บริเวณสแต็กที่เก็บตัวนับโปรแกรม

2.1.7 เรจิสเตอร์คำแสดงสถานะโปรแกรม (Program Status Word PSW)

คำแสดงสถานะโปรแกรมมีขนาด 8 บิต สามารถอ่านและเขียนผ่านแอกคิวมิวเลเตอร์ได้ เมื่อคูหรือเซตสถานะเริ่มต้นก่อนที่จะทำตามชุดคำสั่งที่ได้เขียนไว้ คำแสดงสถานะของโปรแกรมเป็นเรจิสเตอร์ที่เก็บตัวชี้สแต็กและสถานะผลลัพธ์ต่าง ๆ ของโปรแกรมที่ทำงานอยู่ ดังรูปที่ 2.10 การที่สามารถเขียนหรืออ่านได้ของคำแสดงสถานะโปรแกรม ทำให้สามารถเก็บสถานะต่าง ๆ หลังการเกิดปัญหาแหล่งจ่ายไฟ และเรียกกลับคืนมาได้หลังจากจ่ายไฟปกติ หรือหลังการใช้บริการการอินเตอร์รัพต์

คำแสดงสถานะของโปรแกรมจะมีส่วนประกอบ ดังนี้

บิต 0-2 เป็นตัวชี้สแต็ก (S0, S1, S2)

บิต 3 เป็นบิตที่ไม่ได้ใช้งาน โดยปกติเมื่อมีการอ่านเข้ามาจะมีค่าเป็น 1 เสมอ

บิต 4 เป็นบิตใช้ในการเลือกชุดเรจิสเตอร์ (REGISTER BANK SWITCH) โดย

0 = เรจิสเตอร์ชุดที่ 1 (แอดเดรส 0 - 7) หรือ แบงค์ 0

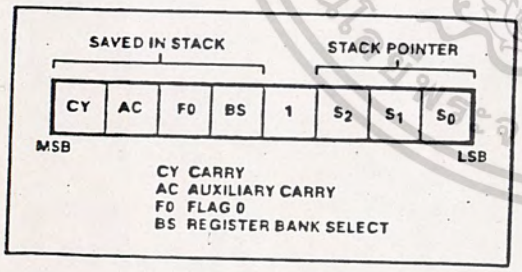
1 = เรจิสเตอร์ชุดที่ 2 (แอดเดรส 24 - 31) หรือ แบงค์ 1

บิต 5 เป็นบิตแฟลกศูนย์ ผู้ใช้สามารถควบคุมโดยเซตหรือรีเซตได้ ตามสถานะของคำสั่ง เมื่อใช้ประโยชน์เป็นเงื่อนไขในการใช้คำสั่งกระโดด JF0

บิต 6 AUXILIARY CARRY (AC) เป็นตัวทดบิต จากบิต 3 ไป บิต4 เมื่อใช้คำสั่ง ADD ใช้สำหรับเป็นเงื่อนไขในการปรับค่า เป็นเลขฐานสิบ DA A (DECIMAL ADJUST)

บิต 7 CARRY BIT บิตตัวทด เป็นบิตที่แสดงสถานะเป็นตัวทด เมื่อผลลัพธ์ ที่แอกคิวมิวเลเตอร์มีค่ามากกว่า 8 บิต

คำแสดงสถานะโปรแกรม บิต 4 - 7 หรือบิบเบิ้ลสูง จะถูกเก็บในสแต็ก เมื่อมีการเรียกโปรแกรมย่อยทำงาน และสามารถนำค่าของคำแสดงสถานะของโปรแกรมกลับมาได้เมื่อใช้คำสั่ง RETR ส่วนคำสั่ง RET จะเป็นการกลับจากโปรแกรมย่อยที่ไม่เก็บคำแสดงสถานะของโปรแกรมเดิมไว้



Device Testable	Jump Conditions (Jump On)	
	All zeros	not all zeros
Accumulator		
Accumulator Bit	—	1
Carry Flag	0	1
User Flags (F0, F1)	—	1
Timer Overflow Flag	—	1
Test Inputs (T0, T1)	0	1
Interrupt Input (INT)	0	—

รูปที่ 2.10 คำแสดงสถานะโปรแกรม ตารางที่ 2.2 แสดงเงื่อนไขสำ หรับใช้ในการกระโด

2.1.8 การใช้ลอจิกเป็นเงื่อนไขในการกระโดด

เงื่อนไขต่าง ๆ ที่เกิดขึ้นทั้งภายในและภายนอกของ MCS-48 สามารถที่จะทำการทดสอบ เพื่อกระโดดไปยังโปรแกรมต่าง ๆ ที่ได้เขียนไว้ตามเงื่อนไขที่ถูกต้อง โดยใช้คำสั่งแบบมีเงื่อนไข (CONDITION JUMP INSTRUCTION) เงื่อนไขต่าง ๆ ที่ใช้สำหรับการกระโดดแบบมีเงื่อนไข แสดงดังตารางที่ 2.2 ใช้ซอฟต์แวร์ด้วยการตรวจ

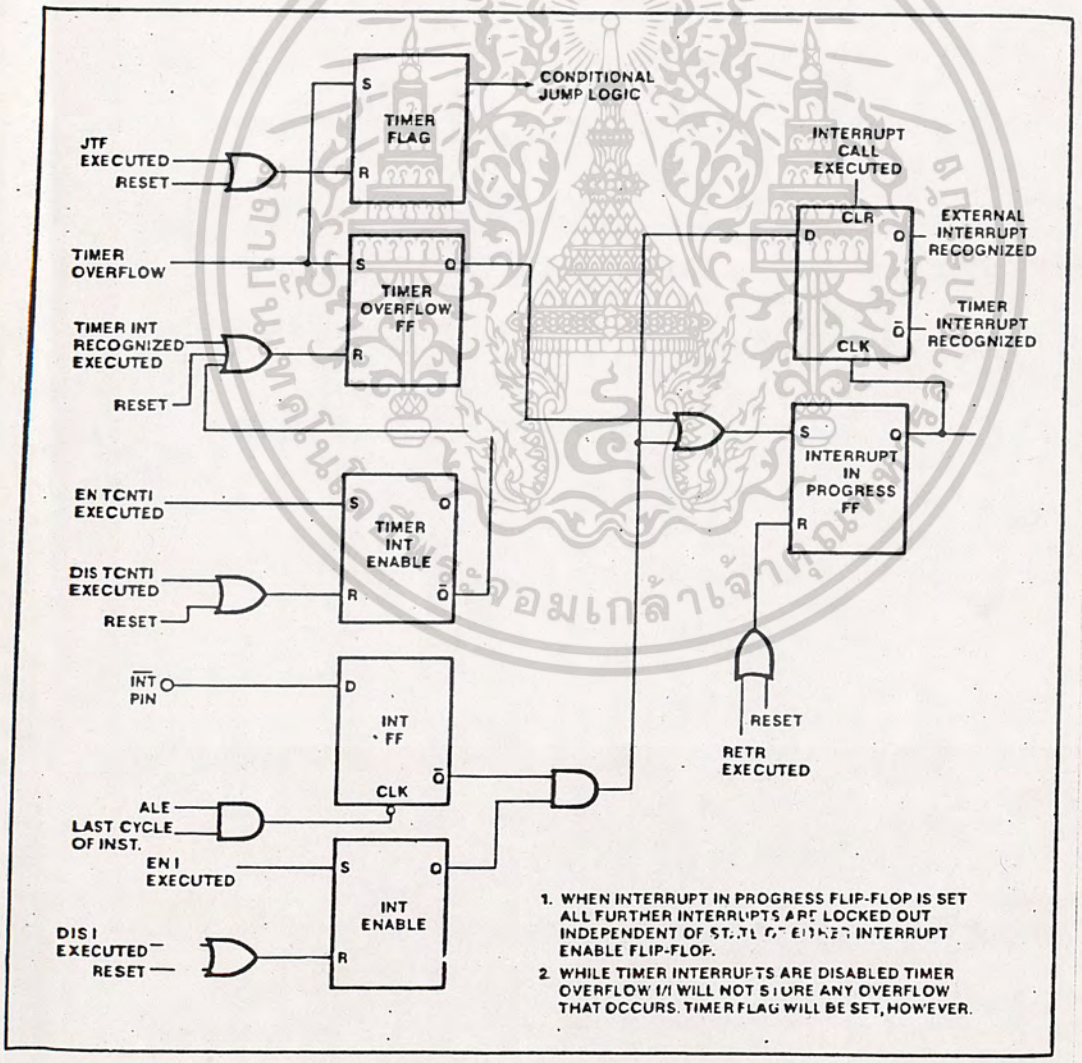
สอบ ค่าศูนย์ทั้งแปดบิตในแอกคิวมิวเลเตอร์ หรือ ไม่เป็นศูนย์ ส่วนการทดสอบเฉพาะบิตสามารถทดสอบได้ด้วยการใช้คำสั่งทางลอจิก และทดสอบผลที่ได้ในแอกคิวมิวเลเตอร์ในค่าที่ไม่เป็นศูนย์ การทดสอบบิตตัวทดสามารถทดสอบได้ทั้งแบบเซตและรีเซตค่านี้ การทดสอบบิต F0 F1 และแฟลก OVERFLOW ของตัวจับเวลา ทดสอบได้เฉพาะการเซตค่า การทดสอบสัญญาณเข้า T0 T1 ทดสอบได้ทั้งแบบเซตและรีเซต ส่วนการ INT ทดสอบได้เฉพาะการรีเซตเท่านั้น

2.1.9 อินเตอร์รัพต์ (INTERRUPT)

MCS-48 จะรับการอินเตอร์รัพต์ต่อเมื่อได้อินาเบิ้ลแล้ว ลำดับการอินเตอร์รัพต์จะเกิดขึ้น เมื่อมีสัญญาณสถานะลอจิกเป็น 0 เข้าที่ขา INT โดยการอินเตอร์รัพต์จะกระตุ้นด้วยสัญญาณระดับต่ำและสามารถที่จะรับการอินเตอร์รัพต์เข้ามาได้หลายแหล่งด้วยกัน ดังแสดงการอินเตอร์รัพต์ทางลอจิกในรูปที่ 2.9 ขาอินเตอร์รัพต์จะถูกตรวจสอบ (Sample) ทุกวัฏจักรคำสั่งในช่วงวัฏจักรการทำงานของ ALE และเมื่อตรวจพบ (ซึ่งสัญญาณอินเตอร์รัพต์จะต้องมีสถานะต่ำอย่างน้อย 3 วัฏจักรแมชีน เพราะสัญญาณจะถูกตรวจที่วัฏจักรที่ 2 ของการทำงานคำสั่ง) ก็จะเรียกโปรแกรมย่อยการอินเตอร์รัพต์ โดยก่อนที่จะเข้าสู่โปรแกรมย่อยบริการอินเตอร์รัพต์ ตัวนับโปรแกรม และบิตที่ 4 - 7 ของคำแสดงสถานะของโปรแกรมจะถูกเก็บไว้ที่สแต็ก และการควบคุมการทำงานต่าง ๆ เมื่อมีการอินเตอร์รัพต์นั้น จะถูกส่งผ่านไปทำงานที่ตำแหน่ง 003 ของหน่วยความจำโปรแกรม ซึ่งโดยปกติจะเขียนด้วยคำสั่งกระโดดแบบไม่มีเงื่อนไข (Unconditional Jump) ไปยังโปรแกรมบริการที่มีต่อบสนองการอินเตอร์รัพต์ การกลับคืนหลังจากบริการโปรแกรมย่อยของการอินเตอร์รัพต์ จะกลับโดยใช้คำสั่ง RETR ซึ่งจะยังคงรักษาคำแสดงสถานะของโปรแกรมไว้ การอินเตอร์รัพต์ของ MCS-48 เป็นการอินเตอร์รัพต์ระดับเดียว ดังนั้น ระหว่างการบริการโปรแกรมอินเตอร์รัพต์อยู่ การอินเตอร์รัพต์จะไม่สามารถได้รับบริการ ดังนั้น จะต้องมีการอินาเบิ้ลอินเตอร์รัพต์ทุกครั้งด้วยคำสั่ง RETR เสมอซึ่งจะอินาเบิ้ลได้ หลังวัฏจักรที่ 2 ของการทำงานคำสั่ง RETR เรจิสเตอร์ของตัวจับเวลา/ตัวนับ ก็จะทำงานเช่นเดียวกับการอินเตอร์รัพต์ภายนอก จะต่างกันก็ตรงที่ว่าการอินเตอร์รัพต์ที่เกิดจากตัวจับเวลา/ตัวนับนั้น จะไปทำต่ำแหน่ง 007 ซึ่งเป็นต่ำ

หน่วยที่โปรแกรมย่อยการอินเตอร์รัพต์ของตัวจับเวลา/ตัวนับอยู่ การอินเตอร์รัพต์จากภายนอกจะมีลำดับสูงกว่าการอินเตอร์รัพต์จากตัวจับเวลา/ตัวนับ หมายความว่าถ้ามีการอินเตอร์รัพต์จากภายนอกพร้อมกับการอินเตอร์รัพต์จากตัวตั้งเวลาแล้ว MCS-48 จะตอบสนองต่อการอินเตอร์รัพต์จากภายนอกก่อนเสมอ

เมื่อต้องการดิสเอเบิ้ลการอินเตอร์รัพต์ สามารถทำได้ 2 วิธี คือ การรีเซต MCS-48 กับการใช้คำสั่งดิสเอเบิ้ล DIS I และ DIS TCNTI ทั้งการอินเตอร์รัพต์ภายนอกกับตัวจับเวลา/ตัวนับ



รูปที่ 2.11 วงจรตรรกการอินเตอร์รัพต์

จังหวะเวลาอินเตอร์รัพต์ (Interrupt Timing)

การอินเตอร์รัพต์จะอนุญาตเอ็นเอ็ลหรือดิสเอเบิ้ลได้ด้วยโปรแกรมควบคุมด้วยการใช้คำสั่ง EN I และ DIS I การอินเตอร์รัพต์จะดิสเอเบิ้ลได้ด้วยการรีเซตและจะยังคงดิสเอเบิ้ลจนกว่าจะถูกอนุญาตเอ็นเอ็ลด้วยผู้ใช้ สัญญาณการร้องขออินเตอร์รัพต์จะต้องสิ้นสุดก่อนที่จะทำงานกับคำสั่ง RETR เพื่อที่จะให้ตัวโพรเซสเซอร์สามารถที่จะบริการการอินเตอร์รัพต์ตัวต่อไปได้ทันที มิฉะนั้น จะเข้าอินเตอร์รัพต์เดิม ฉะนั้น อุปกรณ์ต่อพ่วงการอินเตอร์รัพต์จะมีการป้องกันลักษณะ เช่นนี้ด้วยการรีเซตการร้องขอการอินเตอร์รัพต์ เมื่อไรที่ตัวโพรเซสเซอร์ได้อ่าน หรือเขียนข้อมูลจาก เรจิสเตอร์บัฟเฟอร์ของอุปกรณ์ต่อพ่วงแล้ว ถ้าอุปกรณ์การอินเตอร์รัพต์ไม่ต้องการเข้าถึงตัวโพรเซสเซอร์ ก็อาจจะมีการใช้ MCS-48 เป็นตัวส่งสัญญาณตอบรับการรับรู้อินเตอร์รัพต์จากการร้องขอของอุปกรณ์ต่อพ่วงนั้น ขา INT ยังอาจใช้ทดสอบด้วยการใช้คำสั่งกระโดดอย่างมีเงื่อนไข JNI คำสั่งนี้จะใช้เป็นตัวตรวจจับการอินเตอร์รัพต์ที่เกิดขึ้นก่อนการอินเตอร์รัพต์ MCS-48 จะเกิดขึ้น ถ้าการอินเตอร์รัพต์อยู่ในสภาวะดิสเอเบิ้ลก็อาจใช้เป็นตัวตรวจสอบสัญญาณอินพุตธรรมดาทั่วไป เช่นเดียวกับขา TO และ T1

2.1.10 ตัวจับเวลา/ตัวนับ (TIMER/COUNTER)

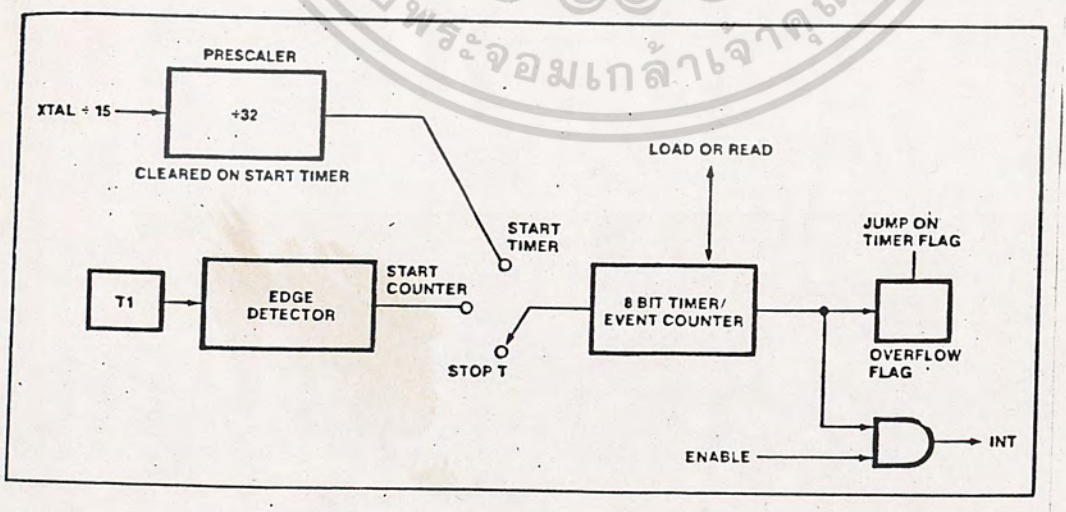
ภายในของ MCS-48 จะประกอบด้วยตัวจับเวลา/ตัวนับ การทำงานของทั้งสองจะคล้ายกันเพียงแต่ว่าตัวนับจะมีการส่งอินพุตเข้าตัวนับเท่านั้น

2.1.10.1 ตัวนับ เป็นตัวนับแบบเลขฐานสองขนาด 8 บิต ที่สามารถกำหนดหรืออ่านค่าได้ด้วยคำสั่ง MOV ที่ส่งผ่านเข้าหรือออกทางแอกคิวมิวเลเตอร์และการรีเซ็ตจะไม่มีผลต่อค่าที่อยู่ในตัวนับ ตัวนับจะหยุดก็ต่อเมื่อมีการรีเซ็ตหรือใช้คำสั่ง STOP TCNT และจะหยุดจนกระทั่งใช้คำสั่ง START T ในการทำงานเป็นตัวจับเวลาหรือ START CNT ในการทำงานเป็นตัวนับอีกครั้ง ทันทีที่ตัวนับเริ่มนับจะเพิ่มค่าตัวเองทีละ 1 จนกระทั่งถึงค่าสูงสุด FFH แล้วเริ่มนับที่ 00 ใหม่ การนับจะเป็นเช่นนี้เรื่อยไปจนกว่าจะมีคำสั่ง STOP TCNT หรือการรีเซ็ตทำงานโปรแกรมย่อยบริการ การทำงานของตัวจับเวลา/ตัวนับที่ตำแหน่ง 007 ทันที ตัวนับสามารถทำหน้าที่เป็นตัวนับเหตุ

การณ์ (EVENT COUNTER) ได้โดยที่ขา T1 จะทำหน้าที่เป็นอินพุต เมื่อมีการเปลี่ยนสถานะจาก"1" ไป "0" ตัวนับจะเพิ่มค่าขึ้นหนึ่งเสมอ และ T1 จะต้องมีสถานะ "0" อย่างน้อย 1 วัฏจักรแมชีน เพื่อให้แน่ใจว่าการนับจะไม่ขาดหายไป ดังนั้นความเร็วของการนับสูงสุด ในแต่ละครั้งจะใช้เวลา 3 รอบ คำสั่ง หรือทุก ๆ 5.7 ไมโครวินาที เมื่อใช้คริสตรอล 6 เมกกะเฮิรตซ์ และอินพุตเข้า T1 จะต้องมีระดับลอจิกเป็น 1 อย่างน้อย 1/5 วัฏจักรแมชีนหลังจากการเปลี่ยนระดับในแต่ละครั้ง

2.1.10.2 ตัวจับเวลา เมื่อใช้คำสั่ง START T ตัวจับเวลา/ตัวนับจะทำหน้าที่เป็นตัวจับเวลา ในการใช้คำสั่ง START T นั้นจะทำให้สัญญาณเวลาภายในถูกหารด้วย 15 และ 32 ตามลำดับจากผลที่ได้จะเพิ่มค่าของตัวจับเวลาที่ได้จะเพิ่มค่าของตัวจับเวลา/ตัวนับ เมื่อไปยังโปรแกรมย่อยที่ต้องการ ตัวอย่างเช่น

