เครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้

PROGRAMMABLE ELECTRIC APPLIANCE CONTROLLER

โดย
นายทศพร ศรีสุข
นายพิพัฒน์ ชีพเจริญรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้

ผู้จัดทำ

1. นายทศพร ศรีสุข
2. นายพิพัฒน์ ชีพเจริญรัตน์

.....อาจารย์ที่ปรึกษา

(.....นายเกียรติศักดิ์ คงชีระ.....)

เครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้

นายทศพร ศรีสุข
นายพิพัฒน์ ชีพเจริญรัตน์
ผศ.ดร.เกียรติศักดิ์ คมวัชระ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์เพื่อใช้ควบคุมเครื่องใช้ไฟฟ้าภายในบ้านให้ทำงานได้ตามเวลาที่กำหนดเพื่อให้เกิดความปลอดภัยและเป็นการใช้พลังงานให้เกิดประโยชน์สูงสุด โดยเป็นการนำไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นส่วนประมวลผลการควบคุมเครื่องใช้ไฟฟ้า ซึ่งมีวงจรฐานเวลาอ้างอิงเป็นตัวอ้างอิงการทำงานของเครื่องใช้ไฟฟ้าให้ทำงานได้ตรงกับเวลาที่แท้จริง ลักษณะของงานจะสามารถควบคุมการทำงานโดยการป้อนค่าเวลาที่ต้องการเปิดปิดเครื่องใช้ไฟฟ้าผ่านทางคีย์บอร์ด แล้วนำค่าที่ได้ไปแสดงผลทางจอ LCD และนำไปประมวลผลเพื่อควบคุมการเปิดปิดวงจรรีเลย์ ให้เครื่องใช้ไฟฟ้าที่ต่ออยู่กับระบบทำงานได้ตามค่าเวลาที่ได้กำหนดไว้

PROGRAMMABLE ELECTRIC APPLIANCE CONTROLLER

Thossaporn Srisuk

Piphat Chepiareanrat

Asst.Prof.Dr. Kiattisak Kumwachara Advisor

2544

Abstract

The purpose of this project is to be able to control and manage electric appliance to work in time fixed for safety and able to use energy profit. That used to be the MCS-51 microcontroller is central processing unit for control electric appliance by real time clock is time reference, real time clock is reference for control electric appliance to work in real time. The control can be select time from keyboard for on or off electric appliance and shows time at LCD display. Finally data from keyboard will process in microcontroller for control relay to give electric appliance on or off along time fixed

สารบัญ

สารบัญรูปภาพ

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 กล่าวนำ

การนำอุปกรณ์ที่ใช้ช่วยอำนวยความสะดวกในการดำเนินชีวิตประจำวันนั้นจะสามารถพบเห็นได้มากมาย ซึ่งส่วนมากจะเป็นการควบคุมแบบอัตโนมัติและกึ่งอัตโนมัติ ในโครงงานนี้ได้กล่าวถึงการควบคุมการทำงานของเครื่องใช้ไฟฟ้าในที่อยู่อาศัยให้ทำงานได้ตามเวลาที่กำหนดเป็นการกำหนดค่าเวลาเปิดหรือปิดให้กับเครื่องใช้ไฟฟ้าให้ทำงานโดยการให้ทำตามโปรแกรมค่าเวลาที่ได้กำหนดไว้ล่วงหน้าได้

การทำงานของโครงงานนี้จะเป็นการการควบคุมการทำงานของเครื่องใช้ไฟฟ้าโดยนำไมโครคอนโทรลเลอร์ตระกูล MCS-51 มาประยุกต์ใช้งานร่วมกับวงจรต่าง ๆ ซึ่งถูกออแบบมาให้ผู้ใช้งานสั่งงานโดยทำการตั้งค่าแวลาการเปิดหรือปิดเครื่องใช้ไฟฟ้าเพื่อเรียกโปรแกรมต่าง ๆ ที่เก็บไว้ในหน่วยความจำของไมโครคอนโทรลเลอร์มาใช้งานได้ตามต้องการ ซึ่งข้อมูลที่ได้จากการสั่งงานจะถูกอ้างอิงกับฐานเวลาอ้างอิงของระบบ (Real Time Clock) เพื่อให้ได้ค่าเวลาที่แม่นยำกว่าการใช้ฐานเวลาที่อยู่ในไมโครคอนโทรลเลอร์ อีกทั้งยังเป็นการลดความยุ่งยากในการเขียนโปรแกรมลงด้วย

