RTC MULTI CONTROLLER

ปริญญานิพนธ์ปีการศึกษา 2533

ภาควิชา เทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง REAL TIME CLOCK MULTICONTROLLERS

ผู้จัดทำ

1. นายเกียรติศักดิ์ สนิกะวาที รหัส 323305
2. นายไพรัช กลิ่นเกษร รหัส 323321

.....อาจารย์ที่ปรึกษา

(ผศ.นิกร สุขุตมตันติ)

ปีการศึกษา 2533

REAL TIME CLOCK MULTICONTROLLERS

โดย

นายเกียรติศักดิ์ สนิกะวาที รหัส 323305

นายไพรัช กลิ่นเกษร รหัส 323321

อาจารย์ที่ปรึกษา

ผศ.นิกร สุขุตมตันติ

REAL TIME CLOCK MULTICONTROLLERS

นายเกียรติศักดิ์ สนิกะวาที
นายไพรัช กลิ่นเกษร

ผศ.นิกร สุขุตมตันติ อาจารย์ที่ปรึกษา
ปีการศึกษา 2533

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการประยุกต์ไมโครโปรเซสเซอร์เบอร์ 8031 และ Real Time Clock เบอร์ MM58167 มาใช้ในการควบคุมเครื่องใช้ไฟฟ้าที่ให้ความเที่ยงตรงสูง อีกทั้งยังสามารถโปรแกรมตั้งเวลาปิด-เปิดเครื่องใช้ไฟฟ้าได้ถึง 8 ช่องอิสระต่อกัน

ในด้านความละเอียดนั้น สามารถโปรแกรมเวลาได้ละเอียดถึง 1 นาที การทำงานของวงจรทั้งหมดถูกกำหนดด้วยซอฟท์แวร์ ซึ่งได้รวบรวมและสรุปไว้ในปริญญานิพนธ์ฉบับนี้แล้ว

REAL TIME CLOCK MULTICONTROLLERS

KIATTISAK SANIKAVATEE
PAIRUCH KLINKESORN

ASSISTANT PROFESSOR NIKORN SUKUTAMATANTI
(ADVISOR)

1990

ABSTRACT

This thesis is design applied microcontroller (#8031) and real time clock (#85167) to control electric device at high efficiency. Detail program on-off time of electric device can program to 8 channel.

Program can set time in 1 minute minimum. Working of circuit is control by software which control all circuit. Detail of circuit and software arein this thesis.

สารบัญ

บทที่ 1
บทนำ

ปัจจุบันเทคโนโลยีด้านไมโครโปรเซสเซอร์ ได้เข้ามามีบทบาทในชีวิตประจำวันของมนุษย์เรามาก โดยใช้ในงานควบคุมต่าง ๆ เช่น ใช้ควบคุมการเปิดไฟในเวลากลางคืนโดยไม่ต้องใช้คนในการเปิด-ปิด, ควบคุมระบบปั๊มน้ำที่ต้องการกระทำเป็นประจำ, เปิด-ปิดแอร์ตามสำนักงานในเวลาทำงานและเลิกงาน เหล่านี้เป็นต้น

ระบบควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าต่าง ๆ ให้ทำงานตามเวลานั้น ในปัจจุบันนี้ส่วนใหญ่จะใช้งานร่วมกับระบบ Real Time Clock หรือนาฬิกาบอกเวลาที่แท้จริงให้กับไมโครโปรเซสเซอร์ ซึ่งทำให้ได้ความเที่ยงตรงสูงเหมาะสำหรับงานควบคุมที่ต้องการความละเอียด

ปริญญานิพนธ์ฉบับนี้ ได้ทำการศึกษาและออกแบบระบบควบคุมขึ้นมาใหม่ทั้งหมดโดยภาคแสดงผลที่ใช้เป็น LCD และใช้ไมโครโปรเซสเซอร์เบอร์ 8031 ซึ่งเน้นที่จะศึกษาเทคโนโลยีใหม่เป็นสำคัญ

สถาปัตยกรรม MCS-51

ในปัจจุบันชิพ MICROCONTROLLER ได้มีการพัฒนาให้มีความสามารถที่จะนำไป ประยุกต์ใช้งานด้านต่างๆได้สูงขึ้นด้วยการเพิ่มองค์ประกอบการทำงานที่จำเป็นมากขึ้นลงบนชิพตัวเดียวกันหรือมีการผลิตให้ SINGLE SHIPในแต่ละตระกูลสามารถที่จะนำไปใช้งานการควบคุมเฉพาะด้านได้มากตัวขึ้น เช่นในตระกูล MCS-51ก็จะมีการผลิต SINGLE SHIPเบอร์หนึ่งที่ใช้งานเฉพาะด้าน

MCS-51 เป็นอุปกรณ์ที่ออกแบบมาสนองความต้องการของผู้ใช้คือมีสาย อินพุทและเอาท์พุท ภายในตัวเอง พอร์ทของ อินพุทและ เอาต์พุทบัพเฟอร์ อินเตอร์เฟส และสายควบคุมอื่นๆ ที่ใช้สำหรับแยกข้อมูล กับแอดเดรส และยังมีชุดคำสั่งเพิ่มขึ้น เป็นพิเศษเพื่อจัดการข้อมูล แถมด้วยวงจรตั้งเวลากับวงจรนับด้วย(ปกติวงจรนับจะสามารถ ทำงานเป็นวงจรตั้งเวลาได้ด้วย จึงเรียกควบคู่กันไปคือวงจรตั้งเวลา/วงจรนับ)

ลักษณะทั่วไป

1. สร้างโดย CHMOSเทคโนโลยีและการทำงานด้วยแหล่งจ่ายไฟ +5V
2. ซีพียู มีขนาด 8 บิท
3. มีวงจรออสซิลเลเตอร์ และวงจรนาฬิกาบนชิพ
4. ชุดแบงค์รีจิสเตอร์มี 4ชุด แต่ละชุดมีรีจิสเตอร์ 8 ตัว
5. มีตัวจับเวลา/ตัวนับขนาด 16 บิท 2ชุด สำหรับ 8032/8052มี 3ชุด
6. มีPORT I/O แบบขนาน 2 ทิศทางจำนวน 4 พอร์ท ๆ ละ 8 บิทรวมทั้งหมด 32เส้นแต่จะเหลือเพียง 16เส้นสำหรับเบอร์ 8031 อีก 16เส้นใช้ในการเข้าถึงทางแอดเดรสและข้อมูล
7. พอร์ทแบบอนุกรมสามารถที่จะโปรแกรมรับส่งแบบ FULL DUPLEX ที่ความเร็วสูง
8. หนึ่งวัฏจักรคำสั่งจะใช้เวลา 1ไมโครวินาทีด้วย CRYSTAL 12 MHz
9. แอดเดรสข้อมูลภายนอกได้ 64 KBYTE
10. แอดเดรสโปรแกรมภายนอกได้ 64 KBYTE
11. สามารถกำหนดที่อยู่ข้อมูลขนาดไบท์หรือบิทได้โดยตรง
12. มีซอฟแวร์แฟลกสำหรับผู้ใช้ที่จะกำหนดเองได้ถึง 128 ตำแหน่งบิท
13. โครงสร้างอินเตอรรัพท์ทำได้ 5 แหล่งและ 6 แหล่งสำหรับ8032/8052พร้อมการจัด PRIORITY ได้ 2 ระดับ
14. ตัวโปรเซสเซอร์สามารถใช้งานแบบบูลลีน เหมาะที่จะใช้ในงานควบคุม
15. มีคำสั่งคูณและหารทาง HARDWARE ทำได้ภายในเวลา 4 ไมโครวินาที

16. ตัวเลขทางคณิตศาสตร์ใช้ได้ทั้งแบบไบนารี่และเดซิมัล

17. การใช้พื้นที่ STACK สำหรับโปรแกรมย่อยต่างๆทำได้ง่ายและกว้าง

18. ชุดคำสั่งของ MCS-51 โดยการกำหนดเลขที่ (ADDRESSING MODE) ได้มากกว่าชุดคำสั่งของ MCS-51

ตระกูล MCS-51จะมีทั้งแบบมี ROMในตัวหรือไม่มี ROMหรือมี EPROM บนชิพเดียวกันและจะมี ตำแหน่งขาที่เหมือนกัน ตารางที่1 แสดงถึงรายละเอียดของเบอร์ต่างๆในตระกูล MCS-51ที่มีขายในท้องตลาด

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5
8031	none	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-8	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

ตารางที่ 1 รายละเอียดของตะกูล MCS 51

8751H อยู่ในกลุ่มรุ่นเดียวกับ 8051AH ที่เราสามารถโปรแกรมได้ด้วยระบบไฟและสามารถลบโปรแกรมออกได้ด้วย แสงอุลตร้าไวโอเล็ท นอกเหนือจากไอซีที่แสดงในตารางที่ 1 ที่ใช้เทคโนโลยี HMOS ที่มีจำหน่ายในขณะนี้ คือเบอร์80C51, 80C31และ87C51

การจัดลักษณะภายนอกของ MCS-51

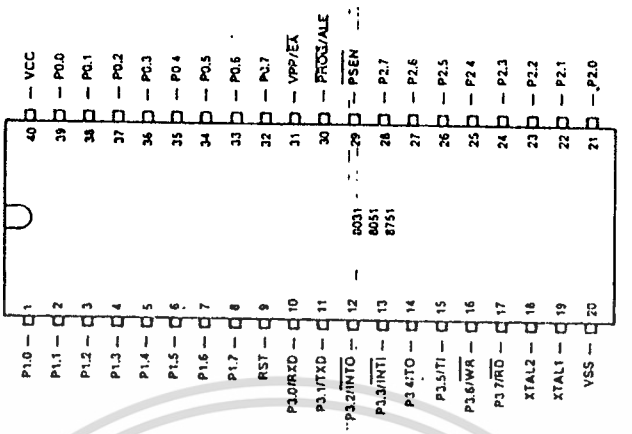
รูปที่1 แสดงการจัดขาตามลักษณะภายนอกของชิพ MCS-51ซึ่งมีรายละเอียดดังนี้คือ

-ขา VSS(ขา20)เป็นขาสำหรับต่อลงดิน

-ขาVCC(ขา 40)เป็นขาที่ต่อแรงดันไฟกระแสตรงขนาด 5V และสำหรับการโปรแกรม

ขา PORT 0 P0.0-P0.7/AD0-AD7)(ขา32-39) เป็นพอร์ทไอโอ 8บิทการเขียนค่า "1" ไปที่พอร์ท จะเป็นการปล่อยลอย ขาของพอร์ททำให้มันสามารถเป็นอินพุทสถานะอิมพีแด๊นซ์สูง ในการให้พอร์ทบริการแบบไอโอพอร์ท 0 จะทำงานเป็น MULTIPLEX ด้วยสัญญาณแอดเดรสไบท์ต่ำกับบัสข้อมูล สำหรับการใช้งานด้านหน่วยความจำภายนอก ในการใช้งานแบบนี้จะใช้ลักษณะภายในเป็นตัวพูลอัพ พอร์ท 0 ยังใช้งานเป็นตัวส่งข้อมูลออกไปทาง

พอร์ทนี้เมื่อใช้บริการ ทางด้านการตรวจสอบโปรแกรม ROM ภายในและการโปรแกรมตัว EPROM ภายในถ้าใช้งานในลักษณะนี้ การพูลอัพจากภายนอกจะต้องต่อด้วยค่า 10 โอห์ม



รูปที่ 1.1 ลักษณะขาภายนอกของ MCS 51

ขาพอร์ท 1(P1.0-P1.7)(ขา1.0-1.7)เป็นพอร์ทไอโอ 8บิท แบบ OPEN DRAIN BIDIRECTIONAL พร้อมด้วยการพูลอัพภายในถ้าเป็น พอร์ท OUTPUT บัพเฟอร์สามารถขับโหลดทีทีแอล ตระกูล LSI ได้ 4 ตัว พอร์ท1 เมื่อถูกเขียนค่า "1" ด้วยโปรแกรม มันจะมีสถานะสูง ด้วยการพูลอัพภายใน การให้สถานะเช่นนี้ จะเป็นการ INITIAL ใช้งานพอร์ทนี้ ให้เป็นอินพุทขณะที่พอร์ท 1 เป็นอินพุท การให้สัญญาณลงต่ำจะเป็นการจ่ายกระแสออกเนื่องจากการพูลอัพภายในเบอร์ 8052 ขา P1.0 และ P 1.7จะใช้งานเป็น T2และ T2 EX จะเป็นอินพุทผ่านเข้าตัวตั้งเวลา 2 ถูกกระตุ้นให้ทำงานแบบปกติตามโปรแกรมที่ตั้งไว้ หรือ แคปเตอร์(CAPTURE)

ขา PORT 2 (P2.0-P2.7 ขา 21-28) เป็นพอร์ทไอโอ 8 บิทแบบพูลอัพภายใน พอร์ท 2 ทำหน้าที่เป็นบัพเฟอร์ OUTPUT สามารถจ่ายโหลด ที่ทีแอลตระกูล LSI ได้ 4 ตัว พอร์ทจะถูกใช้งานเป็นตัวส่งแอดเดรสไบท์สูงด้วย เมื่อใช้งานร่วมกับหน่วยความจำภายนอก เพื่อให้แอดเดรสได้ถึง 16 บิท ด้วยการใช้งานแบบนี้มันจะมีพูลอัพภายในที่ช่วยในการส่งค่า "1" ได้ระดับที่แน่นอน นอกจากการใช้งานสำหรับแอดเดรส อันดับสูงยังใช้เป็นขาควบคุมในการใช้งานและเขียนโปรแกรม เบอร์ 8751 และตรวจสอบโปรแกรมภายใน 8051

ขา PORT 3 (P3.0-P3.7 ขา 10-17) เป็นพอร์ทไอโอ 8 บิท แบบพูลอัพภายใน นอกจากทำเป็นพอร์ทไอโอที่สามารถรับโหลด ทีทีแอล ตระกูล แอลเอสได้ 4 ตัวแล้ว ยังใช้งานเป็นพิเศษสำหรับตระกูล MCS-51 ตามรายการข้างล่างนี้ด้วย

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

ตารางที่ 2 รายละเอียดของพอร์ท 3

การที่จะให้ทำงานตามฟังก์ชันข้างบน จะต้องเริ่มโปรแกรมด้วยการส่งค่า "1" ไปแลทช์ที่บิท รีจิสเตอร์ควบคุมต่างๆไว้ก่อนที่ให้ทำงานตามฟังก์ชันข้างบน

ขา RST(ขา 9) ต้องคงสถานะค่าสูงเป็นเวลา 2 วัฏจักรระหว่างที่ออสซิลเลเตอร์ทำงาน โดยจะต่อรีจิสเตอร์พูลดาวน์(8.2k)จากขา RST ไปลงดินและเพื่อให้ตัวชิพ รีเซทได้โดยอัตโนมัติ ขณะเปิดไฟจะใช้คาปาซิเตอร์ (10 MF) ต่อคร่อมระหว่างขา RSTกับขา VCC

ขา ALE/PROG (ขา 30) เป็นขาแอดเดรส แล็ทช์ อีนาเบิ้ลด้วยการส่งพัลซ์ออกไปใช้สำหรับแล็ทช์ค่า แอดเดรสไบท์ต่ำจาก พอร์ท 0 ในระหว่างการเข้าถึง ข้อมูลจากภายใน ดังนั้นจึงสามารถที่จะ ใช้สัญญาณจากขานี้เป็นตัวตั้งเวลาภายนอก หรือเป็นความถี่ สัญญาณนาฬิกาแต่อย่างไรก็ตามความถี่สัญญาณนี้ช้าลงไปเท่าหนึ่งระหว่างการทำงานแบบการเข้าถึง ของหน่วยความจำข้อมูลภายนอกขานี้ยังใช้เป็น สัญญาณพัลซ์ เข้าสำหรับการควบคุมการโปรแกรม EPROM ภายในชิพ

ขาPSEN(ขา 29)PROGRAM STORAGE ENABLE เป็นสโตรปอ่านข้อมูลจากหน่วยความจำภายนอก ขา PSEN จะสร้างสโตรปต่ำสองครั้งภายในแต่ละวัฏจักร แมชชีน สัญญาณจะมีสถานะสูงหรือพัลซ์ต่ำทั้งสองลูกจะหายไปเมื่อทำงานในช่วง การอ่านหรือเขียนข้อมูลจาก หน่วยความจำข้อมูลภายนอกและ PSEN จะไม่มีพัลซ์ส่งออก ถ้าชิพทำงานด้วยโปรแกรมหน่วยความจำภายใน

ขา EA/VPP (ขา 31) มีสถานะสูง ตัวซีพียู ในชิพจะทำงานตามโปรแกรมที่อยู่ในหน่วยความจำภายใน (โดยโปรแกรมจะต้องไม่ยาวกว่า 4 กิโลไบท์) สำหรับเบอร์ 8051AH และ 8 กิโลไบท์ สำหรับเบอร์ 8052 AH) การทำให้ EA มีสถานะต่ำจะเป็นการควบคุมให้ ซี พี ยู ทำงานตามโปรแกรมหน่วยความจำภายนอก ซึ่งขยายโปรแกรม ได้ยาวถึง 64 กิโลไบท์ ในตัว 8031AH และ 8032AH ขา EA จะต้องต่อลงดินเช่นกันแม้ว่าจะไม่มี ROM อยู่ภายในก็ตาม ในตัว 8751H จะใช้ขานี้จ่ายแรงดันขนาด 21 V ขณะทำการเขียนโปรแกรมเข้า EAPROM ของชิพ 8751H ตัวนี้

ขา XTAL 1,ขา 19 ใช้เป็นตัวอินพุท เข้าสู่ตัวออสซิลเลเตอร์ขยายแบบ INVERT

ขา XTAL 2 ขา 18 ใช้เป็นตัวเอ๊าต์พุทจากตัวออสซิลเลเตอร์ขยายแบบ INVERT

ตามตาราง 1 MCS-51ทั้งสามกลุ่มคือ กลุ่มที่มี ROM ไม่มี ROM และพวก EPROM จะมีขาใช้งานเหมือนกันหมด ยกเว้น ขา 1 จะใช้งานเป็น T2 EX ในเบอร์ 8032/8052 ตลอดถึงจังหวะเวลาและคุณสมบัติทางไฟฟ้าทั้งสามจะแตกต่างกันเฉพาะ การโปรแกรมบนชิพ MCS-51 เท่านั้นซึ่งแต่ละแบบจัดไปตาม ความต้องการ ของผู้ใช้ เช่น 8751 จะมี 4 กิโลไบท์ ของ ULTRAVIOLET-ERASABLE PROGRAMMABLE READ ONLY MEMORY (EAPROM) เหมาะสำหรับ การพัฒนาต้นแบบ และ การผลิตอุปกรณ์ที่มีจำนวนจำกัด เมื่อต้องการเขียนโปรแกรมเข้า EAPROM จะมีตัวเขียนโปรแกรมพิเศษ สำหรับเขียนโปรแกรมที่ผู้ออกแบบเขียนขึ้นมา ถ้าโปรแกรมมีบักหรือส่วนผิดพลาดที่ต้องการแก้ไข ก็สามารถแก้ไขได้โดยนำตัว 8751 นี้ไปล้างโปรแกรมเดิมออกด้วยแสงอุลตร้าไวเล็ท และอัดข้อมูลโปรแกรมที่ได้แก้ไขแล้วเข้าไปใหม่ ทำเช่นนี้จนกระทั่งได้โปรแกรมสมบูรณ์ และเมื่อ ต้องการผลิตจำนวนมากก็สามารถที่จะใช้ 8051 ที่มี 4 กิโลไบท์ของ ROM ซึ่งจะถูกอัดข้อมูลโปรแกรมตามความต้องการของผู้ออกแบบ โดยโรงงานของผู้ผลิตชิพเบอร์นี้การผลิตลักษณะนี้จะถูกกว่าการใช้เบอร์ 8751 แต่โปแกรมภายในจะไม่สามารถลบและโปรแกรมใหม่หลังการผลิตไปแล้ว

ส่วนเบอร์ 8031จะไม่มีหน่วยความจำของโปรแกรมบนชิพ แต่อาจต่อหน่วยความจำโปรแกรมจากภายนอกด้วย ROM, EPROM หรือ PROM ได้ถึง 64 กิโลไบท์ ดังนั้น 8031 จึงเหมาะสำหรับการใช้งานที่โปแกรมมีขนาดใหญ่กว่า 4 กิโลไบท์ และสำหรับ ผู้ออกแบบ ที่ต้องการแยกส่วนของ โปรแกรมออกจากชิพ

การจัดหน่วยความจำ

MCS-51 จะแยกแอดเดรสสำหรับหน่วยความจำของโปรแกรม และหน่วยความจำของข้อมูลออกจากกัน หน่วยความจำของโปแกรมขยายได้ถึง 64 กิโลไบท์ และจำนวนไบท์ต่ำ 4 กิโลไบท์จะอยู่ใน 8051 หน่วยความจำของข้อมูลมี 128 ไบท์ใช้สำหรับ รีจิสเตอร์ฟังก์ชันพิเศษ และหน่วยความจำข้อมูลภายนอกอีก 64 กิโลไบท์

MCS-51 จะจัดแบ่งตำแหน่งสำหรับ SFR ให้ทำงานเป็น รีจิสเตอร์ต่างๆ ดังนี้

		ตำแหน่ง
*ACC	ACCUMULATOR	0E0H
*B	B รีจิสเตอร์	0F0H
*PSW	PROGRAM STATUS WORD	0D0H
SP	STACK POINTER	081H
DPTR	ตัวชี้ข้อมูล ประกอบด้วย DPH	083H
	และ DPL	082H
*P0	พอร์ท 0	080H
*P1	พอร์ท 1	090H
*P2	พอร์ท 2	0A0H
*P3	พอร์ท 3	0B0H
*IP	ตัวควบคุมการอินเตอร์รัพท์ตามลำดับ	0B8H
*IE	ตัวควบคุมการอินเตอร์รัพท์อินาเบิ้ล	0A8H
TMOD	ตัวควบคุมการเลือกโหมดตัวจับเวลา/ตัวนับ	089H
*T2CON	ตัวควบคุมตัวจับเวลา/ตัวนับ 2	088H
TCON	รีจิสเตอร์ตัวจับเวลา/ตัวนับ	0C8H
TH0	รีจิสเตอร์ตัวจับเวลา/ตัวนับ0 (ไบท์สูง)	08CH
TL0	รีจิสเตอร์ตัวจับเวลา/ตัวนับ0(ไบท์ต่ำ)	08AH
TH1	รีจิสเตอร์ตัวจับเวลา/ตัวนับ1(ไบท์สูง)	08DH
TL1	รีจิสเตอร์ตัวจับเวลา/ตัวนับ1(ไบท์ต่ำ)	08BH
+TH2	รีจิสเตอร์ตัวจับเวลา/ตัวนับ2(ไบท์สูง)	0CDH
+TL2	รีจิสเตอร์ตัวนับเวลา/ตัวนับ2(ไบท์ต่ำ)	0CCH

+RLDH	รีจิสเตอร์ตัวจับเวลา/ตัวนับ2 ประจุใหม่อัตโนมัติ(ไบท์สูง)	0CBH
+RLDL	รีจิสเตอร์ตัวจับเวลา/ตัวนับ 2 ประจุใหม่อัตโนมัติ(ไบท์ต่ำ)	0CAH
*SCON	ควบคุมการส่งข้อมูลอนุกรม	098H
SBUF	บัพเฟอร์ข้อมูลการส่งแบบอนุกรม	099H
PCON	ควบคุมการใช้พลังงาน	097H

เครื่องหมาย * หน้าตัวรีจิสเตอร์ แสดงว่า รีจิสเตอร์นั้น สามารถที่จะแอดเดรสข้อมูลได้ทั้งข้อมูล ไบท์และบิท และเครื่องหมาย+ นั้นแสดงว่าจะมีค่าเฉพาะในเบอร์ 8032/8052 เท่านั้น

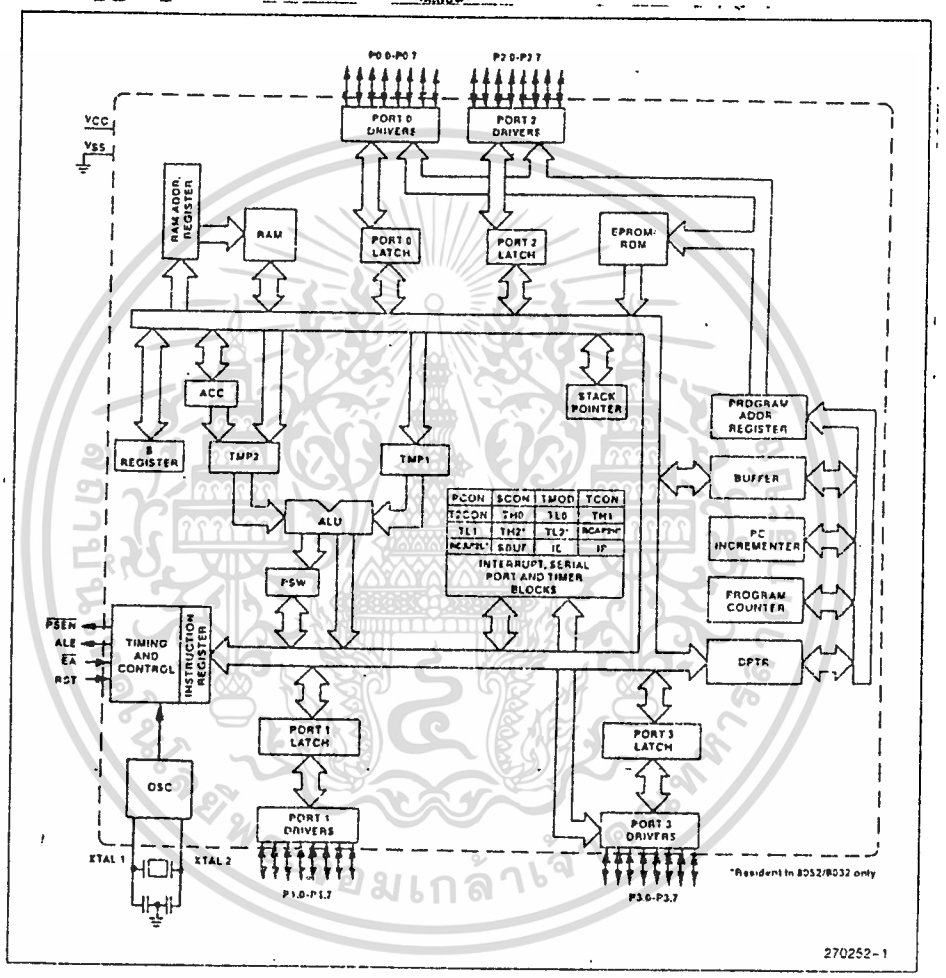
การจัดทางสถาปัตยกรรม

รูปที่ 2 เป็นบล็อคไดอะแกรมที่แบ่งตามลักษณะงานในการจัดภายในของ MCS-51 โดยชิงเกิ้ลชิพแต่ละตัวของตระกูลนี้ จะประกอบด้วยหน่วยศูนย์กลางประมวลผล หน่วยความจำ 2 ชนิดคือ แบบ RAM หรือ ROM หรือ EAPROM พอร์ทเอ้าท์พุท อินพุท โหมด รีจิสเตอร์สถานะและข้อมูลขนาด 8 บิท และจะมีบัพเฟอร์สำหรับการติดต่อข้อมูลภายนอกผ่านพอร์ทไอโอ เมื่อต้องการขยายหน่วยความจำ หรือพอร์ท ไอโอ

หน่วยศูนย์กลางประมวลผลหรือ ซีพียู

ซี พี ยู เป็นมันสมองของระบบไมโครคอมพิวเตอร์ การอ่านโปรแกรมและทำงานตามคำสั่ง โปรแกรมจะกระทำที่ส่วนนี้ โดยการใช้ส่วน คณิตศาสตร์และตรรกศาสตร์ทำงานร่วมกับ รีจิสเตอร์ A,B,PSW,ตัวนับ โปรแกรม ขนาด 16 บิท และตัวชี้ตำแหน่งข้อมูล ส่วนคณิตสาตร์และตรรกศาตร์ ALU นี้ทำงานในฟังก์ชันคณิตศาสตร์และ ตรรกศาสตร์ ด้วยตัวแปรต่างๆขนาด 8 บิท ที่มีลักษณะการทำงานทางคณิตศาสตร์เป็น บวก ลบ คูณ หาร รวมทั้งตรรกศาสตร์เช่น AND OR XOR รวมทั้งเลื่อนและวนรอบบิท การเคลียร์ค่าและกลับค่าเป็นต้นALU ยังสามารถที่จะตัดสินใจในการกระโดดไปทำคำสั่งของโปรแกรมในส่วนอื่น ๆ ตามเงื่อนไขที่ตั้งขึ้น และยังแบ่งรีจิสเตอร์ชั่วคราวใช้สำหรับทางผ่านชั่วคราวของข้อมูลในการถ่ายเทภายใน ระบบ คำสั่งอื่นที่มีการใช้ ALU ยังมีความสามารถที่จะเพิ่มค่าในรีจิสเตอร์ ในลักษณะการบวกด้วย 1 หรือค่านวณ เลขที่อยู่ของ ข้อมูลที่จะนำ ไปเก็บหรือลดค่าลงครั้งละ 1 ในลักษณะการลบด้วยค่า 1 โดยอัตโนมัติ หรือใช้ใน การเปรียบเทียบค่าของตัวแปรทั้งสองสิ่งสำคัญในการทำงานทางสถาปัตยกรรมของ MCS--51 คือความสามารถในการ

ทำงานสำหรับข้อมูล 8 บิทและ 1 บิท การใช้งานในระดับบิทในการเซ็ท เคลียร์ หรือกลับค่า การเคลื่อนย้าย การทดสอบ และใช้ในการคำนวณทางตรรก ขนาด 1 บิทความสามารถเช่นนี้ เหมาะสมสำหรับใช้งานควบคุมสัญญาณเข้าและออกที่มี การคิดและออกแบบทางตรรกด้วยพีชคณิต BOOLEAN ซึ่งโดยปกติทำได้ลำบากสำหรับไมโครโปรเซสเซอร์ทั่วๆไปงานในลักษณะเช่นนี้จึงได้ชื่ออีกอย่างหนึ่งว่า ตัวประมวลผล บูลลีน



รูปที่ 1.2 สถาปัตยกรรม MCS 51

โครงสร้างพอร์ทและการทำงาน

MCS-51 มีพอร์ท 4 พอร์ท เป็นแบบ 2 ทิศทาง แต่ละพอร์ทจะประกอบด้วยแลทช์เป็น P0 ถึง P3 ของ SFR จะมีตัวขับเอ๊าต์พุทและบัฟเฟอร์อินพุท ตัวขับเอ็าต์พุท ของพอร์ท 0 และ 2 และบัฟเฟอร์อินพุทของพอร์ท 0จะใช้งานสำหรับการเข้าถึง หน่วยความจำภายนอก ในการใว้งานลักษณะนี้เอ๊าต์พุทพอร์ท 0 จะทำหน้าที่เป็นตัวกำหนดไบท์ต่ำ ของแอดเดรสหน่วยความจำภายนอก โดยที่ค่าแอดเดรส และค่าข้อมูลจะถูก MULTIPLEX ด้วยช่วงจังหวะการเฟทช์และการอ่านหรือเขียนข้อมูล ส่วนเอ๊าต์พุทพอร์ท 2 จะทำหน้าที่เป็นตัวกำหนดส่งไบท์สูงของแอดเดรสในการเข้าถึง หน่วยความจำภายนอก