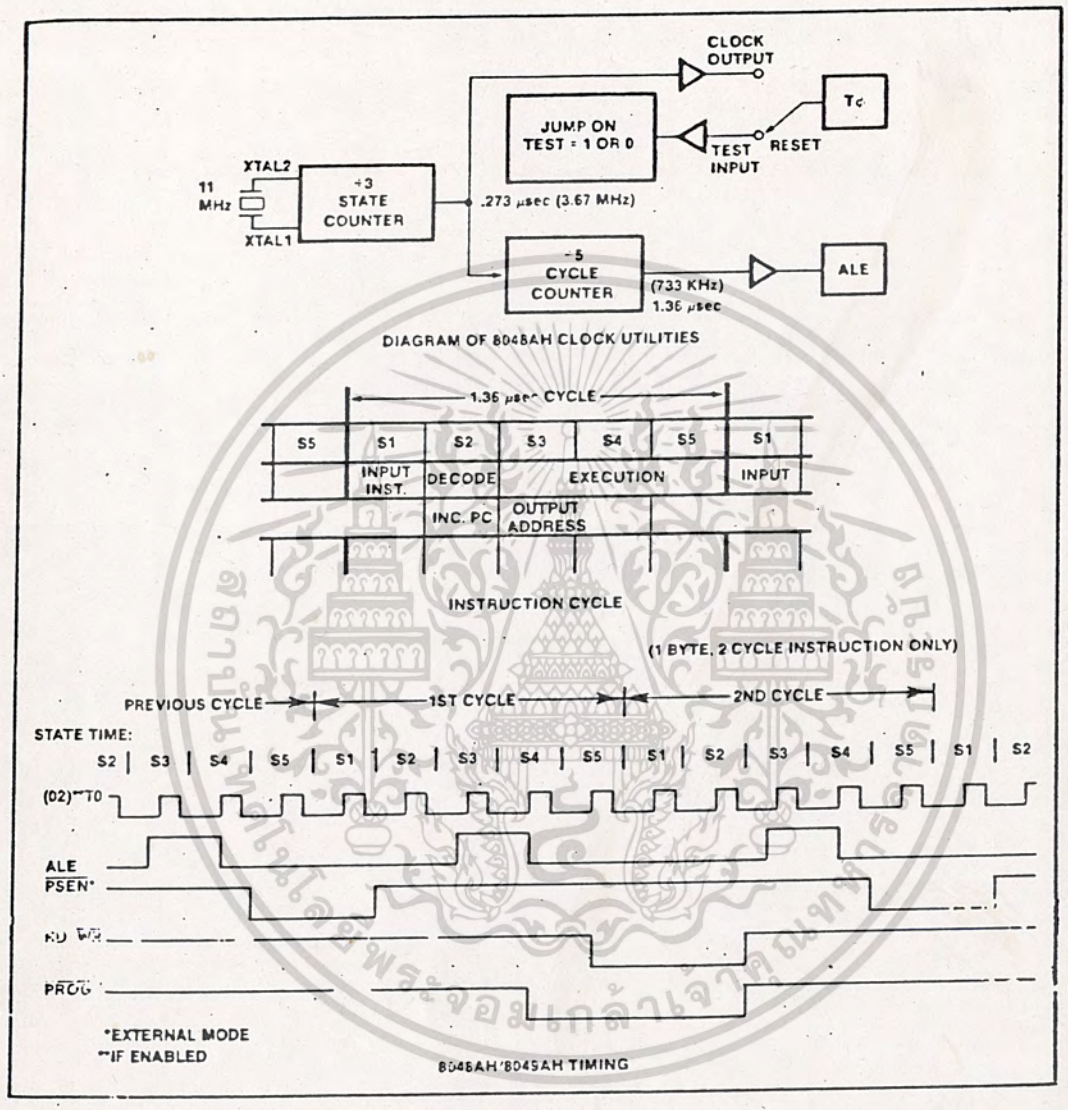
เมื่อใช้คริสตรอลขนาด 11 เมกกะเฮิรตซ์เป็นฐานเวลาให้ MCS-48 เมื่อผ่านตัวหาร 15 จะได้ความถี่ 738 กิโลเฮิรตซ์ และความถี่นี้จะถูกหารด้วย 32 อีก จะได้ 22.917 กิโลเฮิรตซ์ ซึ่งแสดงว่าตัวตั้งเวลาจะเพิ่มขึ้นทุก ๆ 44 ไมโครวินาที รูปที่ 2.12 เป็นการแสดงให้เห็นถึงส่วนประกอบการทำงานของตัวจับเวลา/ตัวนับภายใน MCS-48



2.2 สัญญาณนาฬิกาและวงจรฐานเวลา (CLOCK AND TIMER CIRCUIT)

แหล่งกำเนิดฐานเวลาสำหรับ MCS-48สามารถสร้างได้จากอุปกรณ์ภายนอก เช่น ครีสตรอล อินดัคเตอร์ หรือกำเนิดสัญญาณนาฬิกาภายนอก การทำงานของสัญญาณนาฬิกาและวงจรฐานเวลาแสดงในรูปที่ 2.13 โดยแยกกล่าวเป็นส่วน ๆดังนี้

2.2.1 ตัวกำเนิดความถี่ (OSCILIATOR) ตัวกำเนิดความถี่เป็นวงจรแบบ SERIES RESONANT ที่มีอัตราการขยายสูงโดยมีช่วงความถี่ระหว่าง 1 ถึง 6 เมกกะเฮิรตซ์ ซึ่งขา X1 เป็นขาอินพุตเข้าสู่วงจรขยาย ขณะที่ขา X2 เป็นขาเอาต์พุต ในการต่อครีสตรอลหรืออินดัคเตอร์ระหว่างขา X1 และ X2 จะทำให้เกิดการป้อนกลับและการเลื่อนเฟสสำหรับการกำเนิดความถี่ในกรณีที่ไม่ต้องการใช้ความถี่ที่แน่นอน สามารถใช้อุปกรณ์พวกอินดัคเตอร์ แทน ครีสตรอลได้ ซึ่งจะให้ความถี่อยู่ในช่วง 3 ถึง 5 เมกกะเฮิรตซ์ได้

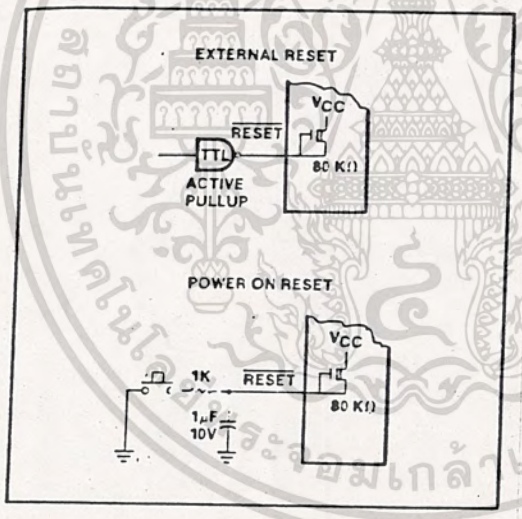


2.2.2 ตัวนับสถานะ (STATE COUNTER) เอาท์พุตของตัวกำเนิดความถี่จะถูกหารด้วย 3 ในตัวนับสถานะ เพื่อสร้างสัญญาณนาฬิกาสำหรับการทำงานของ MCS-48 สัญญาณนาฬิกาสามารถส่งไปยังขา T0 ได้โดยใช้คำสั่ง ENT0 CLK สัญญาณนาฬิกาที่ขาT0จะหยุดเมื่อถูกรีเซ็ต

2.2.3 ตัวนับรอบ (CYCLE COUNTER) สัญญาณขาฟีกาจะถูกหารด้วย 5 ในตัวนับรอบเพื่อสร้างสัญญาณขาฟีกาที่ประกอบด้วยรอบการทำงาน 5 สถานะ สัญญาณขาฟีกานี้เรียกว่า ADDRESS LATCH ENABLE (ALE) เพื่อใช้เป็นสัญญาณติดต่อกับหน่วยความจำโปรแกรมภายนอก สัญญาณนี้จะออกมาอย่างต่อเนื่องตลอดเวลาขณะที่ MCS-48 ยังทำงานอยู่

2.3 การรีเซ็ต (RESET)

การรีเซ็ตจะทำให้ MCS-48 เริ่มการทำงานใหม่เมื่อขา รีเซ็ต มีลอจิกศูนย์ซึ่งขานี้จะต้องทำงานแบบ SCHMITT-TRIGGER โดยจะมีความต้านทานที่ใช้พูลอัพอยู่ภายใน MCS-48 และภายนอกจะต่อตัวเก็บประจุค่า 1 ไมโครฟารัดอยู่เพื่อทำให้เกิดช่วงเวลาเพียงพอในการรีเซ็ต การรีเซ็ต MCS-48 สามารถทำได้ดังแสดงในรูปที่ 2.14



รูปที่ 2.14 แสดงการรีเซ็ต MCS-48

หากมีสัญญาณการรีเซ็ตจากภายนอกที่ขา รีเซ็ต จะต้องมีค่าลอจิกศูนย์อย่างน้อย 50 มิลลิวินาที หลังจากที่แหล่งจ่ายไฟอยู่ในสภาพที่พร้อมจะใช้งานแล้ว

เมื่อมีการรีเซ็ตให้ MCS-48 จะทำให้เกิด

- ตัวนับโปรแกรมถูกรีเซ็ตให้เป็นศูนย์
- ตัวชี้สแต็กถูกเซ็ตเป็นศูนย์
- เรจิสเตอร์แบงค์ 0 จะถูกเลือก

- หน่วยความจำโปรแกรมแบงค์ 0 จะถูกเลือก
- บัสจะถูกรีเซ็ตเป็น HIGH IMPEDANCE ยกเว้นเมื่อขา EA มีสถานะลอจิกหนึ่ง
- พอร์ต 1 และ พอร์ต 2 จะอยู่ในกรณีเป็นอินพุต
- อินเตอร์รัพต์ถูกดิสเอเบิ้ล
- ตัวจับเวลาจะหยุด
- แฟลกของตัวจับเวลาถูกทำให้เป็นศูนย์
- แฟลกศูนย์และแฟลกหนึ่งถูกทำให้เป็นศูนย์

ชุดคำสั่งของตระกูล MCS-48

ชุดคำสั่งสำหรับตระกูล MCS-48 สามารถแบ่งตามชนิดการทำงานได้ดังนี้

- คำสั่งควบคุม (CONTROL INSTRUCTION)
- คำสั่งเคลื่อนย้ายข้อมูล (DATA MOVE INSTRUCTION)
- คำสั่งเกี่ยวกับตัวจับเวลา/ตัวนับ (TIMER/COUNTER INSTRUCTION)
- คำสั่งเกียวกับนอกคิวมิวเลเตอร์ (ACCUMULATOR INSTRUCTION)
- คำสั่งการกระโดด (BRANCH INATRUCTION)
- คำสั่งเกี่ยวกับอุปกรณ์ภายนอก (INPUT/OUPUT INSTRUCTION)
- คำสั่งเกี่ยวกับเรจิสเตอร์ (REGISTER INSTRUCTION)
- คำสั่งเกี่ยวกับโปรแกรมย่อย (SUBROUTINE INSTRUCTION)
- คำสั่งเกี่ยวกับแฟลก (FLAGS INSTRUCTION)
- คำสั่งเบ็ดเตล็ด (MISCELLANEOUS INATRUCTION)

ชุดคำสั่งของตระกูล MCS-48 ประกอบด้วยคำสั่งทั้งหมด ๖ คำสั่ง ลักษณะคำสั่งเป็นคำสั่งที่ใช้ง่ายและจำได้ง่าย โดยคำสั่งเหล่านี้จะประกอบด้วยคำ สั่งไบต์เดียวและคำสั่งสองไบต์ ประมาณ 70 % ของคำสั่ง จะเป็นคำสั่งไบต์ เดียว สำหรับคำสั่งสองไบต์จะเป็นทั้งแบบคำสั่งโดยทันที (IMMEDIATE INSTRUCTIONS) คำสั่งติดต่ออุปกรณ์ภายนอกที่แน่นอน (ยกเว้น MOVE PP,A) และคำสั่งเกี่ยวกับการกระโดดทั้งหมด การทำงานของคำสั่งจะใช้เวลาหนึ่งหรือ สองวัฏจักรแมชีน (MACHINE CYCLE) โดยกว่า 60% เป็นคำสั่งที่ใช้เวลาหนึ่ง วัฏจักรแมชีน ซึ่งหมายความว่าการทำงานของคำสั่งแต่ละคำสั่งใช้เวลาน้อย ย่อมทำให้มีความเร็วการทำงานเร็วขึ้น

นอกจากนี้ตระกูล MCS-48 ยังมีความสามารถการทำงานทางคณิตศาสตร์ทั้งแบบเลขฐานสองและเลข ECD อย่างมีประสิทธิภาพ และยังสามารถทำงานควบคุมด้วยบิตเพียงบิตเดียวได้ง่าย

รายละเอียดของการทำงานของชุดคำสั่งแต่ละแบบมีดังนี้

1. **คำสั่งควบคุม** (CONTROL INSTRUCTION)

คำสั่งควบคุมจะทำให้โปรแกรมที่ใช้งานสามารถควบคุมการอินเตอร์รัพต์ การเลือกชุดของหน่วยความจำโปรแกรม และการให้สัญญาณนาฬิกาออกมา เพื่อใช้ในงานควบคุม

เมื่อตระกูล MCS-48 เริ่มทำงานนั้น สัญญาณอินเตอร์รัพต์ที่เข้าทาง ขา INT จะถูกดิสเอเบิ้ลโดยอัตโนมัติ การอินเตอร์รัพต์จากภายนอกสามารถควบ คุมการอินเตอร์รัพต์ได้โดย คำสั่งเกี่ยวกับการอินเตอร์รัพต์ แต่อย่างไรก็ตามการร้องขอด้วยสัญญาณอินเตอร์รัพต์ที่ขา INT จะไม่ได้รับการตอบรับ เมื่อการทำงานของตระกูล MCS-48 ยังอยู่ในโปรแกรมการอินเตอร์รัพต์อยู่ เนื่องจากการอินเตอร์รัพต์ของตระกูล MCS-48 เป็นการอินเตอร์รัพต์ระดับเดียว หรือภายในโปรแกรมที่ทำงานอยู่ไม่ได้ ซึ่งให้มีการรับอินเตอร์รัพต์สำหรับคำสั่งที่ใช้เลือกชุดการทำงาน 4 คำสั่งนั้น 2 คำสั่งจะใช้เลือกเรจิสเตอร์ทำงานที่อยู่ในส่วนของแรมภายในตระกูล MCS-48 และอีก2 คำสั่งใช้สำหรับเลือกหน่วยความจำโปรแกรมซึ่งเป็นรอม

คำสั่งที่ใช้เลือกชุดของหน่วยความจำโปรแกรม จะทำหน้าที่เซตค่า MEMORY BANK FLIP-FLOP (DBF) เพื่อใช้เลือกชุดของหน่วยความจำโปรแกรมค่าของDBF จะยังค้างค่าคงเดิมอยู่จนกว่าจะมีคำสั่งเลือกชุดใหม่เข้ามา บิต 11ของตัวนับโปรแกรมจะไม่เพิ่มตาม ตามการเพิ่มของบิตอื่นแต่บิตนี้จะเปลี่ยนแปลงตาม DBF เมื่อคำสั่ง CALLและคำสั่งJUMP ทำงานซึ่งจะมีประโยชน์เมื่อมีโปรแกรมย่อยอยู่ในส่วนของหน่วยความจำอีก ชุดหนึ่งทำงานได้ทันทีโดยที่ข้อมูลเดิมที่มีอยู่ในเรจิสเตอร์เก่า ยังคงเก็บรักษาอยู่เหมือนเดิมคำสั่งเลือกชุดเรจิสเตอร์นี้มีประโยชน์มากในการใช้อินเตอร์รัพต์ เพราะสามารถใช้เรจิสเตอร์หนึ่งชุดสำหรับ การอินเตอร์รัพต์และอีกชุดหนึ่งสำหรับการทำงานปกติ

สำหรับคำสั่งที่ใช้ควบคุมสัญญาณนาฬิกาที่จะส่งออกไปยังขา TO นั้น มีประ

โยชน์สำหรับให้สัญญาณนาฬิกากับอุปกรณ์ภายนอกที่ติดต่ออยู่กับ ตระกูลMCS-48 ซึ่งสามารถควบคุมสัญญาณนาฬิกาด้วยคำสั่งได้

2. คำสั่งเคลื่อนย้ายข้อมูล (MOVE DATA INSTRUCTION)

ชุดของคำสั่งเคลื่อนย้ายข้อมูล เป็นชุดคำสั่งแรกในการเคลื่อนย้ายระหว่างแอกคิวมิวเลเตอร์ กับเรจิสเตอร์ต่าง ๆ หรือหน่วยความจำของระบบการเคลื่อนย้ายข้อมูลระหว่างแอกคิวมิวเลเตอร์กับ เรจิสเตอร์จะเป็นการเคลื่อนย้ายโดยตรง โดยกำหนดทิศทางการเคลื่อนย้ายในคำสั่งได้ แต่การเคลื่อนย้ายข้อมูลระหว่างแอกคิวมิวเลเตอร์กับหน่วยความจำนั้น เป็นการ เคลื่อนย้ายทางอ้อม โดยจะมีตัวชี้ตำแหน่งที่ต้องการเคลื่อนย้ายอยู่ในเรจิส เตอร์ RO หรือ R1 ชุดเรจิสเตอร์แต่ละชุด การเคลื่อนย้ายระหว่างหน่วยความจำข้อมูลภายในจะใช้เวลาเพียงหนึ่งรอบการทำงาน ขณะที่การเคลื่อนย้ายกับหน่วยความจำข้อมูลภายนอกจะใช้เวลาสองรอบการทำงาน ขณะเดียวกันข้อมูลที่เก็บอยู่ในส่วนหน่วยความจำโปรแกรมสามารถอ่านเข้ามาเก็บไว้ในแอกคิวมิวเลเตอร์ได้โดยตรง ซึ่งมีประโยชน์ในการจัดข้อมูลในตารางไว้สำหรับเปรียบเทียบได้ นอกจากนี้ยังสามารถเคลื่อนย้ายข้อมูลของตัวนับ/ตัวจับเวลา หรือเรจิสเตอร์บอกสถานะโปรแกรม เข้าสู่หรือออกจากแอกคิวมิวเลเตอร์ได้ ความสามารถในการเคลื่อนย้ายข้อมูลระหว่างตัวแอกคิวมิวเลเตอร์กับเรจิสเตอร์บอกสถานะโปรแกรม ทำให้สามารถเปลี่ยนสถานะการทำงานของโปรแกรมได้ หรือเปลี่ยนตัวชี้สแตคที่อยู่ในเรจิสเตอร์บอกสถานะโปรแกรมได้ เมื่อมีความจำเป็นที่ต้องสลับข้อมูล ในตระกูล MCS-48จะมีคำสั่งพิเศษอีกหนึ่งคำสั่งคือ XCHD A เป็นคำสั่งที่ทำงานร่วมกับคำสั่ง SWAP A เพื่อความสะดวกในการย้ายครั้งละ 4 บิตหรือการเคลื่อนย้ายของเลข BCD คำสั่งนี้จะมีการแลกเปลี่ยนระหว่างข้อมูล 4 แรกของแอกคิวมิวเลเตอร์กับ 4 บิตแรกของหน่วยความจำข้อมูลภายใน เมื่อทำงานร่วมกับคำสั่ง SWAP A จะทำให้สามารถเก็บรักษาข้อมูลครั้งละ 4 บิตหรือเลข BCD ไว้ในหน่วยความจำข้อมูลภายใน

3.คำสั่งเกี่ยวกับตัวจับเวลา/ตัวนับ (TIMER/COUNTER INSTRUCTION)

ชุดคำสั่งเกี่ยวกับตัวจับเวลา/ตัวนับ จะทำหน้าที่เริ่มการทำงานของตัวจับเวลา/ตัวนับ เป็นตัวเคลื่อนย้ายข้อมูลระหว่างแอกคิวมิวเลเตอร์กับตัวจับเวลา/ตัวนับ การทำงานของตัวจับเวลา/ตัวนับ สามารถควบคุมได้โดยคำสั่งโดยตรง

ตัวจับเวลา/ตัวนับ สามารถใช้เป็นตัวจับเวลาที่ใช้ฐานเวลาจากสัญญาณนาฬิกาภายในหรือสัญญาณนาฬิกาภายนอก กับตัวนับสัญญาณที่เข้ามาทางขาT1 การเลือกการทำงานทั้งสองแบบเลือกได้โดยคำสั่ง นอกจากนี้ตัวจับเวลา/ตัวนับสามารถอ่านหรือเขียนผ่านทางแอกคิวมิวเลเตอร์ได้ไม่ว่าตัวนับหรือตัวจับเวลาทำงานอยู่หรือไม่ ทำให้สามารถตรวจสอบได้ทุกเวลาที่ต้องการ

4.คำสั่งเกี่ยวกับแอกคิวมิวเลเตอร์(ACCUMULATOR INSTRUCTION)

ชุดคำสั่งเกี่ยวกับแอกคิวมิวเลเตอร์จะประกอบด้วย การบวกข้อมูลที่มีตัวทดและไม่มีตัวทด การทำงานเกี่ยวกับลอจิก คำสั่งเหล่านี้คือ ADD,AND,OR,XOR หรือ ROTATE เพื่อสร้างรูปแบบของข้อมูลตรง ข้อมูลสำหรับหน่วยความจำหรือการเลือกชุดของเรจิสเตอร์ ข้อมูลสามารถเคลื่อนย้ายระหว่างแอกคิวมิวเลเตอร์กับเรจิสเตอร์หรือหน่วยความจำ ขึ้นกับคำสั่งที่ใช้งาน นอกจากนี้ยังมีคำสั่งพิเศษสำหรับแอกคิวมิวเลเตอร์ คือ SWAP A และ XCHD A ซึ่งได้กล่าวไปแล้วในกรณีที่แอกคิวมิวเลเตอร์คำนวณให้อยู่ในรูปแบบของ BCD ได้ นอกจากการทำงานทางคณิตศาสตร์แล้วชุดคำสั่งของแอกคิวมิวเลเตอร์ยังประกอบด้วย คำสั่งเกี่ยวกับวนรอบซ้ายหรือขวารวมหรือไม่รวมบิตตัวทด COMPLEMENT, INCREMENT, DECREMENT หรือการทำให้เป็นศูนย์

เนื่องจากชุดคำสั่งของตระกูล MCS-48 ไม่มีคำสั่งการลบ ดังนั้นเมื่อต้องการลบเลขขึ้น จะต้องใช้คำสั่ง 3 คำสั่งที่ทำหน้าที่การลบโดยผ่านแอกคิวมิวเลเตอร์ คือ

```
CPL A COMPLEMENT THE ACCUMULATOR
ADD A ADD THE VALUE TO THE ACCUMULATOR
INC A ADD 1 TO ACCUMULATOR
```

หลังจากทำคำสั่งดังกล่าว ผลลัพท์จะเก็บไว้ในแอกคิวมิวเลเตอร์ เมื่อต้องการเปลี่ยนแปลงข้อมูลของเรจิสเตอร์แสดงสถานะโปรแกรม สามารถทำได้โดยการเขียนผ่านแอกคิวมิวเลเตอร์ได้

5.คำสั่งการกระโดด(BRACH INSTRUCTION)