1.2 วัตถุประสงค์ของโครงงาน

โครงงานนี้มีวัตถุประสงค์เพื่ออำนวยความสะดวกและให้การใช้ชีวิตประจำวันเกิดความปลอดภัยมากยิ่งขึ้นอีกทั้งยังเป็นการช่วยอนุรักษ์พลังงานการใช้ทรัพยากรพลังงานไฟฟ้าให้คุ้มค่าอีกรูปแบบหนึ่งด้วย ซึ่งเครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้นี้จะสามารถทำการกำหนดค่าเวลาในการทำงานของเครื่องใช้ไฟฟ้าให้ทำงานได้ตามเวลาที่กำหนดถึงแม้ว่าผู้ใช้จะไม่อยู่บ้านโดยผู้ใช้จะสามารถทำการตั้งค่าเวลาต่าง ๆ ไว้ให้กับเครื่องใช้ไฟฟ้าล่วงหน้าได้เป็นวันหรือเป็นสัปดาห์ได้

1.3 ขอบเขตของโครงงาน

จากลักษณะของโครงงานที่ได้กล่าวมาจะประกอบไปด้วยอุปกรณ์แต่ละส่วนที่มีหน้าที่ต่างๆ กัน เพื่อทำการควบคุมการเปิดหรือปิดเครื่องใช้ไฟฟ้าดังนี้

1. วงจรไมโครคอนโทรลเลอร์ (microcontroller) เป็นหน่วยประมวลผลของระบบ (CPU) ใช้คำนวณและวิเคราะห์การทำงานโดยอาศัยโปรแกรม (Program) ที่เขียนขึ้นมาเป็นอินพุท เพื่อทำ

ให้วงจรเอาท์พุทรีเลย์สามารถไปควบคุมการเปิดหรือปิด (ON/OFF) ของเครื่องใช้ไฟฟ้าที่ต่ออยู่กับระบบได้ โดยตระกูลที่นำมาเลือกใช้ คือ MCS-51 เนื่องจาก เป็นที่นิยมอย่างแพร่หลายในงานควบคุม ใช้งานง่าย มีความยืดหยุ่นในการทำงาน

2. วงจรฐานเวลาอ้างอิง (Real Time Clock) เลือกใช้ IC เบอร์ DS1307 โดยภาคนี้จะทำหน้าที่เป็นฐานเวลาอ้างอิงของระบบซึ่งสามารถแสดงเป็น วัน / เดือน / ปี และ เวลา

3.วงจรรีเลย์เอาท์พุท (Relay output) ในส่วนนี้เป็นเอาท์พุทของระบบ จะนำสัญญาณที่ประมวลผลจาก CPU มาสั่งงานให้รีเลย์ทำงานเพื่อไปเปิดหรือปิด (ON/OFF) เครื่องใช้ไฟฟ้าที่ต่ออยู่กับระบบ

4.โปรแกรมที่ใช้ควบคุม (Program) เป็นส่วนที่เป็นอินพุทของระบบ โดยในส่วนนี้จะเป็นการเขียนโปรแกรมในแบบต่าง ๆ เพื่อมาควบคุมการทำงานของเครื่องใช้ไฟฟ้าที่ต่ออยู่ ซึ่งจะถูกเก็บอยู่ในหน่วยความจำแบบแฟลชของไมโครคอนโทรลเลอร์ (Memory)

5. วงจรสแกนคีย์บอร์ด (Scankey) เป็นส่วนที่ใช้สำหรับการป้อนค่าต่างๆ ให้กับระบบ ไม่ว่าจะเป็นการตั้งเวลา, การกำหนดค่าเวลาให้กับเอาท์พุท โดยในส่วนนี้จะทำหน้าที่ติดต่อระหว่างผู้ใช้กับโปรแกรมที่ถูกบรรจุไว้ในหน่วยความจำแบบแฟลชของไมโครคอนโทรลเลอร์เพื่อทำการเรียกโปรแกรมต่าง ๆ ขึ้นมาใช้งานได้ตามต้องการ