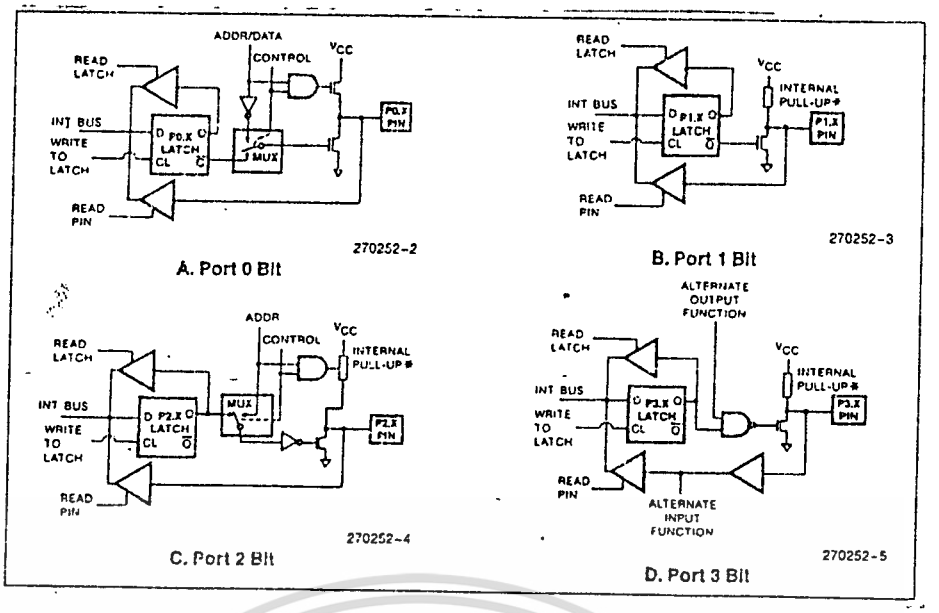
บางขาของตัวขับเอ๊าต์พุทและบัฟเฟอร์อินพุทของขา 1.0,1.1 และพอร์ท 3 ทั้งหมดสามารถนำไปใช้งานเป็นแบบหลายฟังก์ชันได้ดังนี้

ขาพอร์ท	การใช้งานตามฟังก์ชัน
P1.0	T2(TIMER/COUTER 2สัญญาณอินพุทจากภายนอก)
P1.1	T2 RST(TIMER/COUTER 2 สัญญาณอินพุทการรีเซ็ทภายนอก
* P3.0	RX(พอร์ทรับข้อมูลอนุกรม)
* P3.1	TXD(พอร์ทส่งข้อมูลอนุกรม)
P3.2	INT0(การใช้อินเตอร์รัพท์ภายนอกตัวที่ 1)
P3.3	INT1(การใช้อินเตอร์รัพท์ภายนอกตัวที่ 2)
P3.4	T0(TIMER/COUNTER 0 สัญญาณอินพุทภายนอก)
P3.5	T1(TIMER/COUNTER 1 สัญญาณอินพุทภายนอก)
P3.6	WR (สโตรปการเขียนหน่วยความจำภายนอก)
P3.7	RD (สโตรปการอ่านหน่วยความจำภายนอก)

ตัวขับเอ๊าต์พุทในการที่จะให้ทำงานตามตารางบนจะต้องเริ่มโปรแกรมด้วยการเซ็ทค่า 1 เก็บในแล็ทช์ก่อน เครื่องหมาย*แสดงถึงการใช้ตัว TIMER/COUTER 2 ซึ่งมีเฉพาะในเบอร์ 8032/8052 เท่านั้น ทั้งโหมดตัวจับเวลาหรือตัวนับการโหลดใหม่แบบอัตโนมัติของ TIMER/COUTER 2 ที่รีจิสเตอร์ RLDH และ RLDL จะเกิดขึ้น ถ้าการโหลดใหม่แบบอัตโนมัติถูกเลือกใช้งานด้วยกำหนดในบิท CP/RL2=0

การกำหนดใช้งาน ไอโอ

พอรท์ 0 มีความแตกต่างจากพอร์ท์อื่นที่ไม่มีพูลอัพภายใน แต่มี FET ตัวบนมาต่อแทนเป็นตัว DRIVER OUTPUT โดยถ้ามันให้วงจรทำงานจะเป็นการทำงานแบบบัส แอดเดรสและข้อมูล ด้วยการเข้าถึงข้อมูลภายนอกด้วยสัญญาณ



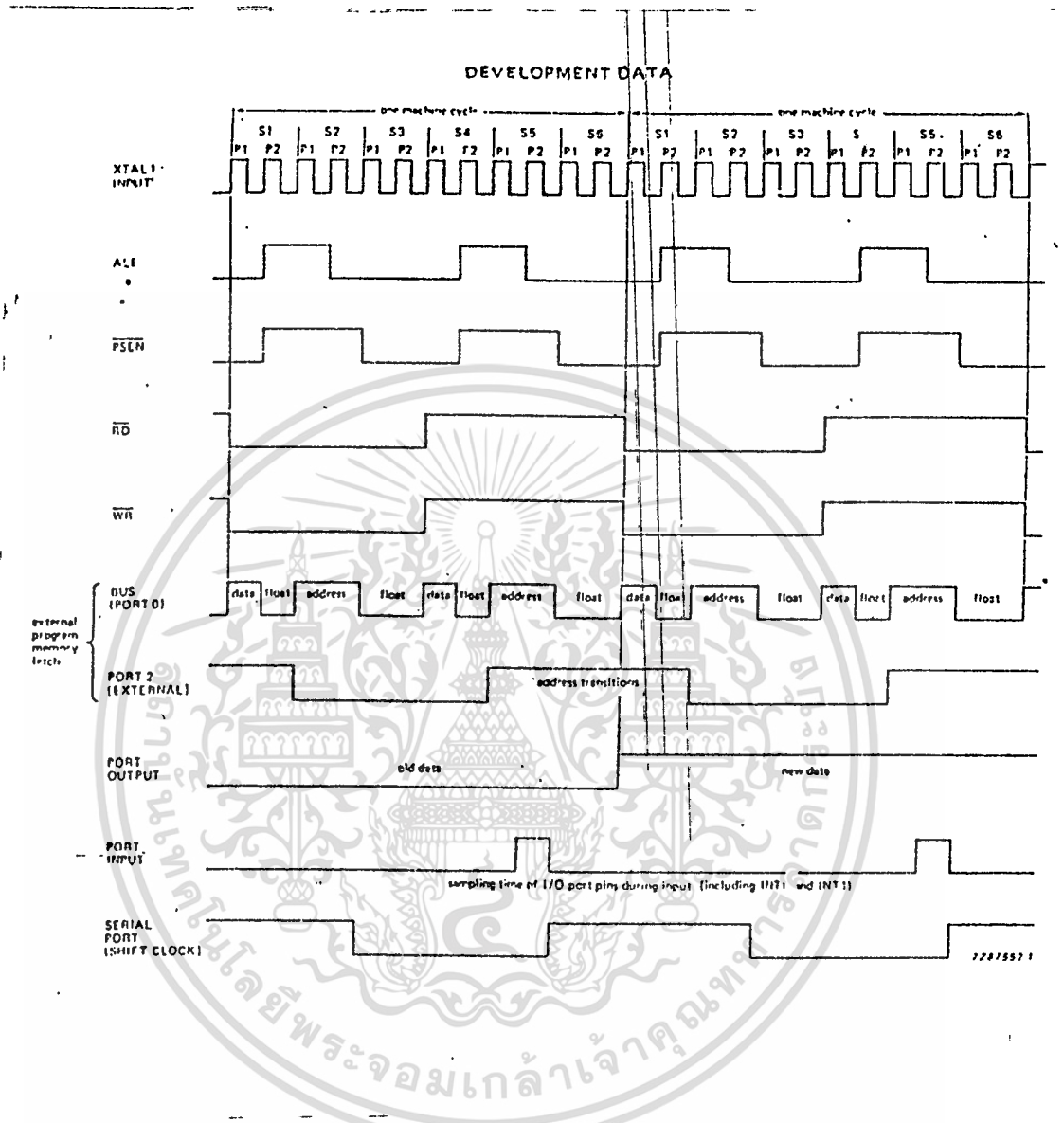
รูปที่ 1.3 พอร์ทการกำหนดแลทช์และบัฟเฟอร์

ควบคุมภายใน แต่งานลักษณะนี้ปกติ FET จะไม่ทำงาน ในการใช้รีจิสเตอร์ P0 แลทช์ค่า 1 ไว้ตามวงจรเกทของ FET ตัวล่างจะต่อที่ Q ซึ่งเป็นลอจิกระดับต่ำทำให้ FET ทั้งสองไม่ทำงานทั้งคู่ จะเป็นการปล่อยขาของพอร์ท 0 ให้ลอยตัวเป็นการให้อินพุทมีค่าอิมพีแดนซ์สูงนั่นเองขณะที่พอร์ท 2 ใน SFR จะมีค่าแอดเดรสที่ไม่เปลี่ยนแปลง

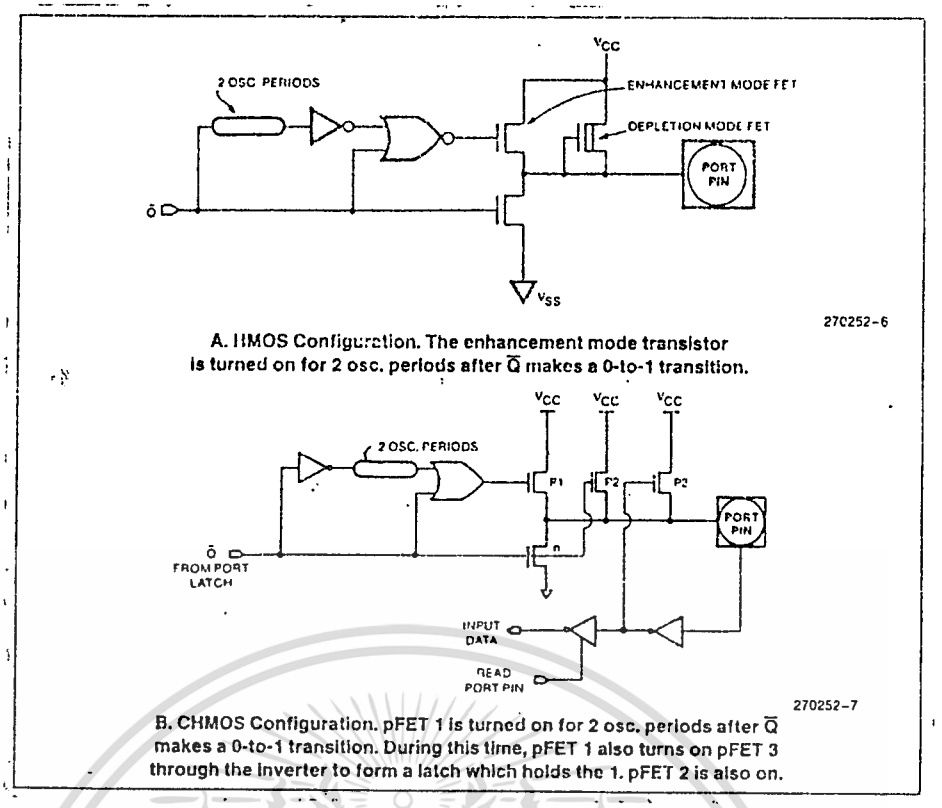
การเขียนไปยังพอร์ท

การทำงานตามคำสั่งเปลี่ยนค่าในแลทช์ของแต่ละพอร์ท ค่าใหม่จะเข้ามาเก็บในช่วงระหว่าง S6P2 ของวัฏจักรสุดท้ายของคำสั่ง อย่างไรก็ตามพอร์ทจะเก็บค่าในแล็ทช์ เมื่อมีการใช้ส่งข้อมูลออกที่บัฟเฟอร์เอ๊าต์พุทระหว่างเฟส 1 ของคาบเวลาใดๆ ของสัญญาณนาฬิกา(ส่วนระหว่างเฟส 2 บัฟเฟอร์เอ๊าต์พุทจะยังคงเก็บค่าเริ่มแรกที่ปรากฏในเฟส 1 ก่อนหน้านี้) โดยลำดับค่าใหม่ที่แล็ทช์ไว้จะยังไม่ปรากฏที่ขาของพอร์ทจนกว่าจะถึงเฟส 1 ตัวใหม่ซึ่งอยู่ในช่วง S1P1 ของวัฏจักรแมชชีนตัวต่อมาดังรูปที่ 4 ถ้าการเปลี่ยนจาก 0 เป็น 1 ของพอร์ท 1,2 หรือ 3 จะเป็นการให้พูลอัพทำงานในระหว่างช่วงระหว่าง S1P1และ S1P2 ของวัฏจักรเกิดการเปลี่ยนแปลง ลักษณะงานเช่นนี้ จะช่วยเพิ่มความเร็วของการเปลี่ยนแปลงการมีพูลอัพเพิ่มขึ้นสามารถที่จะจ่ายกระแสได้เพิ่มขึ้น 100เท่าตัวของการพูลอัพปกติ จะสังเกต

ได้ว่าการพูลอัพภายในเป็นเฟท ซึ่งไม่ใช่ตัวต้านทานเส้นตรง การจัดการพูลอัพมีแสดงในรูปที่ 5



รูปที่ 1.4 วัฏจักรการอ่านและเขียนไปยังพอร์ท



รูปที่ 1.5 การจัดพูลอัพภายในเริ่มแรกของพอร์ท 1,2,3

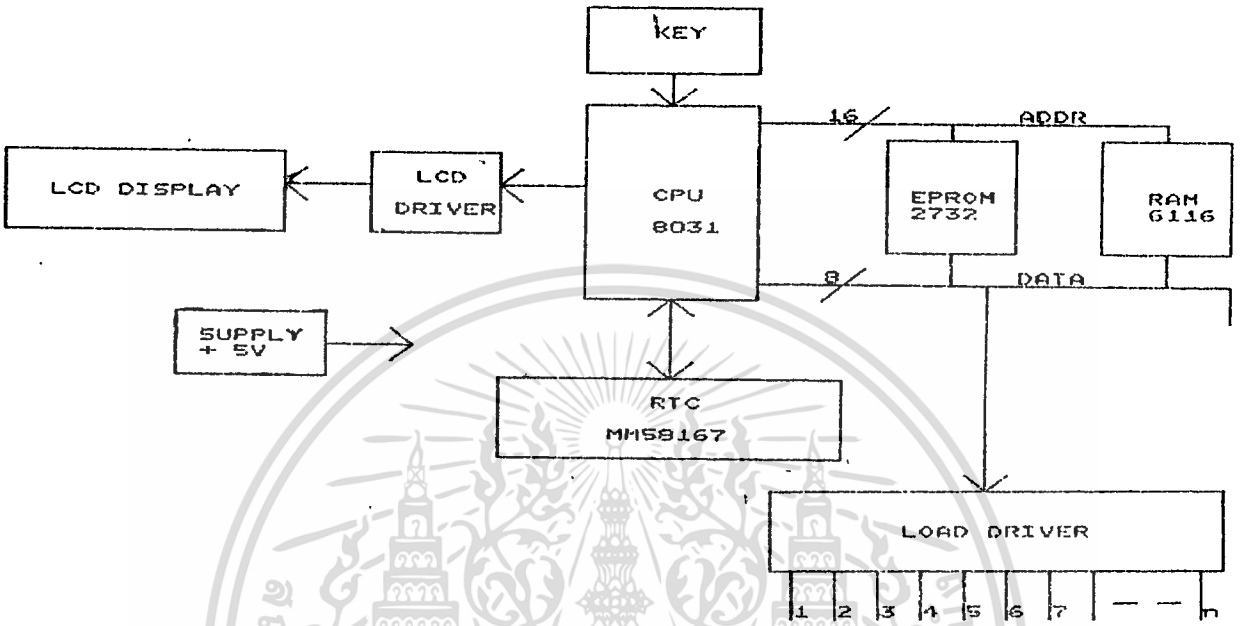
การบรรจุข้อมูลและการต่อพ่วงกับพอร์ท

เอ๊าต์พุทของพอร์ท 1,2 หรือ 3 สามารถขับทีทีแอลอินพุทแบบ LSได้ 4 ตัว พอร์ทเหล่านี้ของHMOS สามารถขับได้ทั้งวงจรทีทีแอล หรือ NMOS MCS-51 ทั้งแบบ HMOS และ CHMOS สามารถทำงาน เป็นตัวขับลักษณะ OPEN COLLECTOR และเอ๊าต์พุท OPENDRAIN โดยไม่ต้องมีพูลอัพจากภายนอก

พอร์ท 0 ตัวบัพเฟอร์ตัวเอ๊าต์พุทที่ขาสามารถขับ ทีทีแอล อินพุทได้ 8 ตัวอย่างไรก็ตาม ขาเหล่านี้ถ้าใช้ในการ ขับอินพุทแบบ NMOS ต้องต่อพูลอัพจากภายนอก ยกเว้นการใช้งานเป็นบัสแอดเดรส/ข้อมูล

บทที่ 3
การออกแบบและวงจรใช้งาน

3.1 บล็อกไดอะแกรมและหลักการทำงาน



รูปที่ 3.1 บล็อกไดอะแกรมของระบบ

จากรูป 3.1 แสดงบล็อกไดอะแกรมของ Real Time Clock Multicontrollers เพื่อความเข้าใจจะขอกล่าวเป็นส่วน ๆ พร้อมแสดงวงจรที่ใช้งานจริงในแต่ละบล็อกโดยละเอียดดังต่อไปนี้

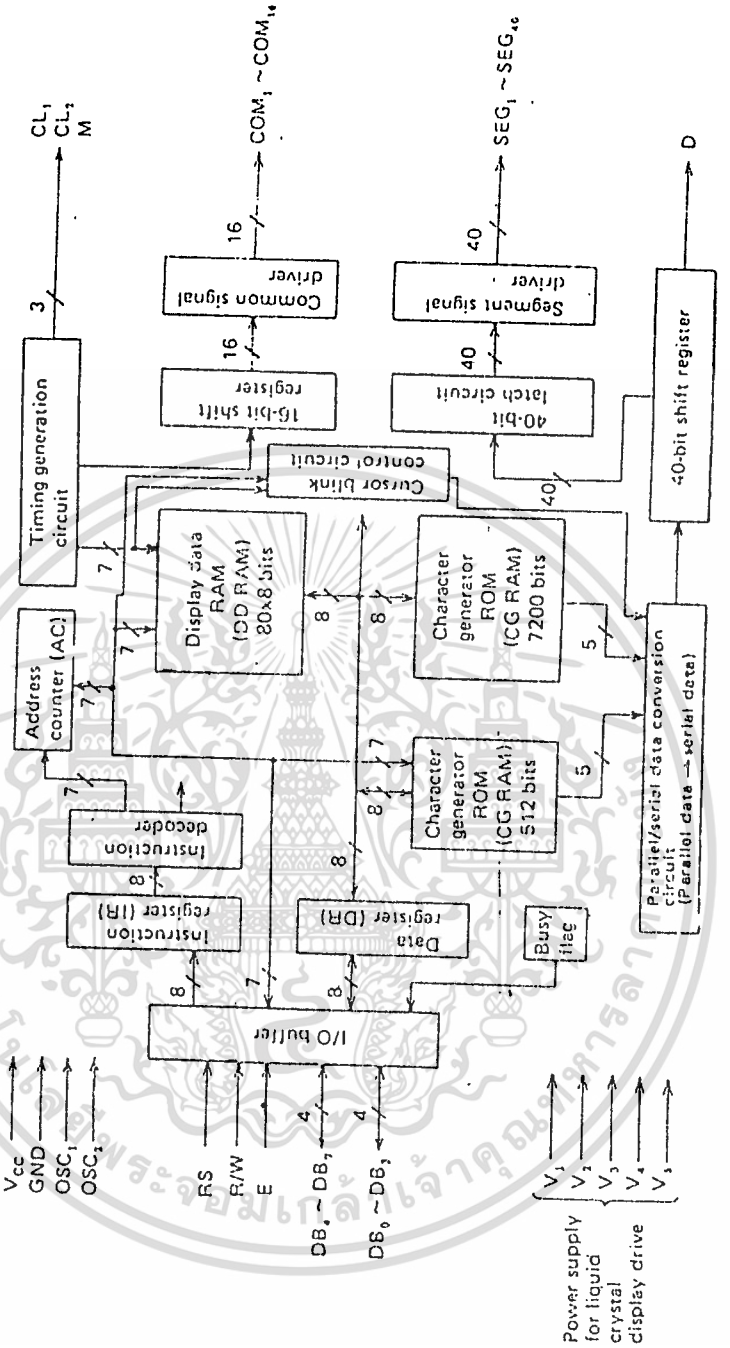
3.1.1 ส่วน LCD DISPLAY

จากรูป 3.2 มีส่วนประกอบใหญ่ ๆ ดังนี้

1. DOT MATRIX LCD เป็นตัวแสดงผลให้เรามองเห็นในลักษณะการปิดและเปิดตัวเองกับแสงก็คือ ส่วนที่เป็นตัวกระจกบรรจุผลึก

2. DRIVER เป็นตัวรับสัญญาณจากตัวควบคุมมาขับผลึก LCD อีกทีหนึ่งโดยมีเบอร์ที่นิยมใช้ใน LCD MODULE เช่น HD 44100

3. CONTROLLER เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมา และจัดการควบคุม LCD MODULE ให้ทำงานแสดงผลต่าง ๆ เช่น การลบจอภาพ, การเกิดตัวอักษร เป็นต้น โดยมีไอซีที่นิยมใช้กันคือ HD44780 ซึ่งจะใช้ในแบบ CHARACTER LCD MODULE เป็นส่วนใหญ่

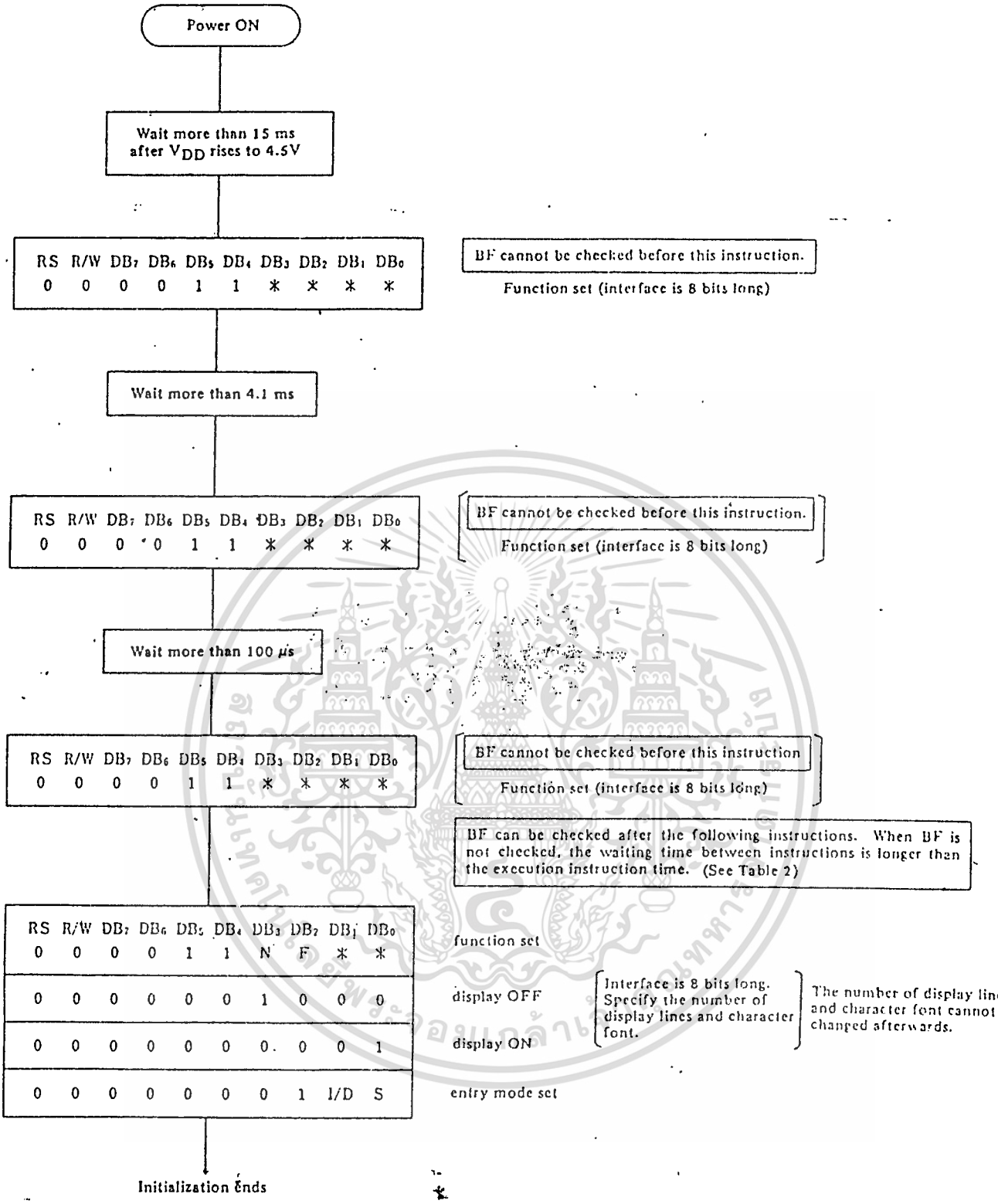


รูปที่ 3.2 BLOCK DIAGRAM OF HD44780

เมื่อเราเริ่มเปิดไฟป้อนให้ HD44780 นั้นก็จะทำการรีเซ็ทตัวมันเอง โดยจะใช้เวลาประมาณ 10 ms หลังจากไฟ VDD ถึง 4.5 Volt แล้ว โดยจะเซ็ทตัวเองดังนี้

1. DISPLAY CLEAR จะทำการลบข้อมูลจอภาพ
2. FUNCTION SET โดยจะเซ็ทค่าภายใน
   - DL = 1 : เป็นการเซ็ทให้เป็น 8 บิท
   - N = 0 : เซ็ทเป็น 1 บรรทัดของการแสดงผล
   - F = 0 : เซ็ท CHR = 5x7 Dot
3. DISPLAY ON/OFF
   - D = 0 : DISPLAY OFF
   - C = 0 : CURSOR OFF
   - B = 0 : BLINK OFF
4. ENTRY MODE SET
   - I/D= 1 : เพิ่มค่า counter ขึ้น1
   - S = 0 : NO SHIFT

เมื่อเริ่มเปิดเครื่องทำงานแล้วก็จะต้องส่งคำสั่งควบคุมให้มันเริ่มทำงานดังตารางที่ 1



รูปที่ 3.3 FLOWCHART OF INITIALIZATION

Instruction	Code										Description	Execution time (when fosc is 250 kHz) Note 1	Execution time (when fosc is 160 kHz) Note 2
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	82 μs ~ 1.64 ms	120 μs ~ 4.9 ms
Return home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	40 μs ~ 1.6 ms	120 μs ~ 4.8 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 μs	120 μs
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40 μs	120 μs
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents	40 μs	120 μs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F).	40 μs	120 μs
Set CG RAM address.	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs	120 μs
Set DD RAM address	0	0	1	A_{DD}							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs	120 μs
Read busy flag & address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	1μs	1 μs
Write data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs	120 μs
Read data to CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs	120 μs
	I/D = 1: Increment (+1) I/D = 0: Decrement (−1) S = 1: Accompanies display shift. S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right. R/L = 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5 x 10 dots F = 0: 5 x 7 dots BF = 1: Internally operating BF = 0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM address A_{DD}: DD RAM address Corresponds to cursor address. AC: Address counter used for both of DD and CG RAM address.	Execution time changes when frequency changes. (Example) When fosc is 270 kHz: $40\ \mu s \times \frac{250}{270} = 37\ \mu s$	

*No effect

Notes 1. Applied to models driven by 1/8 duty or 1/11 duty.
2. Applied to models driven by 1/16 duty.

ตารางที่ 1 ชุดคำสั่งของ HD 44780

รายละเอียดของคำสั่ง

1. CLEAR DISPLAY

	RS	R/W	DB_7							DB_0
CODE	0	0	0	0	0	0	0	0	0	1

คำสั่งนี้จะเป็นการเขียนช่องว่างหรือ SPACE (ASCII 20H) เข้าไปใน DD RAM ทั้งหมดและทำการเซ็ท DD RAM ADDRESSER เป็นศูนย์ ตัว CURSOR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพ SET I/D=1 , S ไม่เปลี่ยน

2. RETURN HOME

	RS	R/W	DB_7							DB_0
CODE	0	0	0	0	0	0	0	0	1	*

* No effect

คำสั่งนี้จะทำการเซ็ท DD RAM ADDRESSER เป็นศูนย์ ตัว CURSER จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพ ข้อมูลในจอภาพไม่เปลี่ยน

3. ENTRY MODE SET

	RS	R/W	DB_7							DB_0
CODE	0	0	0	0	0	0	0	1	I/D	S

BIT I/D : โดยจะเป็นตัวกำหนดให้ว่าเมื่อเขียน หรืออ่านข้อมูลแล้วจะทำให้ DD RAM ADDRESSเพิ่มขึ้นหนึ่งหรือลดลงหนึ่งโดย 1=เพิ่ม, 0=ลด

BIT S : เป็นตัวกำหนดแสดงผลโดยถ้า S=1 จะเป็นการใส่ข้อมูลแล้วตัว CURSOR อยู่กับที่ข้อมูลถูกดันไปทางซ้าย ถ้า S=0 ข้อมูลจะอยู่กับที่ ตัว CURSOR จะถูกดันไปทางขวามือ

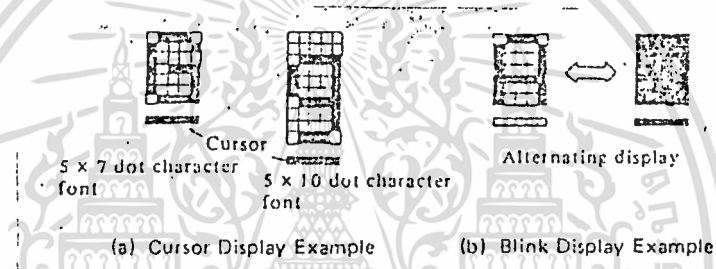
4. DISPLAY ON/OFF CONTROL

	RS	R/W	DB_7							DB_0
CODE	0	0	0	0	0	0	1	D	C	B

BIT D : เป็นบิทให้เปิด-ปิดหน้าจอภาพโดยถ้า D=1 จะ ON และ ถ้า D=0 จะ OFF

BIT C : จะให้แสดง CURSOR ให้บิท C=1 และถ้าไม่ต้องการแสดง CURSOR บิท C=0 โดยตัว CURSOR จะอยู่ที่ LINE 8 ในแบบ 5x7 DOT และจะอยู่ที่ LINE 11 ในแบบ5x10 DOT

BIT B : เป็นบิทเซ็ทการกระพริบของ CURSOR โดย B=1 มีการกระพริบ มีระยะกระพริบประมาณ 379.2 ms



5. CURSOR OR DISPLAY SHIFT

	RS	R/W	DB_7							DB_0
CODE	0	0	0	0	0	1	S/C	R/L	*	*

เป็นคำสั่งกำหนดตำแหน่งให้ CURSOR หรือข้อมูลไปเกิดทางซ้ายหรือขวาโดยไม่ต้องใช้คำสั่งเขียนหรืออ่าน โดย

S/C	R/L	
0	0	ทำการย้าย CURSOR จากตำแหน่งเดิมไปซ้าย
0	1	ทำการย้าย CURSOR จากตำแหน่งเดิมไปขวา
1	0	เป็นการดันตัวอักษรที่เกิดไปทางซ้าย
1	1	เป็นการดันตัวอักษรที่เกิดไปทางขวา

6. FUNCTION SET

	RS	R/W	DB_7							DB_0
CODE	0	0	0	0	1	DL	N	F	*	*

BIT DL : เป็นการ SET การติดต่อว่าจะให้เป็นแบบ 8 บิทหรือ 4 บิท โดยถ้าต้องการติดต่อ 4 บิท DL=0

BIT N : เป็นการเซ็ทบรรทัดแสดงผล N=0 แสดง 1 บรรทัด ถ้า N=1 จะแสดง 2 บรรทัด (หรือมากกว่า)

BIT F : เป็นการเซ็ทขนาด DOT การแสดงผล โดยถ้า F=0 เป็นแบบ 5x7 และ F=1 เป็นแบบ 5x10

N	F	No. of display lines	Character font	Duty factor	Remarks
0	0	1	5 x 7 dots	1/8	
0	1	1	5 x 10 dots	1/11	
1	*	2	5 x 7 dots	1/16	Cannot display 2 lines with 5 x 10 dot character font.