ชุดคำสั่งกระโดดนี้ จะทำหน้าที่การกระโดดที่มีเงื่อนไขหรือไม่มีไปยังหน่วยความจำโปรแกรมใด ๆ เพื่อให้ทำคำสั่งในส่วนของหน่วยความจำโปรแกรมนั้น ๆ ในกรณีของคำสั่งกระโดดแบบไม่มีเงื่อนไขนั้น จะสามารถกระโดดไปยังส่วนใดส่วนหนึ่งของหน่วยความจำโปรแกรมภายในชุดของหน่วยความจำโปรแกรมใดโปรแกรมหนึ่ง หมายความว่า การกระโดดของโปรแกรมจะอยู่ในช่วง 2 กิโลไบต์เท่านั้น สำหรับกรณีที่ต้องการกระโดดข้ามระหว่างชุดของหน่วยความจำโปรแกรม สามารถทำได้โดยการใช้คำสั่งเลือกชุดของหน่วยความจำโปรแกรมที่ต้องการกระโดดก่อนที่ใช้คำสั่งกระโดด คำสั่งการเลือกชุดของหน่วยความจำโปรแกรมจะมีผลทำงานเมื่อทำคำสั่งกระโดด

ในกรณีของการใช้ชุดคำสั่งกระโดดแบบมีเงื่อนไขนั้น สามารถตรวจสอบสัญญาณอินพุตหรือสถานะการทำงานก่อนทำการกระโดดได้ เมื่ออินพุตหรือสถานะการทำงานถูกต้องตามเงื่อนไขหรือไม่ เงื่อนไขที่ใช้กับคำสั่งการกระโดดมีดังนี้

T0 = 1 หรือ 0

T1 = 1 หรือ 0

INT = 0

แอกคิวมิวเลเตอร์ = 0 หรือ <>0

แอกคิวมิวเลเตอร์บิต = 1

CARRY = 1 หรือ 0

F0 = 1

F1 = 1

TIMER FLAG = 1

การกระโดดแบบมีเงื่อนไขนี้ จะมีช่วงกระโดดอยู่ในเพจเดียว ขณะที่ทำงานคำสั่งนั้นอยู่หรือภายในช่วง 256 เมื่อเงื่อนที่ทดสอบถูกต้องการทำงานจะกระโดดไปยังส่วนที่กำหนดทันที นอกจากเงื่อนไขที่กล่าวมาแล้วยังมีการกระโดดเมื่อค่าเรจิสเตอร์ตัวใดตัวหนึ่งไม่เป็นศูนย์หลังจากมีการลดค่าเรจิสเตอร์นั้น ๆ

6.คำสั่งเกี่ยวกับอุปกรณ์ภายนอก (INPUT/OUTPUT INSTRUCTION)

ชุดคำสั่งเกี่ยวกับอุปกรณ์ภายนอกประกอบด้วย การเคลื่อนย้ายข้อมูลระหว่างแอกคิวมิวเลเตอร์กับพอร์ต โดยที่พอร์ตจะแลทช์ค่าที่แอกคิวมิวเลเตอร์ส่งให้กับเอาพุตพอร์ต แต่กรณีเป็นอินพุตพอร์ตนั้นจะไม่มีการแลทช์ นอกจากนี้ยังสามารถทำการแอนด์หรือออร์โดยตรงกับพอร์ต โดยผลลัพท์ของการแอนด์และออร์จะเก็บไว้ที่พอร์ตนั้นอยู่ จากวิธีการแอนด์หรือออร์ได้โดยตรงนี้เอง ทำให้สามารถเซตค่าของแต่ละบิตของพอร์ตได้โดยตรง ในกรณีต้องการเป็นอินพุตพอร์ตนั้น จะต้องทำให้บิตที่ต้องการทำเป็นอินพุตมีลอจิกเป็น 1 เสมอ

บัสของตระกูล MCS - 48 ยังสามารถทำเป็นพอร์ตได้ โดยจะเป็นพอร์ตชนิด BI-DIRECTIONAL และยังสามารถแอนด์หรือออร์ได้โดยตรงเช่นเดียวพอร์ต 1 และ พอร์ต 2 การทำงานของบัส เมื่อทำหน้าที่เป็นพอร์ตจะต่างกับพอร์ต 1และพอร์ต 2 ตรงที่ว่าการทำงานจะเป็นอินพุตหรือเอาท์พุตอย่างใดอย่างหนึ่งเท่านั้น เมื่อใช้บัสเป็นอินพุต/เอาท์พุตพอร์ต จะหน้าที่เป็นอินพุตตลอดเวลา หลังทำการรีเซ็ตแล้ว และจะเป็นเอาท์พุตเมื่อใช้คำสั่ง OUT BUS,A เมื่อใช้คำสั่งนี้แล้วไม่สามารถกลับไปทำเป็นอินพุตได้ จนกว่าจะใช้เป็นบัสภายนอกหรือระบบถูกรีเซ็ตใหม่

นอกจากนี้บัสยังทำหน้าที่ติดต่อกับหน่วยความจำข้อมูลภายนอกก็ได้ โดยใช้คำสั่ง MOVX ซึ่งจะทำให้เกิดสัญญาณ RD หรือ WR ขึ้นกับหน้าที่ที่ใช้ เมื่อบัสไม่ได้ใช้งานจะอยู่ในสถานะ TRI-STATEหรือลอยตัว

อินพุตเอาท์พุตของตระกูล MCS - 48 สามารถขยายเพิ่มได้จาก 3 พอร์ตเป็น 7 พอร์ต โดยอาศัย 4 บิตแรกของพอร์ต 2เพื่อทำหน้าที่ในการขยาย พอร์ตที่ขยายออกมานี้จะมีคำสั่งในการแอนด์หรือออร์แยกออกต่างหาก แต่การแอนด์หรือออร์ของพอร์ตที่เพิ่มขึ้นนี้ไม่สามารถแอนด์หรือออร์ข้อมูลโดยตรง ต้องผ่านแอกคิวมิวเลเตอร์เสมอ

คำสั่ง OUTL BUS,A จะใช้ได้ในกรณีที่ใช้ตระกูล MCS - 48 ทำงานเพียงตัวเดียว และคำสั่งนี้สามารถทำงานกับคำสั่ง MOVX ได้ แต่ต้องระวังในการใช้คำสั่ง เพราะจะทำให้รูปแบบของพอร์ตที่ส่งข้อมูลออกไปแล้วจะสูญหายได้

7.คำสั่งเกี่ยวกับเรจิสเตอร์(REGISTER INSTRUCTION)

ชุดคำสั่งเกี่ยวกับเรจิสเตอร์ ใช้สำหรับทำการเพิ่มหรือลดค่าของเรจิสเตอร์ครั้งละหนึ่ง นอกจากนี้ยังสามารถเพิ่มข้อมูลในหน่วยความจำข้อมูลภายใน ครั้งละหนึ่งโดยใช้เรจิสเตอร์ RO หรือ R1 เป็นตัวชี้ได้อีกด้วย

8.คำสั่งเกี่ยวกับโปรแกรมย่อย(SUBROUTINE INSTRUCTION)

ชุดคำสั่งของโปรแกรมย่อย ใช้สำหรับการเรียกและกลับจากโปรแกรมย่อย ในการเรียกโปรแกรมย่อย สามารถเรียกได้จากหน่วยความจำโปรแกรมชุดหนึ่งไปยังอีกชุดหนึ่งได้ โดยต้องใช้คำสั่งเลือกชุดหน่วยความจำโปรแกรมก่อนใช้คำสั่ง CALL ในชุดคำสั่งนี้จะมีคำสั่งการกลับของโปรแกรมย่อยสองชนิดคือ ชนิดกลับโดยไม่เก็บสถานะกลับเข้าเรจิสเตอร์ของสถานะโปรแกรม และชนิดกลับโดยเก็บสถานะเรจิสเตอร์บอกสถานะโปรแกรม สำหรับการกลับชนิดหลังนี้ จะใช้สำหรับการกลับจากการอินเตอร์รัพต์

9.คำสั่งเกี่ยวกับแฟลก(FLAG INSTRUCTION)

ชุดคำสั่งเกี่ยวกับแฟลกจะให้ความสะดวกในการที่จะ COMPLEMENTหรือทำให้เป็นศูนย์ สำหรับหน้าที่ของแฟลกมีดังนี้

- CARRY ใช้แสดงเมื่อเกิด OVERFLOW เมื่อผลการทำ งานของแอกคิวมิวเลเตอร์เกินจำนวนบิต ของ
แอกคิวมิวเลเตอร์
- AUXILIARY ใช้แสดงเมื่อเกิด OVERFLOW ขึ้นระหว่าง4บิต
ในกรณีทำงานคณิตศาสตร์แบบ BCD เมื่อใช้
คำสั่งการปรับให้เป็นเลข BCD
- F0 และ F1 เป็นแฟลกที่ใช้ทั่วไปสำหรับกระโดดแบบมีเงื่อนไข
- CARRY และ F0 เป็นส่วนหนึ่งของเรจิสเตอร์บอกสถานะโปรแกรม
และจะเก็บรักษาไว้ในสแต็กเมื่อมีการเรียกโปรแกรมย่อย

10.คำสั่งเบ็ดเตล็ด(MISCELLANEOUS INSTRUCTION)

ชุดคำสั่งเบ็ดเตล็ดเป็นคำสั่งที่นอกเหนือจากที่กล่าวมาแล้ว ซึ่งได้แก่คำสั่ง

ที่ให้ตระกูล MCS - 48 ไม่มีการทำงานเพียงแต่ข้ามไปทำคำสั่งถัดไป

ชุดคำสั่งทั้งหมดของตระกูล MCS - 48

คำอธิบายความหมายของคำย่อต่างๆ ที่ใช้ในชุดคำสั่งที่จะอธิบาย ต่อไป

A,ACC แทน แอกคิวมิวเลเตอร์

AC แทน ตัวสำรอง(auxiliary carry)

addr แทน ค่าแอดเดรสในหน่วยความจำขนาด 12 บิต

Bd แทน ตำแหน่งบิตที่ใช้งาน โดย b มีค่า 0 ถึง 7

BUS แทน พอร์ตของบัส

C แทน บิตทด

CLK แทน สัญญาณนาฬิกา

CNT แทน ตัวนับจำนวนเหตุการณ์

CRR แทน การแปลงเลขฐานจากผลลัพท์ในเรจิสเตอร์

D แทน นิวมอนิคที่ใช้ตัวเลขขนาด 4 บิต

data แทน ข้อมูลตัวเลขขนาด 8 บิตหรือนิพจน์

DBF แทน ตัวฟลิบฟลอบของหน่วยความจำแบงค์

F0,F1 แทน แฟลก 0,แฟลก 1

I แทน การอินเตอร์รัพต์

P แทน นิวมอนิคที่ทำงานภายในเพจ(Page)

PC แทน ตัวนับโปรแกรม

Pp แทน พอร์ตที่กำหนด(p = 1,2 หรือ 4-7)

PSW แทน คำแสดงสถานะโปรแกรม

Ri แทน ตัวชี้ข้อมูลในหน่วยความจำ(i= 0 หรือ 1)

Rr แทน ตัวเรจิสเตอร์ที่ถูกใช้(r=0,1หรือ 0-7)

SP แทน ตัวชี้สแต็ก

T แทน ตัวจับเวลา

TF แทน แฟลกตัวจับเวลา

TO,T1 แทน บิตทดสอบ0,บิตทดสอบ1

X แทน นิวมอนิคที่ใช้แรมภายนอก

แทน ตัวบอกการใช้โหมดการกำหนดเลขที่อยู่ข้อมูลโดยทันที

@ แทน ตัวบอกการใช้โหมดการกำหนดเลขที่อยู่ข้อมูลโดยทางอ้อม

$ แทน ค่าของตัวนับโปรแกรมในปัจจุบัน

(X) แทน ค่าข้อมูลที่อยู่ใน X

((X)) แทน ค่าข้อมูลตำแหน่งแอดเดรสที่ถูกชี้โดย X

<--- แทน ถูกแทนด้วย

รายละเอียดชุดคำสั่ง MCS - 48

ADD A,Rr : การบวกข้อมูลในเรจิสเตอร์กับข้อมูลในแอกคิว มิวเลเตอร์

การทำงาน : (A) <--- (A)+(Rr) r=0-7

ADD A,@Rr : เป็นการบวกข้อมูลของหน่วยความจำภายในกับข้อมูลใน ACC

การทำงาน : (A) <--- ((Ri))

ADD A,#DATA: บวกข้อมูลโดยทันทีกับ ACC

การทำงาน : (A) <--- (A) + DATA

ADDC A,@Ri : บวกตัวทดร่วมกับข้อมูลในหน่วยความจำกับACC

การทำงาน : (A) <--- (A) + ((Ri)) + (C) i=0,1

ANL A,Rr : การ AND กันทางตรรกระหว่าง ACC กับ เรจิสเตอร์

ANL A,@Ri : การ AND กันทางตรรกระหว่าง ACC กับค่า
ข้อมูลมาส์ในหน่วยความจำ

การทำงาน : (A) <--- (A) AND ((Ri)) i=0,1

ANL A,#data: การ ANDกันทางตรรกระหว่าง ACC กับข้อมูล
มาสค์โดยทันที

การทำงาน : (A) <--- (A) AND data

ANL BUS,#data: การ AND กันทางตรรกระหว่างข้อมูลบนพอร์ต
บัสกับข้อมูลมาสค์โดยทันที

การทำงาน : (BUS) <--- (BUS) AND data

ANL Pp,#data: การ AND กันระหว่างพอร์ต1 หรือ 2 ด้วยข้อ
มูลมาสค์โดยทันที

การทำงาน :

ANL Pp,A : การ AND ทางตรรกระหว่างพอร์ต 4-7 กับ
ค่ามาสค์ ใน ACC

การทำงาน : (Pp) <--- Pp AND (A0 - 3) P=4-7

Call address : การเรียกโปรแกรมย่อย

การทำงาน : ((SP)) <--- (PC),(PSW4-7)

(SP) <--- (SP) + 1

(PC 8-10) <--- (addr 8-10)

(PC 0-7) <--- (addr 0-7)

(PC 11) <--- DBF

CLR A : เคลียร์ค่าใน ACC

การทำงาน : (A) <--- 0

CLR C : เคลียร์บิตทด

การทำงาน : (C) <--- 0

CLR F1 : เคลียร์แฟลก 1

การทำงาน : (F1) <--- 0

CLR F0 : เคลียร์แฟลก 0

การทำงาน : (F0) <--- 0

CPL A : เป็นการกลับค่า(complement)ลง ACC

การทำงาน : (A) <--- NOT(A)

CPL C : กลับค่าในบิตทด

การทำงาน : (C) <--- NOT(C)

CPL F0 : กลับค่าในแฟลก 0

การทำงาน : (F0) <--- NOT(F0)

CPL F1 : กลับค่าในแฟลก 1

การทำงาน : (F1) <--- NOT(F1)

DA A : การปรับค่าใน ACC เป็นเลขฐานสิบ

การทำงาน :

DIS I : ไม่ให้เกิดอินเตอร์รัพต์จากภายนอก

DIS TCNTI : ไม่ให้เกิดอินเตอร์รัพต์จากตัวจับเวลา/ตัวนับ

EN I : การทำให้เกิดอินเตอร์รัพต์จากภายนอก

EN TCNTI : การทำให้เกิดการอินเตอร์รัพต์จากตัวจับเวลา /ตัวนับ

ENTO CLK : ให้มีความสามารถส่งสัญญาณนาฬิกา ออกสู่ภายนอก

INC A : เพิ่มค่าที่ ACC อีกหนึ่ง

การทำงาน : (A) <--- (A) + 1

INC @Rr : เพิ่มค่าข้อมูลที่หน่วยความจำข้อมูลที่ถูกชี้ตำแหน่ง Rr

การทำงาน : ((Rr)) <--- ((Rr)) + 1 r=0,1

IN A,PO : ข้อมูลที่อินพุตพอร์ต 0 ย้ายเข้า ACC

JBb Address : กระโดดถ้าบิตใดบิตหนึ่งของ b เซตเป็นหนึ่ง

การทำงาน : (PC0-7) <--- addr ถ้าบิต b=1

JC Address : จะกระโดดถ้าบิตทดเซตเป็นหนึ่ง

การทำงาน : (PC0-7) <--- addr ถ้าบิต c=1

(PC) = (PC) + 2 ถ้าบิต c=0

JFO Address : กระโดดถ้าบิตแฟลก 0 เซตเป็นหนึ่ง

การทำงาน : (PC0-7) <--- addr ถ้าบิตF0=1

(PC) = (PC) + 2 ถ้าบิตF0=0

JF1 Address : กระโดดถ้าแฟลก 1 เซตเป็นหนึ่ง

การทำงาน : (PC0-7) <--- addr ถ้าบิตF1=1

(PC) = (PC) + 2 ถ้าบิตF1=0

JMP Address : กระโดดโดยตรงภายใน 2 กิโลไบต์

การทำงาน : (PC8-10) <--- addr 8 - 10

(PC0-7) <--- addr 0 - 7

(PC11) <--- DBF

JMPP @A : การกระโดดไปภายของในเพจของคำสั่งนี้

การทำงาน : (PC0-7) <--- ((A))

JNC Address : กระโดดถ้าบิตทดไม่เซตเป็นหนึ่ง

การทำงาน : (PC0-7) <--- addr ถ้าบิตทด= 0

(PC) = (PC) + 2 ถ้าบิตทด= 1

JNI Address : กระโดดไปถ้าสัญญาณอินเตอร์รัพต์ ที่เข้ามามี
สถานะต่ำ

การทำงาน : (PC0-7) <--- addr ถ้าI = 0

(PC) = (PC) + 2 ถ้าI = 1

JNT0 Address : กระโดดไปถ้าสัญญาณที่เข้ามาที่ขา TEST 0
(T0) มีสถานะต่ำ

การทำงาน : (PC0-7) <--- addr ถ้าT0 = 0

(PC) = (PC) + 2 ถ้าT0 = 1

JNT1 Address : กระโดดไปถ้าสัญญาณที่เข้ามาที่ขา TEST 1
(T1) มีสถานะต่ำ

การทำงาน : (PC0-7) <--- addr

(PC) = (PC) + 2

JNZ Address : กระโดดไปถ้าใน ACC ไม่เป็นศูนย์

การทำงาน : (PC0-7) <--- addr ถ้าA <> 0

(PC) = (PC) + 2 ถ้าA = 0

JTF Address : กระโดดถ้าแฟลกตัวตั้งเวลาเซตเป็นหนึ่ง

การทำงาน : (PC0-7) <--- addr ถ้าTF = 2

(PC) = (PC) + 2 ถ้าTF = 0

JT0 Address : กระโดดไปถ้าสัญญาณที่เข้ามาที่ขา TEST 0

(T0) มีสถานะสูง

การทำงาน : (PC0-7) <--- addr ถ้า T0 = 1

(PC) = (PC) + 2 ถ้า T0 = 0

JT1 Address : กระโดดไปถ้าสัญญาณที่เข้ามาที่ขา T1มีสถานะ

สูง

การทำงาน : (PC0-7) <--- addr ถ้า T1 = 1

(PC) = (PC) + 2 ถ้า T1 = 0

JZ Address : กระโดดไปถ้าแอกคิวมิวเลเตอร์มีค่าเป็นศูนย์

การทำงาน : (PC0-7) <--- addr ถ้า T1 = 1

(PC) = (PC) + 2 ถ้า T1 = 0

MOV A,#data ย้ายข้อมูลโดยทันที (ข้อมูลที่ติดมากับคำสั่ง)

เข้าสู่ ACC

การทำงาน : (A) <--- DATA

MOV A,PSW : ย้ายข้อมูลในเรจิสเตอร์ PSW เข้าสู่ ACC

การทำงาน : (A) <--- (PSW)

MOV A,Rr : ย้ายข้อมูลภายในเรจิสเตอร์ทำงานไปยัง ACC

การทำงาน : (A) <--- (Rr) r = 0 - 7

MOV A,@Ri : ย้ายข้อมูลจากหน่วยความจำข้อมูลไปยัง ACC

การทำงาน : (A) <--- ((Ri)) i = 0 - 1

MOV A,T : ย้ายข้อมูลภายในเรจิสเตอร์ตัวจับเวลา/ตัวนับ
ไปยัง ACC

การทำงาน : (A) <--- (T)

MOV PSW,A ย้ายข้อมูลภายใน ACC ไปยังเรจิสเตอร์ PSW

การทำงาน : (PSW) <--- A

MOV Rr,A : ย้ายข้อมูลใน ACC ไปยังเรจิสเตอร์ทำงาน

การทำงาน : (Rr) <--- (A) r = 0 - 7

MOV Rr,#data : ย้ายข้อมูลโดยทันทีไปยัง ACC

การทำงาน : (Rr) <--- data

MOV @Ri,A : ย้ายข้อมูลใน ACC ไปยังหน่วยความจำภายใน

การทำงาน : ((Ri)) <--- (A)

MOV @R, #data : ย้ายข้อมูลโดยทันทีไปยังหน่วยความจำข้อมูลภายใน

การทำงาน : ((R)) <--- data i = 0 - 1

MOV T,A : ย้ายข้อมูลในแอกคิวมิวเลเตอร์ไปยังเรจิสเตอร์
ตัวจับเวลา/ตัวนับ

การทำงาน : (T) <--- (A)

MOVD A,P : ย้ายข้อมูลจากพอร์ต 4-7 ไปยังแอกคิวมิวเลเตอร์

การทำงาน : (0 - 3) <--- (Pp)
(4-7) <--- 0

MOVD Pp,A : ย้ายข้อมูลจากแอกคิวมิวเลเตอร์ไปยังพอร์ต

การทำงาน : (Pp) <--- (A0-3) P=4-7

MOVP A, @A : ย้ายข้อมูลที่ถูกกำหนดด้วยเพจเดียวกับคำสั่ง
ไปยังแอกคิวมิวเลเตอร์

การทำงาน : (PC0-7) <--- (A)
(A) <--- (PC)

MOV P3,@A : ย้ายข้อมูลที่ถูกกำหนดด้วยเพจ 3 ไปยังแอกคิว
มิวเลเตอร์

การทำงาน : (PCO-7) <--- (A)

MOVX A,@R : ย้ายข้อมูลที่อยู่หน่วยความจำภายนอกไปยังแอกคิว
มิวเตอร์

การทำงาน : (A) <--- (R)

MOVX @Ri<A : ย้ายข้อมูลที่อยู่ใน แอกคิวมิวเลเตอร์ไปยังหน่วย
ความจำภายนอก

การทำงาน : ((R)) <--- (A)

ORL A,Rr : การทำงานตรรก OR กันระหว่างข้อมูลแอกคิวมิว
เลเตอร์กับค่า MASX ในเรจิสเตอร์ทำงาน

การทำงาน : (A) <--- (A) OR (Rk) X= 0-7

ORL A,@Ri : การทำงานตรรก OR กันระหว่างข้อมูล แอกคิวมิว
เลเตอร์ กับค่า MASK ในหน่วยความจำ