6. ชุดแสดงผล (Display) เลือกใช้ Display แบบ LCD เนื่องจากสามารถแสดงผลได้หลากหลายกว่าแบบ 7-segment ซึ่งจะสามารถทำให้ผู้ใช้ได้ง่ายขึ้นและเห็นเป็นรูปธรรมมากขึ้น

บทที่ 2
ทฤษฎี

2.1 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

2.1.1 โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่กล่าวถึงนี้จะอ้างถึงไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลช (flash memory) ของ Atmel Corporation มีเบอร์ขึ้นต้นด้วย AT89 เหตุผลที่ใช้ไมโครคอนโทรลเลอร์แบบนี้ในการกล่าวถึงเพื่อใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายประการดังนี้

1. หน่วยความจำโปรแกรมภายในตัวไมโครคอนโทรลเลอร์เป็นแบบแฟลช ทำให้สามารถลบและเขียนใหม่ได้นับพันครั้ง จึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดี่ยวไม่ต้องใช้หน่วยความจำภายนอกส่งผลให้สามารถใช้งานพอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

2.ต้นทุนและเวลาในการพัฒนาระบบไมโครคอนโทรลเลอร์ลดลงอย่างมากเนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูเลเตอร์และเครื่องโปรแกรมอีพรอม

3.บริษัทผู้ผลิตได้ทำการผลิตไมโครคอนโทรลเลอร์ตระกูลนี้ออกมาหลายเบอร์ และมีความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง

4.ด้วยการใช้หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ทำให้สามารถป้องกันการคัดลอกข้อมูลของหน่วยความจำโปรแกรมได้เป็นอย่างดี

5.ในบางเบอร์ของไมโครคอนโทรลเลอร์ที่ผลิตโดย Atmel สามารถทำการโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยที่ไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาทำกาโปรแกรมใหม่หรือเรียกว่า การโปรแกรมในวงจร หรือในระบบ (In-system programming) โดยใช้ลักษณะการติดต่อแบบ SPI (Serial Peripheral Interface) ทำให้การพัฒนาหรือการซ่อมบำรุง ตลอดจนการปรับปรุงหรืออัพเกรดข้อมูลในหน่วยความจำโปรแกรมทำได้อย่างสะดวกภายใต้งบประมาณที่ไม่สูงมากนัก

6.ชุดคำสั่งและสถาปัตยกรรมพื้นฐานเหมือนกับไมโครคอนโทรลเลอร์ MCS-51 ของผู้ผลิตไม่ว่าจะเป็นอินเทล, ซีเมนส์, หรือดัลลัส

คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต

- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง

-หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม

- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต

- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์

- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว

- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท

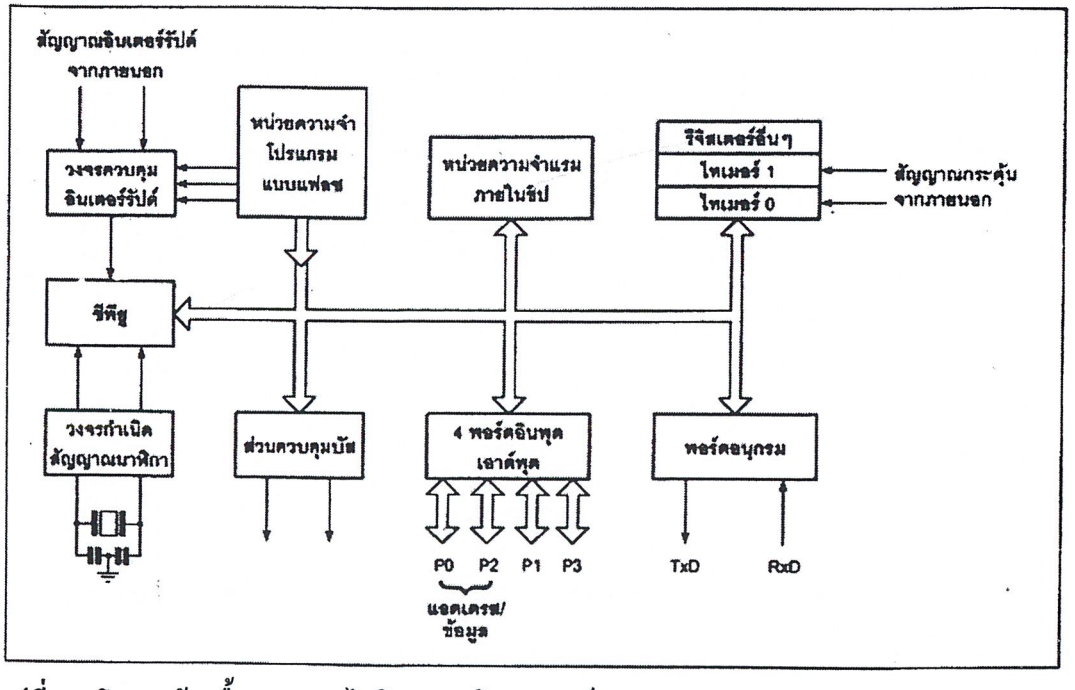
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์

- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป

- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx

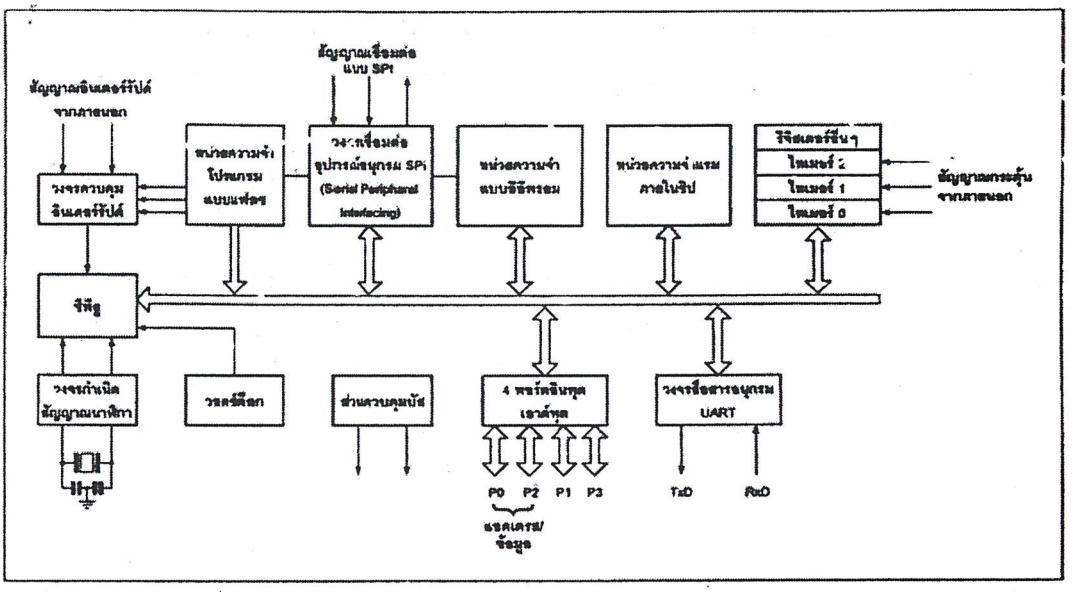
- มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

ในรูปที่ 2.1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx จะเห็นได้ว่า โครงสร้าง AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐานหากแต่แตกต่างกันเฉพาะหน่วยความจำโปรแกรมแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ในอนุกรม 87xx หน่วยความจำโปรแกรมภายในจะเป็นแบบอีพรอม และบางเบอร์สามารถโปรแกรมได้เพียงครั้งเดียว



รูปที่ 2.1 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89Cxx

สำหรับในรูปที่ 2.2 เป็นโครงสร้างพื้นฐานของอนุกรม AT89Sxx จะเห็นได้ว่ามีส่วนประกอบที่เพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน อาทิ วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์อนุกรมนี้ใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบหรือเรียกว่า การโปรแกรมในวงจร ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตที่เพิ่มเติมเข้ามาอีกหนึ่งตัวเป็นไทเมอร์ 2 และวงจรวอตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียู



รูปที่ 2.2 โครงสร้างของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89SXX