* No effect

7. SET CG RAM ADDRESS

	RS	R/W	DB_7							DB_0
CODE	0	0	0	1	A	A	A	A	A	A

— high order bit low order bit—

ใน HD44780 นั้นจะมีหน่วยความจำอยู่ 2 ชุดคือ DISPLAY DATA RAM (DD RAM) จำนวน 80x8 บิท และ CHARACTER GENERATOR RAM (CG RAM) จำนวน 512 บิท และ 7200 บิท คำสั่งนี้จะเป็นการเซ็ท ADDRESS ใน CG RAM โดยต้องทำการเซ็ท ADDRESS ก่อนเขียนหรืออ่านข้อมูลจาก CG RAM

8. SET DD RAM ADDRESS

	RS	R/W	DB_7							DB_0
CODE	0	0	1	A	A	A	A	A	A	A

— high order bit low order bit—

เป็นคำสั่งเซ็ทค่า ADDRESS ใน DD RAM ในการเขียนหรืออ่านค่าจาก DD RAM (DD RAM คือส่วนที่จะแสดงผลหน้าจอ LCD) โดยจำนวน ADDRESS ที่เกิดขึ้นบนจอจะอยู่กับการเซ็ทค่า N ด้วย

ถ้า N=0 (1บรรทัด) ADDRESS จะอยู่ 00-4FH

ถ้า N=1 (2บรรทัด) ADDRESS จะอยู่ 00-27H สำหรับบรรทัดที่ 1 และ 40H-67H สำหรับบรรทัดที่ 2

ตัวอย่างการจัด ADDRESS ของ DDRAM หน้าจอ LCD แบบ 16 ตัวอักษร 4 บรรทัด

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	-- display position
1-line	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	-- DD RAM address.
2-line	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	
3-line	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	
4-line	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	

HDM-16416H

9. READ BUSY FLAG AND ADDRESS

	RS	R/W	DB_7							DB_0
CODE	0	1	BF	A	A	A	A	A	A	A

— high order bit low order bit—

เป็นคำสั่งอ่านค่า BUSY FLAG ซึ่งจะเป็นตัวบอกว่าตัว HD44780 นี้อยู่ในขบวนการทำงานภายในอยู่ หรืออยู่ในสภาพพร้อมจะรับข้อมูล โดย

BF=1 อยู่ในขบวนการทำงานภายใน ไม่พร้อมจะรับข้อมูล

BF=0 พร้อมที่จะรับข้อมูลหรือคำสั่งได้

นอกจากนี้ยังเป็นคำสั่งอ่านค่าข้อมูล ADDRESS ของ CG RAM,DD RAM

10. WRITE DATA TO CG or DD RAM

	RS	R/W	DB_7							DB_0
CODE	1	0	D	D	D	D	D	D	D	D

— high order bit low order bit—

เป็นคำสั่งเขียนข้อมูลเข้าไปใน CG หรือ DD RAM โดยเมื่อเขียนข้อมูลและ ADDRESS จะเพิ่มหรือลดโดยอัตโนมัติตามคำสั่งที่เซ็ทใน ENTRY MODE ข้อกำหนดที่จะรู้ว่าเป็นการเขียนข้อมูลของ CG RAM หรือ DD RAM ทำได้โดยการเซ็ท ADDRESS ของ CG หรือ DD RAM ขึ้นมาก่อนจะเขียนข้อมูล

11. READ DATA FROM CG or DD RAM

	RS	R/W	DB_7							DB_0
CODE	1	1	D	D	D	D	D	D	D	D

— high order bit low order bit—

เป็นคำสั่งอ่านข้อมูลจาก CG หรือ DD RAM โดยก่อนอ่านค่าจาก DD หรือ CG RAM นี้ควรจะใช้คำสั่งเซ็ท ADDRESS ก่อนเพื่อให้รู้ว่าข้อมูลที่อ่านได้นั้นเป็น DD หรือ CG RAM

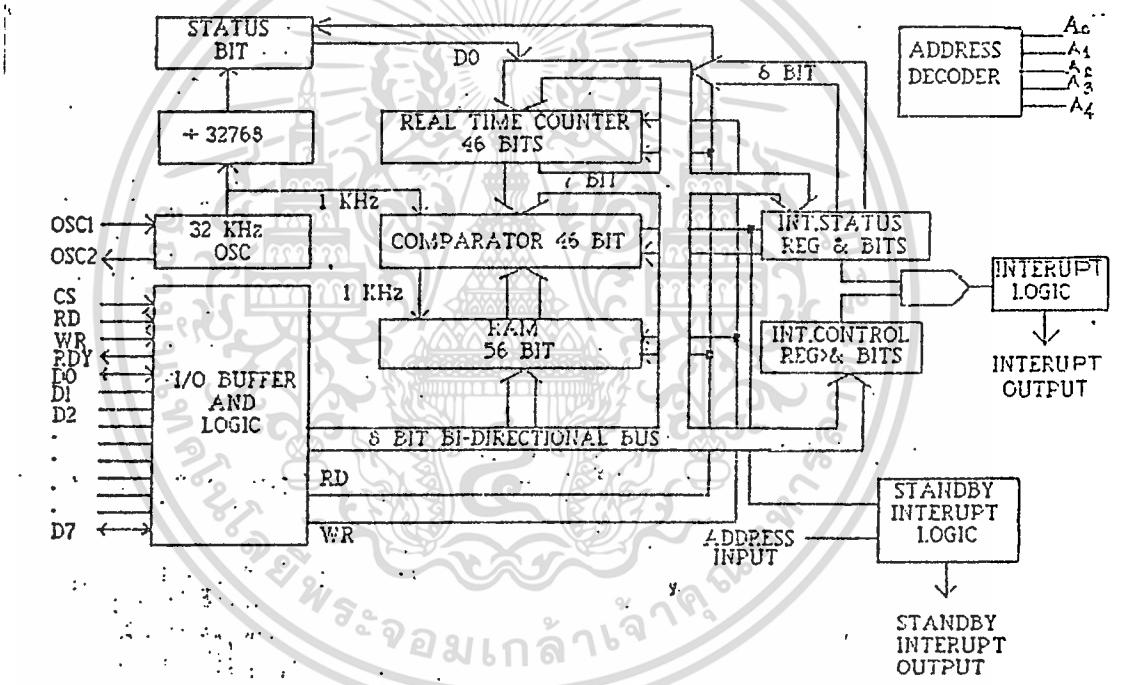
3.1.2 ส่วน REAL TIME CLOCK (RTC)

RTC #MM58167 นี้เปรียบเสมือนอินพุท-เอาท์พุทพอร์ท ซึ่งสามารถอ่านและเขียนไปที่พอร์ทนั้นได้

จากบล็อกไดอะแกรมในรูปที่ 3.4 มีส่วนประกอบที่สำคัญ ๆ ก็คือ

REAL TIME COUNTER

เป็นตัวนับและจัดการเกี่ยวกับเวลา ถูกแบ่งเป็นดิจิทละ 4 บิท ซึ่งการเข้าถึง REAL TIME COUNTER จะกระทำครั้งละ 2 ดิจิท (ในขณะ READ และ WRITE) ซึ่งแต่ละดิจิทจะให้ค่า BCD ดังแสดงในตารางที่ 2 บิทที่ไม่ใช้จะถูก HOLD ด้วยลอจิก 0 ซึ่งเราไม่ต้องสนใจในขณะที่ทำการเขียนข้อมูลลงบน DATA BUS เหตุที่บางบิทไม่ใช้ก็เนื่องจากว่า ไม่จำเป็นต้องใช้ในการให้ข้อมูลแบบ BCD ของบางหลัก ตัวอย่างเช่นในหลักสิบของชั่วโมงจะไม่เกินเลข 2 ฉะนั้นเราจะใช้เพียง 2 บิทเท่านั้น ไม่ต้องใช้ในบิทที่ 6 และ 7



รูปที่ 3.4 BLOCKDIAGRAM OF MM58167

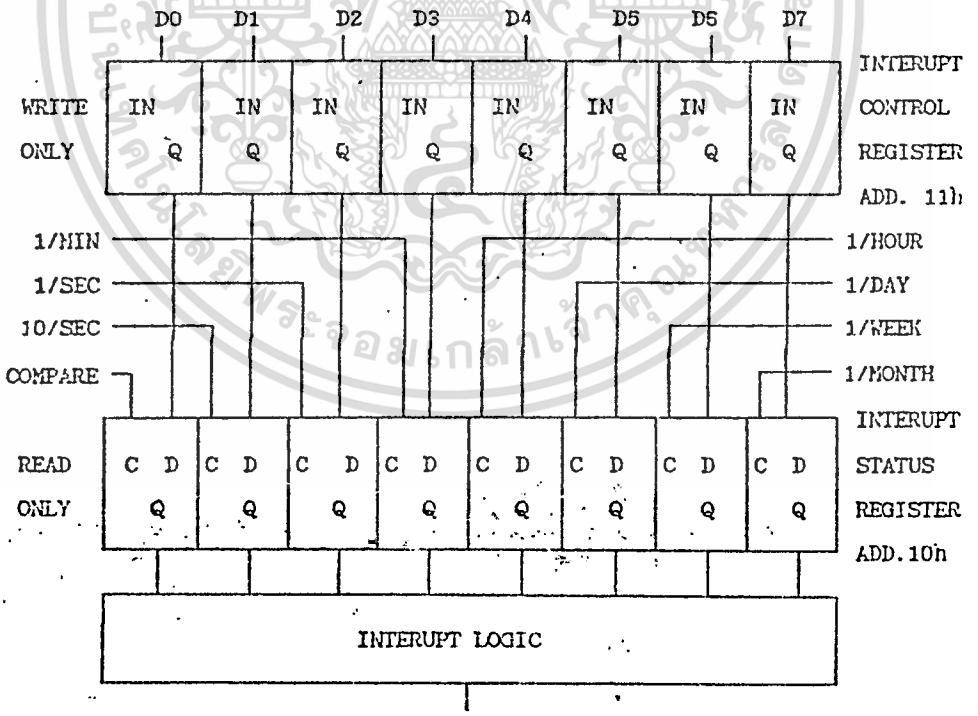
RAM

MM58167 มี RAM ขนาด 56 บิทซึ่งใช้ในการเก็บข้อมูลเมื่อไฟตกหรือใช้เก็บข้อมูลการตั้งปลุกเพื่อที่จะเปรียบเทียบกับ REAL TIME COUNTER ข้อมูลใน RAM จะสามารถเปรียบเทียบกับ REAL TIME COUNTER และมีดิจิทที่ไม่ใช้คือ หลักหน่วยของ 1/10,000 ของ SEC หลักสิบของวันในสัปดาห์

COUNTER ADDRESS		UNIT D0	D1	D2	D3	MAX BCD CODE	TENS D4	D5	D6	D7	MAX BCD CODE
1/10,000 OF SEC.	(00h)	-	-	-	-	.	D4	D5	D6	D7	9
HUNDREDTHS & TENS SEC	(01h)	D0	D1	D2	D3	9	D4	D5	D6	D7	9
SECOND	(02h)	D0	D1	D2	D3	9	D4	D5	D6	-	5
MINUTE	(03h)	D0	D1	D2	D3	9	D4	D5	D6	-	5
HOURS	(04h)	D0	D1	D2	D3	9	D4	D5	-	-	2
DAY OF THE WEEK	(05h)	D0	D1	D2	-	7	-	-	-	-	0
DAY OF THE MONTH	(06h)	D0	D1	D2	D3	9	D4	D5	-	-	3
MONTH	(07h)	D0	D1	D2	D3	9	D4	-	-	-	1

ตารางที่ 2

RAM จะถูกกำหนดให้มีรูปแบบเหมือนกับ REAL TIME COUNTER แต่อย่างไรก็ตามยังมีบิทที่ไม่ใช้อยู่ซึ่งบิทที่ไม่ใช้ใน REAL TIME COUNTER นี้จะทำการ COMPARE กับ 0 ใน RAM



รูปที่ 3.5 INTERUPT CONTROL REGISTER & STATUS

INTERUPT and COMPARATOR

มีสัญญาณอินเตอร์รัพท์อยู่ 2 อย่าง, อย่างแรกคือ INTERUPT OUTPUT (ACTIVE HI) OUTPUT นี้สามารถจะโปรแกรมให้เกิดสัญญาณออกได้ถึง 8 อย่าง คือ 10 Hz , 1 Hz , 1 นาที/ครั้ง , 1 ชม./ครั้ง , 1 วัน/ครั้ง, 1 สัปดาห์/ครั้ง , 1 เดือน/ครั้ง และเมื่อ RAM กับ REAL TIME COUNTER เกิดการเปรียบเทียบกันขึ้น

วิธีการที่จะ ENABLE สัญญาณ INTERUPT คือให้ลอจิก 1 แก่ INTERUPT CONTROL REGISTER ในบิทที่ตรงกับความถี่ที่เราต้องการจะให้เกิดสัญญาณอินเตอร์รัพท์ ดูรูปที่ 3.5ประกอบ เช่นต้องการให้สัญญาณอินเตอร์รัพท์ทุกๆ 1 วินาที ก็ให้ D2 เป็น 1 เขียนไปที่ INTERUPT CONTROL REGISTER

เราสามารถเซ็ท INTERUPT CONTROL REGISTER ครั้งละ 1 บิท หรือมากกว่านั้นก็ได้ ตัวอย่างเช่นต้องการให้มีสัญญาณอินเตอร์รัพท์ทุกวินาที และทุกนาที ก็เซ็ทบิทที่ 3 กับบิทที่ 2 ในที่นี้คือ 0CH โดยเขียนไปที่ ADDRESS ของ INTERUPT CONTROL REGISTER (ดูตารางที่ 3)

เมื่อเวลานับมาถึงค่าสูงสุดของแต่ละภาคจะทำให้เกิด CLOCK ให้กับ INTERUPT CONTROL REGISTER ซึ่งจะทำให้INTERUPT OUTPUT เป็น HIGH การอ่าน INTERUPT CONTROL REGISTERทำให้เราทราบว่าสัญญาณอินเตอร์รัพท์เป็นสัญญาณของบิทใด อีกทั้งยังเป็นการรีเซ็ท INTERUPT CONTROL REGISTER อีกด้วย การอ่าน INTERUPT CONTROL REGISTER นี้จะได้ข้อมูลบน DATA BUS ซึ่งประกอบด้วยบิทที่ทำให้เกิดการอินเตอร์รัพท์โดยจะให้ค่าเป็น 1 ที่บิทนั้นหลังจาก CYCLE ของการอ่านจะทำให้ INTERUPT CONTROL REGISTER ถูกรีเซ็ท

อินเตอร์รัพท์อีกอย่างหนึ่งคือ STANDBY INTERUPT (OPEN DRAIN ACTIVE LOW) อินเตอร์รัพท์ตัวนี้จะเกิดขึ้นเมื่อเราได้ทำการ ENABLE ไว้ และเกิดการเปรียบเทียบใน RAM กับ REAL TIME COUNTER การ ENABLE ทำได้โดยเขียน 01H ไปที่ ADDRESS 16H ในทางตรงกันข้ามถ้าให้ 00 ที่ ADDRESS 16H จะเป็นการ DISABLE

POWER DOWN MODE

ขา POWER DOWN เป็น CHIP SELECT ที่สำคัญตัวที่สอง มันจะ DISABLE สัญญาณออกทั้งหมดยกเว้น สัญญาณ STANDBY INTERUPT เมื่อขา POWER DOWN ได้รับลอจิก 0, MM58167 จะไม่ตอบสนองต่อสัญญาณจากภายนอก แต่นาฬิกาจะยังเดินตามปกติ และจะยังให้สัญญาณ STANDBY INTERUPT ถ้าได้มีการโปรแกรมให้ขานี้ทำงานไว้ก่อนแล้ว

เมื่อต้องการเปลี่ยนจากโหมดการทำงานปกติมาเป็นSTANDBY MODE ควรจะให้ขา POWER DOWN เป็นลอจิก 0 อย่างน้อย 1 us ก่อนที่จะทำการลดระดับลงมาเป็น STANDBY MODE

เมื่อต้องการเปลี่ยนกลับมาสู่การทำงานปกติ ผู้ใช้ต้องแน่ใจว่าขาอินพุทอื่น ๆ ต้องเป็นสัญญาณที่ถูกต้องก่อนที่จะกลับมาสู่โหมดการทำงานปกติ ทั้งนี้เพื่อป้องกันข้อมูลของนาฬิกาเสียไป จะทำให้นาฬิกาเดินผิด ตัวอย่างนี้ได้แก่การที่ขา CS,RD,WR ของ MM58167 มีสัญญาณเปลี่ยนแปลงในขณะที่กลับสู่โหมดปกติ จะทำให้มีการเขียนข้อมูลไปที่ REAL TIME COUNTER หรือ ใน RAM

COUNTER and RAM RESET ; GO COMMAND

ตัวนับเวลา และ RAM สามารถรีเซ็ทได้โดยเขียน FFH ที่ADDRESS 12H,13H ตามลำดับ การให้ PULSE ของการเขียนไปที่ ADDRESS 15H จะรีเซ็ทตัวนับของวินาที ขณะทำการเขียนไปที่ ADDRESS 15H นี้ MM58167 จะไม่สนใจข้อมูลบน DATA BUS แต่ผลของคำสั่ง GO มีดังนี้

ถ้าตัวนับของวินาทีนับได้มากกว่า 39 เมื่อเราใช้คำสั่ง GO จะทำให้หลักของนาทีเพิ่มขึ้น ในกรณีอื่น ๆ จะไม่มีผลต่อหลักนาที

STATUS BIT

STATUS BIT จะบอกผู้ใช้ว่าขณะที่ทำการอ่านตัวนับนั้น ตัวนับกำลังอยู่ในช่วงของการ UPDATE เวลาข้อมูลที่อ่านได้อาจมีการผิดพลาดเกิดขึ้นSTATUS BIT นี้จะอ่านได้จาก ADDRESS 14H ของ RTC โดยจะให้ลอจิก 1 ที่บิท 0 ของ DATA BUS ในขณะที่บิทอื่น ๆ เป็น 0 หมด ถ้าสัญญาณนี้ปรากฏขึ้นภายหลังการอ่านตัวนับควรมีการอ่านตัวนับใหม่ ที่ขอบขาลงของสัญญาณ RD ที่ ADDRESS 14H จะรีเซ็ท STATUS BIT ด้วย

OSCILLATOR

OSCILLATOR เป็นออสซิลเลเตอร์แบบเรโซแนนท์ขนาน โดยใช้อุปกรณ์ภายนอกเพียง คอนเดนเซอร์ 1 ตัว, ตัวต้านทาน 1 ตัว, X-TAL 1 ตัว โดยความต้านทานจะต่ออยู่ระหว่างขั้ว OSC IN (ขา 10)และ OSC OUT(ขา11) เพื่อที่จะ BIAS ตัวอินเวอเตอร์ที่อยู่ภายในให้ทำงานในช่วงที่เป็นเชิงเส้น สำหรับแร่แบบ MICRO POWER CRYSTAL จะใช้ความต้านทานต่ออนุกรมกับขา OSC OUT โดยใช้ความต้านทานมีค่า 200 K ส่วนคอนเดนเซอร์โดยปกติจะใช้ประมาณ25pf X-TAL ที่ใช้มีค่า 32.768 KHz

CONTROL LINE

สัญญาณ RD, WR, CS เป็นสัญญาณอินพุทลิจิก 0 และสัญญาณ READY เป็นสัญญาณออก (OPEN DRAIN OUTPUT) ที่จุดเริ่มต้นของการอ่าน หรือ การ

เขียนขา READY จะให้สัญญาณ OUTPUT เป็น 0 อยู่จนกระทั่งข้อมูลปรากฏบน DATA BUS เรียบร้อย หรือข้อมูลได้ถูก LATCH ไว้แล้วในขณะช่วงการเขียน

TEST MODE

ในโหมดนี้ใช้เป็นเพียงการทดสอบ RTC CHIP ให้ทำงานที่ความถี่สูงกว่าการทำงานปกติ ในโหมดนี้ความถี่ 32 KHz จะถูกต่อตรงเข้าที่ 1/1000 s ขา CS และ WR ต้องเป็น LOW และให้ ADDRESS เป็น 1FH

A4	A3	A2	A1	A0	FUNCTION
0	0	0	0	0	COUNTER THOUSANDTHS OF SECONDS
0	0	0	0	1	COUNTER HUNDREDTHS AND TENTHS OF SECOND
0	0	0	1	0	COUNTER SECONDS
0	0	0	1	1	COUNTER MINUTES
0	0	1	0	0	COUNTER HOURS
0	0	1	0	1	COUNTER DAY OF WEEK
0	0	1	1	0	COUNTER DAY OF MONTH
0	0	1	1	1	COUNTER MONTH
0	1	0	0	0	RAM THOUSANDTHS OF SECONDS
0	1	0	0	1	RAM HUNDREDTHS AND TENTHS OF SECOND
0	1	0	1	0	RAM SECONDS
0	1	0	1	1	RAM MINUTES
0	1	1	0	0	RAM HOURS
0	1	1	0	1	RAM DAY OF WEEK
0	1	1	1	0	RAM DAY OF MONTH
0	1	1	1	1	RAM MONTH
1	0	0	0	0	INTERUPT STATUS REGISTER
1	0	0	0	1	INTERUPT CONTROL REGISTER
1	0	0	1	0	COUNTER RESET
1	0	0	1	1	RAM RESET
1	0	1	0	0	STATUS BIT
1	0	1	0	1	"GO" COMMAND
1	0	1	1	0	STANDBY INTERUPT
1	1	1	1	1	TEST MODE

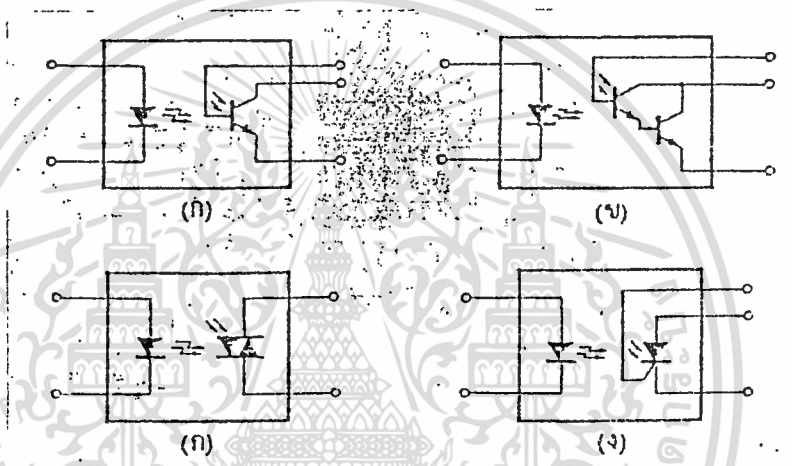
ตารางที่ 3 แสดง ADDRESS ที่ต้องเซ็ทค่าสั่งควบคุม

3.1.3 ส่วน LOAD DRIVER

- ตัวเชื่อมโยงทางแสง (OPTOCOUPLER)

เป็นอุปกรณ์เดี่ยวที่ประกอบด้วยแหล่งกำเนิดแสง และตัวตรวจรับแสง โดยที่ทั้งสองชิ้นส่วนนี้แยกจากกันและกัน โดยมีฉนวนที่โปร่งใสขั้นกลาง และชิ้นส่วนทั้งหมดจะถูกบรรจุอยู่ในตัวถังทึบแสง

แหล่งกำเนิดแสงสำหรับตัวเชื่อมโยงทางแสง ส่วนมากแล้วจะใช้ไดโอดเปล่งแสงอินฟราเรด ที่ทำจากสารแกลเลียม อาร์เซไนด์ (GaAs) ส่วนตัวตรวจจับหรืออุปกรณ์ทางภาคเอาท์พุทนั้น อาจจะเป็นโฟโต้ทรานซิสเตอร์, โฟโต้ดาร์ลิงตัน, SCR ที่ถูกกระตุ้นด้วยแสงและสวิทช์สองทางที่ทำงาน เมื่อมีแสงมากระตุ้น ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 ตัวเชื่อมโยงทางแสงทั่ว ๆ ไป

- ส่วนเชื่อมต่อกับเครื่องใช้ไฟฟ้า

เนื่องจากวงจรควบคุมไม่สามารถจ่ายกำลังงานสูง ๆ แก่เครื่องใช้ไฟฟ้าได้โดยตรง จึงต้องใช้วงจรส่วนนี้ทำหน้าที่จ่ายกำลังงานแก่เครื่องใช้ไฟฟ้าเหล่านั้นโดยการควบคุมจากสัญญาณควบคุมที่ส่งมาจากชิงเกิลชิพซึ่งเป็นสัญญาณลอจิก "1" เมื่อต้องการให้เครื่องใช้ไฟฟ้าทำงานสัญญาณลอจิก 1 นี้จะนำไปขับออพโต้ไดแอคเพื่อต้องการแยกส่วนควบคุม ซึ่งเป็นสัญญาณแรงดันต่ำ จากแหล่งจ่ายไฟที่จ่ายให้แก่เครื่องใช้ไฟฟ้า จากสัญญาณลอจิก 1 จะได้เอาท์พุทเป็นกระแสสลับ ใช้เป็นสัญญาณทริกให้กับขาเกทของไตรแอค เพื่อให้ไตรแอคทำงานและจ่ายกระแสไฟให้กับเครื่องใช้ไฟฟ้าได้

จากการคำนวนการจ่ายพลังงานแก่เครื่องใช้ไฟฟ้า คิดได้จากกำลังวัตต์ของเครื่องใช้ไฟฟ้านั้น เพื่อคำนวนขนาดของกระแสที่จะผ่านไตรแอค

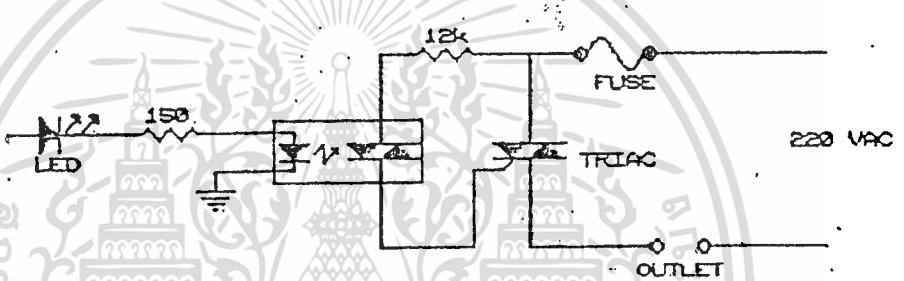
จาก $P = V \times I$

จะได้ $I = P/V$

โดย P คือ กำลังวัตต์ของเครื่องใช้ไฟฟ้า (วัตต์)

V คือ ความต่างศักย์ที่ใช้ (220 โวลท์)

I คือ กระแสที่จะไหลผ่านไตรแอค และไตรแอคจะต้องทนกระแสได้เป็นอย่างน้อย



รูปที่ 3.7 แสดงวงจรที่ใช้เชื่อมต่อกับเครื่องใช้ไฟฟ้า

บทที่ 4

การทำงาน

วงจรนี้ใช้ cpu เบอร์ 8031 ซึ่งเป็นตัวหลักในการทำงานของวงจรดังแสดงดังรูป

การทำงานทั้งหมดถูกกำหนดด้วยซอฟแวร์ หรือโปรแกรมภาษาเครื่อง ซึ่งสามารถอธิบายการทำงานได้ด้วย flow chart ดังรูป

หลังจากเปิดเครื่องแล้วจะทำการหน่วงเวลาเอาไว้ประมาณ 30 ms. เพื่อให้ภาคจ่ายไฟเสถียรภาพเสียก่อน จึงจะให้เริ่มการทำงาน จากนั้นจะทำการกำหนดค่าเริ่มต้นให้กับ LCD และ RTC เพื่อกำหนดโหมดการทำงานตามที่ต้องการ แล้วจึงแสดง title แนะนำผู้ร่วมงาน เสร็จแล้วจะเข้าโหมด time

ในโหมด time นี้จะแสดง วัน วันที่ เดือน และเวลาในปัจจุบันขณะเดียวกันก็จะทำการปิด-เปิด เอาท์พุทตามโปรแกรมต่างๆ ที่ได้ตั้งเอาไว้ หลักการของโหมดนี้คือ จะนำข้อมูลในรีจิสเตอร์ที่เก็บเวลาของนาฬิกา และสภาวะเอาท์พุทมาทำ การถอดรหัสแล้วส่งไปแสดงผล นอกจากนี้ยังสามารถรับข้อมูลจากคีย์เพื่อเปิด-ปิด เอาท์พุทได้โดยตรงอีกด้วย ถ้ากดคีย์ mode อีกจะเข้าสู่โหมด LIST

ในโหมด LIST นี้ จะสามารถดูโปรแกรมในแต่ละ channel ได้อย่างอิสระ และถ้ากด key mode อีก จะเข้าสู่โหมด set time