การทำงาน : (A) <--- (A) OR ((R)) I = 0-1

46

ORL BUS,#data : การทำงานตรรก OR กันระหว่างข้อมูลที่พอร์ตบัส
กับข้อมูล MASK โดยทันที ไม่มีใน 8021 ; 8022

การทำงาน : (BUS) <--- (BUS) OR data

ORL Pp ,#data : การทำงานตรรก OR ระหว่างข้อมูลที่พอร์ต 1 หรือ 2 กับข้อมูล MASK โดยทันที

การทำงาน : (Pp) <--- (Pp) OR data P = 1-2

ORLD P , A : การทำงานตรรก OR กันระหว่างข้อมูลที่พอร์ต 4-7 กับข้อมูล MASK ในแอกคิวมิวเลเตอร์

การทำงาน : (Pp) <--- (Pp) OR (A0-3) P = 4-7

OUTL BUS,A : ส่งข้อมูลจากแอกคิวมิวเลเตอร์ไปยังพอร์ตบัส

การทำงาน : (BUS) <--- (A)

OUTL P, A : การส่งข้อมูลในแอกคิวมิวเลเตอร์พอร์ต 1 หรือ 2

การทำงาน : (Pp) <--- (A) P= 1-2

RET : กลับคืนสู่โปรแกรมหลักโดยไม่นำค่า PSW เก่ากลับคืน

การทำงาน : (SP) <--- (SP) - 1
(PC) <--- ((SP))

RETR : กลับคืนสู่โปรแกรมหลักด้วยการนำ PSW เก่ากลับคืน

การทำงาน : (SP) <--- (SP) - 1
(PC) <--- ((SP))
(PSW 4-7)) <--- ((SP))

RL A : วนบิตทางซ้ายโดยไม่ใช้บิตทด

การทำงาน : (An+1) <--- (An)

(A0) <--- (A7)

RLC A : วนบิตทางซ้ายโดยการใช้บิตทดด้วย

การทำงาน : (An+1) <--- (An) n = 0-6

(A0) <--- (C)

RR A : วนบิตรอบทางขวาโดยไม่ใช้บิตทด

การทำงาน : (An) <--- (An+1) n = 0-6

(A7) <--- A0

RRC A : วนบิตทางขวา โดยการใช้บิตทดด้วย

การทำงาน : (An) <--- (An+1) n = 0-6

(A7) <--- (C)

(C) <--- (A0)

SEL MB0 : การเลือกทำงานในหน่วยความจำแบงก์ 0

การทำงาน : (DBF) <--- 0

SEL MB1 : การเลือกทำงานในหน่วยความจำแบงก์ 1

การทำงาน : (DBF) <--- 1

SKL RBO : การเลือกชุดเรจิสเตอร์ทำงานแบงก์ 0

SEL RB1 : การเลือกชุดเรจิสเตอร์ทำงานแบงก์ 1

STOP TCNT : หยุดการทำงานของตัวจับเวลา/ตัวนับ

STRT CNT : สั่งตัวนับจำนวนเริ่มทำงาน

STRT T : สั่งตัวจับเวลาเริ่มทำงาน

SWAP A : การสลับค่าขนาดนิบเบิลภายในแอกคิวมิวเลเตอร์

XCH A,Rr : การสลับข้อมูลขนาดหนึ่งไบต์ระหว่างข้อมูลที่อยู่ใน
แอกคิวมิวเลเตอร์กับข้อมูลในเรจิสเตอร์ทำงาน

XCH A,@Ri : สลับข้อมูลขนาด 1 ไบต์ในแอกคิวมิวเลเตอร์กับ
ข้อมูลในหน่วยความจำ

XCHD A, @Ri : สลับค่าข้อมูลขนาด 4 บิตแรกในแอกคิวมิวเลเตอร์
กับข้อมูลในหน่วยความจำ

XRL A,Rr : การทำงานตรรก XOR ระหว่างข้อมูลที่ ACCกับ
ข้อมูล MASK ในเรจิสเตอร์

XRL A,@Ri : การทำงานตรรก XOR ระหว่างข้อมูลที่ ACCกับ
ข้อมูล MASK ในหน่วยความจำภายใน

XRL A,#DATA : การทำงานตรรก XOR ระหว่างข้อมูล ACC กับ

ข้อมูล MASK โดยทันที การทำงาน : (A) <--- (A)
AND(Rr) r=0-7

บทที่ 3

การพัฒนา EMULATOR

EMULATOR 48 สร้างขึ้นเพื่อเป็นเครื่องมือช่วยผู้ใช้ที่ต้องการจัดการกับ PROGRAM MCS -48 ที่อยู่ในหน่วยความจำข้อมูลภายนอก (EXTERNAL DATA MEMORY) มากกว่าอยากให้ใช้ใน INTERNAL ROM หรือ EPROM EMULATOR 48 จะเป็นตัว debug โปรแกรมสามารถทำได้ง่ายและรวดเร็ว

ผู้เขียนโปรแกรมอาจเข้าไปข้างในระบบตามคู่มือบอกไว้ หรือ เข้าโดยตรงโดยการ serial link มาจากไมโครฯ และในขณะโหลดโปรแกรมอยู่ก็ยังสามารถที่จะแก้ไขโปรแกรมบน board ได้โดยใช้ keyboard และ ใช้ display เป็นตัวแสดงผล EMULATOR 48 ในตัวมันจะประกอบไปด้วย RAM , ACCUMULATOR , TIMER /COUNTER และสภาวะความจุของรีจิสเตอร์ สามารถอ่านและแก้ไขได้โดย keyboard และ display ดังแสดงไว้ในรูปข้างล่าง

หลักการทำงาน

เมื่อมีไฟจ่ายเข้าสู่ระบบโปรแกรมเคาท์เตอร์(Program counter)จะถูกเซต(set)ให้เป็นศูนย์ โดยอัตโนมัติและนำค่านี้ไปเฟทช์(fetch)เอาข้อมูลจาก EPROM#2732 เพื่อที่จะนำข้อมูลไปประเมินผล

เมื่อCPUต้องการจะนำข้อมูลจาก EPROMมาใช้ CPUจะส่งสัญญาณ ALEที่เป็น "1" ไปยังขา Gของ 74LS373เพื่อใช้เป็นตัวพักค่าแอดเดรส (ADDRESS)ที่ส่งมาจาก CPU จากนั้นCPUก็จะสร้างสัญญาณออกมาที่ขา PSEN มีค่า "0"ไปยังขา CE ของ EPROM เบอร์ #2732เพื่อนำข้อมูลของแอดเดรสที่กำหนดส่งออกมาไปยังบัส (BUS) ซึ่งในขณะนี้ที่ ALE จะมีค่า "0" เพื่อเป็นการกำหนดว่าPORTที่ใช้ส่งแอดเดรสในตอนต้นได้เปลี่ยนมาเป็นPORT ที่จะรับข้อมูลแล้ว ข้อมูลที่เข้ามาจากบัสจะเข้าสู่ CPUโดยใช้ PORTนี้ จากนั้น CPU จะนำข้อมูลที่ได้ไปประเมินผลต่อไป

การติดต่อระหว่าง CPU กับ PORT ต่าง ๆ

ในอิมูเลเตอร์ตัวนี้จะใช้IC#8255 เป็นตัวรองรับในการติดต่อระหว่างPORT กับ CPU ซึ่งต้องประกอบด้วยฮาร์แวร์และซอฟแวร์(HARDWARD AND SOLFWARD)ควบคู่กัน เพื่อให้สามารถเข้าใจได้ง่ายเราจะกล่าวถึงโครงสร้างทางด้านฮาร์แวร์ก่อนและจะกล่างถึงซอฟแวร์ในตอนท้าย

เริ่มแรกเราต้องกำหนดให้ว่าเราต้องการให้PORT A,B,C เป็นINPUT หรือ OUTPUT PORTโดยการส่งสัญญาณไปที่ขา A0,A1ให้เป็น "1"จากนั้นเราก็ส่ง CONTROL WORD เข้าไปเก็บไว้ในรีจิสเตอร์(REGISTER)
ควบคุมของ 8255 PORTต่าง ๆ ที่อยู่ใน8255ก็จะมีคุณสมบัติตามที่เรากำหนดไว้ แต่การส่งต่าง ๆ ของ CONTROL WORD เป็นการกระทำทางซอฟแวร์ซึ่งจะกล่าวถึงต่อไป

ในอิมูเลเตอร์ตัวนี้เราได้กำหนดให้ PORT AและPORT Bเป็น OUTPUT PORT และ PORTCเรากำหนดให้เป็น INPUT PORT

เมื่อเรากำหนดการใช้งานของ PORTแต่ละ PORTได้แล้วต่อไปเรากล่าวถึงการติดต่อระหว่างCPU กับ PORTต่อไป เมื่อCPUต้องการติดต่อกับ PORTจะต้องกำหนดว่าต้องเป็นPORT ไหนต้องการจะทำอะไรกับ PORT(อ่านหรือเขียน)โดยจะส่งสัญญาณไปที่ขาRD,WR,A0และA1 ดังตารางข้างล่าง

DEVICE PIN				REGISTER
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 data
0	1	1	1	Illegal read register

จากตารางข้างบนเป็นการกำหนดรูปแบบการทำงานต่าง ๆ ของPORTใน 8255 โดยรับสัญญาณมาจาก CPU เช่นถ้าเราต้องการส่งข้อมูลไปยังPORT Aเราก็ส่งสัญญาณให้ RD เป็น "1" WR เป็น "0"และค่า A1,A0 เป็น "0"หรือถ้าต้องการจะอ่านข้อมูลจาก PORT Cเราก็ส่งสัญญาณให้ขา RD เป็น"0"ขา WR เป็น "1"ขา A1 เป็น "1" ขา A0 เป็น "0" เป็นต้น

เมื่อเรากำหนด PORT ที่จะใช้งานได้แล้วต่อไปเราก็กำหนดให้ CPUส่งสัญญาณไปที่ขาCEของ 8255 โดยผ่านตัว DECODER เบอร์ 74LS139 และนำสัญญาณ RD และ WR มา AND กันแล้ว ส่งสัญญาณที่ได้ไปที่ขา Gของ 74LS139เพื่อที่จะให้ตัว 74LS139 เริ่มทำงาน จากนั้น CPUจะส่งข้อมูลเข้าไปในบัสหรืออาจจะรับข้อมูลจากบัส โดยมี BUFFER2ทางเบอร์ 74LS245 เป็นตัวเก็บข้อมูล DIR เป็นตัวกำหนดทิศทางของมูล

เมื่อทราบถึงขั้นตอนการทำงานแล้ว ที่นี้เราจะมาพิจารณาถึงการเขียนคำสั่งควบคุมการทำงานต่อไปดังตัวอย่างโปรแกรมข้างล่าง

```
                ;-----------------------------------------
                ;SCAN KEYBOARD AND DISPLAY
                ;DATA SEND TO PORT A OF 8255
                ;CONTROL DIGIT SEND TO PORT B OF 8255
                ;INPUT FOR MATRIX KEYBOARD TO PORT C OF 8255
                ;
041F            SCAN1:
041F 23FF               MOV     A,#PORT_CS
0421 B82D               MOV     R0,#STATUS
0423 50                 ANL     A,@R0
0424 39                 OUTL    P1,A
0425 B801               MOV     R0,#PORTB
0427 FA                 MOV     A,R2
```

```
0428 90                 MOVX    @RO,A
                ;
0429 FD                 MOV     A,R5
042A A8                 MOV     RO,A
042B F0                 MOV     A,@RO
042C B800               MOV     RO,#PORTA
042E 90                 MOVX    @RO,A
                ;
042F B902               MOV     R1,#PORTC
0431 81                 MOVX    A,@R1
0432 37                 CPL     A
0433 C63E               JZ      PASS
0435 BBFF               MOV     R3,#OFFH
0437 EB37               DJNZ    R3,$
                ;
0439 81                 MOVX    A,@R1
043A 37                 CPL     A
043B AE                 MOV     R6,A
043C 8442               JMP     $+6
043E BB80       PASS:   MOV     R3,#080H
0440 EB40               DJNZ    R3,$
0442 83                 RET
```

DISPLAY

ในอีมูเลเตอร์ตัวนี้เราใช้ 7-Segment เป็นตัวแสดงผลมีทั้งหมด 8ตำแหน่ง เราจะใช้ PORT A และ PORT B เป็นเส้นทางในการติดต่อ โดย PORT Aจะเป็นตัวส่งข้อมูลในลักษณะการวนลูป (LOOP) ข้อมูลที่ส่งมายัง PORT Aจะมีการเปลี่ยนแปลงข้อมูล

ตามลูปไปเรื่อย ๆ จนกว่าจะมีการป้อนข้อมูลเข้ามาในลูปใหม่ การแสดงผลถึงจะมีการเปลี่ยนแปลง การที่เราส่งข้อมูลเป็นลูปใน PORT A เหตุผลเพราะทำให้เราเห็นการแสดงผลในแต่ละตำแหน่งได้พร้อมกัน ทั้งที่ความจริงตำแหน่งแต่ละตำแหน่งจะทำงานสลับกันโดยไม่มีตำแหน่งทำงานพร้อมกันทั้งนี้เพราะตำแหน่งต่าง ๆ จะทำงานเร็วมากจนเราไม่สามารถมองด้วยตาเปล่าเห็นว่ามันติด ๆ ดับ ๆ ทำให้ดูคล้ายว่ามันทำงานพร้อมกัน ส่วน PORT B จะเป็นตัวเลือกที่จะกำหนดให้ตำแหน่งไหนทำงาน โดยที่ PORT B จะทำการส่งสัญญาณในลักษณะชิฟ(Shift)สัญญาณไปเรื่อย ๆ เช่น ลูปแรก PORT B มีค่าใน PORT เป็น 01111111 พอวนลูปครั้งต่อไปจะมีค่า 10111111 ซึ่งเป็นอย่างนี้ไปเรื่อย ๆ พอถึงตัวสุดท้ายก็จะกลับมาเริ่มใหม่อีก

KEYBOARD

ในอีมูเลเตอร์ตัวนี้จะมีทั้งหมด 24 คีย์(KEY)สามารถที่จะเพิ่มได้ถึง 64 คีย์ใน 24 คีย์นี้ 16 คีย์ใช้เป็นตัวป้อนข้อมูล ส่วนคีย์ที่เหลือจะเป็นคีย์ฟังก์ชั่น (function)

การตรวจสอบตำแหน่งคีย์ที่กด(SCANKEY) อีมูเลเตอร์ตัวนี้ใช้ PORT B และ PORT C ของ 8255 ในการตรวจสอบคีย์ โดยมีข้อกำหนดว่า ถ้า PORT C มีค่าเป็น F(1111)ให้ถือว่าไม่มีการกดคีย์ แต่ถ้าเป็นค่าอื่น ๆ ให้ถือว่ามีการกดคีย์ และเอาค่าที่ PORT B และ PORT C ในขณะนั้นจากตารางที่สร้างไว้ ซึ่งเมื่อเปรียบเทียบเสร็จทำให้สามารถรู้ได้ว่า คีย์ที่กดนั้นเป็นคีย์อะไร ต้องไปทำงานอะไรเมื่อมีการกดคีย์นี้ ส่วน PORT B จะเป็นการตรวจที่ละ columnโดยจะส่งค่า "0"ไปแต่ละ column เรื่อย ๆ เมื่อถึง column สุดท้ายก็จะมาที่ columnแรกใหม่อีกเป็นเช่นนี้ไปเรื่อย ๆ ส่วนลักษณะการต่อของคีย์บอร์ดจะเป็นการต่อแบบพูลอัพ ซึ่งมีผลดีที่ช่วยในการยกระดับของสัญญาณที่ต่ำให้สูงได้อีกด้วย

จากที่กล่าวมาข้างต้นเป็นแค่เพียงหลักการ ซึ่งเราสามารถนำเอาหลักการมาเขียนเป็นโปรแกรมได้ดังโปรแกรมข้างล่าง

```
                ;-----------------------------------------
                ;       SCAN DISPLAY AND KEYBOADR
                ;
0443 FA         SCAN:   MOV     A,R2
0444 77                 RR      A
0445 AA                 MOV     R2,A
0446 FF                 MOV     A,R7
0447 0304               ADD     A,#04
0449 AF                 MOV     R7,A
044A CD                 DEC     R5
044B FD                 MOV     A,R5
044C 37                 CPL     A               ;2'COMPREMENT
044D 17                 INC     A
044E 031F               ADD     A,#1FH
0450 9658               JNZ     EXIT
0452 BA7F               MOV     R2,#7FH
0454 BD27               MOV     R5,#VIDEO
0456 BFFF               MOV     R7,#NULL
0455 83         EXIT:   RET                     ;RETURE
```

โปรแกรมที่ใช้ใน EMULATOR 48

ในเวลานี้โปรแกรม Emulator ที่ใช้สำหรับปฏิบัติงานนั้น จะนำไปเก็บไว้ใน RAM ภายนอก โครงสร้างหลักๆของ EMULATOR จะต้องประกอบด้วย

- 8048 Microprocessor
- 74373 Address latch
- RAM อย่างต่ำ 2 Kb ใช้สำหรับเก็บโปรแกรมที่ผู้ใช้ต้องการเขียนและปฏิบัติงานจริง ซึ่ง RAM สามารถที่จะขยายได้ถึง 4 Kbytes

ระบบการจัดการ (SYSTEM SUPERVISION)

MCS -48 กับ 74373 Address Latch และ โปรแกรมที่เก็บไว้ใน EPROM เบอร์ 2716 เราสามารถนำคำสั่งใน EPROM มาใช้เพื่อทำงานกับ คีย์บอร์ด, จอภาพ, แปลคำสั่งและควบคุม Serial Interfaces เป็นต้น

โดยทั่วไป MASTER PROCESSOR จะเป็นตัวติดต่อข้อมูลภายนอกให้กับหน่วยความจำระบบและยังเป็นตัวควบคุมการทำงานของขบวนการ

การเข้าถึงข้อมูลของ RAM ภายในของ EP

เราสามารถเข้าถึงข้อมูลได้โดยอาศัย Link Register เป็นสื่อในการเข้าถึงข้อมูล ซึ่งจากที่กล่าวในส่วนของ Program Break Sequence นั้นจะมีช่วงหนึ่งที่มีการเขียนโปรแกรมสำหรับ อ่านและส่งข้อมูลจาก RAM ภายในของ EP มายัง MP โดยที่สามารถทำการ RUN โปรแกรมเหล่านั้น ได้โดยการ CALL จากตำแหน่งภายใน Registerดังนั้นจะเห็นว่า Link Register เป็นตัวชี้ไปยังตำแหน่งโปรแกรม Monitorว่าจะให้ทำการ RUN ที่ตำแหน่งไหน ซึ่งแต่ละตำแหน่งจะมีคำสั่งที่อ่านข้อมูลของแต่ละตัว ดังนั้นหากต้องข้อมูลตำแหน่งใด ภายใน RAM ก็กำหนด Link Register ชี้ไปยังตำแหน่งของข้อมูลที่ ต้องการทำงาน

การใช้งาน

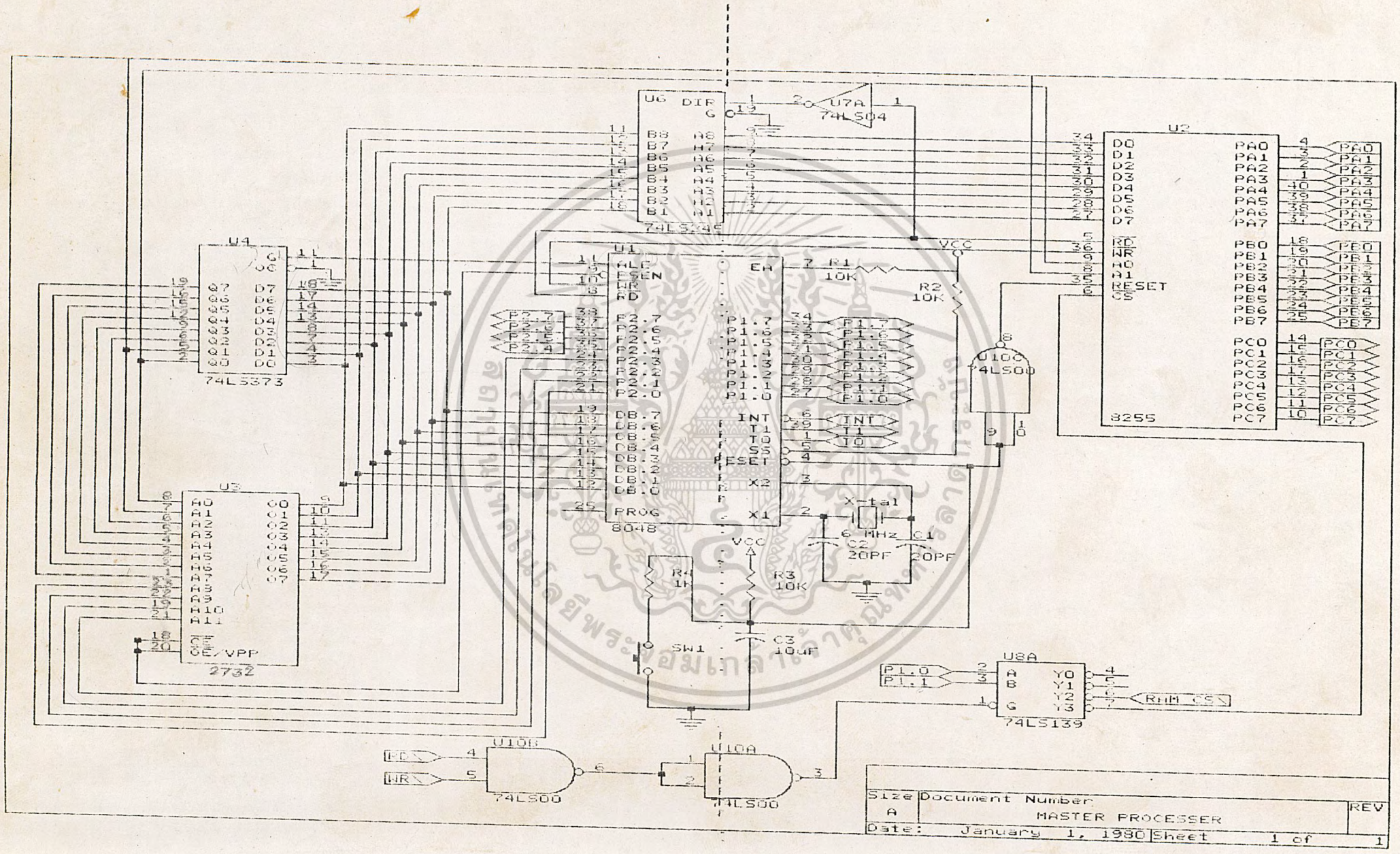
เมื่อป้อนไฟ 5 V. เข้าไปในเครื่อง เครื่องก็จะแสดง MCS-48 เป็นการบอกว่าพร้อมที่จะรับคีย์ต่อไป ซึ่งแต่ละคีย์มีการทำงานต่อไปนี้