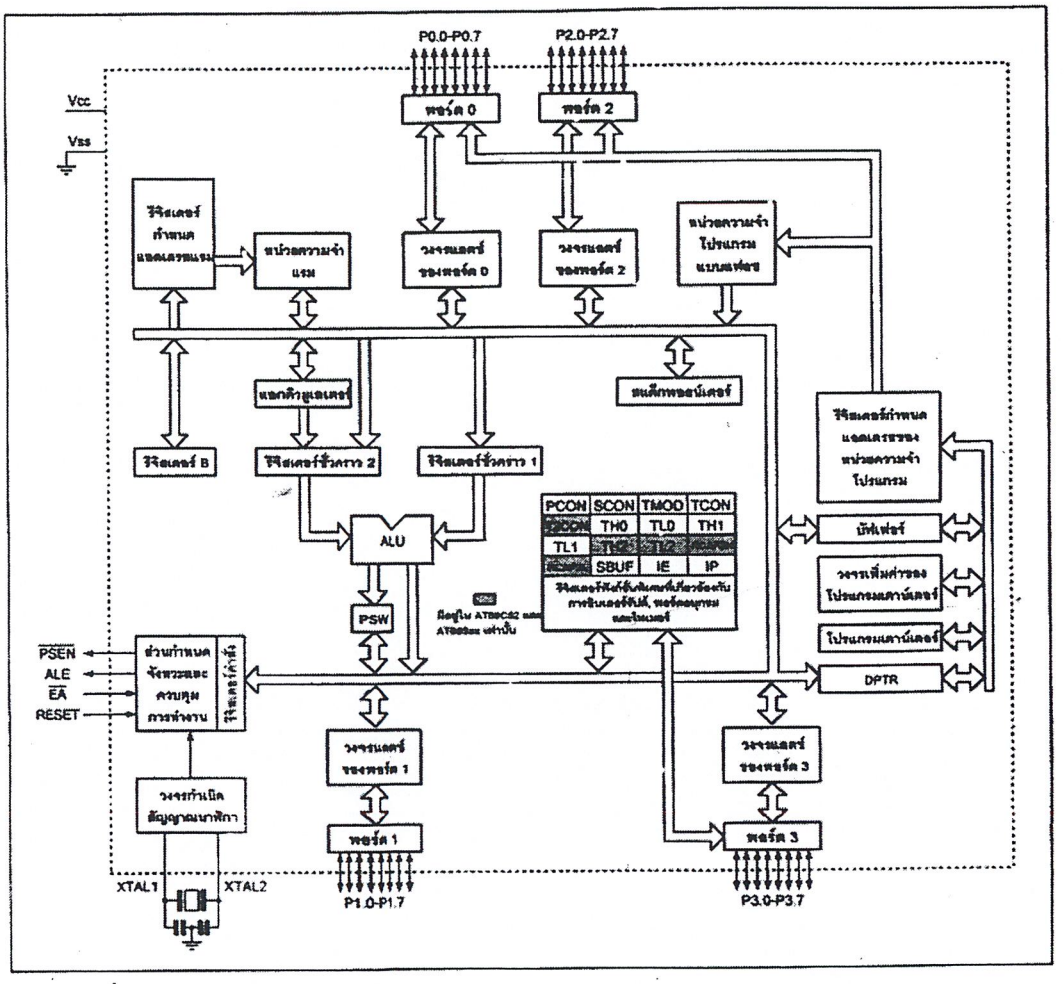
เบอร์ของไมโครคอนโทรลเลอร์	หน่วยความจำโปรแกรม	หน่วยความจำข้อมูล	จำนวนไทม์เมอร์/เคาน์เตอร์ 16 บิต
AT89C1051	แบบแฟลชขนาด 1 กิโลไบต์	แรม 64 ไบต์	1
AT89C2051	แบบแฟลชขนาด 2 กิโลไบต์	แรม 128 ไบต์	2
AT89C51	แบบแฟลชขนาด 4 กิโลไบต์	แรม 128 ไบต์	2
AT89C52	แบบแฟลชขนาด 8 กิโลไบต์	แรม 256 ไบต์	3
AT89C55	แบบแฟลชขนาด 20 กิโลไบต์	แรม 256 ไบต์	3
AT89S8252	แบบแฟลชขนาด 8 กิโลไบต์	แรม 256 ไบต์ อีอีพรอม 2 กิโลไบต์	3
AT89S53	แบบแฟลชขนาด 12 กิโลไบต์	แรม 256 ไบต์	3