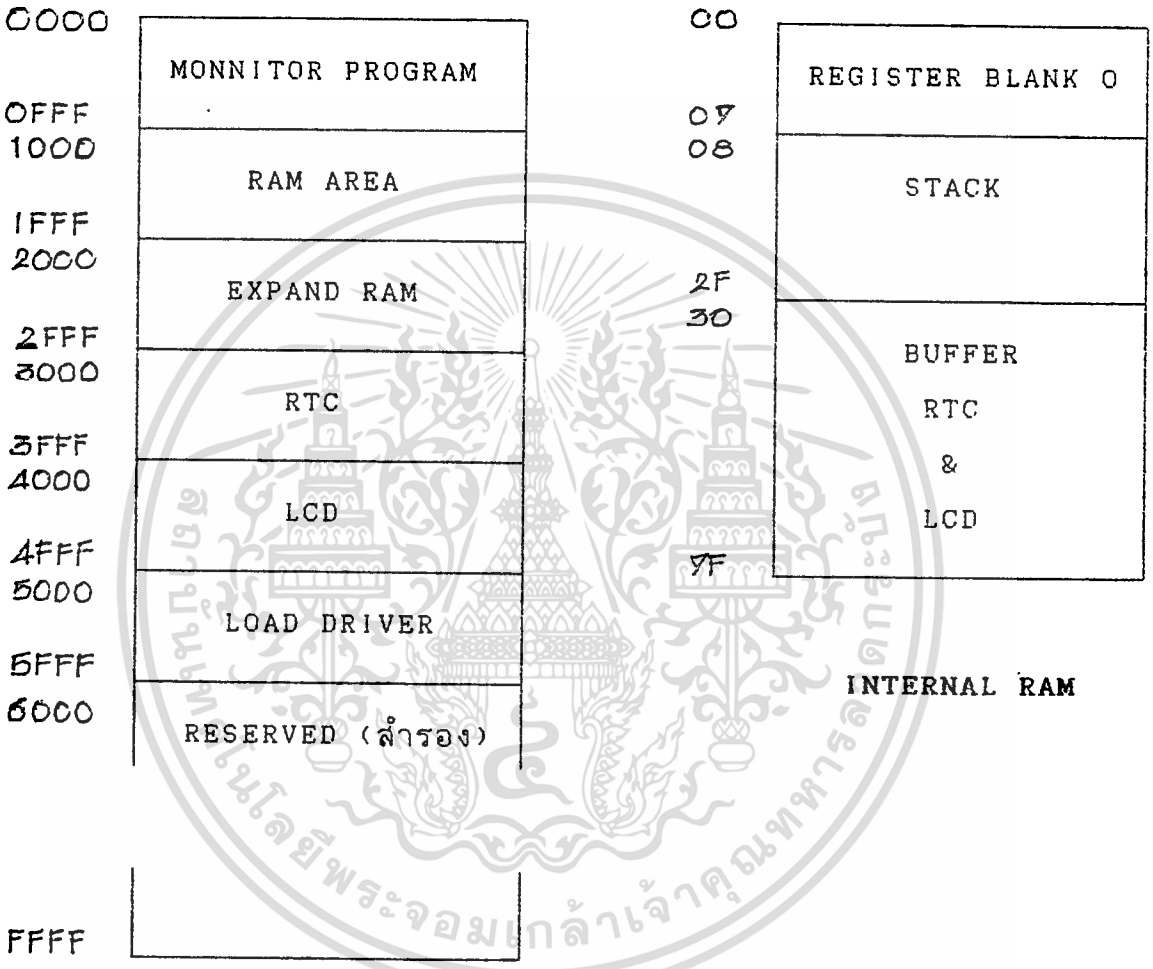
ในโหมด set time นี้ จะทำการตั้งวันเวลาได้ตามต้องการ โดยจะ set ค่า register ควบคุมใน RTC ใหม่ แล้วเก็บทับข้อมูลเดิม ก็จะได้เวลาใหม่ตามที่ต้องการ ถ้ากดโหมดอีกครั้งหนึ่ง จะเข้าสู่โหมดสุดท้ายคือโหมด program โหมดนี้จะทำการตั้ง program ปิดเปิด output ตามที่ต้องการ

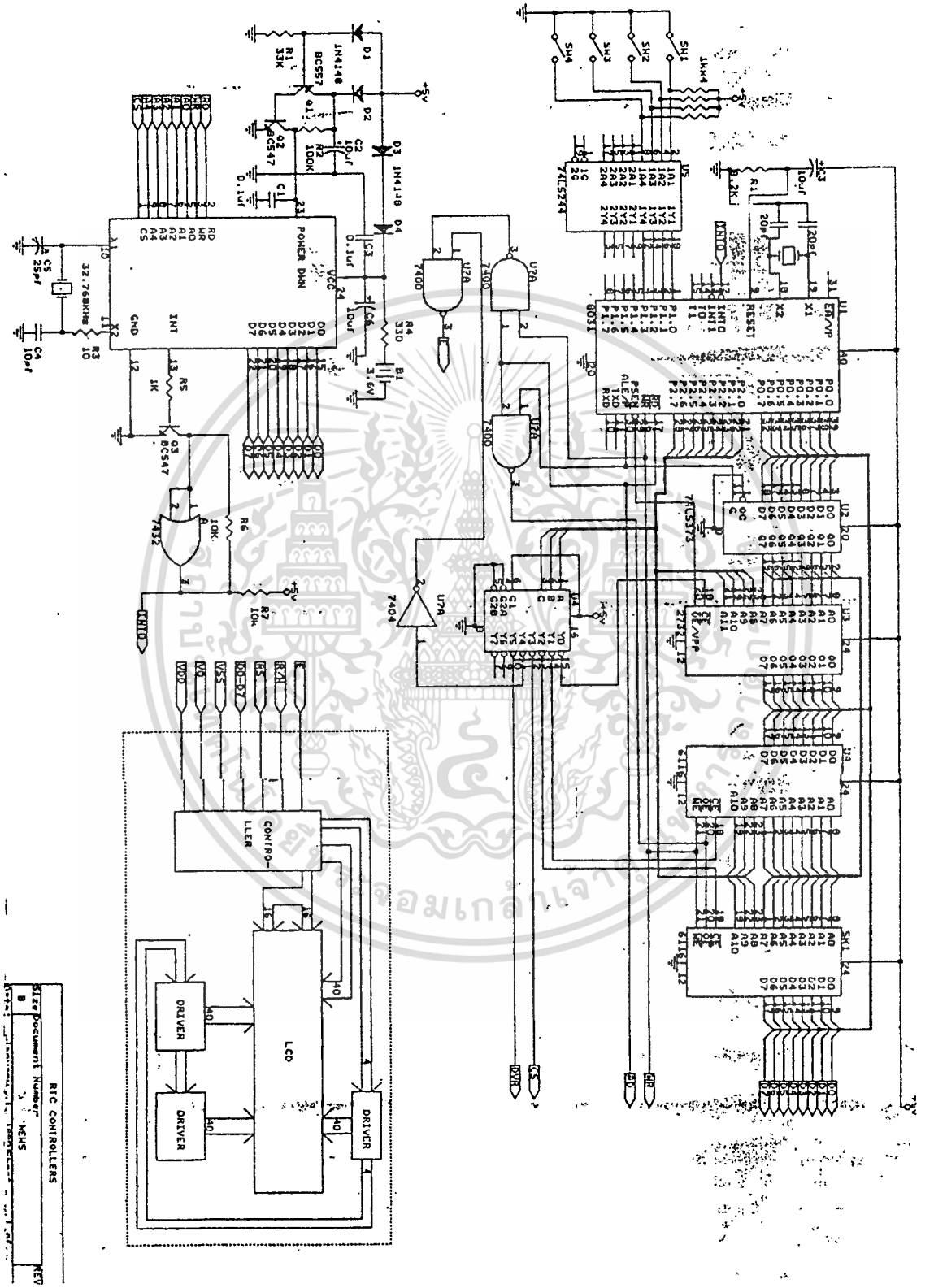
การใช้เครื่อง

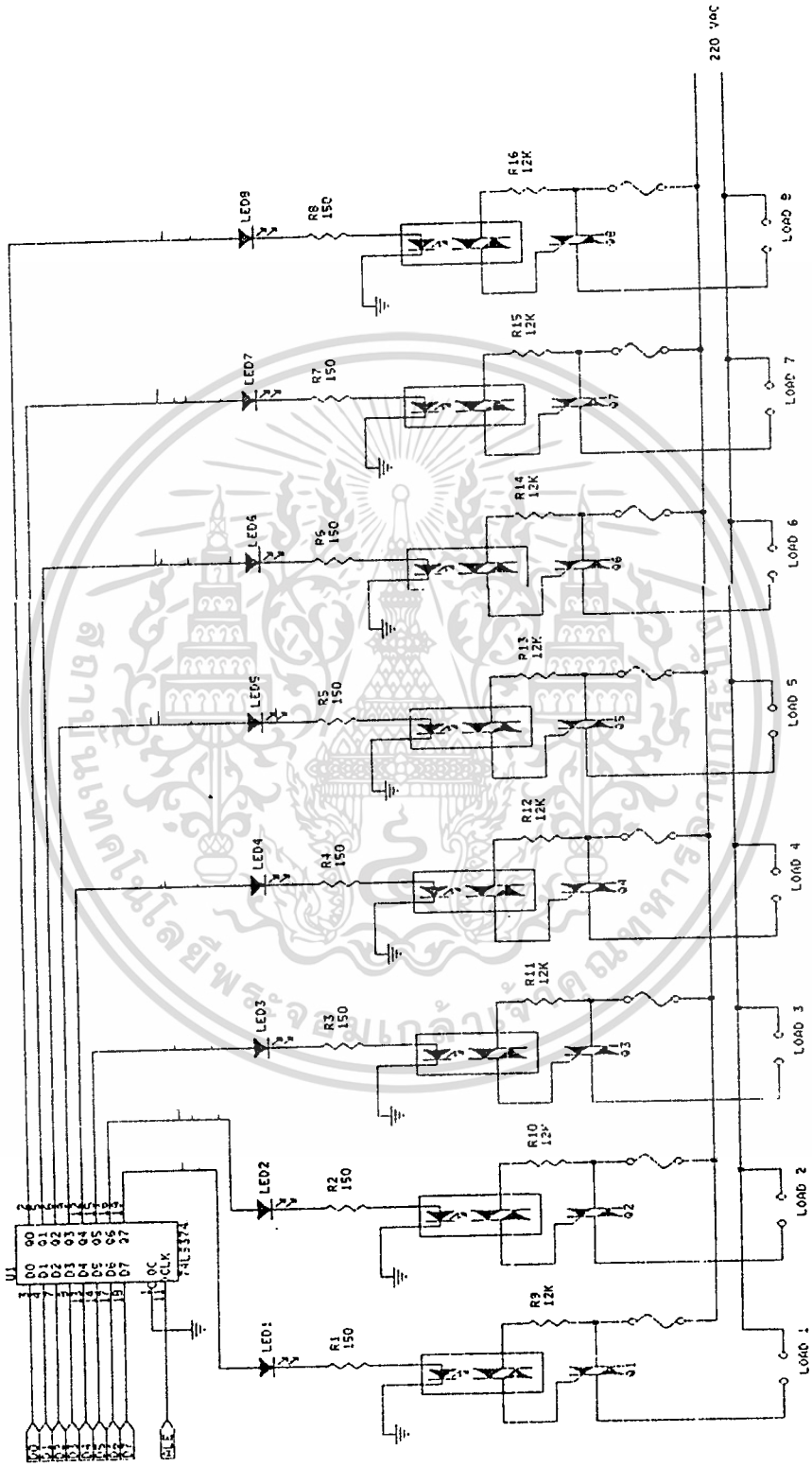
หลังจากเปิดเครื่องแล้ว LCD จะแสดง title ต่างๆ ที่เราออกแบบไว้(ในส่วนนี้อาจพัฒนาเพิ่มเติมโดยออกแบบ display ให้แสดงถึงวิการใช้เครื่องก็ได้) หลังจากนั้นจะเข้าสู่โหมด time , list , set time และ program ตามลำดับ ถ้าเรากด key mode ส่วนวิธีการ program หรือตั้งเวลานั้น จะใช้ key inc , dec เป็นตัวเปลี่ยนตัวเลขไปเรื่อยๆ จนกว่าจะได้ค่าที่ต้องการ จากนั้นกด key enter เพื่อรับข้อมูลใหม่เข้ามา

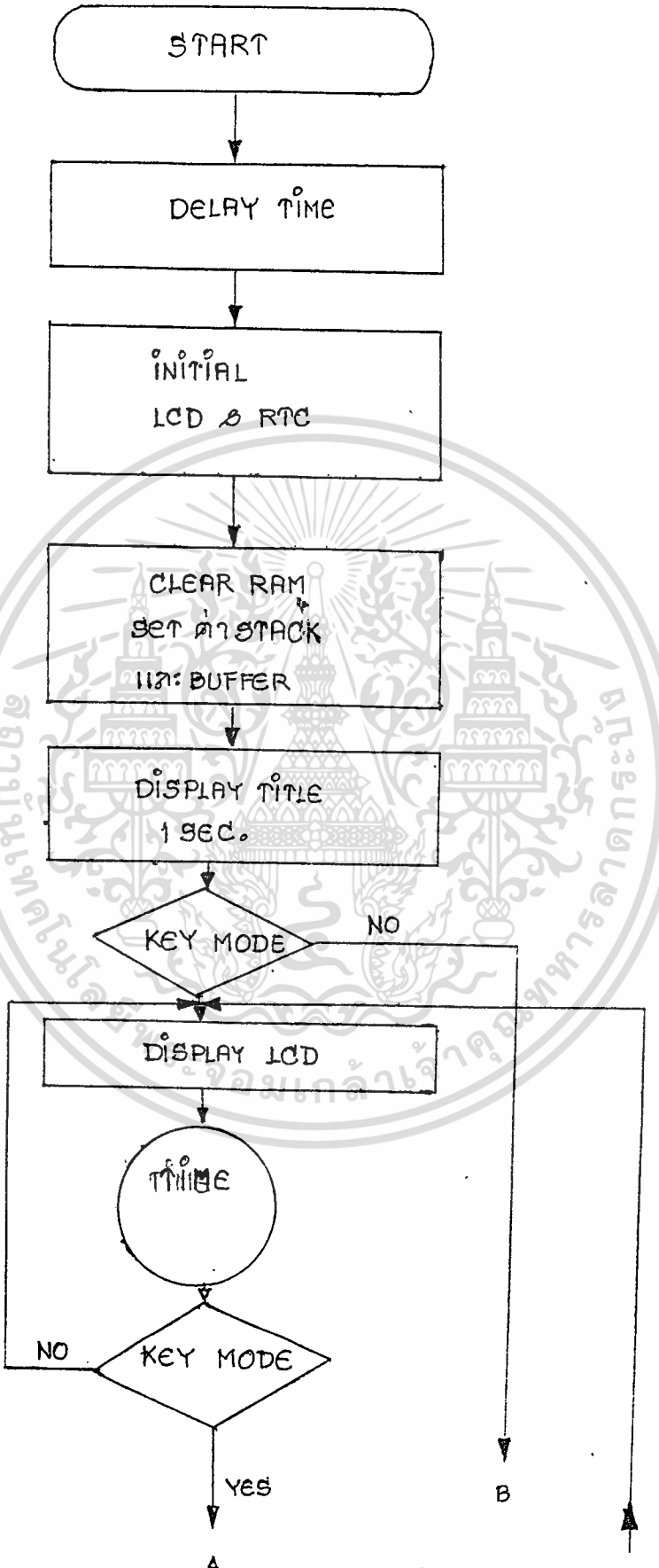
การใช้งานส่วนใหญ่จะ display ให้เห็นที่หน้าจอ จึงง่ายต่อการใช้เครื่อง อย่างไรก็ตาม การพัฒนา program ในส่วนนี้ ยังเป็นสิ่งที่ต้องการอยู่เพื่อให้ project นี้สมบูรณ์ยิ่งขึ้น

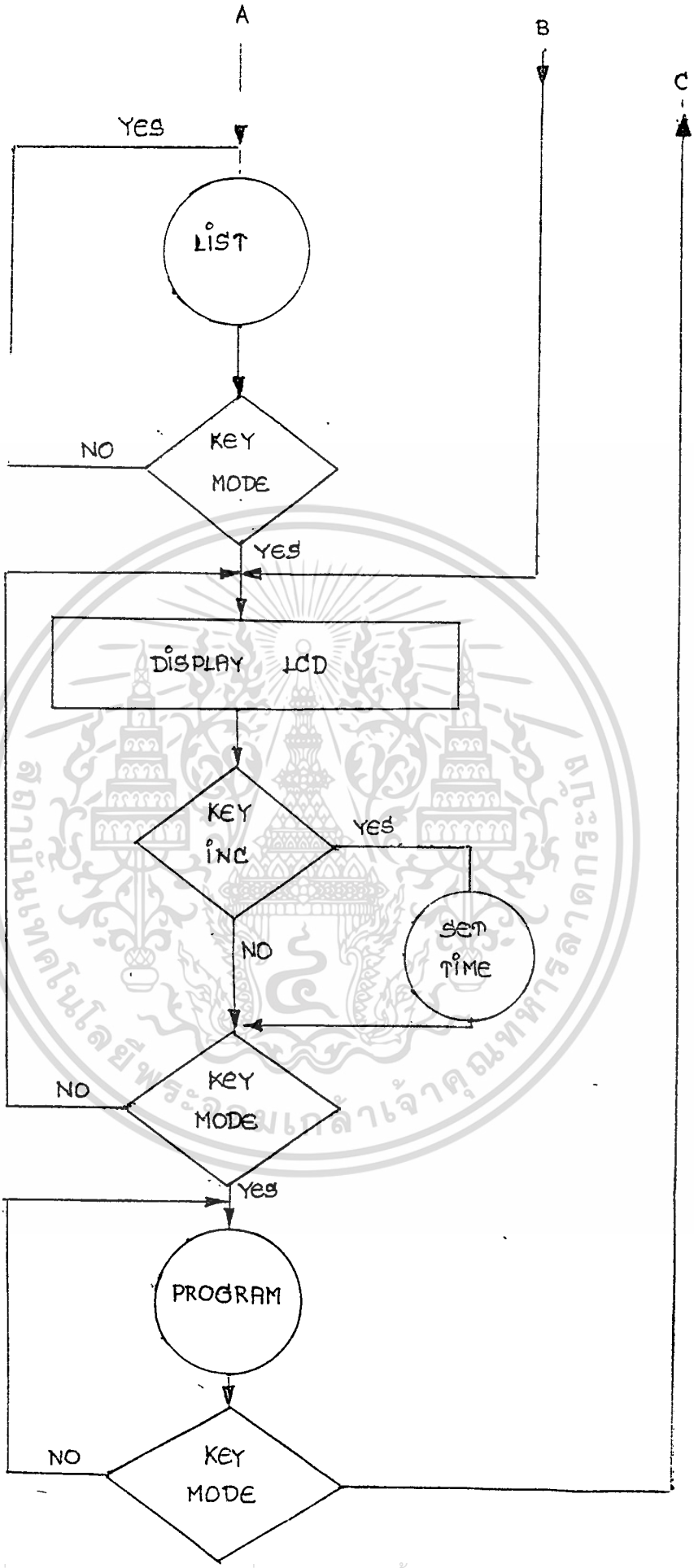
การจัดวางตำแหน่ง MEMORY











บทที่ 5
สรุปและวิจารณ์

จากการทำ project ชิ้นนี้ได้พบกับปัญหามากมาย ซึ่ง เป็นอุปสรรค ต่อการทำงานเป็นอย่างยิ่ง อาทิเช่นวงจร hard ware ทำงานไม่เสถียรภาพอัน เนื่องมาจากการบัดกรีไม่แน่นพอ อีกทั้งยังมีปัญหาการ interface ระหว่าง แรม 2 ทางกับ project ซึ่งในตอนแรกไม่สามารถติดต่อกันได้ทำให้การพัฒนาโปรแกรม ในช่วงนั้นต้องล่าช้าต่อมา ได้วิเคราะห์ปัญหานี้ อย่างละเอียด จึงพบว่าการใช้แรม 2 ทางแทน eaprom บน project จะใช้การดีโค้ดเพื่อเลือกชิพด้วยไอซี 74138 ไม่ได้ต้องใช้สัญญาณ psen ที่อยู่บนชิพ 8031 แทนและในการ link ข้อมูลระหว่าง et.board กับเครื่อง pc ก็มีปัญหาในเรื่องระดับของสัญญาณไม่เท่ากัน คือเอาท์พุท ของเครื่อง pc เป็น 12v แต่ของ et.board เป็น 5v จึงทำให้ไม่สามารถส่ง ข้อมูลติดต่อกันได้จึงต้องแก้ปัญหาด้วยการใส่ buffer เบอร์ 1488 เพื่อแปลง สัญญาณจากเครื่อง pc ให้อยู่ในระดับเดียวกันกับ et.board ปัญหาสำคัญที่ติดอยู่ ในตอนนี้คือ สัญญาณ interrupt ที่ใช้มีขอบขาของสัญญาณไม่ตรงกับ timing ของ cpu ดังนั้นเมื่อมีสัญญาณ interrupt มาแล้ว cpu จะไม่ไปทำโปรแกรม interrupt ที่ display จึงไม่ปรากฏ time ให้เห็นปัญหา อีกอย่างหนึ่งคือ โปรแกรมรับ key ยังไม่สมบูรณ์ต้องพัฒนาเพิ่มเติมขึ้นไปอีก

จากปัหาต่างๆเหล่านี้ผู้จัดทำคิดว่า น่าจะเป็นประโยชน์ สำหรับผู้ที่จะ พัฒนาวงจรนี้ให้ดียิ่งขึนต่อไป