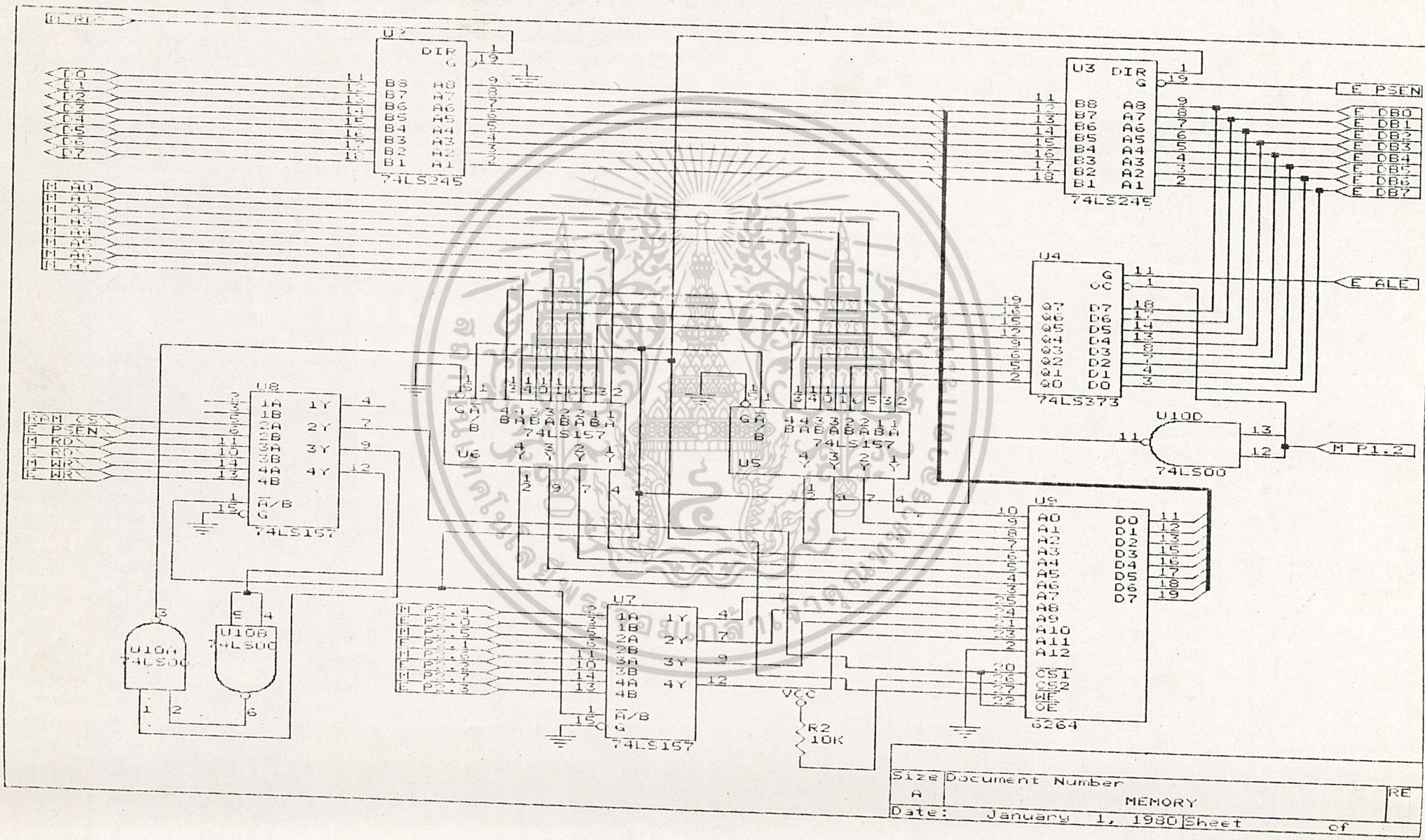
1. เมื่อต้องการดูข้อมูลหรือต้องการแก้ไขข้อมูล ในตำแหน่งที่เรากำหนด เราก็กดคีย์ PD เครื่องจะแสดง Pd ขึ้นมาให้เราใส่แอดเดรสที่ต้องการจะดูข้อมูล ในกรณีที่เราป้อนแอดเดรสถูกต้องตามที่เราต้องการแล้ว เราก็กดคีย์ MON เครื่องก็จะแสดงข้อมูลในแอดเดรสที่เรากำหนด แต่ในกรณีที่เราป้อนค่าแอดเดรสผิด เราต้องการจะลบแอดเดรสที่ผิด ทำได้โดยการกดคีย์ DEC ซึ่งเป็นคีย์ที่สามารถลบได้ที่ละหลัก เพื่อลบค่าแอดเดรสที่เราป้อนค่าผิด จากนั้นก็ป้อนแอดเดรสที่ถูกต้องเข้าไป แล้วกดคีย์ MON เพื่อที่จะดูข้อมูลต่อไป เมื่อ DISPLAY แสดงข้อมูลที่แอดเดรสนั้นแล้วเราสามารถแก้ไขข้อมูลได้โดยใส่ข้อมูลเข้าไปแทนที่ข้อมูลเก่าได้เลย

2. เมื่อมีการแสดงข้อมูลอยู่นั้น เราสามารถที่จะดูข้อมูลในแอดเดรสถัดไปได้โดยการกดคีย์ INC ซึ่งเป็นการเพิ่มค่าแอดเดรสหนึ่งแอดเดรส ทำให้สามารถเรียกดูข้อมูลถัดไปได้ หรือถ้าต้องการดูข้อมูลย้อนหลังเราสามารถทำได้โดยการกดคีย์ DEC ค่าแอดเดรสจะย้อนหลังไปหนึ่งแอดเดรส ซึ่งจะแสดงค่าข้อมูลที่แอดเดรสนั้น และสามารถทำการแก้ไขข้อมูลได้

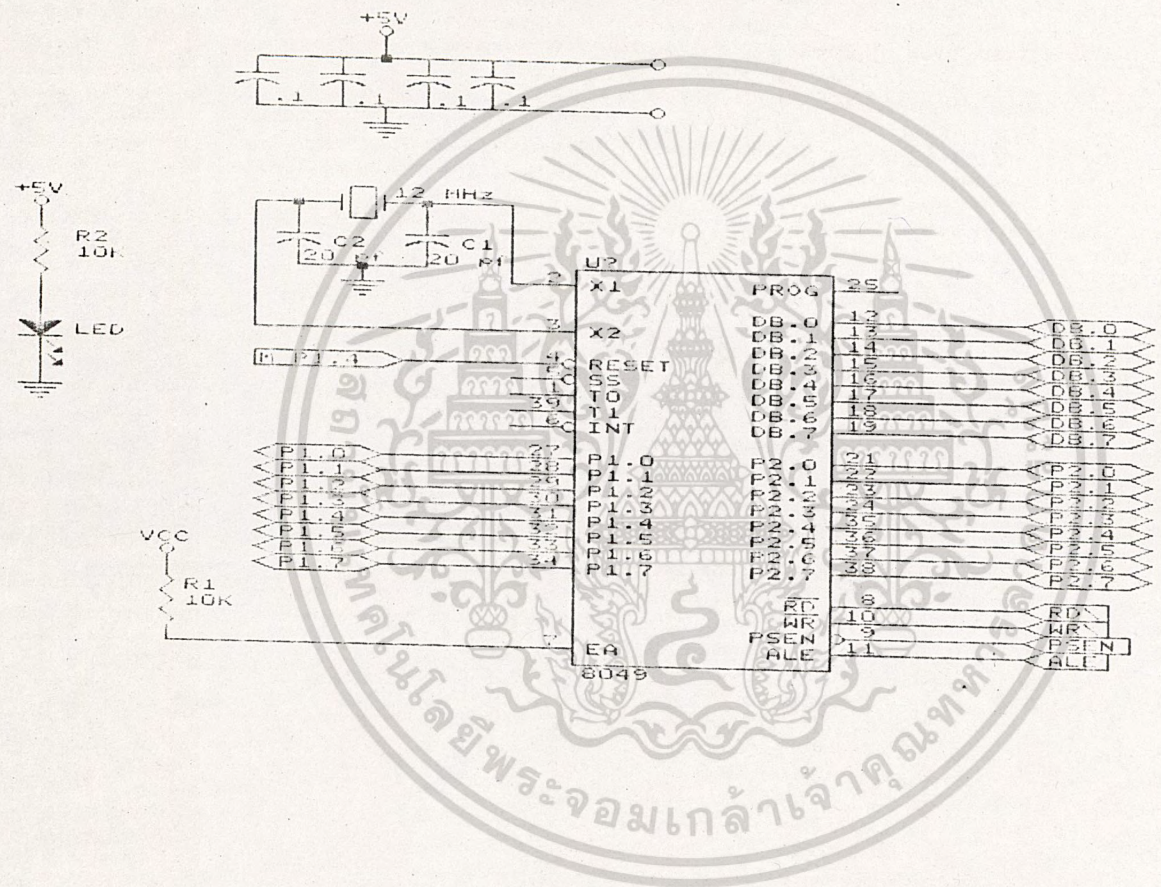
3. เมื่อเราต้องการจะ RUN โปรแกรมเราก็กดคีย์ GO ตามด้วยแอดเดรสเริ่มต้นที่ต้องการจะ RUN ถ้าเกิดต้องการแก้ไขค่าแอดเดรสที่ใส่ตามหลัง ซึ่งสามารถทำได้เหมือนข้อ 1. เมื่อแอดเดรสตรงตามต้องการให้กดคีย์ MON เพื่อเป็นการสั่งให้ RUN

4. เมื่อต้องการเริ่มต้นการทำงานใหม่เราสามารถกดคีย์ RST ได้เลยเครื่องจะเริ่มทำงานใหม่ทันที และจะแสดง MCS-48 ออกมาทาง DISPLAY

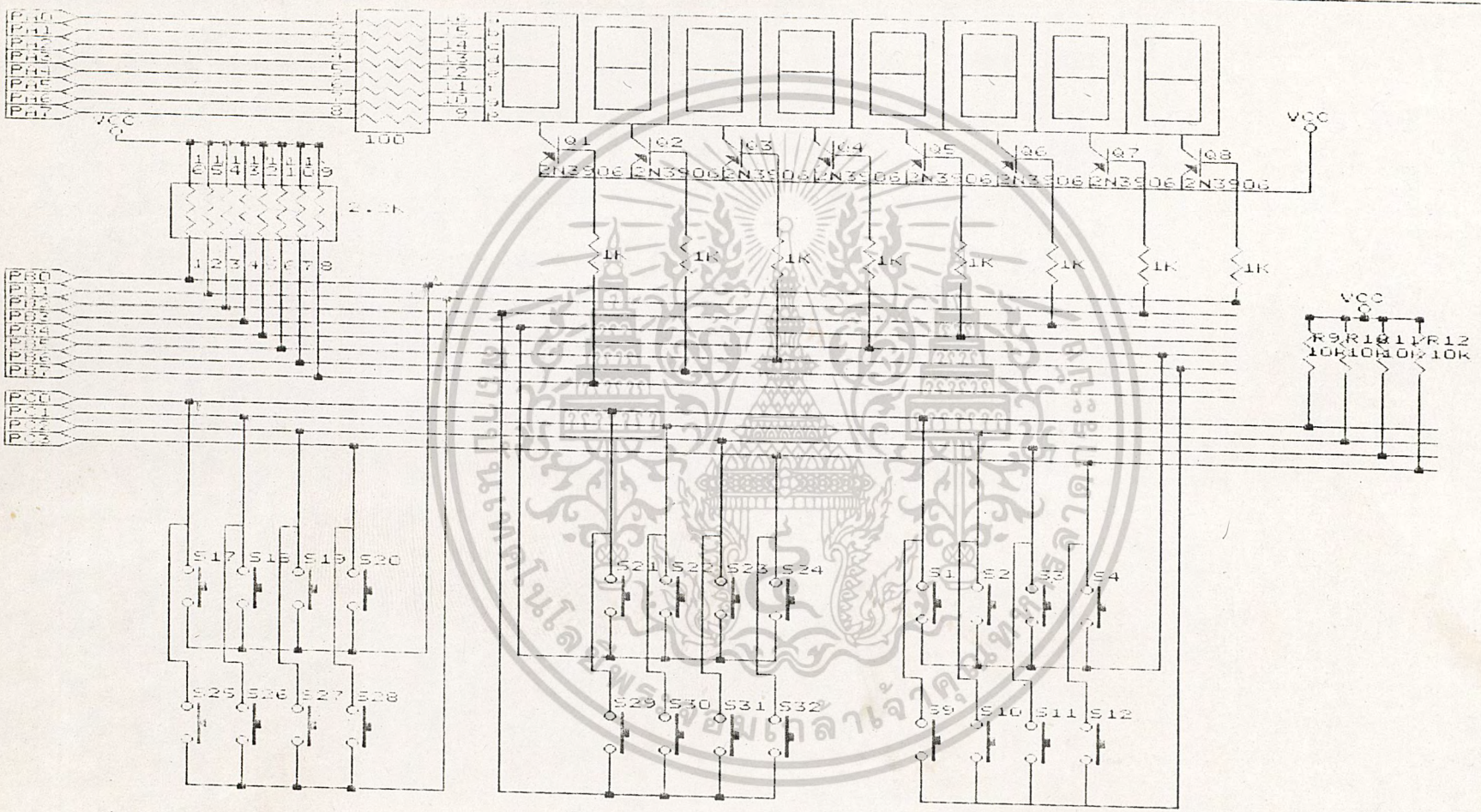

U6 DIR
74LS245
U7A
74LS04
U2
8255
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
RD
WR
A0
A1
RESET
CS
U4
74LS373
U3
2732
U1
8048
ALE
PSEN
WR
RD
EA
RESET
X1
X2
PROG
R1
10K
R2
10K
R3
10K
R4
1K
X-tal
6 MHz
C1
20PF
C2
20PF
C3
10uF
SW1
VCC
U10C
74LS00
U10B
74LS00
U10A
74LS00
U8A
74LS139
P1.0
P1.1
RAM CS
INT
T1
T0
Size
A
Document Number
MASTER PROCESSER
REV
Date: January 1, 1980
Sheet 1 of 1



Size	Document Number	RE
A	MEMORY	
Date: January 1, 1980	Sheet of	



Size A	Document Number	REV
Date: January 1, 1980	Sheet of	



Title	KEYBOARD and DISPLAY	
Size A	Document Number MCS-48	REV A
Date: January 1, 1980	Sheet 1 of 2	

บทที่ 4

ลำดับขั้นการทำงาน

1.ศึกษาหาข้อมูลและเทคนิคต่าง ๆ เกี่ยวกับการทำงาน MCS-48 SINGLEBOARD STAND ALONE

2.ทำแรม 2 ทางเพื่อเป็นอุปกรณ์ช่วยในการทดลอง เพื่อต้องการให้ CPU มองแรม 2 ทาง เหมือนกับ EROM ที่เก็บโปรแกรมมอนิเตอร์ไว้ เพื่อจะได้สามารถเปลี่ยนแปลงและแก้ไขโปรแกรมมอนิเตอร์ที่อยู่ในแรม 2 ทางได้สะดวก

3.ทำชุด DISPLAY และ KEYBOARD เพื่อที่จะนำมาทดลองการส่งข้อมูลไปยัง DISPLAY และรับข้อมูลมาจาก KEYBOARD โดยเขียนโปรแกรมมอนิเตอร์ที่อยู่ในแรม 2 ทางควบคุมผ่านทางCPU

4.นำCPU (MASTER)มาต่อโดยเอาโปรแกรมมอนิเตอร์จากแรม 2 ทางและเอา DISPLAY และ KEYBOARD เป็นตัวแสดงผลและรับค่า

5.ทำชุด SLAVE และนำ แรมขนาด 2k มาต่อเพิ่มเติม

6.เพิ่มเติมโปรแกรมมอนิเตอร์ให้สามารถอ่าน และเขียนลงบนแรมที่ต่อเพิ่มได้และสั่งให้ CPU (SLAVE) นำค่าที่อยู่ในแรมไปประมวลผล และนำค่าที่ได้จากการประมวลผล ส่งออกไปยังportเพื่อนำไปควบคุมสิ่งที่เราต้องการ

7.นำDot Matrixมาต่อที่ Port outของ CPU (slave)เพื่อแสดงว่า MCS-48 SINGLEBOARD STAND ALONE สามารถทำงานได้จริง ๆ

บทที่ 5

บทสรุป

แผงวงจร MCS-48 ทำงานด้วยตัวเอง ทำขึ้นเพื่อนำไปใช้เป็นเครื่องมือช่วยในการใช้ Microcontroller เป็นตัวควบคุมกระบวนการ ซึ่งสามารถใช้งานได้จริง จากการศึกษาการใช้งานพบว่า Function Key มีไม่กี่ Key ทำให้ไม่สะดวกในการป้อนโปรแกรมและทดลองโปรแกรมอยู่บ้าง ด้านอื่นก็เกี่ยวกับ Hardware ไม่มีความยืดหยุ่นพอ ทำให้ขีดความสามารถในการเขียน Soflware มีขีดความสามารถจำกัด นอกจากนี้ก็ยังไม่พบอะไร

ปัญหาที่พบในการทำ ส่วนใหญ่จะเป็นเรื่องของอุปกรณ์ ที่ไม่สามารถหาในท้องตลาดได้ตามต้องการ ส่วนแนวทางในการพัฒนาจะมุ่งไปทางทำให้ Hardware มีความยืดหยุ่นได้สูงและพัฒนา Soflware ควบคู่กันไป จะเพิ่ม Function Key ให้มากขึ้นเพื่อเพิ่มความสะดวกให้แก่ผู้ใช้

ภาคผนวก

คุณสมบัติของแรม 2 ทาง

คุณสมบัติของแรม 2 ทางบน IBM-PC ที่จะออกแบบมีดังนี้

- ประหยัด
- ใช้งานไม่ต้องมีสวิตช์เลือก
- ใช้งานได้ทั้ง4กิโลไบต์(2732) และ 8กิโลไบต์ (2764) หากต้องการใช้ 2 กิโลไบต์ (2716) ก็ได้โดยดัดแปลงนิดหน่อย (จะกล่าวในหัวข้อใช้งาน)แรม 2 ทาง หรือเรียกอีกอย่างหนึ่งว่า "อีพรอม อิมูเลเตอร์" ประกอบด้วยหน่วยความจำแรมขนาด 8 กิโลไบต์ IBM-PC จะมองเป็นตำแหน่ง ODE00:0000H ถึงตำแหน่ง ODE00:1FFFH (เป็นตำแหน่งขยายรอมไบออส) สามารถเขียนและอ่านได้ แต่ส่วนที่ต่อไปที่ ซ็อกเกตจะอ่านได้อย่างเดียว (เพราะคิดว่าคงไม่จำเป็น หากต้องการเขียนข้อมูลเข้ามาก็ควรจะเขียนข้อมูลเข้ามาที่ IBM-PC จะสะดวกกว่า) ในการเลือกทิศทางของแรมจะใช้สัญญาณเลือกชิพ (Chip select) จาก IBM-PC เป็นตัวเลือก โดยปกติทิศทางจะชี้ไปที่ ซ็อกเกต และหาก IBM-PC ติดต่อมา ทิศทางจะกลับมาที่ IBM-PC ทันที

การทำงานของวงจร

จากรูปที่ 2 การทำงานใช้หลักการสวิตช์สัญญาณโดยใช้ IC4-IC7เบอร์74LS157 ทำหน้าที่เลือกแอดเดรส โดยที่ขา SEL ถ้าเป็นลอจิก "0" จะนำสัญญาณชุด A ออกที่ Y เข้าที่ขาแอดเดรสของ IC3 เบอร์ 6264 ส่วนขา STBของ IC4-IC7ต่อลงกราวด์เพื่อให้มีสัญญาณออกที่ เอาต์พุต หากให้ขานี้เป็นลอจิก "1" จะทำให้ไม่มีสัญญาณออกIC1 เป็นบัฟเฟอร์สองทิศทางใช้ต่อระหว่างขาข้อมูลของ IBM-PC กับ IC3 ที่เอาต์พุตของ IC1 นี้เป็นแบบสามสถานะ ซึ่งเลือกได้ด้วยขาGโดยหากขา G เป็นลอจิก "1" ที่เอาต์พุตจะเป็นอิมพีแดนซ์สูง ดังนั้นเมื่อต้องการเลือกจะต้องส่ง "0" มาที่ขานี้ IC8เบอร์ 74LS30เป็น NANDเกตทำหน้าที่ถอดรหัสแอดเดรสจาก IBM-PC แอดเดรสที่เลือกคือ ODE00:0000H ถึงตำแหน่ง ODE00:1FFFH การถอดรหัสแอดเดรสมี

หลักการดังตารางที่ 1

จากตารางที่ 1จะเห็นว่าถ้าต้องการแอดเดรสที่ DE000H เราก็เพียงให้ A17 เป็นลอจิก "0" โดยผ่าน NOT เกตหนึ่งตัวนอกนั้นต่อโดยตรง ส่วน A12-A0 นำไปต่อกับแรม จึงไม่จำเป็นต้องถอดรหัสIC10/1 ทำหน้าที่อีนาเบิ้ลการถอดรหัสอีกทีหนึ่ง โดยจะส่ง เอาต์พุทออกมาเฉพาะเมื่อ IBM-PC ต้องการติดต่อกับหน่วยความจำเท่านั้น ไม่ว่าจะเป็นการอ่านหรือเขียนก็ตามIC10/2 ทำหน้าที่ส่งสัญญาณไปที่ขา WE ของ IC3 เมื่อ IBM-PC ต้องการเขียนข้อมูลเข้าไปในแรม คือสัญญาณMEMW และเอาต์พุทของ IC8 เป็นลอจิก "0" ทั้งสองอันIC10/3 ทำหน้าที่นำสัญญาณถอดรหัสของทั้งสองมารวมกัน โดยหากมีสัญญาณทางใดทางหนึ่ง มันจะส่งสัญญาณไปที่ขา CE1 ของ IC3 ทันที นอกจากนั้นยังส่งไปเลือกทิศทาง(DIR) ให้กับ IC1 ด้วยIC2 ทำหน้าที่เป็นบัฟเฟอร์ให้กับซ็อกเก็ตภายนอก ซึ่งมีขา G1 และ G2 เลือกใช้เมื่อมีการเลือกใช้จากซ็อกเก็ตภายนอก ข้อมูลจากแรมจะถูกส่งออกไปทันทีR1 ทำหน้าที่ ทำให้ A12 เป็นลอจิก"0" เมื่อเราใช้งาน 4 กิโลไบต์(ซ็อกเก็ต 24 ขา)R2 ทำหน้าที่ตัวต้านทานพูลอัพขาสัญญาณเลือกภายนอกไม่ให้มีสัญญาณรบกวน ขณะที่ยังไม่ใช้ส่วนชุดจ่ายไฟของวงจรทั้งหมดใช้แหล่งจ่ายของ IBM-PC