ตารางที่ 2.1 รายละเอียดโดยสรุปบางส่วนของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่ Atmel ผลิตขึ้นและใช้ในการอ้างอิง

ในตารางที่ 2.1 แสดงรายละเอียดบางส่วนของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 แต่ละเบอร์ที่ Atmel ผลิตขึ้น และมีใช้งานอยู่ในปัจจุบัน

การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกันดังแสดงในรูปที่ 2.3 และ 2.4 โดยมีรายละเอียดขั้นต้น ดังนี้



รูปที่ 2.3 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต (P0.0 - P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียน ข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมี

สถานะปล่อยลอย (float) จึงมีอินพุทอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D 0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานให้เป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

ขาพอร์ต1 (P1.0 - 1.1) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89SXX จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทเมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

ขาพอร์ต2 (P2.0 – 2.7) มี 8 ขา แต่ละขาสามารถให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8 – A15)

ขาพอร์ต3 (P3.0 - P3.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นดังต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตร์รัปต์จากภายนอกช่องที่ 0 หรือขา $\overline{INT}$ 0

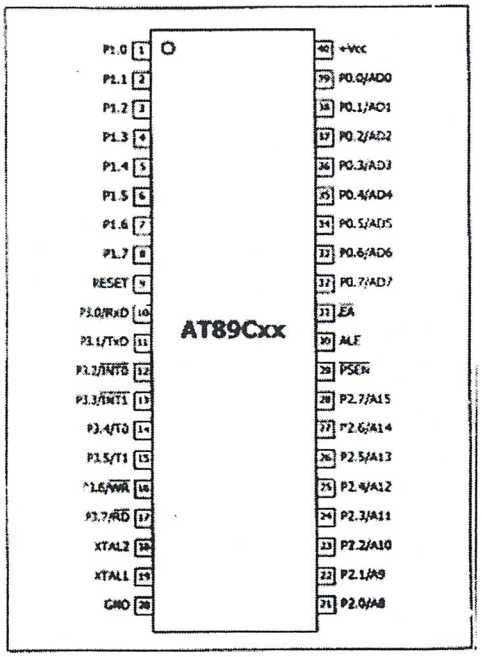
P3.3 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตร์รัปต์จากภายนอกช่องที่ 1 หรือขา $\overline{INT}$ 1

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่องที่ 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่ 1 หรือขา T1

P3.6 ใช้เป็นขาสัญญาณ $\overline{WR}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

P3.6 ใช้เป็นขาสัญญาณ $\overline{RD}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก



รูปที่ 2.4 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51

ขารีเซต ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซต สถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา ALE/$\overline{PROG}$ (Address Latch Enable/Program pulse input) เป็นขาที่ใช้ในการควบคุมการแลตช์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขาสำหรับรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีอีพรอม

$\overline{PSEN}$ **(Program Store Enable)** ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใด ๆ ออกมา

ขา $\overline{EA}$/Vpp (External Access enable/Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขา

นี้เป็น“1”เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ ที่ขานี้นยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12V