อนึ่งจากเวลาเท่าที่มีอยู่ เราได้เขียนโปรแกรม ซึ่งใช้งานในระดับ หนึ่งเท่านั้น ซึ่งยังต้องการพัฒนาโปรแกรมให้อยู่ในขั้นที่สมบูรณ์ยิ่งขึ้นต่อไป

```
                        ;*****************************************************
                        ;*                                                   *
                        ;*  THESE ARE SUBROUTINES FOR THE 8051 MICROCONTROLLER *
                        ;*                                                   *
                        ;*****************************************************
0000                            CPU     "8051.TBL"
0000                            HOF     "INT8"
                        ;
                        ;MCS-51 INTERNAL REGISTERS
                        ;
00F0 =          B:      EQU     0F0H    ;B REGISTER
00E0 =          ACC:    EQU     0E0H    ;ACCUMULATOR
00D0 =          PSW:    EQU     0D0H    ;PROGRAM STATUS WORD
00B8 =          IPC:    EQU     0B8H    ;INTERRUPT PRIORITY
00B0 =          P3:     EQU     0B0H    ;PORT 3
00A8 =          IEC:    EQU     0A8H    ;INTERRUPT ENABLE
00A0 =          P2:     EQU     0A0H    ;PORT 2
0099 =          SBUF:   EQU     99H     ;SEND BUFFER
0098 =          SCON:   EQU     98H     ;SERIAL CONTROL
0090 =          P1:     EQU     90H     ;PORT 1
008D =          TH1:    EQU     8DH     ;TIMER 1 HIGH
008C =          TH0:    EQU     8CH     ;TIMER 0 HIGH
008B =          TL1:    EQU     8BH     ;TIMER 1 LOW
008A =          TL0:    EQU     8AH     ;TIMER 0 LOW
0089 =          TMOD:   EQU     89H     ;TIMER MODE
0088 =          TCON:   EQU     88H     ;TIMER CONTROL
0087 =          PCON:   EQU     87H     ;POWER CONTROL REGISTER
0083 =          DPH:    EQU     83H     ;DATA POINTER HIGH
0082 =          DPL:    EQU     82H     ;DATA POINTER LOW
0081 =          SP:     EQU     81H     ;STACK POINTER
0080 =          P0:     EQU     80H     ;PORT 0
0000 =          R0:     EQU     00H     ;REGISTER0
0001 =          R1:     EQU     01H     ;REGISTER1
0002 =          R2:     EQU     02H     ;REGISTER2
0003 =          R3:     EQU     03H     ;REGISTER3
0004 =          R4:     EQU     04H     ;REGISTER4
0005 =          R5:     EQU     05H     ;REGISTER5
0006 =          R6:     EQU     06H     ;REGISTER6
0007 =          R7:     EQU     07H     ;REGISTER7
0020 =          R8:     EQU     20H
0021 =          R9:     EQU     21H
                        ;
                        ;MCS-51 INTERNAL BIT ADDRESSES
                        ;
00D7 =          CY:     EQU     0D7H    ;CARRY FLAG
00D6 =          AC:     EQU     0D6H    ;AUXILIARY-CARRY FLAG
00D5 =          F0:     EQU     0D5H    ;USER FLAG 0
00D4 =          RS1:    EQU     0D4H    ;REGISTER SELECT MSB
00D3 =          RS0:    EQU     0D3H    ;REGISTER SELECT LSB
00D2 =          OV:     EQU     0D2H    ;OVERFLOW FLAG
00D0 =          P:      EQU     0D0H    ;PARITY FLAG
00BC =          PS:     EQU     0BCH    ;PRIORITY SERIAL PORT
00BB =          PT1:    EQU     0BBH    ;PRIORITY TIMER 1
00BA =          PX1:    EQU     0BAH    ;PRIORITY EXTERNAL 1
00B9 =          PT0:    EQU     0B9H    ;PRIORITY TIMER 0
00B8 =          PX0:    EQU     0B8H    ;PRIORITY EXTERNAL 0
00AF =          EA:     EQU     0AFH    ;ENABLE ALL INTERRUPT
00AC =          ES:     EQU     0ACH    ;ENABLE SERIAL INTERRUPT
00AB =          ET1:    EQU     0ABH    ;ENABLE TIMER 1 INTERRUPT
00AA =          EX1:    EQU     0AAH    ;ENABLE EXTERNAL 1 INTERR
00A9 =          ET0:    EQU     0A9H    ;ENABLE TIMER 0 INTERRUPT
00A8 =          EX0:    EQU     0A8H    ;ENABLE EXTERNAL 0 INTERR
009F =          SM0:    EQU     09FH    ;SERIAL MODE 0
```

```
009E =          SM1:        EQU     09EH        ;SERIAL MODE 1
009D =          SM2:        EQU     09DH        ;SERIAL MODE 2
009C =          REN:        EQU     09CH        ;SERIAL RECEPTION ENABLE
009B =          TB8:        EQU     09BH        ;TRANSMITT BIT 9
009A =          RB8:        EQU     09AH        ;RECEIVE BIT 8
0099 =          TI:         EQU     099H        ;TRANSMIT INTERRUPT FLAG
0098 =          RI:         EQU     098H        ;RECEIVE INTERRUPT FLAG
008F =          TF1:        EQU     08FH        ;TIMER 1 OVERFLOW FLAG
008E =          TR1:        EQU     08EH        ;TIMER 1 RUN CONTROL BIT
008D =          TF0:        EQU     08DH        ;TIMER 0 OVERFLOW FLAG
008C =          TR0:        EQU     08CH        ;TIMER 0 RUN CONTROL BIT
008B =          IE1:        EQU     08BH        ;EXT INTERR. 1 EDGE FLAG
008A =          IT1:        EQU     08AH        ;EXT INTERR. 1 TYPE FLAG
0089 =          IE0:        EQU     089H        ;EXT INTERR. 0 EDGE FLAG
0088 =          IT0:        EQU     088H        ;EXT INTERR. 0 TYPE FLAG

6000 =          RTC:        EQU     6000H
4000 =          LCD:        EQU     4000H
2000 =          RAM:        EQU     2000H
8000 =          DRIVER:     EQU     8000H
0070 =          CN_S:       EQU     70H
0071 =          CN_M:       EQU     71H
0072 =          CN_H:       EQU     72H
0073 =          CN_D:       EQU     73H
0074 =          CN_DAT:     EQU     74H
0075 =          CN_MO:      EQU     75H

0000                        ORG     00H
0000 020100                 LJMP    INIT_LCD
                ;*************************************************
                ;*                  INTERUPT 0                   *
                ;*************************************************

0003                        ORG     03H
0003 4142                   AJMP    INT0
                ;*************************************************
                ;*          INITIALIZE LCD DISPLAY & RTC         *
                ;*************************************************

0100                        ORG     100H
0100            INIT_LCD:
0100 318F                   ACALL   DELAY1          ;10ms
0102 904000                 MOV     DPTR,#LCD       ;wr addr
0105 7438                   MOV     A,#38H          ;data 8bit,5x7dot
0107 F0                     MOVX    @DPTR,A
0108 318F                   ACALL   DELAY1
010A 7401                   MOV     A,#01H          ;clear display
010C F0                     MOVX    @DPTR,A
010D 318F                   ACALL   DELAY1
010F 740F                   MOV     A,#0FH          ;display&cursor on
0111 F0                     MOVX    @DPTR,A
0112 318F                   ACALL   DELAY1
0114 7406                   MOV     A,#06H          ;cursor shift right
0116 F0                     MOVX    @DPTR,A
0117 318F                   ACALL   DELAY1
                                                    ;END INIT_LCD

0119 90600C                 MOV     DPTR,#RTC+12    ;reset count addr
011C 74FF                   MOV     A,#0FFH         ;data for reset
011E F0                     MOVX    @DPTR,A
                                                    ;END INIT RTC
011F 3123                   ACALL   CLRAM
0121 41D7                   AJMP    DISP_TIT
                ;*************************************************
                ;*                  CLEAR RAM                    *
```

```
                        ;*****************************************
0123 902000     CLRAM:  MOV   DPTR,#RAM         ;ram addr
0126 E4                 CLR   A                 ;data for res.t
0127 7808               MOV   R0,#08H           ;loop 1000 time
0129 7900       CLR:    MOV   R1,#00H
012B F0         CLR1:   MOVX  @DPTR,A           ;put data
012C A3                 INC   DPTR
012D D9FC               DJNZ  R1,CLR1
012F D8F8               DJNZ  R0,CLR
                                                ;END CLR RAM

                        ;*****************************************
                        ;*          SUBRUTINE OF LCD DISPLAY     *
                        ;*****************************************
0131 7830       DISPY:  MOV   R0,#30H           ;lcd buff addr
0133 7840               MOV   R3,#40H           ;64 char
0135 313A               ACALL LCD_BUF
0137 3145               ACALL DOWNLOAD
0139 22                 RET
013A 7400       LCD_BUF: MOV  A,#00H
013C F9                 MOV   R1,A
013D 93         BUF1:   MOVC  A,@A+DPTR
013E F6                 MOV   @R0,A
013F 08                 INC   R0
0140 09                 INC   R1
0141 E9                 MOV   A,R1
0142 DBF9               DJNZ  R3,BUF1
0144 22                 RET

0145 318F       DOWNLOAD: ACALL DELAY1          ;30ms
0147 904000             MOV   DPTR,#LCD         ;wr comman
014A 7480               MOV   A,#80H            ;line 1
014C 7830               MOV   R0,#30H
014E 3188               ACALL DOWN1
0150 3182       LINE1:  ACALL SUBLINE
0152 DAFC               DJNZ  R2,LINE1
0154 318F               ACALL DELAY1
0156 904000             MOV   DPTR,#LCD         ;wr command
0159 740C               MOV   A #0CH            ;line 2
015B 7840               MOV   R0,#40H
015D 3188               ACALL DOWN1
015F 3182       LINE2:  ACALL SUBLINE
0161 DAFC               DJNZ  R2,LINE2
0163 318F               ACALL DELAY1
0165 904000             MOV   DPTR,#LCD         ;wr comman
0168 7490               MOV   A,#90H            ;line 3
016A 7850               MOV   R0,#50H
016C 3188               ACALL DOWN1
016E 3182       LINE3:  ACALL SUBLINE
0170 DAFC               DJNZ  R2,LINE3
0172 318F               ACALL DELAY1
0174 904000             MOV   DPTR,#LCD         ;wr command
0177 74D0               MOV   A,#0D0H           ;line 4
0179 7860               MOV   R0,#60H
017B 3188               ACALL DOWN1
017D 3182       LINE4:  ACALL SUBLINE
017F DAFC               DJNZ  R2,LINE4
0181 22                 RET

0182 318F       SUBLINE: ACALL DELAY1
0184 E6                 MOV   A,@R0
0185 F0                 MOVX  @DPTR,A
0186 08                 INC   R0
0187 22                 RET
```

```
0189 904002                     MOV   DPTR,#LCD+2      ;wr data
018C 7A10                       MOV   R2,#10H         ;16 char/line
018E 22                         RET

018F AC0A       DELAY1:         MOV   R4,0AH
0191 AD00       EY:             MOV   R5,00H
0193 DDFE       EY1:            DJNZ  R5,EY1
0195 DCFA                       DJNZ  R4,EY
0197 22                         RET

0198 F9         DELAY:          MOV   R1,A            ;BEFORE CALL MUST LOAD ACC
0199 7A00       DY1:            MOV   R2,#00H
019B 7B00       DY2:            MOV   R3,#00H
019D DBFE       DY3:            DJNZ  R3,DY3
019F DAFA                       DJNZ  R2,DY2
01A1 D9F6                       DJNZ  R1,DY1
01A3 22                         RET


0200                            ORG   200H

0200 4041424344TABT:            DFB 40H,41H,42H,43H,44H,45H,46H,47H
0208 48494A4B4C                 DFB 48H,49H,4AH,4BH,4CH,4DH,4EH,4FH    ;line1
0210 5051525354                 DFB 50H,51H,52H,53H,54H,55H,56H,57H
0218 58595A5B5C                 DFB 58H,59H,5AH,5BH,5CH,5DH,5EH,5FH    ;line2
0220 6061626364                 DFB 60H,61H,62H,63H,64H,65H,66H,67H
0228 68696A6B6C                 DFB 68H,69H,6AH,6BH,6CH,6DH,6EH,6FH    ;line3
0230 7071727374                 DFB 70H,71H,72H,73H,74H,75H,76H,77H
0238 78797A7B7C                 DFB 78H,79H,7AH,7BH,7CH,7DH,7EH,7FH    ;line4
0240 11         TAB1:           DFB 11H
0241 22         TAB2:           DFB 22H
                ;*****************************************
                ;*                INITO                  *
                ;*****************************************
0242 C083       INTO:           PUSH  DPH
0244 C082                       PUSH  DPL
0246 C0E0                       PUSH  ACC
0248 C000                       PUSH  R0
024A C007                       PUSH  R7
024C 90600A                     MOV   DPTR,#RTC+10      ;rd status addr
024F E0                         MOVX  A,@DPTR
0250 30E20D                     JNB   ACC+2,INT1
0253 7870                       MOV   R0,#70H           ;rtc buff addr
0255 7F06                       MOV   R7,#06H
0257 906002     RTC_COUN:       MOV   DPTR,#RTC+2       ;coun sec
025A E0                         MOVX  A,@DPTR
025B F6                         MOV   @R0,A
025C 08                         INC   R0
025D A3                         INC   DPTR
025E DFF7                       DJNZ  R7,RTC_COUN
0260 D007       INT1:           POP   R7
0262 D000                       POP   R0
0264 D0E0                       POP   ACC
0266 D082                       POP   DPL
0268 D083                       POP   DPH
026A 32                         RETI

                ;*****************************************
                ;*              BCD TO ASCII             *
                ;*****************************************
026B C000       BCD_ASCI:       PUSH  R0
026D C001                       PUSH  R1
026F C002                       PUSH  R2
0271 C0E0                       PUSH  ACC
0273 7A06                       MOV   R2,#06H
0275 7970                       MOV   R1,#70H         ;BCD RTC BUF
```

```
0277 78B9                MOV   R0,#0B9H       ;ASCII RTC BUF
0279 E7        BD_BUF:   MOV   A,@R1
027A 518D                ACALL H_TO_A
027C F6                  MOV   @R0,A
027D 08                  INC   R0
027E A6F0                MOV   @R0,B
0280 08                  INC   R0
0281 09                  INC   R1
0282 DAF5                DJNZ  R2,BD_BUF
0284 D0E0                POP   ACC
0286 D002                POP   R2
0288 D001                POP   R1
028A D000                POP   R0
028C 22                  RET

028D C0E0      H_TO_A:   PUSH  ACC
028F 5199                ACALL H_A
0291 F5F0                MOV   B,A
0293 D0E0                POP   ACC
0295 C4                  SWAP  A
0296 5199                ACALL H_A
0298 22                  RET

0299 C0D0      H_A:      PUSH  PSW
029B C2D7                CLR   CY
029D 540F                ANL   A,#0FH
029F B40902              CJNE  A,#09H,H_A1
02A2 D2D7                SETB  CY
02A4 30D704    H_A1:     JNB   CY,H_A2
02A7 4430                ORL   A,#30H
02A9 8004                SJMP  END_HTOA
02AB 9409      H_A2:     SUBB  A,#09H
02AD 4440                ORL   A,#40H
02AF D0D0      END_HTOA: POP   PSW
02B1 22                  RET

                ;*****************************************
                ;*          ASCII TO LCD BUF             *
                ;*****************************************
02B2 85BF36     ASC_BUF:  MOV   36H,0BFH      ;ascii day to buf
02B5 85C037               MOV   37H,0C0H
02B8 85C139               MOV   39H,0C1H      ;date
02BB 85C23A               MOV   3AH,0C2H
02BE 85C33C               MOV   3CH,0C3H      ;month
02C1 85C43D               MOV   3DH,0C4H
02C4 85BD46               MOV   46H,0BDH      ;hr
02C7 85BE47               MOV   47H,0BEH
02CA 85BB49               MOV   49H,0BBH      ;min
02CD 85BC4A               MOV   4AH,0BCH
02D0 85B94C               MOV   4CH,0B9H      ;sec
02D3 85BA4D               MOV   4DH,0BAH
02D6 22                   RET

                ;*****************************************
                ;*             DISPLAY TITLE             *
                ;*****************************************
02D7 900200     DISP_TIT: MOV   DPTR,#TABT    ;dispy title
02DA 3131                 ACALL DISPY
02DC 7404                 MOV   A,#04H      ;DISPLAY 1sec
02DE 3198                 ACALL DELAY
02E0 7590FF               MOV   P1,#11111111B
02E3 209017               JB    P1+0,ST_SETME
02E6 D2AF                 SETB  EA                ;MODE TIME
02E8 D2A8                 SETB  EX0
02EA D2B8                 SETB  PX0
02EC 758803               MOV   TCON,#00000011B
```

```
02EF 900240          MOV   DPTR,#TAB1          ;display mode time
02F2 3131            ACALL DISPY
02F4 516B            ACALL BCD_ASCI
02F6 51B2            ACALL ASC_BUF
02F8 3145            ACALL DOWNLOAD
02FA 020405          LJMP  SUBTIME
02FD C2AF  ST_SETME: CLR   EA
02FF 900241          MOV   DPTR,#TAB2     ;display mode settime
0302 3131            ACALL DISPY
0304 516B            ACALL BCD_ASCI
0306 51B2            ACALL ASC_BUF
0308 3145            ACALL DOWNLOAD
030A 7590FF          MOV   P1,#11111111B
030D 30910A          JNB   P1+1,SETTIME    ;press key inc
0310 7590FF          MOV   P1,#11111111B
0313 309002          JNB   P1+0,PROG       ;OUT TO PROGRAM MODE
0316 80E5            SJMP  ST_SETME
0318 A194  PROG:     AJMP  PROGRAM

           ;*************************************************
           ;*                SUBRUTINE SETTIME              *
           ;*************************************************
031A 7590FF SETTIME: MOV   P1,#11111111B
031D 309102          JNB   P1+1,A2         ;if press go A2
0320 8008            SJMP  A20
0322 7872    A2:     MOV   R0,#CN_H
0324 06              INC   @R0
0325 7418            MOV   A,#24D
0327 96              SUBB  A,@R0
0328 4008            JC    A21             ;if over 24
032A 7590FF  A20:    MOV   P1,#11111111B
032D 309306          JNB   P1+3,A22        ;press key enter
0330 80E8            SJMP  SETTIME
0332 7600    A21:    MOV   @R0,#00H        ;clear hr=0
0334 80F4            SJMP  A20
0336 516B    A22:    ACALL BCD_ASCI
0338 51B2            ACALL ASC_BUF
033A 3145            ACALL DOWNLOAD
033C 906004          MOV   DPTR,#RTC+4     ;count hr
033F E6              MOV   A,@R0
0340 F0              MOVX  @DPTR,A

0341 7590FF  ST3:    MOV   P1,#11111111B
0344 309102          JNB   P1+1,A3         ;if press go a3
0347 8008            SJMP  A30
0349 7871    A3:     MOV   R0,#CN_M
034B 06              INC   @R0
034C 743C            MOV   A,#60D
034E 96              SUBB  A,@R0
034F 4008            JC    A31             ;if over 60
0351 7590FF  A30:    MOV   P1,#11111111B
0354 309306          JNB   P1+3,A32        ;press key enter
0357 80E8            SJMP  ST3
0359 7600    A31:    MOV   @R0,#00H        ;clear min=0
035B 80F4            SJMP  A30
035D 516B    A32:    ACALL BCD_ASCI
035F 51B2            ACALL ASC_BUF
0361 3145            ACALL DOWNLOAD
0363 906003          MOV   DPTR,#RTC+3     ;coun min
0366 E6              MOV   A,@R0
0367 F0              MOVX  @DPTR,A

0368 7590FF  ST4:    MOV   P1,#11111111B
036B 309102          JNB   P1+1,A4         ;if press go a4
036E 8008            SJMP  A40
0370 7870    A4:     MOV   R0,#CN_S
```

```
0372 06                 INC   @R0
0373 743C               MOV   A,#60D
0375 96                 SUBB  A,@R0
0376 4008               JC    A41
0378 7590FF     A40:    MOV   P1,#11111111B
037B 309306             JNB   P1+3,A42          ;KEY ENTER?
037E 80E8               SJMP  ST4
0380 7600       A41:    MOV   @R0,#00H          ;clear sec=0
0382 80F4               SJMP  A40
0384 516B       A42:    ACALL BCD_ASCI
0386 51B2               ACALL ASC_BUF
0388 3145               ACALL DOWNLOAD
038A 906002             MOV   DPTR,#RTC+2       ;coun sec
038D E6                 MOV   A,@R0
038E F0                 MOVX  @DPTR,A

038F 7590FF     ST5:    MOV   P1,#11111111B
0392 309102             JNB   P1+1,A5           ;if press go a5
0395 8008               SJMP  A50
0397 7875       A5:     MOV   R0,#CN_MO
0399 06                 INC   @R0
039A 740C               MOV   A,#12D
039C 96                 SUBB  A,@R0
039D 4008               JC    A51
039F 7590FF     A50:    MOV   P1,#11111111B
03A2 309306             JNB   P1+3,A52          ;KER ENTER?
03A5 80E8               SJMP  ST5
03A7 7600       A51:    MOV   @R0,#00H          ;clear month=0
03A9 80F4               SJMP  A50
03AB 516B       A52:    ACALL BCD_ASCI
03AD 51B2               ACALL ASC_BUF
03AF 3145               ACALL DOWNLOAD
03B1 906007             MOV   DPTR,#RTC+7
03B4 E6                 MOV   A,@R0
03B5 F0                 MOVX  @DPTR,A

03B6 7590FF     ST6:    MOV   P1,#11111111B
03B9 309102             JNB   P1+1,A6           ;if press go a6
03BC 8008               SJMP  A60
03BE 7873       A6:     MOV   R0,#CN_D
03C0 06                 INC   @R0
03C1 7407               MOV   A,#07D
03C3 96                 SUBB  A,@R0
03C4 4008               JC    A61
03C6 7590FF     A60:    MOV   P1,#11111111B
03C9 309306             JNB   P1+3,A62          ;KER ENTER?
03CC 80E8               SJMP  ST6
03CE 7600       A61:    MOV   @R0,#00H
03D0 80F4               SJMP  A60
03D2 516B       A62:    ACALL BCD_ASCI
03D4 51B2               ACALL ASC_BUF
03D6 3145               ACALL DOWNLOAD
03D8 906005             MOV   DPTR,#RTC+5
03DB E6                 MOV   A,@R0
03DC F0                 MOVX  @DPTR,A

03DD 7590FF     ST7:    MOV   P1,#11111111B
03E0 309102             JNB   P1+1,A7           ;if press go a7
03E3 8008               SJMP  A70
03E5 A874       A7:     MOV   R0,CN_DAT
03E7 06                 INC   @R0
03E8 741F               MOV   A,#31D
03EA 96                 SUBB  A,@R0
03EB 4008               JC    A71
03ED 7590FF     A70:    MOV   P1,#11111111B
03F0 309306             JNB   P1+3,A72          ;KER ENTER?
```

```
03F3 80E8                SJMP   ST7
03F5 7600      A71:      MOV    @R0,#00H
03F7 80F4                SJMP   A70
03F9 516B      A72:      ACALL  BCD_ASCI
03FB 51B2                ACALL  ASC_BUF
03FD 3145                ACALL  DOWNLOAD
03FF 906006              MOV    DPTR,#RTC+6
0402 E6                  MOV    A,@R0
0403 F0                  MOVX   @DPTR,A
0404 22                  RET
                ;*****************************************
                ;*          SUBRUTINE OF TIME            *
                ;*****************************************
0405 901000    SUBTIME:  MOV    DPTR,#1000H
0408 7A07                MOV    R2,#07H
040A 7900      T:        MOV    R1,#00H
040C 7590FF    TIME:     MOV    P1,#11111111B
040F 30911B              JNB    P1+1,TIME20          ;IF  PRESS INC GO TIME
0412 E573                MOV    A,73H                ;MOV CONTENT IN 73 TO
0414 03                  RR     A                    ;------XXX
0415 03                  RR     A
0416 03                  RR     A                    ;XXX------
0417 4548                ORL    A,72                 ;DAY+HR TO A
0419 F8                  MOV    R0,A                 ;DDDHHHHH
041A E0                  MOVX   A,@DPTR              ;LOAD FIRST PROGRAM
041B B50017              CJNE   A,R0,NEXT1
041E A3                  INC    DPTR
041F E0                  MOVX   A,@DPTR              ;LOAD MIN IN PROG
0420 B54716              CJNE   A,71,NEXT2
0423 A3                  INC    DPTR
0424 E0                  MOVX   A,@DPTR              ;LOAD ON/OFF
0425 30E60D              JNB    ACC+6,NEXT1            ; 8888888888888888
0428 7F00                MOV    R7,#00H
042A FE                  MOV    R6,A
042B 5407                ANL    A,#00000111B         ;MASK OFF LSB
042D A109      TIME20:   AJMP   TIME2
042F A3        NEXT:     INC    DPTR                 ;LOAD PROGRAM2
0430 D9DA                DJNZ   R1,TIME
0432 DAD6                DJNZ   R2,T
0434 22                  RET
0435 A3        NEXT1:    INC    DPTR
0436 A3                  INC    DPTR
0437 80F6                SJMP   NEXT
0439 A3        NEXT2:    INC    DPTR
043A 80F3                SJMP   NEXT
                                                     ;KEY
043C B40009              CJNE   A,#00H,AA0             ;JMP IF NO CH_0
043F 202704              JB     R8+7,CH00             ;ON CH_0
0442 91E1                ACALL  OFF_0                 ;OFF CH_0
0444 8002                SJMP   AA0
0446 91AA      CH00:     ACALL  ON_0                  ;ON CH_0
0448 B40109    AA0:      CJNE   A,#01H,AA1             ;JMP IF NO CH_1
044B 202704              JB     R8+7,CH01             ;ON CH_1
044E 91E6                ACALL  OFF_1
0450 8002                SJMP   AA1
0452 91AF      CH01:     ACALL  ON_1
0454 B40209    AA1:      CJNE   A,#02H,AA2
0457 202704              JB     R8+7,CH02
045A 91EB                ACALL  OFF_2
045C 8002                SJMP   AA2
045E 91B4      CH02:     ACALL  ON_2
0460 B40309    AA2:      CJNE   A,#03H,AA3
0463 202704              JB     R8+7,CH03
0466 91F0                ACALL  OFF_3
0468 8002                SJMP   AA3
046A 91B9      CH03:     ACALL  ON_3
```

```
046C B40409      AA3:       CJNE   A,#04H,AA4
046F 202704                 JB     R8+7,CH04
0472 91F5                   ACALL  OFF_4
0474 8002                   SJMP   AA4
0476 91BE        CH04:      ACALL  ON_4
0478 B40509      AA4:       CJNE   A,#05H,AA5
047B 202704                 JB     R8+7,CH05
047E 91FA                   ACALL  OFF_5
0480 8002                   SJMP   AA5
0482 91C3        CH05:      ACALL  ON_5
0484 B40609      AA5:       CJNE   A,#06H,AA6
0487 202704                 JB     R8+7,CH06
048A 91FF                   ACALL  OFF_6
048C 8002                   SJMP   AA6
048E 91C8        CH06:      ACALL  ON_6
0490 B40709      AA6:       CJNE   A,#07H,AA7
0493 202704                 JB     R8+7,CH07
0496 B104                   ACALL  OFF_7
0498 8002                   SJMP   AA7
049A 91CD        CH07:      ACALL  ON_7
049C C083        AA7:       PUSH   DPH
049E C082                   PUSH   DPL
04A0 905000                 MOV    DPTR,#5000H
04A3 F0                     MOVX   @DPTR,A              ;ON/OFF BIT IN A
04A4 D082                   POP    DPL                  ;FOR CONTROL LOAD
04A6 D083                   POP    DPH
04A8 812F                   AJMP   NEXT

                 ;*************************************************
                 ;*                  SUB IN TIME                  *
                 ;*************************************************
04AA D221        ON_0:      SETB   R9+0                 ;ON CH_0
04AC 91D2                   ACALL  ON_OFF
04AE 22                     RET
04AF D222        ON_1:      SETB   R9+1                 ;ON CH_1
04B1 91D2                   ACALL  ON_OFF
04B3 22                     RET
04B4 D223        ON_2:      SETB   R9+2
04B6 91D2                   ACALL  ON_OFF
04B8 22                     RET
04B9 D224        ON_3:      SETB   R9+3
04BB 91D2                   ACALL  ON_OFF
04BD 22                     RET
04BE D225        ON_4:      SETB   R9+4
04C0 91D2                   ACALL  ON_OFF
04C2 22                     RET
04C3 D226        ON_5:      SETB   R9+5
04C5 91D2                   ACALL  ON_OFF
04C7 22                     RET
04C8 D227        ON_6:      SETB   R9+6
04CA 91D2                   ACALL  ON_OFF
04CC 22                     RET
04CD D228        ON_7:      SETB   R9+7
04CF 91D2                   ACALL  ON_OFF
04D1 22                     RET

04D2 E521        ON_OFF:    MOV    A,R9
04D4 C083                   PUSH   DPH
04D6 C082                   PUSH   DPL
04D8 905000                 MOV    DPTR,#5000H
04DB F0                     MOVX   @DPTR,A
04DC D082                   POP    DPL
04DE D083                   POP    DPH
04E0 22                     RET

04E1 C221        OFF_0:     CLR    R9+0                 ;OFF CH_0
```

```
04E5 22                 RET
04E6 C222       OFF_1:  CLR   R9+1
04E8 91D2               ACALL ON_OFF
04EA 22                 RET
04EB C223       OFF_2:  CLR   R9+2
04ED 91D2               ACALL ON_OFF
04EF 22                 RET
04F0 C224       OFF_3:  CLR   R9+3
04F2 91D2               ACALL ON_OFF
04F4 22                 RET
04F5 C225       OFF_4:  CLR   R9+4
04F7 91D2               ACALL ON_OFF
04F9 22                 RET
04FA C226       OFF_5:  CLR   R9+5
04FC 91D2               ACALL ON_OFF
04FE 22                 RET
04FF C227       OFF_6:  CLR   R9+6
0501 91D2               ACALL ON_OFF
0503 22                 RET
0504 C228       OFF_7:  CLR   R9+7
0506 91D2               ACALL ON_OFF
0508 22                 RET

0509 C0E0       TIME2:  PUSH  ACC
050B C000               PUSH  R0
050D C001               PUSH  R1
050F C002               PUSH  R2
0511 C083               PUSH  DPH
0513 C082               PUSH  DPL
0515 309102     TIME1:  JNB   P1+1,T1
0518 8008               SJMP  T10
051A 7876       T1:     MOV   R0,#76H
051C 06                 INC   @R0
051D 7407               MOV   A,#07H
051F 96                 SUBB  A,@R0
0520 4005               JC    T11               ;OVER 7
0522 309306     T10:    JNB   P1+3,T12
0525 80EE               SJMP  TIME1
0527 7600       T11:    MOV   @R0,#00H
0529 80F7               SJMP  T10
052B 309202     T12:    JNB   P1+2,T13          ;KEY DEC?
052E 8004               SJMP  T14
0530 7C00       T13:    MOV   R4,#00H
0532 8002               SJMP  T15               ;PROG OFF
0534 7C01       T14:    MOV   R4,#01H
0536 309302     T15:    JNB   P1+3,T16
0539 80F0               SJMP  T12
053B C0E0       T16:    PUSH  ACC
053D E6                 MOV   A,@R0
053E B40007             CJNE  A,#00H,AAA0
0541 200402             JB    R4+0,BBB0
0544 91E1               ACALL OFF_0
0546 91AA       BBB0:   ACALL ON_0
0548 B40107     AAA0:   CJNE  A,#01H,AAA1
054B 200402             JB    R4+0,BBB1
054E 91E6               ACALL OFF_1
0550 91AF       BBB1:   ACALL ON_1
0552 B40207     AAA1:   CJNE  A,#02H,AAA2
0555 200402             JB    R4+0,BBB2
0558 91EB               ACALL OFF_2
055A 91B4       BBB2:   ACALL ON_2
```

```
055C B40307     AAA2:     CJNE   A,#03H,AAA3
055F 200402               JB     R4+0,BBB3
0562 91F0                 ACALL  OFF_3
0564 91B9       BBB3:     ACALL  ON_3
0566 B40407     AAA3:     CJNE   A,#04H,AAA4
0569 200402               JB     R4+0,BBB4
056C 91F5                 ACALL  OFF_4
056E 91BE       BBB4:     ACALL  ON_4
0570 B40507     AAA4:     CJNE   A,#05H,AAA5
0573 200402               JB     R4+0,BBB5
0576 91FA                 ACALL  OFF_5
0578 91C3       BBB5:     ACALL  ON_5
057A B40607     AAA5:     CJNE   A,#06H,AAA6
057D 200402               JB     R4+0,BBB6
0580 91FF                 ACALL  OFF_6
0582 91C8       BBB6:     ACALL  ON_6
0584 B40707     AAA6:     CJNE   A,#07H,AAA7
0587 200402               JB     R4+0,BBB7
058A B104                 ACALL  OFF_7
058C 91CD       BBB7:     ACALL  ON_7
058E 5199       AAA7:     ACALL  H_A
0590 F56C                 MOV    6CH,A              ;PUT CHANEL IN LCD
0592 3145                 ACALL  DOWNLOAD

0594 E9         PROGRAM:  MOV A,R1

                  END
```

0322	A2	032A	A20	0332	A21
0336	A22	0349	A3	0351	A30
0359	A31	035D	A32	0370	A4
0378	A40	0380	A41	0384	A42
0397	A5	039F	A50	03A7	A51
03AB	A52	03BE	A6	03C6	A60
03CE	A61	03D2	A62	03E5	A7
03ED	A70	03F5	A71	03F9	A72
0448	AA0	0454	AA1	0460	AA2
046C	AA3	0478	AA4	0484	AA5
0490	AA6	049C	AA7	0548	AAA0
0552	AAA1	055C	AAA2	0566	AAA3
0570	AAA4	057A	AAA5	0584	AAA6
058E	AAA7	00D6	AC	00E0	ACC
02B2	ASC_BUF	00F0	B	0546	BBB0
0550	BBB1	055A	BBB2	0564	BBB3
056E	BBB4	0578	BBB5	0582	BBB6
058C	BBB7	026B	BCD_ASCI	0279	BD_BUF
013D	BUF1	0446	CH00	0452	CH01
045E	CH02	046A	CH03	0476	CH04
0482	CH05	048E	CH06	049A	CH07
0129	CLR	012B	CLR1	0123	CLRAM
0073	CN_D	0074	CN_DAT	0072	CN_H
0071	CN_M	0075	CN_MO	0070	CN_S
00D7	CY	0198	DELAY	018F	DELAY1
0131	DISPY	02D7	DISP_TIT	0188	DOWN1
0145	DOWNLOAD	00B3	DPH	00B2	DPL
8000	DRIVER	0199	DY1	019B	DY2
019D	DY3	00AF	EA	02AF	END_HTOA
00AC	ES	00A9	ET0	00AB	ET1
00A8	EX0	00AA	EX1	0191	EY
0193	EY1	00D5	F0	0299	H_A
02A4	H_A1	02AB	H_A2	028D	H_TO_A
0089	IE0	008B	IE1	00A8	IEC
0100	INIT_LCD	0242	INT0	0260	INT1
00B8	IPC	0088	IT0	008A	IT1
4000	LCD	013A	LCD_BUF	0150	LINE1
015F	LINE2	016E	LINE3	017D	LINE4
042F	NEXT	0435	NEXT1	0439	NEXT2
04E1	OFF_0	04E6	OFF_1	04EB	OFF_2
04F0	OFF_3	04F5	OFF_4	04FA	OFF_5
04FF	OFF_6	0504	OFF_7	04AA	ON_0
04AF	ON_1	04B4	ON_2	04B9	ON_3
04BE	ON_4	04C3	ON_5	04C8	ON_6
04CD	ON_7	04D2	ON_OFF	00D2	OV
00D0	P	0080	P0	0090	P1
00A0	P2	00B0	P3	0087	PCON
0318	PROG	0594	PROGRAM	00BC	PS
00D0	PSW	00B9	PT0	00BB	PT1
00B8	PX0	00BA	PX1	0000	R0
0001	R1	0002	R2	0003	R3
0004	R4	0005	R5	0006	R6
0007	R7	0020	R8	0021	R9
2000	RAM	009A	RB8	009C	REN
0098	RI	00D3	RS0	00D4	RS1
6000	RTC	0257	RTC_COUN	0099	SBUF

0098	SCON	031A	SETTIME	009F	SM0
009E	SM1	009D	SM2	0081	SP
0341	ST3	0368	ST4	038F	ST5
03B6	ST6	03DD	ST7	02FD	ST_SETME
0182	SUBLINE	0405	SUBTIME	040A	T
051A	T1	0522	T10	0527	T11
052B	T12	0530	T13	0534	T14
0536	T15	053B	T16	0240	TAB1
0241	TAB2	0200	TABT	009B	TB9
0088	TCON	008D	TF0	008F	TF1
008C	TH0	008D	TH1	0099	TI
040C	TIME	0515	TIME1	0509	TIME2
042D	TIME20	008A	TL0	008B	TL1
0089	TMOD	008C	TR0	008E	TR1

ภาคผนวก

ACALL addr11

ความหมาย: เป็นคำสั่งเรียกแบบไม่มีเงื่อนไข ไปยังตำแหน่งที่ ชี้ด้วยค่าแอดเดรสที่กำหนด โปรแกรมเคาเตอร์จะเพิ่มค่าขึ้นสองครั้ง เพื่อรับแอดเดรสในคำสั่งที่ตามมา จากนั้นใส่ผลลัพธ์ 16 บิท ลงบนสแตค(8บิทต่ำก่อน) แล้วเพิ่มค่าสแตคพอน์ยเตอร์ขึ้นสองครั้ง แอดเดรสปลายทางได้รับจาก 5 บิทสูงของโปรแกรมเคาเตอร์ที่เพิ่มขึ้น ออปโค๊ดบิทที่ 5-7 และไบท์ที่สองต่อจากคำสั่ง โปรแกรมย่อยที่ถูกเรียกจะต้องเริ่มภายในบล็อก(2K)ของหน่วยความจำโปรแกรมเดียวกัน ซึ่งอยู่ในไบท์แรก ของคำสั่งต่อจาก ACALL คำสั่งนี้ไม่มีผลต่อแฟลก

ตัวอย่าง: เริ่มต้นให้สแตคพอน์ยเตอร์มีค่า 07H และ label"SUBRTN" ในหน่วยความจำโปรแกรมอยู่ที่ตำแหน่ง 0345H หลังจากทำคำสั่ง ACALL SUBRTN แล้ว ที่ตำแหน่ง 0123H สแตคพอน์ยเตอร์จะมีข้อมูล 09H หน่วยความจำข้อมูลภายในตำแหน่ง 08H และ 09H จะมีค่า 23H และ 01H ตามลำดับ โปรแกรมเคาเตอร์จะมีค่า 0345H

ENCODE:

a_{10}	a_9	a_8	1	0	0	0	1	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

ADD A,<src-byte>

ความหมาย: จะทำการบวกค่าในตัวแปร(8บิท) กับแอคคิวมูเลเตอร์ แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์ ถ้ามีตัวทดจากบิท 7 C-แฟลกจะเซ็ทค่าเป็น 1 ถ้ามีตัวทดออกจากบิท 3 AC-แฟลกจะเซ็ทค่าเป็น 1 มิฉะนั้นจะถูกเคลียร์ เมื่อทำการบวกจำนวนเต็มแบบไม่คิดเครื่องหมาย C-แฟลกจะเป็นตัวชี้โอเวอร์โฟล์วที่เกิดขึ้น OV-แฟลกจะถูกเซ็ทเมื่อมีตัวทดออกจากบิท 6 แต่ไม่ออกจากบิท 7 หรือมีตัวทดออกจากบิท 7 แต่ไม่ออกจากบิท 6 มิฉะนั้นแล้ว OV-แฟลกจะถูกเคลียร์ เมื่อทำการบวกเลขจำนวนเต็มแบบไม่คิดเครื่องหมาย OV-แฟลกจะเป็นตัวชี้ค่าลบของผลลัพธ์ จากการบวกค่าทั้งสองที่เป็นบวก หรือจะชี้ผลบวกที่เป็นบวกจากการบวกเลขลบทั้งสองก็ได้ โอเปอแรนด์ที่ใช้อ้างแอดเดรสมี 4โหมด : REGISTOR,DIRECT,REG-INDIRECT,IMMEDIATE

ตัวอย่าง: แอคคิวมูเลเตอร์มีค่า 0C3H(11000011B)และR0มีค่า0AAH(10101010B) หลังจากทำคำสั่ง ADD A,R0 จะได้ผลลัพธ์ 6DH(01101101B) เก็บไว้ในแอคคิวมูเลเตอร์ AC-แฟลกจะถูกเคลียร์ Cและ0V-แฟลกจะเซ็ทค่าเป็น 1

ตัวอย่าง: label"JMPADR" ในหน่วยความจำโปรแกรมอยู่ที่ตำแหน่ง 0123H คำสั่ง AJMP JMPADR อยู่ที่แอดเดรส 0345H ผลลัพธ์จะโหลดค่าโปรแกรมเคาเตอร์ด้วย 0123H

ENCODE:

a_{10} a_9 a_8 0	0 0 0 1

a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

ANL <dest-byte>,<src-byte>

ความหมาย: กระทำlogic-AND ระหว่างค่าในตัวแปรทั้งสองและเก็บผลลัพธ์ไว้ในตัวแปรปลายทาง คำสั่งนี้ไม่มีผลต่อแฟลก โอเปอแรนด์ทั้งสองอ้างแอดเดรสได้6โหมด ขณะที่ปลายทางเป็นแอคคิวมูเลเตอร์ src-byteสามารถอ้างแอดเดรสได้ 4 โหมดคือ register,diretc,reg-indiretc,immediate และเมื่อปลายทางอ้างแอดเดรสแบบ diretc, source สามารถอ้างแอดเดรสแบบimmediate หรือแอคคิวมูเลเตอร์

หมายเหตุ: เมื่อใช้คำสั่งนี้เพื่อเปลี่ยนแปลงพอร์ทเอาพุท ค่าที่ใช้เป็นข้อมูลพอร์ทเริ่มต้นจะอ่านได้จากข้อมูลเอาพุทที่แลทช์ ไว้ไม่ใช่จากขาอินพุท

ตัวอย่าง: ถ้าแอคคิวมูเลเตอร์มีค่า 0C3H(11000011B) และR0 มีค่า 55H (01010101B) เมื่อทำคำสั่ง ANL A,R0 จะได้ผลลัพธ์ 41H(01000001B)เก็บไว้ในแอคคิวมูเลเตอร์