โปรแกรมประยุกต์การใช้งาน

เมื่อมีฮาร์แวร์แล้วแต่ขาดซอฟต์แวร์ก็ใช้งานยังไม่ได้ ต่อไปจะเป็นโปรแกรม

ใช้งานโดยจะอ่านข้อมูลจากแผ่นดิสค์มาเก็บที่แรม 2 ทางนี้ ลักษณะการใช้งานมีรูปแบบดังนี้

fomat : RAM(ชื่อไฟล์)

ตัวอย่าง เช่น A>RAM DATA.COM

จากรูปแบบจะเห็นว่าชื่อไฟล์ที่ต้องการจะนำข้อมูลไปเก็บไว้ในแรม 2 ทาง จะผ่านโดยส่วนท้ายของชื่อโปรแกรม โปรแกรมนี้จะตรวจสอบข้อผิดพลาดต่าง ๆ ดังนี้

- หากผู้ใช้ไม่ระบุชื่อไฟล์ จะแสดง -> fomat : RAM (ชื่อไฟล์)
- หากไม่พบไฟล์ที่ต้องการ จะแสดงคำว่า -> File not found
- หากเขียนข้อมูลลงหน่วยความจำแรม 2 ทาง แล้วมีการผิดพลาดจะแสดงคำว่า ->Card memory

สำหรับโปรแกรมการย้ายข้อมูลจากไฟล์ลงในแรม 2 ทาง ในตำแหน่ง ODE00:0000H อยู่ในหน้าซ้ายมือนี้

การนำไปใช้งาน

การใช้งานหากต้องการใช้ 2กิโลไบต์ (2716) จะต้องพิจารณาที่ซ็อกเก็ต (24ขา) ที่จะต่อว่าขา 21 ของซ็อกเก็ตนั้นต่อกับลอจิกอะไร หากเป็นลอจิก "0" ก็สามารถใช้ได้ทันที แต่หากเป็นลอจิก "1" ก็ต้องตัดและต่อเป็นลอจิก "0" หรือหากไม่ต้องการตัดก็เพียงเปลี่ยนตำแหน่งที่จะนำข้อมูลไปเก็บ จากODE00:0000H เป็น ODE00:0800H

ในการต่อสายแพระหว่างซ็อกเก็ตกับการ์ด ความยาวไม่ควรเกิน 3 ฟุตเพราะอาจทำให้ ข้อมูลผิดพลาดได้จากที่กล่าวมาแล้วคิดว่าคงจะประหยัดเวลาขึ้นในการพัฒนาโปรแกรม

CROSS-32

เป็น SOFTWARE ที่เป็น COMPILER สามารถเปลี่ยน MNEMONICS ใน SOURCE FILE บน DISK ให้เป็น machine code ใน output file แล้วเก็บลงไปบนDISK ใช้นามสกุล".PRN"ซึ่งเป็นรูปแบบของภาษา Assembly โดยมี parameter ในการแสดงความต้องการโดยมีรูปแบบดังนี้

```
C32 SOURCE FILE [-L list file] [-H hexfile]
```

-L list file เป็น parameter ที่ต้องการให้ file เป็นภาษา mnemonic ของ CPU ตระกูลนั้น ๆ

-H Hexfile เป็น parameterที่ต้องการให้ file มีการแสดงค่าเป็น Hexdecimalรวมอยู่ด้วย

Cross-32 สามารถที่จะ compileภาษา Assembly ของ CPU ตระกูลต่าง ๆ ได้หลายตระกูล ขึ้นอยู่กับการกำหนดว่าต้องการใช้ CPU ตระกูลอะไร โดยใช้รูปแบบดังตัวอย่างข้างล่าง

```
CPU     "B:8048.TBL";cpu table
```

รูปแบบของ Source file ที่ใช้ใน Cross-32

1.line แต่ละ lineจะมรูปแบบดังนี้

```
Line#   LABEL:    OPERATION     OPERAND(S) ;COMMENT
```

Line# เป็นตัวบอกว่าเป็น Line ที่เท่าไรโดยจะมีค่าอยู่ในช่วง 0 - 65535 Line# นี้ Cross-32 จะไม่สนใจในการทำการ compileแต่จะสร้างไว้ในListfile

Label ใช้อักษร(A-Z)และ "_",".","?" ในการขึ้นต้นและสิ้นสุดโดยการใส่ ":" ส่วนตัวที่ตามมาอาจจะใช้เป็นตัวเลขก็ได้

Operation ใน Cross-32 ได้มีการกำหนดoperationไว้แล้วซึ่งไว้ในตารางคำสั่ง(instruction table)คำสั่งนี้จะต้องขึ้นต้นด้วยตัวอักษร(A-Z)เท่านั้น

Operands จะเป็นlabel,ค่าคงที่,ตัวแปรของค่าคงที่ก็ได้ โดยค่าจะอยู่ในช่วง -2,147,483,648 ถึง 2,147,483,648 และ Stringe