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

โครงสร้างและการทำงานของพอร์ต

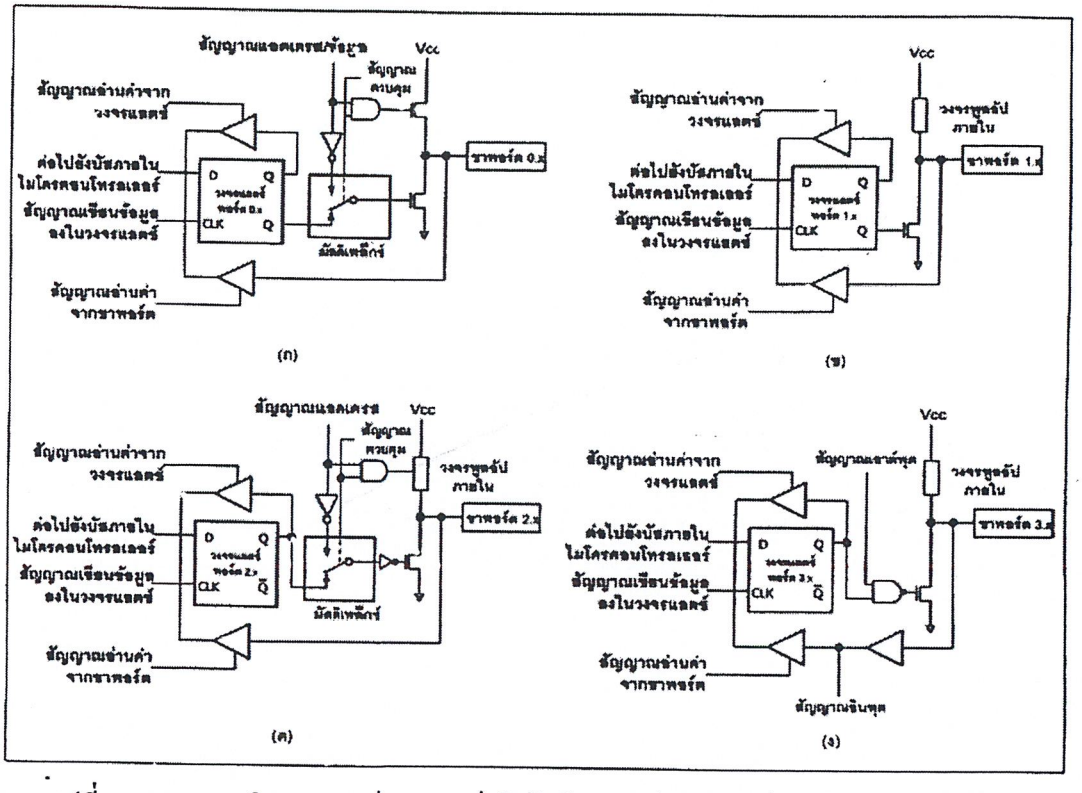
ไมโครคยนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึงพอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้าและเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงจรแลตช์และวงจรขับตลอดจนรบัฟเฟอร์อินพุต ดังแสดงให้เห็นในสถานปัตยกรรมในรูปที่ 2.3

ขา	เบอร์ของไมโครคอนโทรลเลอร์	หน้าที่พิเศษ
P1.0	AT89C52/AT89SXX	ขา T2 เป็นขาอินพุตนับค่าของไทเมอร์/เคาน์เตอร์ 2 และเป็นขาเอาต์พุตของการกำเนิดสัญญาณนาฬิกาโดยไทเมอร์ 2 (clock out)
P1.1	AT89C52/AT89SXX	ขา T2EX เป็นขาอินพุตทริกเกอร์สำหรับการแคปเจอร์/รีโหลดและควบคุมทิศทางของสัญญาณ
P1.4	AT89SXX	ขา $\overline{SS}$ (Slave Select) เป็นขาเลือกการติดต่อในกรณีที่ไมโครคอนโทรลเลอร์เป็นอุปกรณ์สเลฟ ในระบบการติดต่อแบบ SPI
P1.5	AT89SXX	ขา MOSI (Master data output, Slave data input) ใช้ในการติดต่อกับพอร์ต SPI
P1.6	AT89SXX	ขา MOSI (Master data output, Slave data input) ใช้ในการติดต่อกับพอร์ต SPI
P1.7	AT89SXX	ขา SCK (Master clock output) เป็นขาสัญญาณนาฬิกาของการติดต่อกับพอร์ต SPI

ตารางที่ 2.2 หน้าที่พิเศษของพอร์ต 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ที่พอร์ต 0 และพอร์ต 2 จะใช้งานเป็นพอร์ตอินพุตและเอาต์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ตและพอร์ต 1 บางขานอกจากจะใช้เป็นขาพอร์ตอินพุตเอาต์พุตเอาต์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด ดังสรุปได้ในตารางที่ 2.2