เมื่อปลายทางอ้างแอดเดรสแบบ direct คำสั่งจะเคลียร์ทุกบิทใน ram บางตำแหน่งหรือ Hardware register, Mask byteจะหารูปแบบบิทที่ถูกเคลียร์ถ้าไม่เป็นค่าคงที่ในคำสั่งก็จะเป็นค่าที่คำนวนในแอคคิวมูเลเตอร์ ขณะrun คำสั่ง ANL P1,#01110011B จะทำการเคลียร์บิท 7,3 และ2 ของพอร์ท 1

ANL A,Rn

ความหมาย: ทำlogic-AND ระหว่างแอคคิวมูเลเตอร์กับ Rn แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 0 1	1 r r r

ANL A,direct

ความหมาย: ทำlogic-AND ระหว่างแอคคิวมูเลเตอร์กับข้อมูลในตำแหน่งที่กำหนดแล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 0 1	0 1 0 1	direct address

ANL A,@Ri

ความหมาย: ทำlogic-AND ระหว่างแอคคิวมูเลเตอร์กับข้อมูลในแอดเดรสที่กำหนดด้วยR0หรือR1 แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 0 1	0 1 1 i

ANL A,#data

ความหมาย: ทำlogic-AND ระหว่างแอคคิวมูเลเตอร์กับข้อมูลที่เข้ามากับคำสั่งแล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 0 1	0 1 0 0	immediate data

ANL direct,A

ความหมาย: ทำlogic-AND ระหว่างข้อมูลในแอดเดรสที่กำหนด กับแอคคิวมูเลเตอร์ แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 0 1	0 0 1 0	direct address

ANL direct,#data

ความหมาย: ทำlogic-AND ระหว่างข้อมูลในแอดเดรสที่กำหนด กับข้อมูลที่เข้ามากับคำสั่ง แล้วเก็บผลลัพธ์ไว้ในแอดเดรสที่กำหนดนั้น

ENCODE:

0 1 0 1	0 0 1 1	direct address	immediate data

ANL C,<src-bit>

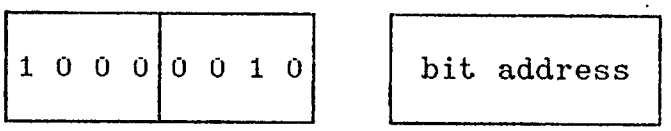
ความหมาย: ถ้าค่าของ src-bit มีlogic 0 จะเป็นการเคลียร์ C-แฟลก มิฉะนั้นจะย้าย C-แฟลกไปสู่สถานะ current state เครื่องหมาย "/" แสดงให้เห็นว่าเป็นการคอมพลีเมนท์บิทของแอดเดรส และใช้เป็น source แต่จะไม่มีผลต่อ source bit และแฟลกอื่นๆ src-bit อ้างแอดเดรสแบบ direct เท่านั้น

ตัวอย่าง: C-แฟลกจะถูกเซ็ทถ้า P1.0=1,ACC.7=1 และ OV-แฟลก=0

ANL C,bit

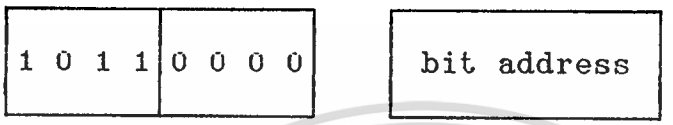
ความหมาย: ทำlogic-AND ระหว่างแอดเดรสบิทที่กำหนด กับc-แฟลก แล้วเก็บ

ผลลัพธ์ไว้ในc-แฟลก



ANL C,/bit

ความหมาย: ทำlogic-AND ระหว่างแอดเดรสบิทที่คอมพลีเมนท์ กับc-แฟลก แล้วเก็บผลลัพธ์ไว้ในc-แฟลก



CJNE<dest-byte>,<src-byte>,rel

ความหมาย: ทำการเปรียบเทียบค่าของโอเปอร์แรนด์แรก และจะกระโดดไปถ้าผลลัพธ์ไม่เท่ากัน แอดเดรสปลายทางจะคำนวนได้โดย การบวกแบบคิดเครื่องหมาย relative แล้วใส่ในคำสั่งไบท์สุดท้ายไปยังโปรแกรมเคาเตอร์ หลังจากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่จุดเริ่มต้นของคำสั่งต่อไป c-แฟลกจะถูกเซ็ทถ้าเลขจำนวนเต็มแบบไม่คิดเครื่องหมายของ<dest-byte>น้อยกว่า<src-byte> นอกนั้นc-แฟลกจะถูกเคลียร์ คำสั่งนี้ไม่มีผลต่อโอเปอร์แรนด์ทั้งสอง ซึ่งในสองโอเปอร์แรนด์แรกสามารถอ้างแอดเดรสได้ 4 โหมด แอคคิวมูเลเตอร์อาจจะถูกเปรียบเทียบกับ ไบท์ที่อ้างแอดเดรสแบบdirect หรือimmediete และอ้างตำแหน่งของram หรือโดยอ้างผ่านregister แล้วเปรียบเทียบกับค่าคงที่immediate

ตัวอย่าง: แอคคิวมูเลเตอร์มีค่า 34H, R7มีค่า56H เมื่อทำคำสั่ง

```
          CJNE R7,#60H,NOTIEQ
          .....                ;R7=60H
NOTIEQ:   JC   REQILOW         ;IF R7<60H
          .....                ;R7>60H
```

จะทำการเซ็ทค่าc-แฟลก และแยกสาขาไปทำคำสั่งที่ label"NOTIEQ" คำสั่งนี้จะกำหนดว่า R7 มากกว่าหรือน้อยกว่า 60H โดยการทดสอบc-แฟลก ถ้าข้อมูลที่พอร์ท1 เป็น34H ดังนั้นเมื่อทำคำสั่ง

```
WAIT:  CJNE  A,P1,WAIT
```

จะเคลียร์c-แฟลก และทำคำสั่งต่อไป เมื่อข้อมูลที่อ่านจากพอร์ท1 เท่ากับแอคคิวมูเลเตอร์ ถ้าพอร์ท1 เป็นค่าอื่นโปรแกรมก็จะวนที่จุดนี้จนกว่าพอร์ท1 จะเปลี่ยนข้อมูลเป็น 34H

CJNE A,direct,rel

ความหมาย: ทำการเปรียบเทียบแอคคิวมูเลเตอร์กับข้อมูลในแอดเดรสที่กำหนด ถ้าไม่เท่ากัน จะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป

CJNE A,#data,rel

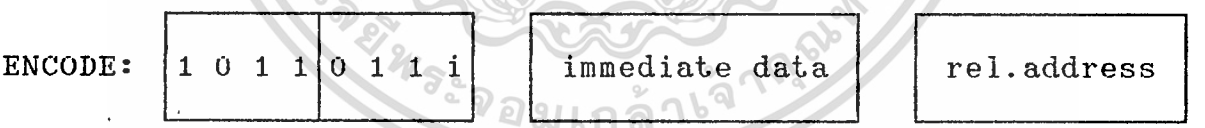
ความหมาย: ทำการเปรียบเทียบแอคคิวมูเลเตอร์ กับข้อมูลที่เข้ามากับคำสั่ง ถ้าไม่เท่ากัน จะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป

CJNE Rn,#data,rel

ความหมาย: ทำการเปรียบเทียบข้อมูลในRn กับข้อมูลที่เข้ามาในคำสั่ง ถ้าไม่เท่ากัน จะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป

CJNE @Ri,#data,rel

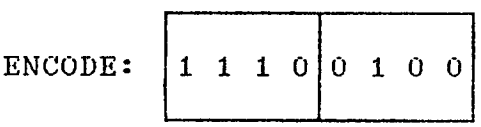
ความหมาย: ทำการเปรียบเทียบข้อมูลในแอดเดรสที่กำหนดด้วย R0 หรือ R1 ถ้าไม่เท่ากันจะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป



CLR A

ความหมาย: เคลียร์แอคคิวมูเลเตอร์ให้ทุกบิทเป็นศูนย์ และไม่มีผลต่อแฟลก

ตัวอย่าง: แอคคิวมูเลเตอร์มีค่า 5CH(01011101B) เมื่อทำคำสั่งCLR A จะเซ็ทค่าแอคคิวมูเลเตอร์เป็น 00H



CLR bit

ความหมาย: เคลียร์บิทที่ถูกชี้ คำสั่งนี้ไม่มีผลต่อแฟลกอื่นๆ และสามารถทำบนc-แฟลก หรือบิทใดๆ ที่อ้างแอดเดรสโดยตรง

ตัวอย่าง: พอร์ท1 ถูกเขียนด้วยข้อมูล 5DH(01011101B) มาก่อน เมื่อทำคำสั่ง CLR P1.2 จะได้ 059H(01011001B) เก็บไว้ในพอร์ท1

CLR C

ความหมาย: เคลียร์c-แฟลกให้เป็นศูนย์

ENCODE:	1 1 0 0	0 0 1 1

CLR bit

ความหมาย: เคลียร์บิทใดๆ ในแอดเดรสที่ต้องการ

ENCODE:	1 1 0 0	0 0 1 0	bit address

CPL A

ความหมาย: คอมพลีเมนต์ค่าในแอคคิวมูเลเตอร์ แต่ละบิทของแอคคิวมูเลเตอร์จะถูกคอมพลีเมนต์โดยบิทซึ่งเคยเป็น1 ก็จะถูกเปลี่ยนเป็น0 และในทางกลับกัน คำสั่งนี้ไม่มีผลต่อแฟลก

ตัวอย่าง: แอคคิวมูเลเตอร์มีค่า 5CH(01011100B) เมื่อทำคำสั่ง CPL A จะได้ค่า 0A3H(10100011B) เก็บไว้ยังแอคคิวมูเลเตอร์

ENCODE:	1 1 1 1	0 1 0 0

CPL bit

ความหมาย: บิทที่กำหนดจะถูกคอมพลีเมนต์ โดยบิทที่เคยเป็น1 ก็จะถูกเปลี่ยนมาเป็น0 และในทางกลับกัน คำสั่งนี้ไม่มีผลต่อแฟลกอื่นๆ ซึ่งคำสั่งนี้ยังสามารถกระทำบนc-แฟลก หรืออ้างแอดเดรสแบบบิทdiretc ก็ได้

หมายเหตุ:- เมื่อใช้คำสั่งนี้เพื่อเปลี่ยนแปลงขาเอาท์พุท ข้อมูลเริ่มต้นจะอ่านจากข้อมูลที่แลทช์ไว้ทางเอาท์พุท ไม่ใช่ขาอินพุท

ตัวอย่าง: พอร์ท1 ถูกเขียนด้วย 5BH(01011011B) มาก่อน เมื่อทำคำสั่ง

```
CLR  P1.1
CLR  P1.2
```

จะเซ็ทข้อมูลที่พอร์ท1 ได้เป็น 59H(01011001B)

CPL C

ความหมาย: คอมพลีเมนท์c-แฟลก

ENCODE:	1 0 1 1	0 0 1 1

CPL bit

ความหมาย: คอมพลีเมนท์บิทใดๆ ในแอดเดรสที่ต้องการ

ENCODE:	1 0 1 1	0 0 1 0	bit address

DA A

ความหมาย: ทำการปรับค่า 8บิท ในแอคคิวมูเลเตอร์ ซึ่งเป็นผลลัพธ์จากการบวกเกิน 2ตัวแปร(แต่ละตัวแปรเก็บค่าBCDไว้) และสร้างBCD 4บิทขึ้นมา2หลัก โดยอาจใช้คำสั่ง ADD หรือ ADDC เพื่อทำการบวกก็ได้

ถ้าแอคคิวมูเลเตอร์บิท0 ถึงบิท3 มีค่าเกิน9(xxxx1010-xxxx1111)หรือถ้า AC-แฟลกถูกเซ็ท จะบวก6เข้ากับแอคคิวมูเลเตอร์ เพื่อสร้างBCDที่ถูกต้องในบิทต่ำ การบวกที่เกิดขึ้นภายในนี้จะเซ็ทc-แฟลก ถ้ามีตัวทดออกจาก4บิทต่ำ ถ่ายทอดออกไปผ่าน 4บิทสูง แต่จะไม่ทำการเคลียร์c-แฟลก

ถ้าc-แฟลกถูกเซ็ทหรือ ถ้า4บิทสูงมีค่าเกิน9(1010xxxx-1111xxxx)จะบวก6 เข้ากับ4บิทสูงนี้ เพื่อสร้างBCDที่ถูกต้อง และจะเซ็ทc-แฟลกถ้ามีตัวทดออกจาก4บิทสูง ดังนั้นc-แฟลกจะชี้ผลบวกของBCD ทั้งสองเกิน100 เพื่อความแม่นยำของการบวก เลขฐานสิบ คำสั่งนี้ไม่มีผลต่อOV-แฟลก

ทุกอย่างที่เกิดขึ้นในระหว่างไซเกิลคำสั่ง จำเป็นที่จะต้องเปลี่ยนเป็นฐานสิบ โดยการบวก 00,06,60,66 เข้ากับแอคคิวมูเลเตอร์ ตามค่าเริ่มแรกของแอคคิวมูเลเตอร์และสภาวะของPSW

หมายเหตุ:- คำสั่งDA A ไม่สามารถแปลงเลขฐานสิบหกในแอคคิวมูเลเตอร์ ให้เป็นค่าBCDได้และใช้กับการลบเลขฐานสิบไม่ได้เช่นกัน

ตัวอย่าง: แอคคิวมูเลเตอร์มีค่า 56H(01010110B) ถูกแบ่งเป็นBCD 2หลัก และ

R3 มีค่า67H(01100111B) ถูกแบ่งเป็นBCD 2หลัก ขณะที่c-แฟลกถูกเซ็ท เมื่อทำคำสั่ง

```
ADDC  A,R3
DA  A
```

ขั้นแรกจะทำการบวกแบบ 2's complement ได้ผลลัพธ์คือ 0BEHเก็บไว้ในแอคคิวมูเลเตอร์ CและAC-แฟลกถูกเคลียร์

คำสั่งนี้จะเปลี่ยนค่าแอคคิวมูเลเตอร์เป็น 24H(00100100B) แบ่งเป็นBCD 2หลัก c-แฟลกจะถูกเซ็ทด้วยคำสั่งนี้แสดงว่าเกิด Overflowในฐานสิบ ผลบวกที่ถูกต้องของ 56,67และ1 คือ 124

ค่าBCD สามารถเพิ่มหรือลดโดยการบวก01H หรือ99H ถ้าแอคคิวมูเลเตอร์เริ่มแรกมีค่า30H(แสดงหลักของ30ฐานสิบ) เมื่อทำคำสั่ง

```
ADD  A,#99H
DA  A
```

จะย้ายข้อมูลใน c-แฟลก และข้อมูล 29H ไว้ในแอคคิวมูเลเตอร์เมื่อ 30+99=129,ไบท์ต่ำของผลบวกนั้นสามารถอธิบายด้วย 30-1=29

ENCODE:

1	1	0	1	0	1	0	0

DEC byte

ความหมาย: ทำการลดค่าตัวแปรลงหนึ่ง ถ้าค่าเริ่มต้นเป็น 00H จะunderflowเป็น0FFH คำสั่งนี้ไม่มีผลต่อแฟลก สามารถอ้างแอดเดรสได้4โหมด คือ accumulator,register,direct,register-indirect

หมายเหตุ:- เมื่อใช้คำสั่งนี้เพื่อเปลี่ยนแปลงเอาท์พุท ค่าที่ใช้เป็นพอร์ทเริ่มต้นจะอ่านจากข้อมูลที่แลทช์ไว้ทางเอาท์พุท ไม่ใช่อ่านจากขาอินพุท

ตัวอย่าง: R0มีค่า 7FH(01111111B) RAMภายในตำแหน่ง 7EH และ7FH มีข้อมูล 00H และ40H ตามลำดับ เมื่อทำคำสั่ง

```
DEC @R0
DEC R0
DEC @R0
```

R0จะมีข้อมูล7EH และRAMภายในตำแหน่ง7EHและ7FHจะมีข้อมูล 0FFHและ03FH ตามลำดับ

DEC A

ความหมาย: ลดค่าแอคคิวมูเลเตอร์ลงหนึ่ง

ENCODE:

0	0	0	1	0	1	0	0

DEC Rn

ความหมาย: ลดค่าRnลงหนึ่ง

ENCODE:

0	0	0	1	1	r	r	r

DEC direct

ความหมาย: ลดค่าของข้อมูลในตำแหน่งที่กำหนดลงหนึ่ง

ENCODE:

								direct address
0	0	0	1	0	1	0	1	

DEC @Ri

ความหมาย: ลดค่าของข้อมูลในแอดเดรสที่กำหนดด้วยR0 หรือR1 ลงหนึ่ง

ENCODE:

1	0	0	0	0	1	0	0

DIV AB

ความหมาย: ทำการหารจำนวนเต็ม 8บิท โดยไม่คิดเครื่องหมายซึ่งอยู่ใน แอคคิวมูเลเตอร์ ด้วยจำนวนเต็มชนิดเดียวกันใน reg B ผลหารที่ได้เก็บไว้ในแอคคิวมูเลเตอร์ และเศษเก็บไว้ใน reg B, cและov-แฟลกจะถูกเคลียร์

ถ้าreg Bมีค่า00H ผลลัพธ์จะไม่มีค่า และov-แฟลกจะถูกเซ็ท c-แฟลกจะถูกเคลียร์

ตัวอย่าง: แอคคิวมูเลเตอร์มีค่า251(0FBH)และมีค่า18(12H)เมื่อทำคำสั่ง DIV AB จะได้ 13เก็บไว้ในแอคคิวมูเลเตอร์(0DH)และ17(11H)เก็บไว้ในreg B

เมื่อ 251=(13*18)+17 cและov-แฟลกจะถูกเคลียร์

ENCODE:

1	0	0	0	0	1	0	0

DJNZ <byte>,<rel-addr>

ความหมาย: ทำการลดค่าในตำแหน่งที่ถูกชี้ลงหนึ่ง และกระโดดไปยังตำแหน่งที่ถูกชี้ด้วยโอเปอแรนด์ที่2 ถ้าผลลัพธ์ไม่เป็น0 ค่าที่เริ่มต้นจาก00Hจะunderflowเป็น 0FFH ค่าสั่งไม่มีผลต่อแฟลก แอดเดรสปลายทางสามารถคำนวนได้โดยการบวกแบบคิดเครื่องหมายrelative ในค่าสั่งไบท์สุดท้ายไปยังโปรแกรมเคาเตอร์ หลังจากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไป ตำแหน่งที่ถูกลดค่าอาจเป็น register หรือaddress byte

หมายเหตุ:- เมื่อใช้คำสั่งนี้เพื่อเปลี่ยนแปลงพอร์ทเอาท์พุท ค่าที่ใช้เป็นพอร์ทเริ่มต้นจะอ่านได้จากข้อมูลที่แลทช์ไว้ทางเอาท์พุท ไม่ใช่ขาอินพุท

ตัวอย่าง: RAMภายในตำแหน่ง40H,50Hและ60H มีข้อมูล 01H,70Hและ15Hตามลำดับ เมื่อทำคำสั่ง

```
        DJNZ  40H,LABEL1
        DJNZ  50H,LABEL2
        DJNZ  60H,LABEL3
```

จะกระโดดไปทำคำสั่งที่ label"LABEL2" ด้วยค่า 00H,6FH,และ15H ในตำแหน่งทั้ง3 ของRAM การกระโดดครั้งแรกจะไม่เกิดขึ้นเพราะผลลัพธ์เป็น0

คำสั่งนี้เป็นตัวอย่างโปรแกรมLOOP โดยให้การหน่วงเวลาพอประมาณ (จาก 2~512machine cycle) ด้วยคำสั่งเดียว เมื่อทำคำสั่ง

```
        MOV  R2,#8
TOOGLE: CPL  P1.7
        DJNZ  R2,TOGGLE
```

จะทำการ togglewอร์ท1.7 8ครั้ง ซึ่งจะได้ 4เอาท์พุทพัลซ์เกิดขึ้นที่บิท7 ของเอาท์พุทพอร์ท1 แต่ละพัลซ์จะมีค่า 3 machine cycle,2mcสำหรับDJNZ และ1mc สำหรับการเปลี่ยนแปลงค่าที่พอร์ท

DJNZ Rn,rel

ความหมาย: ลดค่าRnลงหนึ่งและจะกระโดดไปยังแอดเดรสที่ต้องการถ้าRnไม่เท่ากับ0

ENCODE:

1 1 0 1	1 r r r	rel address

DJNZ direct,rel

ความหมาย: ลดค่าในแอดเดรสที่กำหนด ถ้าค่ายังไม่เป็น0 จะบวกค่าโปรแกรมเคาเตอร์เดิมเข้ากับ rel-address แล้วเก็บไว้ในโปรแกรมเคาเตอร์ใหม่

ENCODE:	1 1 0 1	0 1 0 1	diretc address	rel.address

INC <byte>

ความหมาย: ทำการเพิ่มค่าตัวแปรที่ถูกชี้ขึ้นหนึ่ง ถ้าค่าเริ่มต้นเป็น 0FFH จะover-flowเป็น 00H และไม่มีผลต่อแฟลก

หมายเหตุ:- เมื่อใช้คำสั่งนี้เพื่อเปลี่ยนแปลงพอร์ทเอาท์พุท ค่าที่ใช้เป็นพอร์ทเริ่มต้นจะอ่านได้จากข้อมูลที่แลทช์ทางเอาท์พุท ไม่ใช่ขาอินพุท

INC A

ความหมาย: เพิ่มค่าแอคคิวมูเลเตอร์ขึ้นหนึ่ง

ENCODE:	0 0 0 0	0 1 0 0

INC Rn

ความหมาย: เพิ่มค่าRnขึ้นหนึ่ง

ENCODE:	0 0 0 0	0 r r r

INC direct

ความหมาย: เพิ่มค่าในแอดเดรสที่กำหนดขึ้นหนึ่ง

ENCODE:	0 0 0 0	0 1 0 1	diretc address

INC @Ri

ความหมาย: เพิ่มค่าในแอดเดรสที่กำหนดด้วยR0หรือR1ขึ้นหนึ่ง

ENCODE:	0 0 0 0	0 1 1 i

INC DPTR

ความหมาย: ทำการเพิ่มค่าตัวชี้ข้อมูลขนาด16บิทขึ้นหนึ่ง คำสั่งนี้ไม่มีผลต่อแฟลก

ตัวอย่าง: reg DPH และ DPL มีข้อมูล12Hและ0FEH ตามลำดับ เมื่อทำคำสั่ง

INC DPTR

INC DPTR

INC DPTR

จะเปลี่ยนDPHและDPLเป็น13และ01ตามลำดับ

ENCODE:

1 0 1 0	0 0 1 1

JB bit,rel

ความหมาย: ถ้าบิทที่ถูกชี้มีค่าเป็น1 จะกระโดดไปสู่แอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป ตำแหน่งปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมายrelative ในไบท์ที่3 ของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์ จากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไป บิทที่ตรวจสอบจะไม่ถูกเปลี่ยนแปลงและไม่มีผลต่อแฟลก

ตัวอย่าง: ข้อมูลที่อินพุทพอร์ท1 เป็น(11001010B) ข้อมูลในแอคคิวมูเลเตอร์มีค่า 56(01010110B) เมื่อทำคำสั่ง

```
JB  P1.2,LABEL1
JB  ACC.2,LABEL2
```

โปรแกรมจะกระโดดไปทำที่ LABEL2

ENCODE:

0 0 1 0	0 0 0 0	bit address	rel.address

JBC bit,rel

ความหมาย: ถ้าบิทที่ถูกชี้มีค่าเป็น1 จะกระโดดไปยังแอดเดรสที่ต้องการ แล้วเคลียร์บิทนั้นด้วย นอกนั้นจะทำคำสั่งต่อไป ถ้าบิทเป็น0แล้วจะไม่ถูกเคลียร์ แอดเดรสปลายทาง คำนวนโดยการบวกแบบคิดเครื่องหมายrelative ในไบท์ที่3 ของคำสั่งเพื่อส่งค่าไปยังโปรแกรมเคาเตอร์ จากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไปและไม่มีผลต่อแฟลก

หมายเหตุ:- เมื่อใช้คำสั่งนี้ตรวจสอบขาเอาท์พุท ค่าที่ใช้เป็นข้อมูลเริ่มต้น จะอ่านมาจากข้อมูลที่แลทช์ทางเอาท์พุท ไม่ใช่ขาอินพุท

ENCODE:

0 0 0 1	0 0 0 0	bit address	rel.address

JC rel

ความหมาย: ถ้า c-แฟลกถูกเซ็ทจะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมาย relative ในไบท์ที่2 ของคำสั่งเพื่อไปยังโปรแกรมเคาเตอร์ จากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ขึ้น 2ครั้ง คำสั่งนี้ไม่มีผลต่อแฟลก

ENCODE:	0 1 0 0	0 0 0 0	rel.address

JMP @A+DPTR

ความหมาย: บวกค่า8บิทแบบไม่คิดเครื่องหมาย ที่อยู่ในแอคคิวมูเลเตอร์ กับข้อมูล 16บิทใน DPTR และเก็บผลลัพธ์ไว้ในโปรแกรมเคาเตอร์เพื่อเป็นตำแหน่งสำหรับfetch คำสั่ง ตัวทดจาก 8บิทต่ำจะส่งผ่านไปทาง 8บิทสูง ซึ่งค่าของแอคคิวมูเลเตอร์และ DPTRจะไม่เปลี่ยนแปลง และไม่มีผลต่อแฟลก

ENCODE:	0 1 1 1	0 0 1 1

JNB bit,rel

ความหมาย: ถ้าบิทที่ถูกชี้มีค่าเป็น0 จะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมายrelative ในไบท์ที่3ของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์ จากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไป บิทที่ตรวจสอบไม่เปลี่ยนแปลงและไม่มีผลต่อแฟลก

ENCODE:	0 0 1 1	0 0 0 0	bit address	rel.address

JNC rel

ความหมาย: ถ้า c-แฟลกเป็น0 จะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมาย relative ในไบท์ที่2ของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์ หลังจากนั้นโปรแกรมเคาเตอร์จะเพิ่มค่าขึ้น2ครั้ง เพื่อชี้คำสั่งต่อไป และc-แฟลกจะไม่เปลี่ยนแปลง

ENCODE:	0 1 0 1	0 0 0 0	rel.address

JNZ rel

ความหมาย: ถ้าบิทใดบิทหนึ่งของแอคคิวมูเลเตอร์มีค่าเป็น1 จะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมาย relative ในไบท์ที่2 ของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์ หลังจากนั้น เพิ่มค่าโปรแกรมเคาเตอร์ขึ้น2ครั้ง แอคคิวมูเลเตอร์และแฟลก จะไม่เปลี่ยนแปลง

ENCODE:

0 1 1 1	0 0 0 0

rel.address

JZ rel

ความหมาย: ถ้าทุกบิทของแอคคิวมูเลเตอร์เป็น0 จะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไปแอดเดรสปลายทางคำนวนโดย การบวกแบบคิดเครื่องหมาย relativeในไบท์ที่2ของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์ หลังจากนั้นโปรแกรมเคาเตอร์จะเพิ่มค่าขึ้น2ครั้ง แอคคิวมูเลเตอร์และแฟลกจะไม่เปลี่ยนแปลง

ENCODE:

0 1 1 0	0 0 0 0

rel.address

LCALL addr16

ความหมาย: ทำการเรียกโปรแกรมย่อย ที่อยู่ในตำแหน่งที่ถูกชี้ โปรแกรมเคาเตอร์จะเพิ่มค่าขึ้น3ครั้ง เพื่อเป็นแอดเดรสของคำสั่งต่อไป และใส่ผลลัพธ์16บิทลงบนสแตค (ใส่8บิทต่ำก่อน) จากนั้นสแตคพอนย์เตอร์จะเพิ่มค่า2ครั้ง แล้วโหลดค่าโปรแกรมเคาเตอร์8บิทสูงและ8บิทต่ำตามลำดับ เป็นไบท์ที่2และ3ของคำสั่งLCALL การexecute โปรแกรม จะต่อเนื่องกับแอดเดรสนี้ โปรแกรมย่อยอาจเริ่มต้นที่แอดเดรสใดๆ ก็ได้ในหน่วยความจำโปรแกรม(64K) และไม่มีผลต่อแฟลก

ENCODE:

0 0 0 1	0 0 1 0

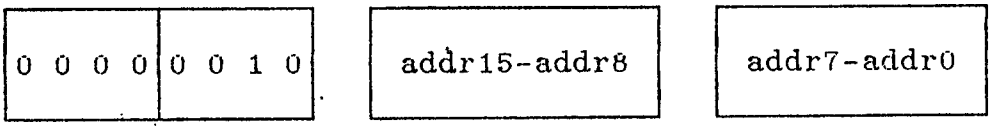
addr15-addr8

addr7-addr0

LJMP addr16

ความหมาย: ทำการกระโดดไปยังแอดเดรสที่ถูกชี้อย่างไม่มีเงื่อนไข โดยการโหลดค่าใส่ไบท์สูงและไบท์ต่ำของโปรแกรมเคาเตอร์ด้วยข้อมูลในไบท์ที่2 และ3ของคำสั่งตามลำดับ แอดเดรสปลายทางจะเป็นตำแหน่งใดๆ ใน64Kของหน่วยความจำโปรแกรม คำสั่งนี้ไม่มีผลต่อแฟลก

ENCODE:



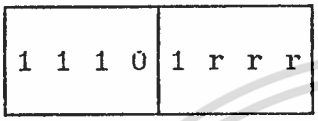
MOV <dest-byte>,<src-byte>

ความหมาย: ย้ายค่าในโอเปอแรนด์ที่2 มาไว้ในโอเปอแรนด์แรก(8บิท) และไม่มีผลต่อ<src-byte>และรีจิสเตอร์อื่นๆ

MOV A,Rn

ความหมาย: โหลดค่าในRnมาไว้ในแอคคิวมูเลเตอร์

ENCODE:



MOV A,direct

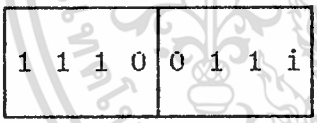
ความหมาย: โหลดค่าในแอดเดรสที่กำหนดมาไว้ในแอคคิวมูเลเตอร์

ENCODE:

MOV A,@Ri

ความหมาย: โหลดค่าในแอดเดรสที่กำหนดด้วยR0หรือR1 มาไว้ในแอคคิวมูเลเตอร์

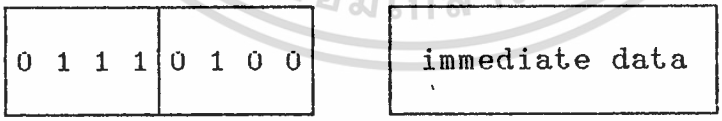
ENCODE:



MOV A,#data

ความหมาย: โหลดข้อมูลที่เข้ามากับคำสั่ง ไว้ในแอคคิวมูเลเตอร์

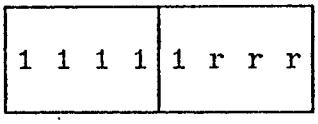
ENCODE:



MOV Rn,A

ความหมาย: โหลดข้อมูลในแอคคิวมูเลเตอร์มาไว้ในRn

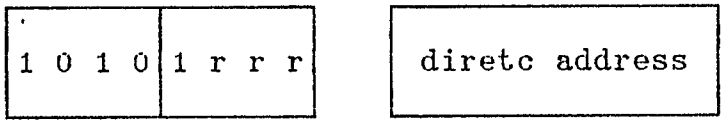
ENCODE:



MOV Rn,direct

ความหมาย: โหลดข้อมูลในแอดเดรสที่กำหนดมาไว้ในRn

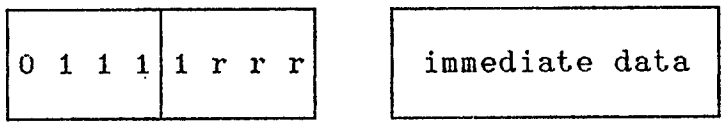
ENCODE:



MOV Rn,#data

ความหมาย: โหลดข้อมูลที่เข้ามากับคำสั่ง เก็บไว้ในRn

ENCODE:



MOV direct,A

ความหมาย: โหลดข้อมูลในแอคคิวมูเลเตอร์มาเก็บไว้ในแอดเดรสที่กำหนด

ENCODE:

MOV direct,Rn

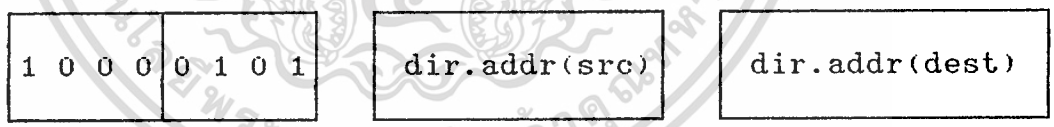
ความหมาย: โหลดค่าRnมาเก็บไว้ยังแอดเดรสที่กำหนด

ENCODE:

MOV direct,direct

ความหมาย: โหลดข้อมูลจากแอดเดรสหนึ่งมาเก็บไว้ในอีกแอดเดรสหนึ่ง

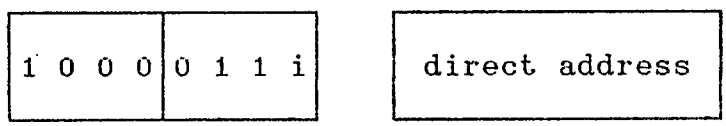
ENCODE:



MOV direct,@Ri

ความหมาย: โหลดข้อมูลจากแอดเดรสที่กำหนดโดยR0 หรือR1 มาเก็บไว้ในแอดเดรสที่กำหนด

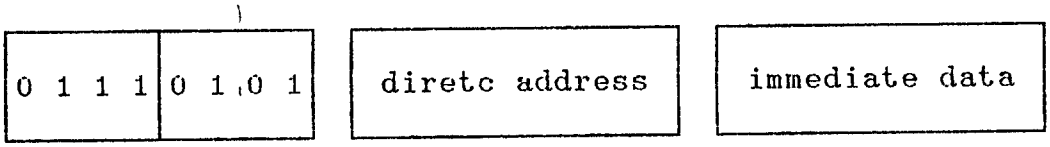
ENCODE:



MOV direct,#data

ความหมาย: โหลดข้อมูลที่เข้ามากับคำสั่ง มาเก็บไว้ในแอดเดรสที่กำหนด

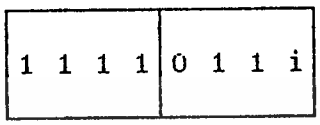
ENCODE:



MOV @Ri,A

ความหมาย: โหลดข้อมูลในแอคคิวมูเลเตอร์ มาเก็บไว้ในแอดเดรสที่กำหนดโดย R0หรือR1

ENCODE:



MOV @Ri,direct

ความหมาย: โหลดข้อมูลจากแอดเดรสที่กำหนด มาเก็บไว้ในแอดเดรสที่กำหนดโดย R0หรือR1

ENCODE:

MOV @Ri,#data

ความหมาย: โหลดข้อมูลที่เข้ามากับคำสั่งมาเก็บไว้ในแอดเดรสที่กำหนดโดยR0หรือR1

ENCODE:

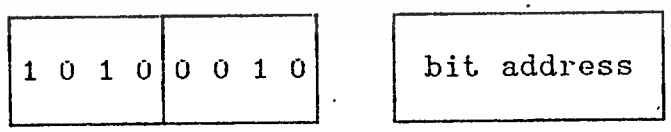
MOV <dest-bit>,<src-bit>

ความหมาย: โหลดค่า1บิทในโอเปอแรนด์ที่2มาไว้ในโอเปอแรนด์แรก คำสั่งนี้ไม่มีผลต่อแฟลกหรือรีจิสเตอร์ใดๆ

MOV C,bit

ความหมาย: โหลดค่า1บิทที่ต้องการ มาเก็บไว้ในc-แฟลก

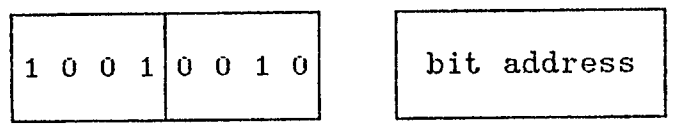
ENCODE:



MOV bit,C

ความหมาย: โหลดค่าจากc-แฟลก มาเก็บไว้ในแอดเดรสบิทที่ต้องการ

ENCODE:



MOV DPTR,#data16

ความหมาย: ทำการโหลด DPTRด้วยข้อมูล 16บิท โดยข้อมูลจะโหลดเข้าไปในไบท์ที่2และ3ของคำสั่ง โดยในไบท์ที่2เก็บ8บิทสูงและไบท์3เก็บ8บิทต่ำ คำสั่งนี้ไม่มีผลต่อแฟลก

ENCODE:	1 0 0 1	0 0 0 0	immed.data15-8	immed.data7-0

MOVC A,@A+<base-reg>

ความหมาย: ทำการโหลดค่าแอคคิวมูเลเตอร์ด้วย code8บิทหรือค่าคงที่จากหน่วยความจำโปรแกรม แอดเดรสดังกล่าวเป็นผลบวกของข้อมูล8บิทแบบไม่คิดเครื่องหมายในแอคคิวมูเลเตอร์กับข้อมูลในbase-regแบบ16บิท ซึ่งอาจเป็นDPTR หรือPC ก็ได้ ถ้าเป็นPC จะทำการเพิ่มค่าเพื่อไปยังแอดเดรสของคำสั่งที่ตามมาก่อนที่จะบวกกับแอคคิวมูเลเตอร์ มิฉะนั้นbase-regจะไม่เปลี่ยนแปลง การบวกแบบ16บิทจะเกิดตัวทดออกจาก8บิทต่ำและอาจจะผ่านเข้าไปยังบิทสูงได้ คำสั่งจะไม่มีผลต่อแฟลก

MOVC A,@A+DPTR

ความหมาย: ทำการบวกข้อมูลในหน่วยความจำที่กำหนดตำแหน่ง โดยแอคคิวมูเลเตอร์และDPTR แบบคิดตัวทดด้วย นำผลลัพธ์ที่ได้เก็บไว้ในแอคคิวมูเลเตอร์

ENCODE:	1 0 0 1	0 0 1 1

MOVC A,@A+PC

ความหมาย: ทำการบวกข้อมูลในหน่วยความจำที่กำหนดตำแหน่งโดยแอคคิวมูเลเตอร์และPC แบบคิดตัวทดด้วย นำผลลัพธ์ที่ได้เก็บไว้ในแอคคิวมูเลเตอร์

ENCODE:	1 0 0 0	0 0 1 1

MOVX <dest-byte>,<src-byte>

ความหมาย: ทำการย้ายข้อมูลระหว่างแอคคิวมูเลเตอร์ และข้อมูล8บิท จากหน่วยความจำข้อมูลภายนอก สังเกตจาก"X"ที่ต่อท้ายMOV โดยมีคำสั่งอยู่2ชนิดคือ ย้ายค่า8บิทหรือ16บิทที่อ้างแอดเดรสแบบdirectไปยังramภายนอก

ชนิดแรก ข้อมูลในR0หรือR1ในreg bank จะให้addr 8บิทmultiplexed กับข้อมูลบนP0 ค่า8บิทนี้เพียงพอสำหรับการdecode การขยายI/Oภายนอก หรือสำหรับ

ramขนาดเล็ก ถ้าพื้นที่ค่อนข้างใหญ่ขาพอร์ทเอาท์พุทจะใช้ต่อไปยังเอาท์พุทของแอดเดรสบิทสูง ขาเหล่านี้จะถูกควบคุมด้วยคำสั่งoutputที่มาก่อนคำสั่งmovx

ชนิดที่สอง DPTRจะสร้างแอดเดรส16บิท P2จะoutแอดเดรส8บิทสูง ในขณะที่ P0 multiplexed address 8บิทต่ำกับdata P2(sfr)จะเก็บค่าไว้ก่อน ขณะที่ P2(o/p buffer)จะปล่อยข้อมูลของDPH ลักษณะนี้จะเร็วและมีประสิทธิภาพเมื่อจัดการกับข้อมูลที่ใหญ่มาก และไม่ต้องเพิ่มคำสั่งในการตั้งเอาท์พุทพอร์ท

บางครั้งอาจรวมคำสั่งmovxทั้ง2ชนิดเข้าด้วยกัน ramขนาดใหญ่จะdriveแอดเดรส8บิทสูงด้วยP2 และอ้างแอดเดรสผ่านDPTR หรือด้วยcodeไบท์สูงไปยังP2 ตามหลังคำสั่งmovxโดยใช้R0หรือR1

MOVX A,@Ri

ความหมาย: โหลดค่าจากหน่วยความจำข้อมูลภายนอก(256byte) โดยกำหนดแอดเดรสผ่าน R0หรือR1เข้ามาเก็บไว้ในแอคคิวมูเลเตอร์

ENCODE:

1	1	1	0	0	0	1	i

MOVX A,@DPTR

ความหมาย: โหลดค่าจากหน่วยความจำข้อมูลภายนอก(64K) โดยกำหนดแอดเดรสผ่านDPTR เข้ามาเก็บไว้ในแอคคิวมูเลเตอร์

ENCODE:

1	1	1	0	0	0	0	0

MOVX @Ri,A

ความหมาย: โหลดค่าจากแอคคิวมูเลเตอร์ ไปไว้ยังหน่วยความจำข้อมูลภายนอกที่กำหนดแอดเดรส โดยR0หรือR1(256byte)

ENCODE:

1	1	1	1	0	0	1	i

MOVX @DPTR,A

ความหมาย: โหลดค่าจากแอคคิวมูเลเตอร์ ไปไว้ยังหน่วยความจำข้อมูลภายนอกที่กำหนดแอดเดรส โดยDPTR(64K)

ENCODE:

1	1	1	1	0	0	0	0

MUL AB

ความหมาย: ทำการคูณจำนวนเต็ม 8บิท แบบไม่คิดเครื่องหมายในแอคคิวมูเลเตอร์และreg B ผลคูณ8บิทต่ำจะเก็บไว้ในแอคคิวมูเลเตอร์ ส่วน8บิทสูงเก็บไว้ในreg B ถ้าผลคูณมีค่ามากกว่า255(0FFH) ov-แฟลกจะเซ็ทค่ามิฉะนั้นจะเคลียร์ c-แฟลกถูกเคลียร์

ENCODE:	1 0 1 0	0 1 0 0

NOP

ความหมาย: ทำการexecuteคำสั่งที่ตามมา นอกจากโปรแกรมเคาเตอร์แล้ว จะไม่มีผลต่อรีจิสเตอร์หรือแฟลกอื่นๆ

ENCODE:	0 0 0 0	0 0 0 0

ORL <dest-byte>,<src-byte>

ความหมาย: ทำlogic-OR ระหว่างสองโอเปอแรนด์และเก็บผลลัพธ์ไว้ที่<dest-byte>และไม่มีผลต่อแฟลก โอเปอแรนด์ทั้งสองอ้างแอดเดรสได้6โหมด เมื่อปลายทางเป็นแอคคิวมูเลเตอร์src-สามารถใช้ register,direct,reg-indirectหรือ immediate data

หมายเหตุ:- เมื่อคำสั่งนี้ถูกใช้เพื่อเปลี่ยนแปลงเอาท์พุทพอร์ทบางส่วน ค่าที่ใช้เป็นพอร์ทข้อมูลเริ่มต้น จะอ่านจากข้อมูลเอาท์พุทที่ถูกแลทช์เอาไว้ ไม่ใช่ขาอินพุท

ORL A,Rn

ความหมาย: ทำlogic orระหว่างค่าในแอคคิวมูเลเตอร์กับRn แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:	0 1 0 0	1 r r r

ORL A,direct

ความหมาย: ทำlogic - or ระหว่างค่าในแอคคิวมูเลเตอร์ กับข้อมูลในแอดเดรสที่กำหนด แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:	0 1 0 0	0 1 0 1	direct address

ORL A,@Ri

ความหมาย: ทำlogic-OR ระหว่างค่าในแอคคิวมูเลเตอร์ กับข้อมูลในแอดเดรสที่กำหนดด้วยR0หรือR1 แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 0 0	0 1 1 i

ORL A,#data

ความหมาย: ทำlogic-OR ระหว่างค่าในแอคคิวมูเลเตอร์ กับข้อมูลที่เข้ามากับคำสั่ง แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 0 0	0 1 0 0	immediate data

ORL direct,A

ความหมาย: ทำlogic-OR ระหว่างค่าในแอคคิวมูเลเตอร์ กับข้อมูลในแอดเดรสที่กำหนด แล้วเก็บผลลัพธ์ไว้ในแอดเดรสนั้น

ENCODE:

0 1 0 0	0 0 1 0	direct address

ORL direct,#data

ความหมาย: ทำlogic-OR ระหว่างข้อมูลที่เข้ามากับคำสั่ง กับข้อมูลในแอดเดรสที่กำหนด แล้วเก็บผลลัพธ์ไว้ในแอดเดรสนั้น

ENCODE:

0 1 0 0	0 0 1 1	direct address	immediate data

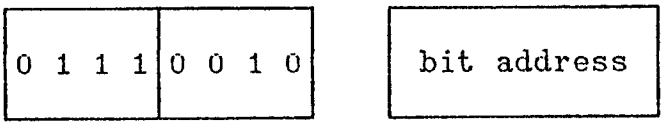
ORL C,<src-bit>

ความหมาย: ทำการเซ็ทค่า c-แฟลก ถ้าค่าทางบูลลีนมีlogic"1"มิฉะนั้นจะย้ายตัวทดไว้ใน current state เครื่องหมาย"/"แสดงว่าเป็นการคอมพลีเมนต์ เมื่อทำคำสั่งนี้ src-bit จะไม่เปลี่ยนแปลง และไม่มีผลต่อแฟลกอื่น

ORL C,bit

ความหมาย: ทำlogic-OR ระหว่างc-แฟลก กับแอดเดรสบิทที่ต้องการ แล้วเก็บผลลัพธ์ไว้ในc-แฟลก

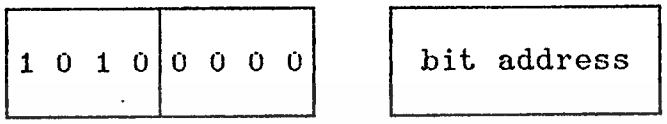
ENCODE:



ORL C,/bit

ความหมาย: ทำlogic-OR ระหว่างc-แฟลก กับแอดเดรสบิทที่คอมพลีเมนต์แล้วเก็บผลลัพธ์ไว้ใน c-แฟลก

ENCODE:



POP direct

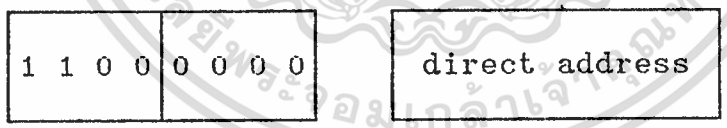
ความหมาย: ข้อมูลในสแตคพอน์ยเตอร์(sp) ถูกอ่านออกมาใส่ไว้ใน ram ภายในและ spก็จะลดค่าลงหนึ่ง ค่าที่อ่านได้จะย้ายไปยังdirect address ที่ถูกชี้ คำสั่งไม่มีผลต่อแฟลก

ENCODE:

PUSH direct

ความหมาย: สแตคพอน์ยเตอร์(sp) จะเพิ่มค่าขึ้นหนึ่ง จากนั้นข้อมูลในdirect address จะย้ายไปอยู่ใน ram ภายใน และเก็บไว้ใน sp คำสั่งไม่มีผลต่อแฟลก

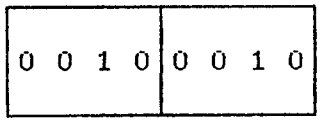
ENCODE:



RET

ความหมาย: เอาค่าในไบท์สูงและไบท์ต่ำของโปรแกรมเคาเตอร์(pc) ออกจากสแตคและลดค่า sp ลง2ค่า โดยทั่วไปค่าสั่งนี้จะใช้ต่อจาก ACALLหรือLCALL และไม่มีผลต่อแฟลก

ENCODE:



RETI

ความหมาย: เอาค่าในไบท์สูงและไบท์ต่ำของโปรแกรมเคาเตอร์ ออกจากสแตคและคืน interupt logic ให้กับตัวรับ interuptที่ระดับความสำคัญเดียวกัน sp จะลดค่าลง 2 ครั้ง ค่าสั่งไม่มีผลต่อแฟลก

ENCODE:	0 0 1 1	0 0 1 0

RL A

ความหมาย: rotate ค่าในแอคคิวมูเลเตอร์ไปทางซ้าย 1บิท โดยบิท7 rotate ไปยังบิท0 ค่าสั่งไม่มีผลต่อแฟลก

ENCODE:	0 0 1 0	0 0 1 1

RLC A

ความหมาย: rotate ทั้งค่าในแอคคิวมูเลเตอร์และc-แฟลก ไปทางซ้าย 1บิท โดยบิท7 rotate ไปยัง c-แฟลก และค่าเริ่มแรกของc-แฟลก rotateไปยังบิท0 ค่าสั่งไม่มีผลต่อแฟลก

ENCODE:	0 0 1 1	0 0 1 1

RR A

ความหมาย: rotate ค่าในแอคคิวมูเลเตอร์ไปทางขวา 1บิท, บิท0 rotate ไปยังบิท7 และไม่มีผลต่อแฟลก

ENCODE:	0 0 0 0	0 0 1 1

RRC A

ความหมาย: rotate ทั้งค่าในแอคคิวมูเลเตอร์และc-แฟลกไปทางขวา 1บิท, บิท0 rotate ไปยังc-แฟลก และค่าเริ่มต้นของc-แฟลกจะ rotate ไปยังบิท7 ค่าสั่งไม่มีผลต่อแฟลก

ENCODE:	0 0 0 1	0 0 1 1

SETB <bit>

ความหมาย: เซ็ทค่าบิทที่ถูกชี้ให้เป็น 1 คำสั่งนี้สามารถทำบน c-แฟลก และไม่มีผลต่อแฟลกอื่น

SETB C

ความหมาย: เซ็ท c-แฟลก ให้เป็น 1

ENCODE:	1 1 0 1	0 0 1 1

SETB bit

ความหมาย: เซ็ทค่าแอดเดรส ในบิทที่ต้องการ

ENCODE:	1 1 0 1	0 0 1 0	bit address

SJMP rel

ความหมาย: โปรแกรมควบคุมการกระโดดไปยัง แอดเดรสที่ถูกชี้โดยไม่มีเงื่อนไข แอดเดรสปลายทางคำนวนได้ โดยการบวกแบบคิดเครื่องหมายซึ่งอยู่ในไบท์ที่2 ของคำสั่ง แล้วจะโหลดไปให้กับโปรแกรมเคาเตอร์ จากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ขึ้น 2ค่า ดังนั้นขอบเขตของแอดเดรสปลายทางจะใช้ได้จาก 128ไบท์ก่อนหน้าคำสั่งนี้จนถึง 127ไบท์ ที่ตามหลังคำสั่งนี้

ENCODE:	1 0 0 0	0 0 0 0	rel.address

SUBB A,<src-byte>

ความหมาย: ทำการลบข้อมูลที่ถูกชี้และc-แฟลก ออกจากแอคคิวมูเลเตอร์ คำสั่งจะเซ็ท c-แฟลกถ้าบิท7 มีตัวยืม มิฉะนั้น c-แฟลกจะถูกเคลียร์ (ถ้าc-แฟลกถูกเซ็ทมาก่อนที่จะทำคำสั่งนี้ จะชี้ให้เห็นว่าตัวยืมนี้สำคัญด้วย ดังนั้นตัวทดจะถูกลบออกจากแอคคิวมูเลเตอร์พร้อมกับ<src-byte>) AC-แฟลกจะถูกเซ็ทถ้าบิท3ต้องการยืม มิฉะนั้นจะถูกเคลียร์ OV-แฟลกจะถูกเซ็ทถ้าบิท6ต้องการยืมแต่บิท7ไม่ต้องการ หรือบิท7ต้องการยืมแต่บิท6ไม่ต้องการ

เมื่อลบจำนวนเต็มแบบคิดเครื่องหมาย OV-แฟลกจะชี้ผลลัพธ์ถ้าเป็นค่าติดลบเมื่อค่าติดลบนี้ถูกลบจากค่าบวก หรือผลลัพธ์ที่เป็นบวกเมื่อค่าบวกนี้ถูกลบออกจากค่าลบ

SUBB A,Rn

ความหมาย: ลบข้อมูลใน c-แฟลกและRn ออกจากแอคคิวมูเลเตอร์แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:	1 0 0 1	1 r r r

SUBB A,direct

ความหมาย: ลบข้อมูลใน c-แฟลกและข้อมูลในแอดเดรสที่ต้องการ ออกจากแอคคิวมูเลเตอร์ แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:	1 0 0 1	0 1 0 1	direct address

SUBB A,@Ri

ความหมาย: ลบข้อมูลในc-แฟลกและข้อมูลในแอดเดรสที่กำหนดด้วยR0 หรือR1 ออกจากแอคคิวมูเลเตอร์ แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:	1 0 0 1	0 1 1 i

SUBB A,#data

ความหมาย: ลบข้อมูลในc-แฟลกและข้อมูลที่เข้ามากับคำสั่งออกจากแอคคิวมูเลเตอร์ แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:	1 0 0 1	0 1 0 0	immediate data

SWAP A

ความหมาย: สลับค่าระหว่าง 4บิทสูง กับ4บิทต่ำของแอคคิวมูเลเตอร์ การทำงานสามารถใช้แทนคำสั่งrotate 4bit ได้ และไม่มีผลต่อแฟลก

ENCODE:	1 1 0 0	0 1 0 0

XCH A,<byte>

ความหมาย: ทำการโหลดค่าแอคคิวมูเลเตอร์ด้วยข้อมูลของตัวแปรที่ถูกชี้ ในขณะเดียวกัน ก็จะโหลดข้อมูลของตัวแปรที่ถูกชี้ ด้วยแอคคิวมูเลเตอร์ ตัวแปร sorce และ destination สามารถใช้การอ้างแอดเดรสแบบ register,directหรือreg-indirect

XCH A,Rn

ความหมาย: สลับค่าระหว่างแอคคิวมูเลเตอร์กับข้อมูลในRn

ENCODE:

1 1 0 0	1 r r r

XCH A,direct

ความหมาย: สลับค่าระหว่างแอคคิวมูเลเตอร์กับข้อมูลในแอดเดรสที่กำหนด

ENCODE:

1 1 0 0	0 1 0 1	direct address

XCH A,@Ri

ความหมาย: สลับค่าระหว่างแอคคิวมูเลเตอร์ กับข้อมูลในแอดเดรสที่กำหนดด้วย R0 หรือR1

ENCODE:

1 1 0 0	0 1 1 i

XCHD A,@Ri

ความหมาย: ทำการแลกเปลี่ยนค่า4บิทต่ำของแอคคิวมูเลเตอร์(ปกติเป็นhexหรือbcd) กับค่าใน ram ภายในที่อ้างแอดเดรสแบบ spacial reg 4บิทสูงของ reg จะไม่เปลี่ยนแปลงและไม่มีผลต่อแฟลก

ENCODE:

1 1 0 1	0 1 1 i

XRL <dest-byte>,<src-byte>

ความหมาย: ทำ logic exclusive-OR ระหว่างตัวแปรที่ถูกชี้ แล้วเก็บผลลัพธ์ใน <dest-byte> ค่าสิ่งไม่มีผลต่อแฟลก โอเปอแรนด์ทั้งสองอ้างแอดเดรสได้6โหมด ถ้าแอดเดรสปลายทางเป็นแอคคิวมูเลเตอร์ sorceจะอ้างแอดเดรสแบบ register,

direct,reg-indirect,immediate เมื่อปลายทางเป็น direct,sorce ควรเป็นแอคคิวมูเลเตอร์ หรือ immediate data

หมายเหตุ: เมื่อใช้คำสั่งนี้เพื่อเปลี่ยนแปลงเอาท์พุทพอร์ท ค่าที่เป็นข้อมูลพอร์ทเริ่มต้นจะอ่านจากข้อมูลเอาท์พุทที่แลทช์ไว้ ไม่ใช่ที่ขาอินพุท

XRL A,Rn

ความหมาย: ทำ logic exclusive-OR ระหว่างแอคคิวมูเลเตอร์ กับRn แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 1 0	1 r r r

XRL A,direct

ความหมาย: ทำ logic exclusive-OR ระหว่างแอคคิวมูเลเตอร์ กับข้อมูลในแอดเดรสที่กำหนด แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 1 0	0 1 0 1	direct address

XRL A,@Ri

ความหมาย: ทำ logic exclusive-OR ระหว่างแอคคิวมูเลเตอร์ กับข้อมูลในแอดเดรสที่กำหนดด้วย R0 หรือ R1 แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 1 0	0 1 1 i

XRL A,#data

ความหมาย: ทำ logic exclusive-OR ระหว่างแอคคิวมูเลเตอร์ กับข้อมูลที่เข้ามากับคำสั่ง แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE:

0 1 1 0	0 1 0 0	immediate data

XRL direct,A

ความหมาย: ทำ logic exclusive-OR ระหว่างข้อมูลในแอดเดรสที่กำหนดกับแอคคิวมูเลเตอร์ แล้วเก็บผลลัพธ์ไว้ในแอดเดรสที่กำหนด

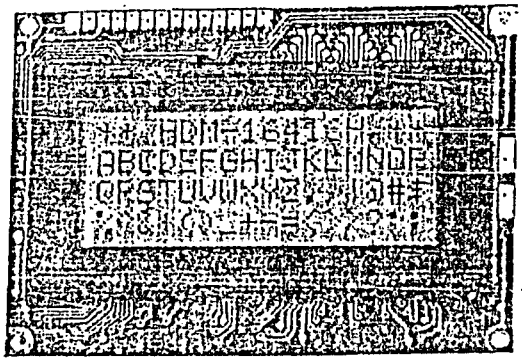
ENCODE:	0 1 1 0	0 0 1 0	direct address

XRL direct,#data

ความหมาย: ทำ logic exclusive-OR ระหว่างข้อมูลในแอดเดรสที่กำหนด กับข้อมูลที่เข้ามากับคำสั่ง แล้วเก็บผลลัพธ์ไว้ในแอดเดรสที่กำหนด

ENCODE:	0 1 1 0	0 0 1 1	direct address	immediate data

HDM-16416H



16 CHARACTERS X 4 LINES MODULE

PHYSICAL DATA

Module size	87.0W x 60.0H x 10.0D mm
Min. view area	61.8W x 25.2H mm
Character construction	5 x 7 dots
Character size	2.95W x 4.15H mm
Character pitch	3.55 mm
Dot size	0.55 x 0.55H mm
Weight	about 40g

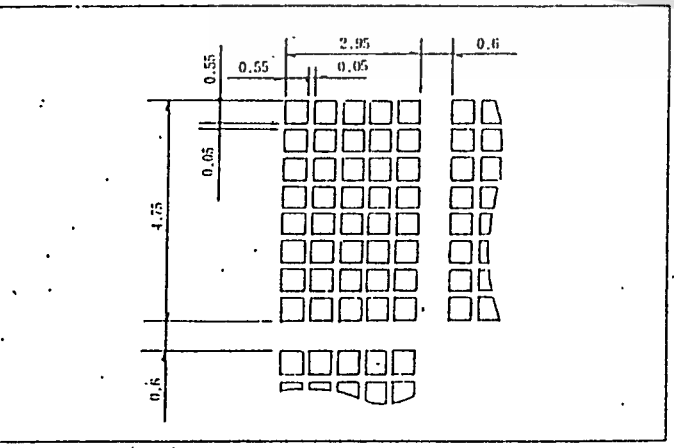
ABSOLUTE MAXIMUM RATINGS

	min	max
Power supply for logic (V_{DD} - V_{SS})	0	7.0 V
Power supply for LCD (V_{DD} - V_L)	0	13.5 V
Input voltage (V_{IN})	V_{SS}	V_{DD} V
Operation temperature (T_{OP})	0°C	50°C
Storage temperature (T_{STG})	-20°C	70°C

ELECTRICAL CHARACTERISTICS (V_{DD} =5.0±0.25V 25°C)

Input high voltage (V_{IH})		2.2V min.
Input low voltage (V_{IL})		0.6V max.
Output high voltage (V_{OH})	(-I_{OH}=0.205mA)	2.4V min.
Output low voltage (V_{OL})	(I_{OL}=1.2mA)	0.4V max.
Power supply current (I_{DD})	(V_{DD}=5.0V)	2.0mA typ.
		3.0mA max.
Drive method		1/16Duty
Power supply for LCD drive (V_{DD} - V_L)		
at Ta = 0°C		4.6V typ.
Ta = 25°C		4.4V typ.
Ta = 50°C		3.6V typ.

DISPLAY PATTERN

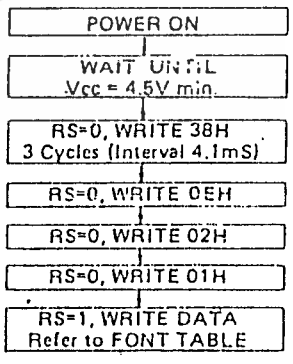


PIN CONNECTIONS

Pin No.	Symbol	Level	Function	
1	VSS	—	0 V	Power supply
2	VDD	—	5 V	
3	VL	—	5V-	
4	RS	H/L	H: Data input L: Instruction data input	
5	R/W	H/L	H: Data read L: Data write	
6	E	H,H→L	Enable signal	
7	D0	H/L	Data bus line*	
8	D1	H/L		
9	D2	H/L		
10	D3	H/L		
11	D4	H/L		
12	D5	H/L		
13	D6	H/L		
14	D7	H/L		

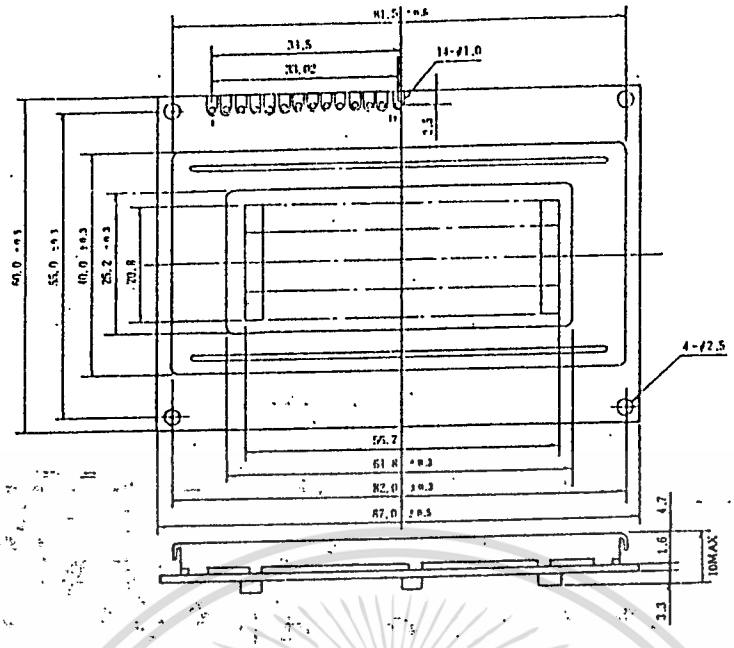
* In case of 4 bits instruction, data is transferred by twice using only 4 buses of D4-D7, and D0-D3 are not used; first operation is higher order 4 bits and second is lower 4 bits of 8 bits, but in case of 8 bits instruction, data is transferred by data bus of D0-D7.