Comment จะเริ่มจาก Semicolon ";" commentใช้สำหรับบอกให้ทราบว่าโปรแกรมเป็นอย่างไร ทำให้ง่ายต่อการทำความเข้าใจโปรแกรมที่เขียนขึ้น

```
                ;*****************************************
                ;*          MONITOR PROGRAM              *
                ;*       FOR MCS-48 STAN ALONE           *
                ;*       PROJECT 2   MARCH 1991          *
                ;*       BY NOPPHARAT TAWANRON           *
                ;*       ROOM 2-0 No 326341              *
                ;*****************************************
0000                    CPU     "8048.TBL"
0000                    HOF     "BIN8"

                ;*************************************************************
                ;
                ;       INVIDUAL PINS OF PORT 1 USED AS FOLLOW
                ;
                ;*************************************************************
                ;       P10-P13
                ;       P10 AND P11 ARE SELECT 74LS139
                ;       P10     P11     MODE
                ;         0       0     SELECT PORT 8255
                ;         0       1     SELECT EXTERNAL RAM
                ;         1       X     UNUSED
                ;
00FF =          PORT_CS: EQU    0FFH
00FE =          RAM_CS:  EQU    0FEH
                ;       P12     LO FOR ENABLE BUS FOR EP
                ;               HI FOR MP BUS (RAM)
00FB =          EPBUS:   EQU    0FBH
                ;       P14 - P17 UNUSED
```

```
                ;*********************************************************
                ;
                ;       INVIDUAL PORT 8255 AS FOLLOW
                ;
                ;*********************************************************
                ;       PORT A IS OUTPUT FOR DATA TO DISPLAY
0000 =          PORTA:      EQU     00H     ; 8255 Port A  Output
                ;       PORT B IS OUTPUT FOR SELECT DIGIT AND SCAN KEYB
0001 =          PORTB:      EQU     01H     ; 8255 Port B  Output
                ;       PORT C (LOWER) FOR INPUT MATRIX KEYBOARD
0002 =          PORTC:      EQU     02H     ; 8255 Port C  Input
                ;       THE CONTROL WORD AND PORT CONTROL
                ;       FOR THE 8255 INITIAL
0003 =          P_CON:      EQU     03H     ; 8255 Port Control
0089 =          CON_W:      EQU     89H     ; 8255 Control Word
                ;
                ;*********************************************************
                ;
                ;       DATA INTERNAL RAM ALLOCATION
                ;
                ;*********************************************************
                ;       20H - 27H FOR VIDEO RAM
0027 =          VIDEO:      EQU  27H        ;STORAGE DATA FOR DISPLAY
                ;
0029 =          KEY_NO:     EQU  29H        ;STORAGE KEYBOARD NUMBER
                ;
002A =          KB_FLAG:    EQU  2AH        ;STORAGE KEYBOARD FLAGS
                ;
```

```
002B =          CURSOR:     EQU  2BH      ;STORAGE POSITION CURSOR
                ;
002C =          FUNC_KEY:   EQU  2CH      ;STORAGE NUMBER FUNCTION KEYS
                ;
002D =          STATUS:     EQU  2DH      ;STORAGE STATUS OF FUNCTION
                ;
0030 =          DO:         EQU  30H      ;STORAGE DATA  LOW NIBBLE
                ;
0031 =          D1:         EQU  31H      ;STORAGE DATA  HIGH NIBBLE
                ;
0033 =          A1:         EQU  33H      ;STROAGE ADDRESS BITS 0-3
                ;
0034 =          A2:         EQU  34H      ;STORAGE ADDRESS BITS 4-7
                ;
0035 =          A3:         EQU  35H      ;STROAGE ADDRESS BITS 8-11
                ;
0036 =          DB:         EQU  36H      ;STORAGE DATA BUFFER 1 BYTE
                ;
0037 =          AD:         EQU  37H      ;STORAGE ADDRESS BUFFER LOWER
                ;
0038 =          AH:         EQU  38H      ;STORAGE ADDRESS BUFFER HIGHER
                ;
                ;*************************************************************
                ;
                ;       KEYBOARD FUNCTION COMPARE DATA FOLOW
                ;
                ;*************************************************************
00EE =          GO:         EQU  0EEH     ;FUNCTION FOR EP EXECUTE PROGRA
```

```
00EB =          MON:        EQU  0EBH    ;FUNCTION FOR MONOTOR DATA IN R
00E9 =          FI:         EQU  0E9H    ;
00EA =          SS:         EQU  0EAH    ;FUNCTION FOR SINGLE STEP
00ED =          RG:         EQU  0EDH    ;
00EF =          INC:        EQU  0EFH    ;FUNCTION FOR INCREMENT
00F0 =          DEC:        EQU  0F0H    ;FUNCTION FOR DECREMENT
00EC =          DTA:        EQU  0ECH    ;FUNCTION FOR CONFIRM DATA
                ;
00FF =          NULL:       EQU  0FFH    ;
                ;*************************************************************
                ;       THE KEYBOARD MAP AS FOLLOW
                ;  ____  ____  ____  ____  ____  ____
                ;  |  |  |  |  |  |  |  |  |  |  |  |
                ;  | C |  | D |  | E |  | F |  |REG |  |RES |
                ;  |___|  |___|  |___|  |___|  |____|  |____|
                ;   ____  ____  ____  ____  ____  ____
                ;  |  |  |  |  |  |  |  |  |  |  |  |
                ;  | 8 |  | 9 |  | A |  | B |  | GO |  |STEP|
                ;  |___|  |___|  |___|  |___|  |____|  |____|
                ;   ____  ____  ____  ____  ____  ____
                ;  |  |  |  |  |  |  |  |  |  |  |  |
                ;  | 4 |  | 5 |  | 6 |  | 7 |  |INC |  |MON |
                ;  |___|  |___|  |___|  |___|  |____|  |____|
                ;   ____  ____  ____  ____  ____  ____
                ;  |  |  |  |  |  |  |  |  |  |  |  |
                ;  | 0 |  | 1 |  | 2 |  | 3 |  |DEC |  |DATA|
                ;  |___|  |___|  |___|  |___|  |____|  |____|
                ;
```

```
                ;*********************************************************
                ;
                ;Page 0
                ;
0000                    ORG     0
0000 E5                 SEL     MB0     ;
0001 0410               JMP     START   ;Reset Jmp

0010            START:  ORG     10H
0010 23FF               MOV     A,#PORT_CS      ;SELECT PORT
0012 39                 OUTL    P1,A
0013 BBFF               MOV     R3,#0FFH        ;DELAY TIME
0015 EB15               DJNZ    R3,$
0017 B803               MOV     R0,#P_CON       ;INITIAL 8255 PORT
0019 2389               MOV     A,#CON_W
001B 90                 MOVX    @R0,A
                ;
                ;INITIAL DISPLAY AND KEYBOARD
                ;

001C B82D               MOV     R0,#STATUS
001E 23F7               MOV     A,#0F7H
0020 A0                 MOV     @R0,A
0021 BA7F               MOV     R2,#7FH         ;SET CONTROL DIGIT DISP
0023 BD27               MOV     R5,#VIDEO       ;Set Address Pointer
0025 BFFF               MOV     R7,#NULL        ;Set Number of Keyboard
0027 34CE               CALL    VERS
0029 5400               CALL    STR1
```

```
002B 54AC              CALL    CLEAR
002D B827              MOV     R0,#27H
002F 54D2              CALL    NORM

                       ;
0031 5400      MAIN:   CALL    STR1
0033 9237              JB4     KB_FUN          ;JMP IF IT IS KEY NUMBE
0035 047F              JMP     L2

0037 B829      KB_FUN: MOV     R0,#KEY_NO
0039 F0                MOV     A,@R0
003A 03EB              ADD     A,#MON
003C 9642              JNZ     $+6
003E 3400              CALL    MONITOR         ;CALL IF " MONITOR " KE
0040 047F              JMP     L2
                       ;
0042 F0                MOV     A,@R0
0043 03EE              ADD     A,#GO
0045 964B              JNZ     $+6
0047 B400              CALL    RUN             ;CALL if " GO " KEY
0049 047F              JMP     L2
                       ;
004B F0                MOV     A,@R0
004C 03E9              ADD     A,#F1
004E 9654              JNZ     $+6
0050 1485              CALL    X0
0052 047F              JMP     L2
                       ;
```

```
0054 F0             MOV     A,@R0
0055 03ED           ADD     A,#RG
0057 965D           JNZ     $+6
0059 1485           CALL    X0
005B 047F           JMP     L2
                    ;
005D F0             MOV     A,@R0
005E 03ED           ADD     A,#RG
0060 9666           JNZ     $+6
0062 1485           CALL    X0
0064 047F           JMP     L2
                    ;
0066 F0             MOV     A,@R0
0067 03F0           ADD     A,#DEC
0069 966F           JNZ     $+6
006B 1485           CALL    X0
006D 047F           JMP     L2
                    ;
006F F0             MOV     A,@R0
0070 03EF           ADD     A,#INC
0072 9678           JNZ     $+6
0074 1485           CALL    X0
0076 047F           JMP     L2
                    ;
0078 F0             MOV     A,@R0
0079 03EC           ADD     A,#DTA
007B 967F           JNZ     $+4
007D 1485           CALL    X0
```

```
                ;
007F 540D       L2:     CALL    STR2
0081 1485               CALL    XO
0083 0431               JMP     MAIN

                ;
                ;-----------------------------------------------------
                ;
0085 54AC       XO:     CALL    CLEAR           ;CLEAR MONITOR
0087 B827               MOV     R0,#VIDEO
0089 54D2               CALL    NORM
008B 83                 RET
                ;
                ;-----------------------------------------------------
                ;
008C 54AC       X3:     CALL    CLEAR           ;CLEAR MONITOR AND
008E 54BC               CALL    FUNC_SHOW       ;SHOW NAME OF FUNCTION
0090 83                 RET
                ;
                ;PAGE 1
                ;
0100            MONITOR:  ORG     100H
                ;       DATA IN EXTERNAL RAM IS DUM TO MONITOR
                ;       BY MP PROCESSER IN FUNCTION 'MON'
                ;
0100 54AC               CALL    CLEAR           ;CLEAR MONITOR AND
0102 54BC               CALL    FUNC_SHOW       ;SHOW THE NAME OF FUNCT
0104 540D               CALL    STR2
```

```
0106 3450              CALL    GET_A           ;GET ADDRESS IN HEX NUM
0108 37                CPL     A
0109 9636              JNZ     FOUT2
010B B822              MOV     R0,#22H
010D 54D2              CALL    NORM
                       ;
                       ;-----------------------------------------------
                       ;
010F                   CONTINUE:
010F 5462              CALL    PROG_READ       ;READ DATA FROM MEMORY
0111 540D              CALL    STR2
0113 349F              CALL    GET_D           ;GET NEW DATA FROM
0115 37                CPL     A               ;KEYBOARD
0116 9620              JNZ     BACK
0118 3437              CALL    PROG_WRITE      ;WRITE NEW DATA IN MEMO
011A 960F              JNZ     CONTINUE
011C 5421              CALL    INCREMENT       ;INCREMENT ADDRESS
011E 240F              JMP     CONTINUE
                       ;
                       ;-----------------------------------------------
                       ;
0120 B829              BACK:   MOV     R0,#KEY_NO
0122 F0                MOV     A,@R0           ;CHACK THE  'DEC' KEY
0123 03F0              ADD     A,#DEC          ;AND CALL SUBRUTINE
0125 962B              JNZ     FORW            ;DECREMENT IF YES
0127 5440              CALL    DECREMENT
0129 240F              JMP     CONTINUE
                       ;
```

```
                ;-----------------------------------------------------
                ;
012B B829       FORW:   MOV     R0,#KEY_NO
012D F0                 MOV     A,@R0           ;CHACK THE 'INC' KEY
012E 03EF               ADD     A,#INC          ;AND CALL SUBRUTINE
0130 9636               JNZ     FOUT2           ;INCREMENT IF YES
0132 5421               CALL    INCREMENT
0134 240F               JMP     CONTINUE
                ;
0136 83         FOUT2:  RET
                ;
                ;-----------------------------------------------------
                ;
0137            PROG_WRITE:
                ;       WRITE A NEW DATA FROM KEYBOARD.
                ;       (D0 & D1) TO MEMORY AT
                ;       ADDRESS FROM A1 - A3
                ;
0137 B831               MOV     R0,#D1
0139 B930               MOV     R1,#D0
013B 7458               CALL    MIX_NIBBLE
                ;
013D B836               MOV     R0,#DB          ;WRITE A NEW DATA
013F A0                 MOV     @R0,A
0140 5483               CALL    OUTADDR
0142 A9                 MOV     R1,A
0143 B836               MOV     R0,#DB
0145 F0                 MOV     A,@R0
```

```
0146 91                 MOVX    @R1,A
0147 81                 MOVX    A,@R1
                ;
0148 37                 CPL     A               ;2'COMPREMENT
0149 17                 INC     A               ;FOR COMPARE DATA
014A 60                 ADD     A,@R0
014B 964F               JNZ     $+4
014D 2300               MOV     A,#0
014F 83                 RET
                ;
                ;-----------------------------------------------------
                ;
0150            GET_A:
                ;       SUBRUTINE GET ADDRESS FROM KEYBOARD
                ;       AND KEEP TO A1 - A3 BUFFER
                ;       IF GET OK IT RETURN NULL
                ;       PROGRAM WAIT GET ADDRESS FROM
                ;       KEYBOARD AND SHOW TO DISPLAY
                ;
0150 B82B               MOV     R0,#CURSOR
0152 2325               MOV     A,#25H
0154 A0                 MOV     @R0,A
0155 5400       GET:    CALL    STR1            ;GET INPUT FROM KEYBOAR
0157 9270               JB4     FOUT            ;JMP IF IS FUNCTION KEY
0159 9414               CALL    KEEP            ;KEEP DATA AND SHIFT
015B 9459               CALL    PATT            ;CURSOR
015D 9464               CALL    CUR_DAT
015F 540D               CALL    STR2
```

```
                ;
0161 B82B               MOV     R0,#CURSOR
0163 F0                 MOV     A,@R0
0164 03DE               ADD     A,#0DEH         ;#0DEH IS 2'COMPREMENT
0166 9655               JNZ     GET
                ;
0168 5400               CALL    STR1            ;WAIT UNTIL
016A 9270               JB4     FOUT            ;KEYBOARD IS FUNCTION K
016C 540D               CALL    STR2
016E 2468               JMP     $-6
                ;
0170 B829       FOUT:   MOV     R0,#KEY_NO
0172 F0                 MOV     A,@R0           ;PROGRAM IS CHACK THE
0173 03EC               ADD     A,#DTA          ;CONFIRM KEY
0175 9682               JNZ     NO_DTA
0177 B82B               MOV     R0,#CURSOR
0179 F0                 MOV     A,@R0
017A 03DE               ADD     A,#0DEH         ;#0DEH IS 2'COMPREMENT
017C 969E               JNZ     EXIT1
                ;
017E 23FF               MOV     A,#NULL         ;SET  STATUS TO NULL
0180 249E               JMP     EXIT1
                ;
0182 B829       NO_DTA: MOV     R0,#KEY_NO
0184 F0                 MOV     A,@R0           ;CHACK THE 'DEC'KEY
0185 03F0               ADD     A,#DEC
0187 969E               JNZ     EXIT1
                ;
```

```
0189 B82B               MOV     R0,#CURSOR
018B F0                 MOV     A,@R0           ;CHACK THE POSITION CUR
018C 17                 INC     A
018D A0                 MOV     @R0,A
018E 03DA               ADD     A,#0DAH         ;#0DA IS 2'COMPREMENT O
0190 9695               JNZ     $+5
                ;
0192 2325               MOV     A,#25H
0194 A0                 MOV     @R0,A
0195 F0                 MOV     A,@R0
0196 A9                 MOV     R1,A
0197 23FF               MOV     A,#NULL
0199 A1                 MOV     @R1,A
019A 540D               CALL    STR2
019C 2455               JMP     GET
019E 83         EXIT1:  RET
                ;
                ;-----------------------------------------------
                ;
019F            GET_D:
                ;       GET DATA FROM KEYBOARD
                ;       AND KEEP TO BUFFER AND
                ;       SHOW TO DISPLAY
                ;       PROGRAM RETRUN NULL IF GET DATA OK!
                ;
019F B82B               MOV     R0,#CURSOR
01A1 2321               MOV     A,#21H
01A3 A0                 MOV     @R0,A
```

```
01A4 5400               CALL    STR1
01A6 92CD               JB4     DA_OK
                ;
01A8 9414               CALL    KEEP            ;KEEP DATA AND
01AA 54E1               CALL    CLRD            ;CLEAR OLD DATA
01AC 9459               CALL    PATT            ;CHANGE TO PATTERN
01AE 9464               CALL    CUR_DAT
01B0 540D               CALL    STR2
01B2 5400               CALL    STR1
01B4 92C0               JB4     CHANGE
                ;
01B6 9414               CALL    KEEP
01B8 9459               CALL    PATT
01BA 540D               CALL    STR2
01BC 23FF               MOV     A,#NULL
01BE 24CD               JMP     DA_OK
                ;
01C0            CHANGE:
01C0 B829               MOV     R0,#KEY_NO
01C2 F0                 MOV     A,@R0
01C3 03F0               ADD     A,#DEC
01C5 96CD               JNZ     DA_OK
01C7 54E1               CALL    CLRD
01C9 540D               CALL    STR2
01CB 249F               JMP     GET_D
01CD 83         DA_OK:  RET
                ;
```

```
01A4 5400               CALL    STR1
01A6 92CD               JB4     DA_OK
                ;
01A8 9414               CALL    KEEP            ;KEEP DATA AND
01AA 54E1               CALL    CLRD            ;CLEAR OLD DATA
01AC 9459               CALL    PATT            ;CHANGE TO PATTERRN
01AE 9464               CALL    CUR_DAT
01B0 540D               CALL    STR2
01B2 5400               CALL    STR1
01B4 92C0               JB4     CHANGE
                ;
01B6 9414               CALL    KEEP
01B8 9459               CALL    PATT
01BA 540D               CALL    STR2
01BC 23FF               MOV     A,#NULL
01BE 24CD               JMP     DA_OK
                ;
01C0            CHANGE:
01C0 B829               MOV     R0,#KEY_NO
01C2 F0                 MOV     A,@R0
01C3 03F0               ADD     A,#DEC
01C5 96CD               JNZ     DA_OK
01C7 54E1               CALL    CLRD
01C9 540D               CALL    STR2
01CB 249F               JMP     GET_D
01CD 83         DA_OK:  RET
                ;
```

```
                ;-----------------------------------------------------
                ;       SHOW TITEL " MCS - 48 "
                ;       WHEN RESET PROGRAM SHOW
                ;
01CE BB08       VERS:   MOV     R3,#8
01D0 B850               MOV     R0,#50H
01D2 B927               MOV     R1,#VIDEO
01D4 F8         LOOPX:  MOV     A,R0
01D5 E3                 MOVP3   A,@A
01D6 A1                 MOV     @R1,A           ;KEEP IN VIDEO RAM
01D7 C9                 DEC     R1
01D8 18                 INC     R0
01D9 EBD4               DJNZ    R3,LOOPX
01DB 83                 RET

                ;PAGE  2
                ;
0200                    ORG     200H
                ;STR1   IS CALL SCAN UNTIL HAVE KEYBOARD
                ;       INPUT AND FIND THE NUMBER KEY
                ;       AND KEEP LINE OF MATRIX KEYBOARD
                ;
0200 9443       STR1:   CALL    SCAN
0202 941F               CALL    SCAN1
0204 C600               JZ      $-4
0206 FA                 MOV     A,R2
0207 B82A               MOV     R0,#KB_FLAG     ;SAVE LINE OF MATRIX
```

```
0209 A0                 MOV     @R0,A           ;KEYBOARD
020A 946A               CALL    KUM_KEY
020C 83                 RET

                ;
                ;-----------------------------------------------------
                ;       IT CALL SCAN KEYBOARD AND DISPLAY
                ;       UNTIL KEYBOARD IS NULL
                ;       AND CLEAR FLAGE OF KEYBOARD
                ;
020D 9443       STR2:   CALL    SCAN
020F 941F               CALL    SCAN1
0211 23FF               MOV     A,#PORT_CS
0213 39                 OUTL    P1,A
0214 B82A               MOV     R0,#KB_FLAG
0216 F0                 MOV     A,@R0
0217 B901               MOV     R1,#PORTB
0219 91                 MOVX    @R1,A
021A B902               MOV     R1,#PORTC
021C 81                 MOVX    A,@R1
021D 37                 CPL     A
021E 960D               JNZ     STR2
0220 83                 RET
                ;
                ;-----------------------------------------------------
                ;
0221            INCREMENT:
                ;       SUBRUTINE INCREMENT
```

```
                ;
                ;
0221 B833               MOV     R0,#A1
0223 5498               CALL    INCR
0225 963D               JNZ     H1
                ;
0227 27                 CLR     A
0228 A0                 MOV     @R0,A           ;SET DATA == 0
0229 54A3               CALL    DATA_TO_PATTREN
022B B834               MOV     R0,#A2
022D 5498               CALL    INCR
022F 963D               JNZ     H1
                ;
0231 27                 CLR     A
0232 A0                 MOV     @R0,A           ;SET DATA == 0
0233 54A3               CALL    DATA_TO_PATTREN
0235 B835               MOV     R0,#A3
0237 5498               CALL    INCR
0239 963D               JNZ     H1
                ;
023B 27                 CLR     A
023C A0                 MOV     @R0,A           ;SET DATA == 0
023D 54A3       H1:     CALL    DATA_TO_PATTREN
023F 83                 RET
                ;
                ;----------------------------------------------------
                ;       SUBRUTINE DECREMENT
                ;
```

```
0240            DECREMENT:
0240 B833               MOV     R0,#A1
0242 549D               CALL    DECR
0244 965F               JNZ     H2
                ;
0246 230F               MOV     A,#0FH          ;SET DATA == F
0248 A0                 MOV     @R0,A
0249 54A3               CALL    DATA_TO_PATTREN
024B B834               MOV     R0,#A2
024D 549D               CALL    DECR
024F 965F               JNZ     H2
                ;
0251 230F               MOV     A,#0FH          ;SET DATA == F
0253 A0                 MOV     @R0,A
0254 54A3               CALL    DATA_TO_PATTREN
0256 B835               MOV     R0,#A3
0258 549D               CALL    DECR
025A 965F               JNZ     H2
                ;
025C 230F               MOV     A,#0FH
025E A0                 MOV     @R0,A           ;SET DATA == F
025F 54A3       H2:     CALL    DATA_TO_PATTREN
0261 83                 RET
                ;
                ;-----------------------------------------------------
                ;
0262            PROG_READ:
                ;       READ DATA FROM MEMORY
```

```
                        ;
0262 5483               CALL    OUTADDR
0264 A8                 MOV     RO,A
0265 80                 MOVX    A,@RO
0266 B936               MOV     R1,#DB
0268 A1                 MOV     @R1,A
0269 BBFF               MOV     R3,#0FFH
026B EB6B               DJNZ    R3,$
026D 80                 MOVX    A,@RO
                        ;
026E 37                 CPL     A                   ;2'COMPREMENT
026F 17                 INC     A
0270 61                 ADD     A,@R1
0271 9662               JNZ     PROG_READ
                        ;
0273 F1                 MOV     A,@R1
0274 B930               MOV     R1,#DO
0276 B831               MOV     RO,#D1
0278 940A               CALL    SUB_NIBBLE
027A B831               MOV     RO,#D1
027C 54A3               CALL    DATA_TO_PATTREN
027E B830               MOV     RO,#DO
0280 54A3               CALL    DATA_TO_PATTREN
0282 83                 RET
                        ;
                        ;-------------------------------------------------------
                        ;       OUTADDR  is outport address
```

```
                ; INPUT  : ADDRESS BUFFER #A1 - #A3
                ; OUTPUT : OUTPORT ADDRESS
0283            OUTADDR:
0283 23FE               MOV     A,#RAM_CS
0285 B82D               MOV     R0,#STATUS
0287 50                 ANL     A,@R0
0288 39                 OUTL    P1,A
0289 B835               MOV     R0,#A3
028B F0                 MOV     A,@R0
028C 47                 SWAP    A
028D 3A                 OUTL    P2,A
028E B834               MOV     R0,#A2
0290 B933               MOV     R1,#A1
0292 7458               CALL    MIX_NIBBLE
0294 B937               MOV     R1,#AD
0296 A1                 MOV     @R1,A
0297 83                 RET
                ;
                ;-----------------------------------------------
                ;
                ;SUBRUTINE INCR IS INCREMENT ADDRESS IN REGISTER R0
                ;INPUT  :  R0 IS POINTER FOR POINT ADDRESS
                ;OUTPUT :  A = 0 IF ADDRESS > #0FH
                ;
0298 10         INCR:   INC     @R0
0299 F0                 MOV     A,@R0           ;SUB WITH 10H
029A 03F0               ADD     A,#0F0H         ;(10h -> 2'comprement)
029C 83                 RET
```

```
                    ;
                    ;-------------------------------------------------------
                    ;
                    ;SUBRUTINE DECR IS DECREMENT ADDRESS IN REGISTER R0
                    ;INPUT  : R0 IS POINTER
                    ;OUTPUT : A = 0 IF ADDRESS < #0H
029D F0             DECR:       MOV     A,@R0
029E 07                         DEC     A
029F A0                         MOV     @R0,A           ;SUB WITH OFFH
02A0 0301                       ADD     A,#1            ;(FFh -> 2'comprement)
02A2 83                         RET
                    ;
                    ;-------------------------------------------------------
                    ;
                    ;SUBRUTINE DATA_TO_PATTREN IS CHANGE DATA TO PATTERN
                    ;FOR DISPLAY
                    ;INPUT  :  R0 = ADDRESS
                    ;OUTPUT :  PATTERN TO VIDEO RAM
                    ;
02A3                DATA_TO_PATTREN:
02A3 F0                     MOV     A,@R0
02A4 A9                     MOV     R1,A            ;PUSH A IN R1
02A5 F8                     MOV     A,R0
02A6 03F0                   ADD     A,#0F0H         ;SUB A EITH 10H
02A8 29                     XCH     A,R1            ;SWAP R1 = address A = Da
02A9 E3                     MOVP3   A,@A            ;CHANGE TO PATTERN
02AA A1                     MOV     @R1,A
02AB 83                     RET
```

```
                ;
                ;-----------------------------------------------------------
                ;
                ;       CLEAR MONITOR AND SET CURSOR TO BEGIN
                ;
02AC 23FF       CLEAR:  MOV     A,#0FFH
02AE BB08               MOV     R3,#8
02B0 B827               MOV     R0,#27H
02B2 A0         L1:     MOV     @R0,A
02B3 C8                 DEC     R0
02B4 EBB2               DJNZ    R3,L1
02B6 B82B               MOV     R0,#CURSOR
02B8 2327               MOV     A,#VIDEO
02BA A0                 MOV     @R0,A
02BB 83                 RET
                ;
                ;-----------------------------------------------------------
                ;
                ;       SHOW NAME OF FUNCTION TO DISPLAY
                ;
02BC            FUNC_SHOW:
02BC B829               MOV     R0,#KEY_NO
02BE F0                 MOV     A,@R0
02BF E3                 MOVP3   A,@A            ;CHANGE TO PATTERN
02C0 A9                 MOV     R1,A
02C1 BB02               MOV     R3,#2           ;COUNTER = 2
02C3 B827               MOV     R0,#VIDEO
02C5 F9         L3:     MOV     A,R1
```

```
02C6 E3                 MOVP3   A,@A            ;CHANGE TO PATTERN
02C7 A0                 MOV     @R0,A
02C8 19                 INC     R1
02C9 C8                 DEC     R0
02CA EBC5               DJNZ    R3,L3
02CC B82B               MOV     R0,#CURSOR      ;SET CURSOR
02CE 2325               MOV     A,#25H
02D0 A0                 MOV     @R0,A
02D1 83                 RET
                ;
                ;-----------------------------------------------
                ;
02D2 23BF       NORM:   MOV     A,#0BFH         ;SET CURSOR TO ' - '
02D4 A0                 MOV     @R0,A
02D5 83                 RET
                ;
                ;-----------------------------------------------
                ;
                ;       CLEAR THE OLD ADDRESS
                ;
02D6 BB03       CLRA:   MOV     R3,#3
02D8 23FF               MOV     A,#0FFH
02DA B825               MOV     R0,#25H
02DC A0                 MOV     @R0,A
02DD C8                 DEC     R0
02DE EBDC               DJNZ    R3,$-2
02E0 83                 RET
                ;
```

```
                ;-------------------------------------------------------
                ;
                ;       CLEAR THE OLD DATA
                ;
02E1 B821       CLRD:   MOV     R0,#21H
02E3 23FF               MOV     A,#0FFH
02E5 A0                 MOV     @R0,A
02E6 C8                 DEC     R0
02E7 A0                 MOV     @R0,A
02E8 83                 RET
                ;*******************************************************
                ;
                ;PAGE  3
                ;
                ;*******************************************************
0300                    ORG     300H
0300 C0                 DFB     0C0H  ;SEGMENT PATTERN FOR DIGIT '0'
0301 F9                 DFB     0F9H  ;SEGMENT PATTERN FOR DIGIT '1'
0302 A4                 DFB     0A4H  ;SEGMENT PATTERN FOR DIGIT '2'
0303 B0                 DFB     0B0H  ;SEGMENT PATTERN FOR DIGIT '3'
0304 99                 DFB     099H  ;SEGMENT PATTERN FOR DIGIT '4'
0305 92                 DFB     092H  ;SEGMENT PATTERN FOR DIGIT '5'
0306 82                 DFB     082H  ;SEGMENT PATTERN FOR DIGIT '6'
0307 F8                 DFB     0F8H  ;SEGMENT PATTERN FOR DIGIT '7'
0308 80                 DFB     080H  ;SEGMENT PATTERN FOR DIGIT '8'
0309 90                 DFB     090H  ;SEGMENT PATTERN FOR DIGIT '9'
030A 88                 DFB     088H  ;SEGMENT PATTERN FOR DIGIT 'A'
```

```
030B 83                 DFB     083H  ;SEGMENT PATTERN FOR DIGIT 'B'
030C C6                 DFB     0C6H  ;SEGMENT PATTERN FOR DIGIT 'C'
030D A1                 DFB     0A1H  ;SEGMENT PATTERN FOR DIGIT 'D'
030E 86                 DFB     086H  ;SEGMENT PATTERN FOR DIGIT 'E'
030F 8E                 DFB     08EH  ;SEGMENT PATTERN FOR DIGIT 'F'
                ;
0310 28                 DFB     028H
0311 28                 DFB     028H
0312 20                 DFB     020H
0313 24                 DFB     024H
0314 20                 DFB     020H
0315 22                 DFB     022H
0316 26                 DFB     026H
                ;
0320                    ORG     320H
0320 C2                 DFB     0C2H    ; 'G ' Charecter
0321 23                 DFB     023H    ; 'o.' Charecter

0322 8C                 DFB     08CH    ; 'P ' Charecter
0323 21                 DFB     021H    ; 'd.' Charecter

0324 AF                 DFB     0AFH    ; 'R ' Charecter
0325 C2                 DFB     0C2H    ; 'G ' Charecter

0326 92                 DFB     092H    ; 'S ' Charecter
0327 12                 DFB     012H    ; 'S.' Charecter
```



```
0328 FF                 DFB     NULL    ;
0329 FF                 DFB     NULL    ;
```

```
0340                    ORG     340H
0340 04                 DFB     04H     ;JMP PAGE 0
0341 24                 DFB     24H     ;JMP PAGE 1
0342 44                 DFB     44H     ;JMP PAGE 2
0343 64                 DFB     64H     ;JMP PAGE 3
0344 84                 DFB     84H     ;JMP PAGE 4
0345 A4                 DFB     0A4H    ;JMP PAGE 5
0346 C4                 DFB     0C4H    ;JMP PAGE 6
0347 E4                 DFB     0E4H    ;JMP PAGE 7

0350                    ORG     350H
0350 FF                 DFB     NULL    ;
0351 C8                 DFB     0C8H    ; ' M ' CHARECTER
0352 C6                 DFB     0C6H    ; ' C '     "
0353 92                 DFB     92H     ; ' S '     "
0354 BF                 DFB     0BFH    ; ' - '     "
0355 99                 DFB     99H     ; ' 4 '     "
0356 80                 DFB     80H     ; ' 8 '     "
0357 FF                 DFB     NULL    ;
                ;
                ;PAGE 4
                ;
                ;-------------------------------------------------------
                ;MIX_NIBBLE
                ;INPUT  : R0 IS HIGH BYTE
                ;       : R1 IS LOW BYTE
                ;OUTPUT : REGISTER A
                ;
```

```
0358                MIX_NIBBLE:
0400                        ORG     400H
0400 F0                     MOV     A,@R0
0401 47                     SWAP    A
0402 53F0                   ANL     A,#0F0H
0404 A8                     MOV     R0,A
0405 F1                     MOV     A,@R1
0406 530F                   ANL     A,#0FH
0408 48                     ORL     A,R0
0409 83                     RET
                    ;
                    ;-----------------------------------------------
                    ;
                    ;SUB_NIBBLE  CHANGE DATA 1 BYTE TO 2 BYTE
                    ;INPUT  : REGISTER A
                    ;OUTPUT : HIGH NIBBLE BITS TO @R0
                    ;       : LOW NIBBLE BITS TO @R1

040A                SUB_NIBBLE:
040A AB                     MOV     R3,A
040B 530F                   ANL     A,#0FH
040D A1                     MOV     @R1,A
040E FB                     MOV     A,R3
040F 53F0                   ANL     A,#0F0H
0411 47                     SWAP    A
0412 A0                     MOV     @R0,A
0413 83                     RET
                    ;
```

```
                ;-----------------------------------------------------
                ;
                ;       KEEP
                ;
0414 B82B       KEEP:   MOV     R0,#CURSOR
0416 2310               MOV     A,#10H
0418 60                 ADD     A,@R0
0419 A8                 MOV     R0,A
041A B929               MOV     R1,#KEY_NO
041C F1                 MOV     A,@R1
041D A0                 MOV     @R0,A
041E 83                 RET
                ;
                ;-----------------------------------------------------
                ;SCAN KEYBOARD AND DISPLAY
                ;DATA SEND TO PORT A OF 8255
                ;CONTROL DIGIT SEND TO PORT B OF 8255
                ;INPUT FOR MATRIX KEYBOARD TO PORT C OF 8255
                ;
041F            SCAN1:
041F 23FF               MOV     A,#PORT_CS
0421 B82D               MOV     R0,#STATUS
0423 50                 ANL     A,@R0
0424 39                 OUTL    P1,A
0425 B801               MOV     R0,#PORTB
0427 FA                 MOV     A,R2
0428 90                 MOVX    @R0,A
                ;
```

```
0429 FD                 MOV     A,R5
042A A8                 MOV     R0,A
042B F0                 MOV     A,@R0
042C B800               MOV     R0,#PORTA
042E 90                 MOVX    @R0,A
                ;
042F B802               MOV     R1,#PORTC
0431 81                 MOVX    A,@R1
0432 37                 CPL     A
0433 C63E               JZ      PASS
0435 BBFF               MOV     R3,#0FFH
0437 EB37               DJNZ    R3,$
                ;
0439 81                 MOVX    A,@R1
043A 37                 CPL     A
043B AE                 MOV     R6,A
043C 8442               JMP     $+6
043E BB80       PASS:   MOV     R3,#080H
0440 EB40               DJNZ    R3,$
0442 83                 RET
                ;
                ;-----------------------------------------------
                ;
                ;       SCAN DISPLAY AND KEYBOADR
                ;
0443 FA         SCAN:   MOV     A,R2
0444 77                 RR      A
0445 AA                 MOV     R2,A
```

```
0446 FF                 MOV     A,R7
0447 0304               ADD     A,#04
0449 AF                 MOV     R7,A
044A CD                 DEC     R5
044B FD                 MOV     A,R5
044C 37                 CPL     A               ;2'COMPREMENT
044D 17                 INC     A
044E 031F               ADD     A,#1FH
0450 9658               JNZ     EXIT
0452 BA7F               MOV     R2,#7FH
0454 BD27               MOV     R5,#VIDEO
0456 BFFF               MOV     R7,#NULL
0458 83         EXIT:   RET
                ;
                ;-----------------------------------------------------------
                ;
                ;CHANG THE NUMBER OF KEYBOARD TO PATTERN
                ;
0459 B829       PATT:   MOV     R0,#KEY_NO
045B F0                 MOV     A,@R0
045C E3                 MOVP3   A,@A
045D 29                 XCH     A,R1
045E B82B               MOV     R0,#CURSOR
0460 F0                 MOV     A,@R0
0461 29                 XCH     A,R1
0462 A1                 MOV     @R1,A
0463 83                 RET
                ;
```

```
                ;-----------------------------------------------------
                ;
                ;       SHIFT THE CURSUR ONE DIGIT
                ;
0464            CUR_DAT:
0464 B82B               MOV     R0,#CURSOR
0466 F0                 MOV     A,@R0
0467 07                 DEC     A
0468 A0                 MOV     @R0,A
0469 83                 RET
                ;
                ;-----------------------------------------------------
                ;
                ;FINE THE NUMBER OF KEYBOARD
                ;INPUT  : NULL
                ;OUTPUT : A = KEYBOARD NUMBER
                ;
046A            NUM_KEY:
046A FF                 MOV     A,R7
046B A8                 MOV     R0,A
046C FE                 MOV     A,R6
046D 97                 CLR     C
046E 18                 INC     R0
046F 67                 RRC     A
0470 E66E               JNC     $-2
0472 F8                 MOV     A,R0
0473 B829               MOV     R0,#KEY_NO
0475 A0                 MOV     @R0,A
```

```
0476 83                 RET
                        ;
                        ;-----------------------------------------------
                        ;
                        ;PAGE 5
                        ;
0500                    ORG     500H
0500                RUN:
0500 54AC               CALL    CLEAR
0502 54BC               CALL    FUNC_SHOW
0504 540D               CALL    STR2
0506 3450               CALL    GET_A
0508 37                 CPL     A
0509 964E               JNZ     OUT_RUN
                        ;
050B 23FF               MOV     A,#PORT_CS
050D B82D               MOV     RO,#STATUS
050F 50                 ANL     A,@RO
0510 39                 OUTL    P1,A
0511 23FF               MOV     A,#NULL         ;OFF MONITOR
0513 B801               MOV     RO,#01
0515 90                 MOVX    @RO,A
                        ;
0516 23FE               MOV     A,#RAM_CS
0518 B82D               MOV     RO,#STATUS
051A 50                 ANL     A,@RO
051B 39                 OUTL    P1,A
051C 27                 CLR     A
```

```
051D 3A                 OUTL    P2,A

                ;

051E B834               MOV     R0,#A2
0520 B933               MOV     R1,#A1
0522 7458               CALL    MIX_NIBBLE
0524 B802               MOV     R0,#2H          ;ADDRESS 02
0526 90                 MOVX    @R0,A           ;LOCLTION FOR ADDRESS J
0527 B835               MOV     R0,#A3
0529 F0                 MOV     A,@R0
052A 530F               ANL     A,#0FH
052C 7235               JB3     BANK1

                ;

052E 23E5               MOV     A,#0E5H         ;OPCODE FOR SEL MB0
0530 B800               MOV     R0,#0           ;ADDRESS 00
0532 90                 MOVX    @R0,A
0533 A43D               JMP     PROC

                ;

0535 5307       BANK1:  ANL     A,#07H
0537 A0                 MOV     @R0,A
0538 23F5               MOV     A,#0F5H         ;OPCODE FOR SEL  MB1
053A B800               MOV     R0,#0           ;ADDRESS 00
053C 90                 MOVX    @R0,A

                ;

053D B835       PROC:   MOV     R0,#A3
053F F0                 MOV     A,@R0
0540 0340               ADD     A,#40H
0542 E3                 MOVP3   A,@A
0543 B801               MOV     R0,#1H          ;ADDRESS 01
```

```
0545 90                 MOVX    @R0,A           ;OPCODE FOR JMP PAGE
                        ;
0546 B82D               MOV     R0,#STATUS
0548 23FB               MOV     A,#EPBUS        ;ENABLE EP BUS
054A A0                 MOV     @R0,A
054B 39                 OUTL    P1,A
054C A44C               JMP     $
054E 83         OUT_RUN:RET

0000                    END
0033  A1                0034  A2                0035  A3
0037  AD                0038  AH                0120  BACK
0535  BANK1             01C0  CHANGE            02AC  CLEAR
02D6  CLRA              02E1  CLRD              010F  CONTINUE
0089  CON_W             002B  CURSOR            0464  CUR_DAT
0030  D0                0031  D1                02A3  DATA_TO_PATTREN
01CD  DA_OK             0036  DB                00F0  DEC
029D  DECR              0240  DECREMENT         00EC  DTA
00FB  EPBUS             0458  EXIT              019E  EXIT1
00E9  F1                012B  FORW              0170  FOUT
0136  FOUT2             002C  FUNC_KEY          02BC  FUNC_SHOW
0155  GET               0150  GET_A             019F  GET_D
00EE  G0                023D  H1                025F  H2
00EF  INC               0298  INCR              0221  INCREMENT
002A  KE_FLAG           0037  KB_FUN            0414  KEEP
0029  KEY_NO            02B2  L1                007F  L2
02C5  L3                01D4  LOOPX             0031  MAIN
0358  MIX_NIBBLE        00EB  MON               0100  MONITOR
```

02D2 NORM	0182 NO_DTA	00FF NULL
046A NUM_KEY	0283 OUTADDR	054E OUT_RUN
043E PASS	0459 PATT	0000 PORTA
0001 PORTB	0002 PORTC	00FF PORT_CS
053D PROC	0262 PROG_READ	0137 PROG_WRITE
0003 P_CON	00FE RAM_CS	00ED RG
0500 RUN	0443 SCAN	041F SCAN1
00EA SS	0010 START	002D STATUS
0200 STR1	020D STR2	040A SUB_NIBBLE
01CE VERS	0027 VIDEO	0085 X0
008C X3		

54139/74139 Dual 2-Line-to-4-Line Decoder/Demultiplexer

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.	SN54S139	J①			W①						SN54LS139	J①			W①										
	SN74S139	J①	N①								SN74LS139	J①	N①												
FAIRCHILD	/FM93S139	D①									FM54LS139/FM93LS139	D①			F①										
	FC74S139/FC93S139	D①									FC74LS139/FC93LS139	D①	P①		F①										
MOTOROLA																									
N.S.C.	DM74S139										SN74LS139		P①												
											DM54LS139		①												
PHILIPS	N74S139		D								DM74LS139		①												
SIGNETICS	S54S139	F①	B①		W①																				
	N74S139	F①	B①		W①						N74LS139		①												
SIEMENS											N74LS139		A①												
FUJITSU																									
HITACHI											74LS139		M①												
MITSUBISHI											HD74LS139		P①												
NEC											M74LS139		P①												
AMD											74LS139		C①												

Electrical Characteristics SN54LS139/SN74LS139

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7 V	Operating free-air temperature range	SN54LS	−55°C to 125°C
Input voltage	7 V		SN74LS	0°C to 70°C
		Storage temperature range		−65°C to 150°C

recommended opering conditions

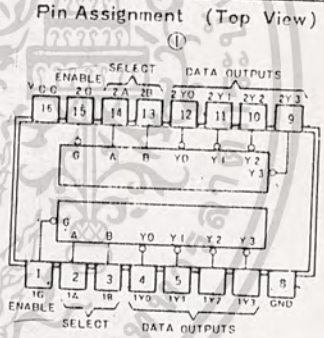
	SN54LS139			SN74LS139			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER*		TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN, I_I = −18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OH} = −1mA	SN54LS	2.5	3.4		V
			SN74LS	2.7	3.4		
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OL} = 8mA			0.35	0.5	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 7 V				0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7 V				20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.5 V				−0.4	mA
I_{OS}	Short circuit output current*	V_{CC} = MAX		−20		−100	mA
I_{CC}	Supply current	V_{CC} = MAX, Outputs enabled and open			6.8	11	mA
t_{PLH}	from Binary select to Any output	V_{CC} = 5 V, T_A = 25°C, C_L = 15pF, R_L = 2kΩ	levels of delay: 2		13	20	ns
t_{PHL}					22	33	
t_{PLH}			3		18	29	ns
t_{PHL}					25	38	
t_{PLH}	from Enable to Any output		2		16	24	ns
t_{PHL}					21	32	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
* Not more than one output should be shorted at a time.
* t_{PLH} = propagation delay time, low-to-high-level output; t_{PHL} = propagation delay time, high-to-low-level output.

Pin Assignment (Top View)



positive logic: see function table

Function Table

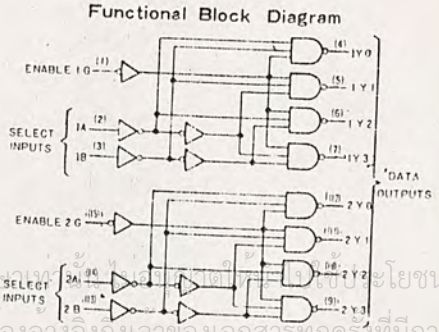
'S139 'LS139

(EACH DECODER/DEMULTIPLEXER)

INPUTS			OUTPUTS			
ENABLE G	SELECT B	SELECT A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = high level, L = low level, X = irrelevant

Functional Block Diagram



'S139 'LS139 DECODER/DEMUTIPLEXER

54157 / 74157 Quadruple 2-Line-to-1-Line Data Selector / Multiplexer

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.	SN54S157	J①			W①						SN54LS157	J①			W①	SN54157	J①			W①	SN54L157	J①			
	SN74S157	J①	N①								SN74LS157	J①	N①			SN74157	J①	N①			SN74L157	J①	N①		
FAIRCHILD		B①									FM54LS157/FM74LS157	B①			F①		B①								
	FC74S157	B①	P①								FC74LS157/FC74LS157	B①	P①		F①	FC74157	B①	P①							
MOTOROLA																									
											MC74LS157		P①			MC74157		P①							
N.S.C.	DM74S157		N①								DM54LS157		①								DM54L157A				
											DM74LS157		①			DM74157		N①			DM74L157A				
PHILIPS																									
	N74S157		①								N74LS157		①			N74157									
SIGNETICS	S54S157															S54157	F①	B①		W①					
	N74S157										N74LS157		A⑦			N74157	F①	B①							
SIEMENS																									
																FLY171		①							
FUJITSU																									
											74LS157		M①												
HITACHI																									
	HD74S157		P①								HD74LS157		P①			HD74157	①	P①							
MITSUBISHI																									
	M5S157		P①								M74LS157		P①			M53357		P①							
NEC																									
											74LS157		C①			74157		C①							
AMD	AM54S157		①								AM54LS157		①												
	AM74S157		①								AM74LS157		①												

Electrical Characteristics SN54LS157/SN74LS157

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55℃ to 125℃
Input voltage	7V		SN74LS	0℃ to 70℃
		Storage temperature range		−65℃ to 150℃

recommended operating conditions

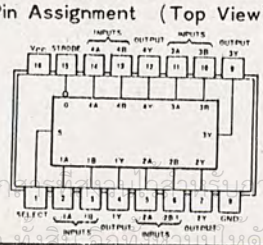
	SN54LS157			SN74LS157			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	℃

electrical characteristics over recommended operating free-air temperature range

PARAMETER*			TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage				2			V
V_{IL}	Low-level input voltage						0.8	V
V_I	Input clamp voltage		V_{CC}=MIN, I_I=−18mA				−1.5	V
V_{OH}	High-level output voltage		V_{CC}=MIN, V_{IH}=2V, V_{IL}=0.8V, I_{OH}=−400μA		2.7	3.4		V
V_{OL}	Low-level output voltage		V_{CC}=MIN, V_{IH}=2V, V_{IL}=0.8V, I_{OL}=8mA			0.35	0.5	V
I_I	Input current at maximum input voltage	S or G input	V_{CC}=MAX, V_I=7V				0.2	mA
		A or B input					0.1	
I_{IH}	High-level input current	S or G input	V_{CC}=MAX, V_I=2.7V				40	μA
		A or B input					20	
I_{IL}	Low-level input current	S or G input	V_{CC}=MAX, V_I=0.4V				−0.8	mA
		A or B input					−0.4	
I_{OS}	Short-circuit output current*		V_{CC}=MAX	SN54LS	−20		−100	mA
				SN74LS	−20		−100	
I_{CC}	Supply current		V_{CC}=MAX, See Note			9.7	16	mA
t_{PLH}	from data input		V_{CC}=5V, T_A=25℃, C_L=15pF, R_L=2kΩ			8	14	ns
t_{PHL}						9	14	
t_{PLH}	from Strobe					14	20	ns
t_{PHL}						14	21	
t_{PLH}	from Select					18	23	ns
t_{PHL}						18	27	

Pin Assignment (Top View)

①



positive logic:
Low logic level at S selects A inputs
High logic level at S selects B inputs

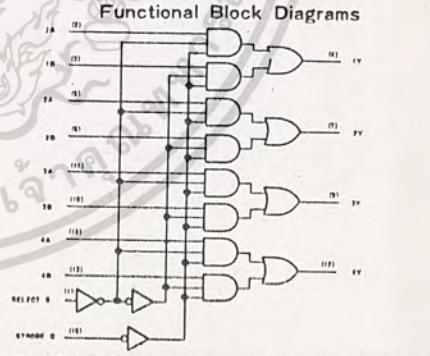
Function Table

'157, 'S157, 'L157, 'LS157

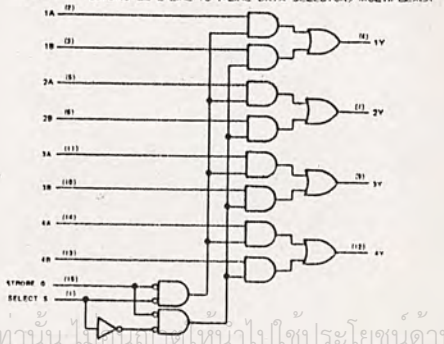
INPUTS				OUTPUT
STROBE	SELECT	A	B	Y
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

H = high level, L = low level, X = irrelevant

Functional Block Diagrams



'157, 'L157 QUADRUPLE 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER



'S157, 'LS157 QUADRUPLE 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER

NOTE: I_{CC} is measured with 4.5V applied to all inputs and all outputs open.

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
‡ All typical values are at V_{CC}=5V, T_A=25℃.
* Not more than one output should be shorted at a time.
* t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output

54244/74244 Octal Buffers/Line Drivers/Line Receivers

	Schttky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF
T. I.											SN54LS244	J ①													
FAIRCHILD											SN74LS244	J ①	N ①												
MOTOROLA																									
N. S. C.																									
PHILIPS																									
SIGNETICS																									
SIEMENS																									
FUJITSU																									
HITACHI																									
MITSUBISHI																									
NEC																									
TOSHIBA																									

Electrical Characteristics SN54LS244/SN74LS244

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	55°C to 125°C
Input voltage	5.5V		SN74LS	0°C to 70°C
Interemitter voltage	5.5V	Storage temperature range		65°C to 150°C

recommended operating conditions

	SN54LS244 MIN	NOM	MAX	SN74LS244 MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			12			15	mA
Low-level output current, I_{OL}			12			24	mA
Operating free-air temperature, T_A	55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS †		SN74LS MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −18mA				−1.5	V
	Hysteresis($V_{T+} - V_{T-}$)	V_{CC} = MIN		0.2	0.4		V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = $V_{IL\,max}$, I_{OH} = −3mA		2.4	3.4		V
		V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.5V, I_{OH} = MAX		2			
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = $V_{IL\,max}$	I_{OL} = 12mA			0.4	V
			I_{OL} = 24mA			0.5	
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC} = MAX, V_{IH} = 2V, V_{IL} = $V_{IL\,max}$	V_O = 2.7V			20	μA
I_{OZL}	Off-state output current, low-level voltage applied		V_O = 0.4V			−20	
I_I	Input current at maximum input voltage	V_{CC} = MAX,	V_I = 7V			0.1	mA
I_{IH}	High-level input current, any input	V_{CC} = MAX,	V_I = 2.7V			20	μA
I_{IL}	Low-level input current	V_{CC} = MAX,	V_{IL} = 0.4V			−0.2	mA
I_{OS}	Short-circuit output current ♦	V_{CC} = MAX		−40		−225	mA
I_{CC}	Supply current, Outputs high	V_{CC} = MAX, Outputs open	All		13	23	mA
	Supply current, Outputs low		'LS244		27	46	
	Supply current, All outputs disabled		'LS244		32	54	

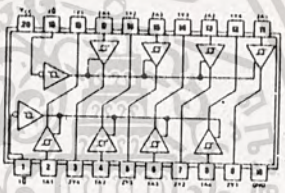
switching characteristics, V_{CC} 5V, T_A 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output	C_L = 45pF, R_L = 667Ω, See Note 2		9	14	ns
t_{PHL}	Propagation delay time, high-to-low-level output			12	18	ns
t_{PZL}	Output enable time to low level			20	30	ns
t_{PZH}	Output enable time to high level			15	23	ns
t_{PLZ}	Output disable time from low level	C_L = 5pF, R_L = 667Ω, See Note 2		15	25	ns
t_{PHZ}	Output disable time from high level			10	18	ns

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} 5V, T_A 25°C
♦ Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.
NOTE 2 : Load circuit and voltage wave forms are shown on page 3-11.

Pin Assignment (Top View)

①



SN54LS244 (J SN74LS244 (J, N)

54245/74245 Octal Bus Transceivers with 3-state Outputs

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.											SN54LS245	J			W										
											SN74LS245	J		N											
FAIRCHILD																									
MOTOROLA																									
N.S.C																									
PHILIPS																									
SIGNETICS																									
SIEMENS																									
FUJITSU																									
HITACHI																									
MITSUBISHI																									
NEC																									
TOSHIBA																									

Electrical Characteristics SN54LS245/SN74LS245

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7 V	Operating free-air temperature range	SN54LS	55°C to 125°C
Input voltage	7 V		SN74LS	0°C to 70°C
		Storage temperature range		65°C to 150°C

recommended operating conditions

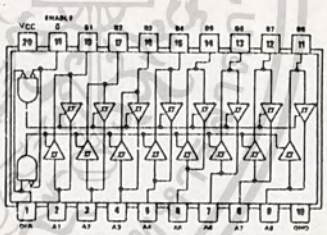
	SN54LS245			SN74LS245			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−12			−15	mA
Low-level output current, I_{OL}			12			24	mA
Operating free-air temperature, T_A	55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER		TEST CONDITIONS†		SN74LS245 MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN,	I_I = −18mA			−1.5	V
	Hysteresis ($V_{T+} - V_{T-}$) A or B input	V_{CC} = MIN		0.2	0.4		V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = V_{IL} max	I_{OH} = −3mA	2.4	3.4		V
			I_{OH} = MAX	2			
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = V_{IL} max	I_{OL} = 12mA			0.4	V
			I_{OL} = 24mA			0.5	
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC} = MAX, G at 2V	V_O = 2.7 V			10	μA
I_{OZL}	Off-state output current, low-level voltage applied		V_O = 0.4V			−200	
I_I	Input current at maximum input voltage: A or B	V_{CC} = MAX,	V_I = 5.5V			0.1	mA
	DIR or $\overline{G}$		V_I = 7V			0.1	
I_{IH}	High-level input current	V_{CC} = MAX, V_{IH} = 2.7V				20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_{IL} = 0.4V				−0.2	mA
I_{OS}	Short-circuit output current	V_{CC} = MAX		−40		−225	mA
I_{CC}	Supply current: Total, outputs high	V_{CC} = MAX, Outputs open			48	70	mA
	Total, outputs low				62	90	
	Outputs at Hi-Z				64	95	

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

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output	C_L = 45pF, R_L = 667Ω, See Note 2		8	12	ns
t_{PHL}	Propagation delay time, high-to-low-level output			8	12	ns
t_{PZL}	Output enable time to low level			27	40	ns
t_{PZL}	Output enable time to high level			25	40	ns
t_{PLZ}	Output disable time from low level	C_L = 5pF, R_L = 667Ω, See Note 2		15	25	ns
t_{PHZ}	Output disable time from high level			15	25	ns



SN54LS245 (J) SN74LS245 (J, N)

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5V, T_A = 25°C.
* Not more than one output should be shorted at a time, and duration of the short-short should not not exceed one second.

54373/74373 Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.	SN54S373	J①									SN54LS373	J①													
	SN74S373	J①N①									SN74LS373	J①N①													
FAIRCHILD																									
MOTOROLA																									
N.S.C.																									
PHILIPS																									
SIGNETICS																									
SIEMENS																									
FUJITSU																									
HITACHI																									
MITSUBISHI																									
NEC																									
TOSHIBA																									

Electrical Characteristics, SN54LS373/SN74LS373

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to 125°C	
Input voltage	7V		SN74LS	0°C to 70°C	
		Storage temperature range		−65°C to 150°C	

recommend operating conditions

		SN54LS373			SN74LS373			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				−1			−2.6	mA
High-level output voltage, V_{OH}				5.5			5.5	V
Pulse width, t_w	Clock enable high	15			15			ns
	Clock enable high	15			15			
Setup time, t_{SU} ††		0↓			0↓			ns
Hold time, t_{hold}		10↓			10↓			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

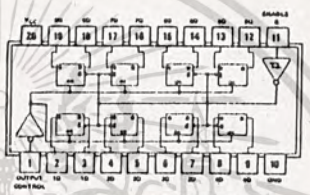
PARAMETER		TEST CONDITIONS‡		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = $V_{IL\,max}$, I_{OH} = MAX		2.4	3.1		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = $V_{IL\,max}$	I_{OL} = 24mA		0.35	0.5	V
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC} = MAX, V_{IH} = 2V, V_O = 2.7V				20	μA
I_{OZL}	Off-state output current, low-level voltage applied	V_{CC} = MAX, V_{IH} = 2V, V_O = 0.4V				−20	μA
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 7V				0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7V				20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4V				−0.4	mA
I_{OS}	Short-circuit output current §	V_{CC} = MAX		−30		−130	mA
I_{CC}	Supply current	V_{CC} = MAX, Output control at 4.5V	'LS373		24	40	mA

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

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_{max}			C_L = 45pF, R_L = 667Ω, See Notes 2 and 3				MHz
t_{PLH}	Data	Any Q			12	18	ns
t_{PHL}					12	18	
t_{PLH}	Clock or enable	Any Q			20	30	ns
t_{PHL}					18	30	
t_{PZH}	Output Control	Any Q			15	28	ns
t_{PZL}					25	36	
t_{PHZ}	Output Control	Any Q	C_L = 5pF, R_L = 667Ω, See Note 3		12	20	ns
t_{PLZ}					15	25	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5V, T_A = 25°C.
§ Not more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Pin Assignments (Top View)



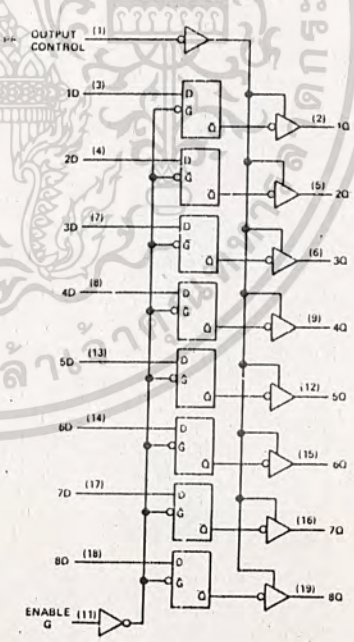
SN54LS373 (J) SN74LS373 (J, N)
SN54S373 (J) SN74S373 (J, N)

'LS373, 'S373 FUNCTION TABLE

OUTPUT CONTROL	ENABLE G	D	OUTPUT
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

'LS373, 'S373 TRANSPARENT LATCHES

'LS373, 'S373 TRANSPARENT LATCHES



NOTES: 2. Maximum clock frequency is tested with all outputs loaded.
3. See load circuits and waveforms on page 3-11.

f_{max} = maximum clock frequency
t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output
t_{PZH} = output enable time to high level
t_{PZL} = output enable time to low level
t_{PHZ} = output disable time from high level
t_{PLZ} = output disable time from low level

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลงได้ก็ด้วยความร่วมมือ และความช่วยเหลือจากหลายฝ่ายด้วยกันโดยเฉพาะ รศ. พิมพันธ์เลาหสงคราม ผู้ซึ่งเป็นอาจารย์ที่ปรึกษา อ.วิทยา ทิพย์สุวรรณพร และ คุณเรืองไกร รังสิพล และอีกหลายท่านที่ยังไม่ได้เอยนามจึงขอขอบพระคุณท่านทั้งหลายไว้ ณ ที่นี้ด้วย

คณะผู้จัดทำ

หนังสืออ้างอิง

1. INTEL,MICROCONTROLLER HANDBOOK,1983

2. INTEL,MCS-48 ARCHITECTURAL OVERVIEW,1983

3. รศ.พิพัฒน์ เลาหสงคราม,ไมโครโปรเซสเซอร์,1985