TEST PROCEDURE

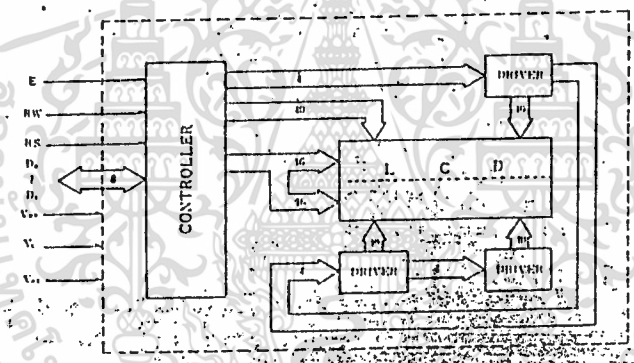


If above instruction is executed by 8 bits, 5 x 7 dots character will be displayed from left side of upper line and cursor moves to right.

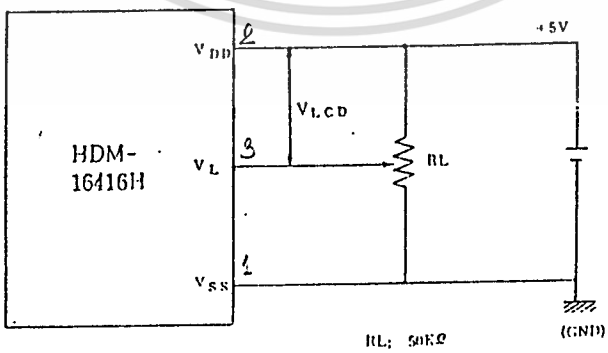
AL DIMENSIONS



OCK DIAGRAM



WER SUPPLY



UM82C8167

Real-Time Clock(RTC)

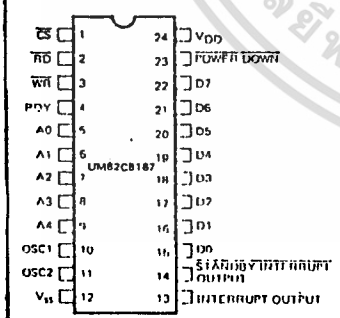
Features

- Microprocessor compatible (8-bit data bus)
- Milliseconds through month counters
- 56 bits of RAM with comparator to compare the real time counter to the RAM data
- 2 INTERRUPT OUTPUTS with 8 possible interrupt signals
- Single +5V power supply
- POWER DOWN input that disables all inputs and outputs except for one of the interrupts
- Status bit to indicate rollover during a read
- 32,768 Hz crystal oscillator
- Four-year calendar (no leap year)
- 24-hour clock
- 24 pin dual-in-line package

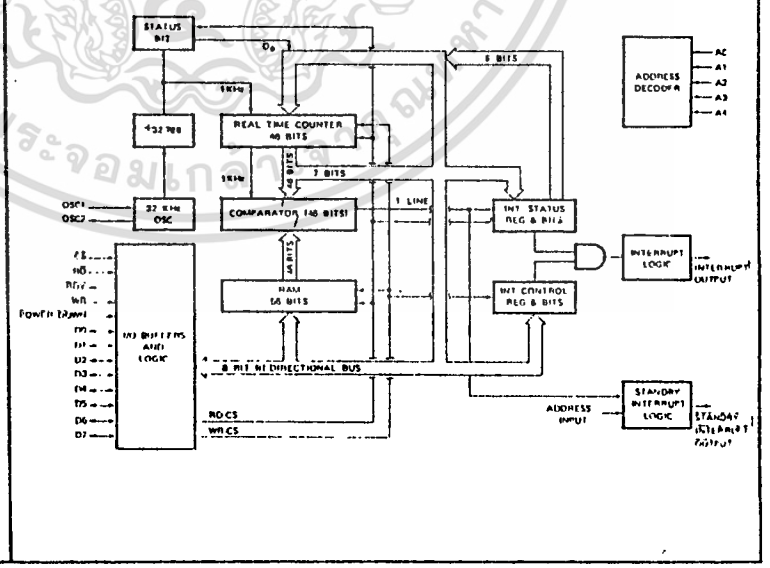
General Description

The UM82C8167 is a Si-gate CMOS LSI used as a real time clock in micro system. This product includes an addressable real time counter, 56 bits of static RAM and two interrupt outputs. User can disable the chip from the rest of the system for standby low power operation by using of a POWER DOWN input. With an on chip oscillation circuit, it can generate the 32,768 Hz time base.

Pin Configuration



Block Diagram



Absolute Maximum Ratings*

Voltage at All Inputs and Outputs
. . . . . V_{DD} +0.3 to V_{SS} −0.3
Operating Temperature. . . . . −25°C to +85°C
Storage Temperature . . . . . −65°C to +150°C
$V_{DD} - V_{SS}$ 6V

Comments*

Stresses above those listed under 'Absolute Maximum Rating' may cause permanent damage to the device. These are stress ratings only. Functional operation of this device at these or any other conditions above those indicated in the operational sections of this specification is not implied and exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Electrical Characteristics

(T_A = −25°C to +85°C, V_{SS} = 0V)

Parameter	Conditions	Min.	Typ.	Max.	Units
Supply Voltage					
V_{DD}	Outputs Enabled	4.0		5.5	V
V_{DD} (Note 1)	Power Down Mode	2.0		5.5	V
Supply Current					
I_{DD}, Static	Outputs TRI-STATE, f_{IN} = DC, V_{DD} = 5.5V			10	μA
I_{DD}, Dynamic	Outputs TRI-STATE, f_{IN} = 32 KHz, V_{DD} = 5.5V $V_{IH} \geq V_{DD} - 0.3V$, $V_{IL} \leq V_{SS} + 0.3V$			20	μA
I_{DD}, Dynamic	Outputs TRI-STATE, f_{IN} = 32 KHz V_{DD} = 5.5V, V_{IH} = 2.0V, V_{IL} = 0.8V			5	mA
Input Voltage					
V_{IL} Logical Low		0 0		0.8	V
V_{IH} Logical High		2.0		V_{DD}	V
I_L Input Leakage Current	$V_{SS} \leq V_{IN} \leq V_{DD}$	−1		1	μA
Output Voltage	(I/O and Interrupt Output)				
V_{OL} Logical Low	V_{DD} = 4.75V, I_{OL} = 1.6mA			0.4	V
V_{OH} Logical High	V_{DD} = 4.75V, I_{OH} = −400μA,	2.4			V
	I_{OH} = −10μA	0.8 V_{DD}			V
TRI-STATE®	V_{OUT} = 0V,			−1	μA
	V_{OUT} = V_{DD}			1	μA
Output Impedance	(Ready and Standby Interrupt Output)				
Logical Low, Sink	V_{DD} = 4.75V, I_{OL} = 1.6mA			0 4	V
Logical High, Leakage	$V_{OUT} \leq V_{DD}$			10	μA

Note 1: To insure that no illegal data is read from or written into the chip during power up, the power down input should be enabled only after all other lines (Read, Write, Chip Select, and Data Bus) are valid.

AC Characteristics

Interrupt Timing

(0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{DD} ≤ 5.5V, V_{SS} = 0V)

Symbol	Parameter	Min.	Max.	Units
t_{INTON}	Status Register Clock to INTERRUPT OUTPUT (Pin 13) High (Note 1)		5	μs
t_{SBYON}	Compare Valid to STANDBY INTERRUPT (Pin 14) Low (Note 1)		5	μs
t_{INTOFF}	Trailing Edge of Status Register Read to INTERRUPT OUTPUT Low		5	μs
t_{SBYOFF}	Trailing Edge of Write Cycle 9d0 = 0, Address = 16 H) to STAND BY BY INTERRUPT Off (high Impedance State)		5	μs

Note 1: The status register clocks are: The corresponding counter's. rollover to its reset state or the compare becoming valid. The compare becomes valid 61μs after the 1/10,000 of a second counter is clocked, if the real time counter data matches the RAM data.

Read Cycle Timing
($0°C \leq T_A \leq 70°C$, $4.5V \leq V_{DD} \leq 5.5V$, $V_{SS} = 0V$)

Symbol	Parameter	Min.	Max.	Units
t_{AR}	Address Bus Valid to Read Strobe	100		ns
t_{CSR}	Chip Select to Read Strobe	0		ns
t_{RRY}	Read Strobe to Ready Strobe		150	ns
t_{RYD}	Ready Strobe to Data Valid		800	ns
t_{AD}	Address Bus Valid to Data Valid		1050	ns
t_{RH}	Data Hold Time From Trailing Edge of Read Strobe	0		ns
t_{HZ}	Trailing Edge of Read Strobe to TRI-STATE Mode		250	ns
t_{RYH}	Read Hold Time after Ready Strobe	0		ns
t_{RA}	Address Bus Hold Time from Trailing Edge of Read Strobe	50		ns
t_{RYDV}	Rising Edge of Ready to Data Valid		100	ns

Note 2: If $t_{AR} = 0$ and Chip Select, Address Valid or Read are coincident then they must exist for 1050 ns.

Write Cycle Timing
($0°C \leq T_A \leq 70°C$ $4.5V \leq V_{DD} \leq 5.5V$, $V_{SS} = 0V$)

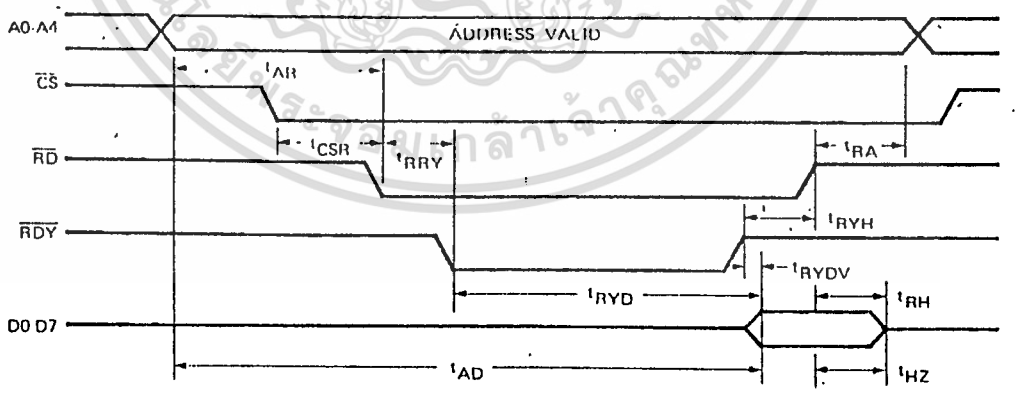
Symbol	Parameter	Min.	Max.	Units
t_{AW}	Address Valid to Write Strobe	100		ns
t_{CSW}	Chip Select to Write Strobe	0		ns
t_{DW}	Data Valid before Write Strobe	100		ns
t_{WRY}	Write Strobe to Ready Strobe		150	ns
t_{RY}	Ready 1 Strobe Width		800	ns
t_{RYH}	Write Hold Time after Ready Strobe	0		ns
t_{WD}	Data Hold Time after Write Strobe	50		ns
t_{WA}	Address Hold Time after Write Strobe	50		ns

Note 3: If data changes while CS and WR are low, then it must remain coincident with 1050 ns after the data change to ensure a valid writing.

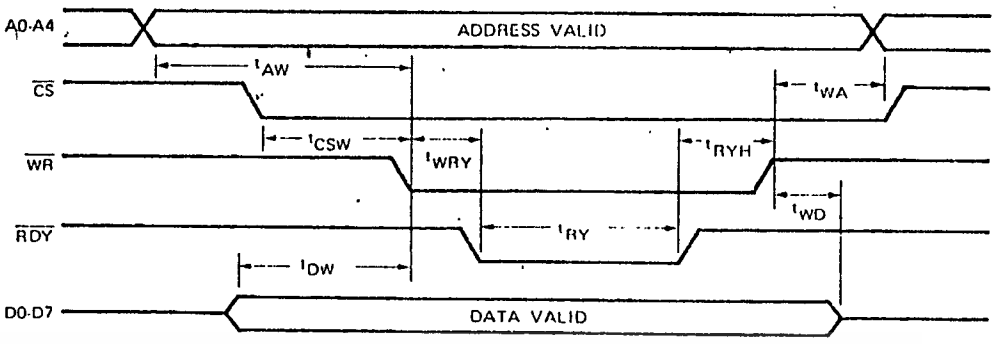
Data bus loading is 100 pF.
Ready output loading is 50 pF and 3 kΩ pull-up.
Input and output AC timing levels:
Logical one = 2.0V
Logical zero = 0.8V

Timing Waveforms

READ CYCLE TIMING



WRITE CYCLE TIMING



Functional Description

Real Time Counter

The real time counter is divided into 4-bit digits with 2 digits being accessed during any read or write cycle. Each digit represents a BCD number and is defined in Table 1. Any unused bits are held at a logical zero and ignored during a write. An unused bit is any bit not necessary to provide a full BCD number. For example, tens of hour can not legally exceed the number 2, thus only 2 bits are necessary to define the tens of hours. The other 2 bits in the tens of hours digit are unused. The unused bits are designated in Table 1 as dashes.

The addressable portion of the counter is from milliseconds to months. The counter itself is a ripple counter. The ripple delay is less than 60μs above 4.0V and 300μs at 2.0V.

Table 1. Real Time Counter Format

Counter Addressed		Units D0	D1	D2	D3	Max. BCD Code	Tens D4	D5	D6	D7	Max. BCD Code
1/10,000 of Seconds	(00_H)	–	–	–	–		D4	D5	D6	D7	9
Hundredths and Tenths Sec	(01_H)	D0	D1	D2	D3	9	D4	D5	D6	D7	9
Seconds	(02_H)	D0	D1	D2	D3	9	D4	D5	D6	–	5
Minutes	(03_H)	D0	D1	D2	D3	9	D4	D5	D6	–	5
Hours	(04_H)	D0	D1	D2	D3	9	D4	D5	–	–	2
Day of the Week	(05_H)	D0	D1	D2	–	7	–	–	–	–	0
Day of the Month	(06_H)	D0	D1	D2	D3	9	D4	D5	–	–	3
Month	(07_H)	D0	D1	D2	D3	9	D4	–	–	–	1

(–) Indicates unused bits

RAM

56 bits of RAM are contained on-chip. These can be used for any necessary power down storage or as an alarm latch for comparison to the real time counter. The data in the RAM can be compared to the real time counter on a digit basis. The only digits that are not compared are the unit ten thounandths of seconds and tens of days of the week (these are unused in the real time counter). If the two most significant bits of any RAM digit are ones then this RAM location will always compare.

The RAM is formatted the same as the real time counter, 4 bits per digits, 14 digits, however there are no nused bits. The unused bits in the real time counter will compare only to zeros in the RAM.

Interrupts and Comparator

There are two interrupt outputs. The first and most flexible is the INTERRUPT OUTPUT (a true high signal). This output can be programmed to provide 8 different output signals. They are: 10 Hz, 1 Hz, once per minut, once per hour, once a day, once a week, once a month, and when a RAM/real time counter comparison occurs. To enable the output a one is written into the interrupt control register at the bit location corresponding the desired output frequency (Figure 1). Once one or more bits have been set in the interrupt control register, the corresponding counter's rollover to in reset state will clock the interrupt status register and cause the interrupt output to go high. To reset the interrupt and to identify which frequency caused the interrupt, the interrupt status register is read. Reading this register places the contents of the status register on the data bus. The interrupt frequency will be identified by a one in the respective bit position. Removing the read will reset the interrupt.

The second interrupt is the STANDBY INTERRUPT (open drain output, active low). This interrupt occurs when enabled and when a RAM/ real time counter comparison occurs. The STANDBY INTERRUPT is enabled by writing a one on the D0 line at address 16H or disabled by writing a zero on the D0 line. This interrupt is not triggered by the edge of the compare signal, but rather by the level. Thus if the compare is enabled when the STANDBY INTERRUPT is enabled, the interrupt will turn on immediately.

The comparator is a cascaded exclusive NOR. Its output is latched 61μs after the rising edge of the 1KHz clock signal (input to the ten thousandth of seconds counter). This allows the counter to ripple through before looking at the comparator. For operation at less than 4.0V, the thousandth of seconds counters should not be included in a compare because of the possibility of having a ripple delay greater than 61μs. (For output timing see interrupt timing.)

Table 2 and 3 are refered for the address input codes and functions and for the counter and latch reset format.

Power Down Mode

The POWER DOWN input is essentially a second chip select. It disables all inputs and outputs except for the STANDBY INTERRUPT. When this input is at a logical zero, the device will not respond to any external signals. It will, however, maintain time keeping and turn on the STANDBY INTERRUPT if programmed to do so. (The programming must be down before the POWER DOWN input go to a logical zero.) When switching V_{DD} to the standby or power down mode, the POWER DOWN input should go to a logical zero at least 1μs before V_{DD} is switched. When switching V_{DD} all other inputs must remain between $V_{SS}-0.3V$ and $V_{DD}+0.3V$. When restoring V_{DD} to the normal operating mode, it is necessary to insure that all other inputs are at valid levels before switching the POWER DOWN input back to a logical one. These precautions are necessary to insure that no data

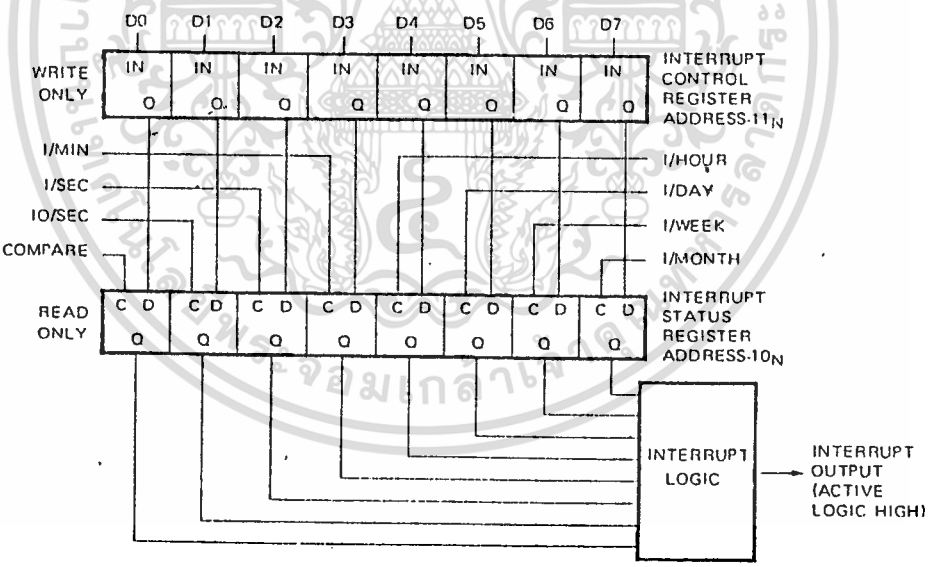


Figure 1. Interrupt Register Format

Table 2. Address Codes and Functions

A4	A3	A2	A1	A0.	Function
0	0	0	0	0	Counter – Thousandths of Seconds
0	0	0	0	1	Counter – Hundredths and Tenths of Seconds
0	0	0	1	0	Counter – Seconds
0	0	0	1	1	Counter – Minutes
0	0	1	0	0	Counter – Hours
0	0	1	0	1	Counter – Day of the Week
0	0	1	1	0	Counter – Day of the Month
0	0	1	1	1	Counter – Months
0	1	0	0	0	Latches – Thousandths of Seconds
0	1	0	0	1	Latches – Hundredths and Tenths of Seconds
0	1	0	1	0	Latches – Seconds
0	1	0	1	1	Latches – Minutes
0	1	1	0	0	Latches – Hours
0	1	1	0	1	Latches – Day of the Week
0	1	1	1	0	Latches – Day of the Month
0	1	1	1	1	Latches – Months
1	0	0	0	0	Interrupt Status Register
1	0	0	0	1	Interrupt Control Register
1	0	0	1	0	Counter Reset
1	0	0	1	1	Latch Reset
1	0	1	0	0	Status Bit
1	0	1	0	1	"GO" Command
1	0	1	1	0	Standby Interrupt
1	1	1	1	1	Test Mode

All others unused.

Table 3. Counter and Latch Reset Format

D0	D1	D2	D3	D4	D5	D6	D7	Counter or Latch Reset
1	0	0	0	0	0	0	0	Thousandths of Seconds
0	1	0	0	0	0	0	0	Hundredths and Tenths of Seconds
0	0	1	0	0	0	0	0	Seconds
0	0	0	1	0	0	0	0	Minutes
0	0	0	0	1	0	0	0	Hours
0	0	0	0	0	1	0	0	Days of the Week
0	0	0	0	0	0	1	0	Days of the Month
0	0	0	0	0	0	0	1	Months

For Counter Reset A4–A0 Must be 10010
For Latch Reset A4–A0 Must be 10011

is lost or altered when changing to or from the power down mode..

Counter and RAM Resets; GO Command

The counter's and RAM can be reset by writing all 1's (FF) at address 12H or 13H respectively.

A write pulse at address 15H will reset the thousandths, hunderedths, tentns, units, and tens of seconds counters. This GO command is used for precise starting of the clock. The data on the data bus is ignored during the writing. If the seconds counter is at a value greater than 39 when the GO is issued, the minute counter will increment; otherwise the minute counter is unaffected. This command is not necessary to start the clock, but merely a convinient way to start precisely at a given minute.

Status Bit

The status bit is provided to inform the user the clock is in the process of rolling over when a counter is read. The

status bit is set if this 1 KHz clock occurs during or after any counter read. This tells the user that the clock is rippling through the real time counter. Because the clock is rippling, invalid data may be read from the counter. If the status bit is set following a counter read, the counter should be reread.

The status bit appears on D0 when address 14H is read. All the other data lines will be zero. The bit is set when a logical one appears. This bit should be read every time a counter read or after a series of counter reads are done. The trailling edge of the read at address 14H will reset the status bit.

Oscillator

The oscillator is the standard parallel resonant oscillator. Externally, 2 capacitors, a 20M Ohms resistor and the crystal are required. The 20M Ohms resistor is connected between OSC IN and OSC OUT to bias the internal inverter in the linear region. For micropower crystals a resistor in series with the oscillator output may be necessary to insure the crystal is not overdriven. This resistor should be approximately 200K Ohms. The capacitor values should be typically 20 pF – 25 pF. The crystal frequency is 32,768 Hz.

The oscillator input can be externally driven, if desired. In this case the output should be left floating and the input level should be within 0.3V of the supplies.

A ground line or ground plane between pins 9 and 10 may be necessary to prevent interference of the oscillator by the A4 address.

Control Lines

The READ, WRITE, CHIP SELECT signals are active low inputs. The READY signal is an open drain output. At the start of each read or write cycle the READY line (open drain) will pull low and remain low until valid data from a chip read appears on the bus or data on the bus is latched in during a writing. READ and WRITE must be accompanied by a CHIP SELECT (see Timing waveforms for read and write cycle).

During a read or write, address bits must not change while chip select and control strobaes are low.

Test Mode

The test mode is merely a mode for production testing. It allows the counters to count at a higher than normal rate. In this mode the 32 KHz oscillator input is connected directly to the ten thousandths of seconds counter. The chip select and write lines must be low and the address must be held at 1 FH.

Typical Application

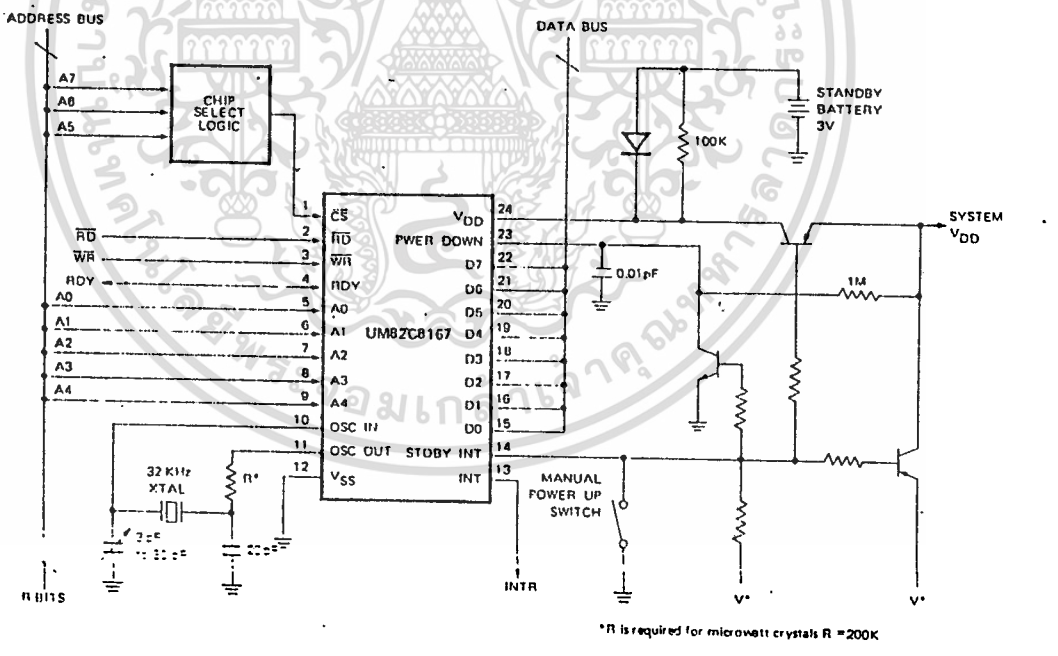


Figure 2. Standby Interrupt is Enable (ON) for Normal Operation and Disabled for Standby Operation

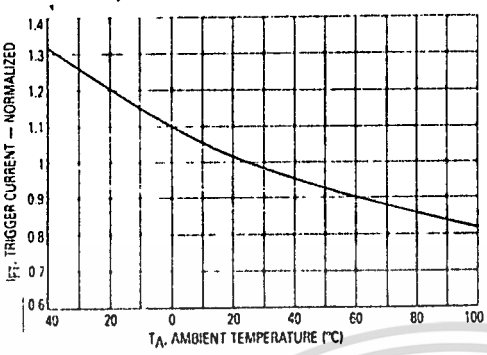


Figure 3. Trigger Current versus Temperature

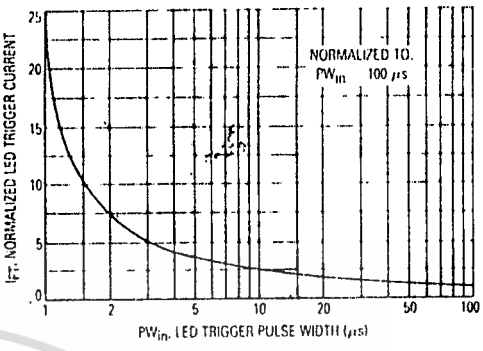


Figure 4. LED Current Required to Trigger versus LED Pulse Width

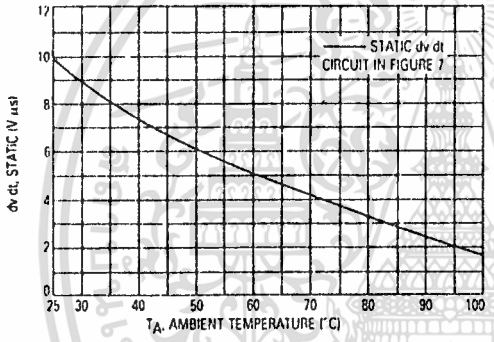


Figure 5. dv/dt versus Temperature

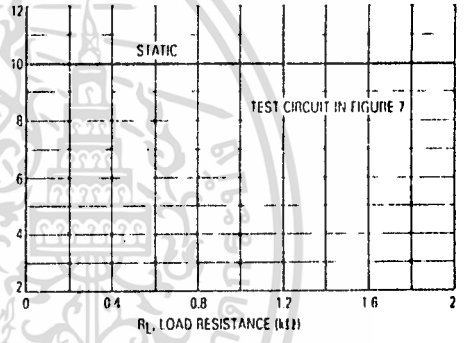
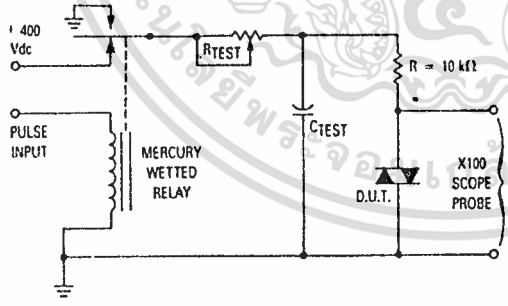


Figure 6. dv/dt versus Load Resistance



1. The mercury wetted relay provides a high speed repeated pulse to the D.U.T.
2. 100x scope probes are used, to allow high speeds and voltages.
3. The worst case condition for static dv/dt is established by triggering the D.U.T. with a normal LED input current, then removing the current. The variable R_{TEST} allows the dv/dt to be gradually increased until the D.U.T. continues to trigger in response to the applied voltage pulse, even after the LED current has been removed. The dv/dt is then decreased until the D.U.T. stops triggering. τ_{RC} is measured at this point and recorded.

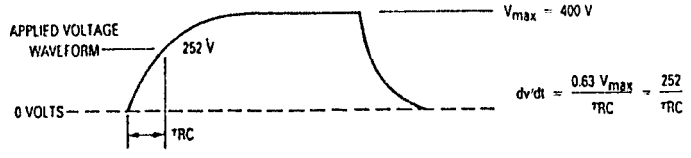


Figure 7. Static dv/dt Test Circuit

กิตติกรรมประกาศ

โครงงานและปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี ทั้งนี้ก็ด้วยความช่วยเหลือจากบุคคลหลาย ๆ ฝ่ายซึ่งต้องขอขอบคุณ ผศ.นิกร สุขุตมตันติ ที่ช่วยชี้แนะในด้านต่าง ๆ ตลอดจน อ.รุ่งนิรัญ เที่ยงธรรม และ คุณศมิทธิ์ เอมสมบัติ ที่ช่วยให้คำปรึกษาทางด้านซอร์ฟแวร์และฮาร์ดแวร์

ขอขอบคุณทางภาควิชาวิศวกรรมคอมพิวเตอร์ ที่ให้การสนับสนุนโครงงาน ในด้านเครื่องมือและอุปกรณ์ในการทดลอง ซึ่งมีส่วนทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

หนังสืออ้างอิง

1. สมยศ ภูงามแปลง. "MCS-51 ไมโครคอนโทรลเลอร์แบบชิพเดี่ยว". วารสารเซมิคอนดัคเตอร์อิเลคทรอนิคส์, ฉบับที่ 93, 2532 หน้า 264-269.
2. ผศ.พิพัฒน์ เลาหสงคราม. "ไมโครคอนโทรลเลอร์ MCS-51 (1)". วิศวสารลาดกระบัง, หน้า 60-68.
3. ผศ.พิพัฒน์ เลาหสงคราม. "ไมโครคอนโทรลเลอร์ MCS-51 (2)". วิศวสารลาดกระบัง, หน้า 49-60.
4. Handbook MCS-51, Intel Corporation, 1985
5. Curtis D. Johnson "Microprocessor-Based Process Control". Pretice-Hall, 1984