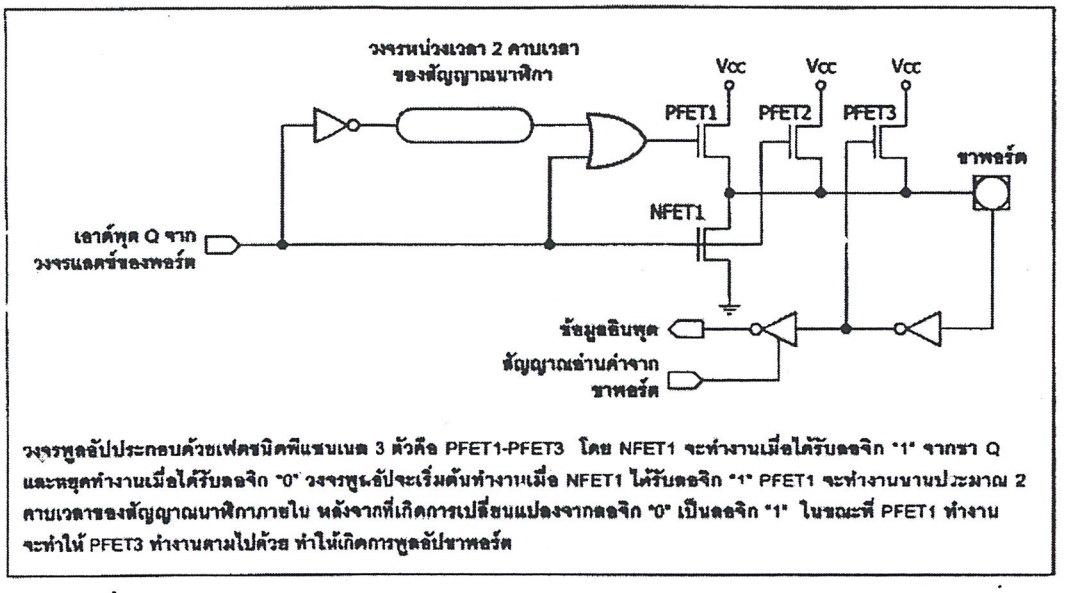
ในรูปที่ 2.5 แสดงวงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดยในรูปที่ 2.5 (ก) เป็นวงจรของพอร์ต 0 วงจรแลตช์ของแต่ละบิตในแต่ละพอรีตก็คือวงจรดีฟลิปฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระต่อกันด้วยสัญญาณที่แยกจากกัน นั่นคือ สัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิปฟลอปในขณะที่ข้อมูลจะส่งผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิปฟลอป



รูปที่ 2.5 วงจรภายในของพอร์ตทุกพอร์ตในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในรูปที่ 2.5 (ข) เป็นวงจรของพอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน สำหรับรายละเอียดของวงจรพูลอัปแสดงในรูปที่ 2.6

ในรูปที่ 2.5 (ค) เป็นวงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัปเพิ่มเติมเข้ามา ส่วนในรูปที่ 2.5 (ง) เป็นวงจรภายในของพอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์ และวงจรอินพุตเอาต์พุตเมื่อทำงานในฟังก์ชั่นพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานในหน้าที่พิเศษได้ทุกขา



รูปที่ 2-6 วงจรพูลอัปภายในพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุต ต้องเริ่มต้นด้วยการเขียนข้อมูล "1" มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้น ๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น"1"สามารถรับสัญญาณลอจิก"0"จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไปเมื่อเป็นเช่นนี้ อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชควรกำหนดให้ทำงานในสภาวะลอจิก"0"จะดีและสะดวกที่สุด (ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก "0" แล้ว)

การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล "0" ออกไปทางเอาต์พุตก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็จะส่งผลต่อไปขับเฟต ทำให้เฟตทำงานที่ขาพอร์ตที่กำหนดให้ทำงานเกิดลอจิก "0" ขึ้น ในทางตรงกันข้ามหากต้องการส่งข้อมูล "1" ออกไป ก็ให้เขียนข้อมูล "1" ไปยังวงจรแลตช์ วงจรขับจะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเป็นพอร์ตเอาต์พุต แต่ละขา (หรือแต่ละบิต) ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแสหรือที่เรียกว่า **กระแสซอร์ส (source current)** ได้สูงสุด 10 mA และทุกขารวมกันในแต่ละพอร์ต (ทั้ง 8 บิต) สูงสุด 26 mA สำหรับพอร์ต 0 และ 15 mA สำหรับพอร์ต 1-3 ในกรณีที่ใช้งานทุกพอร์ตเอาต์พุตจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาต์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาต์พุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง

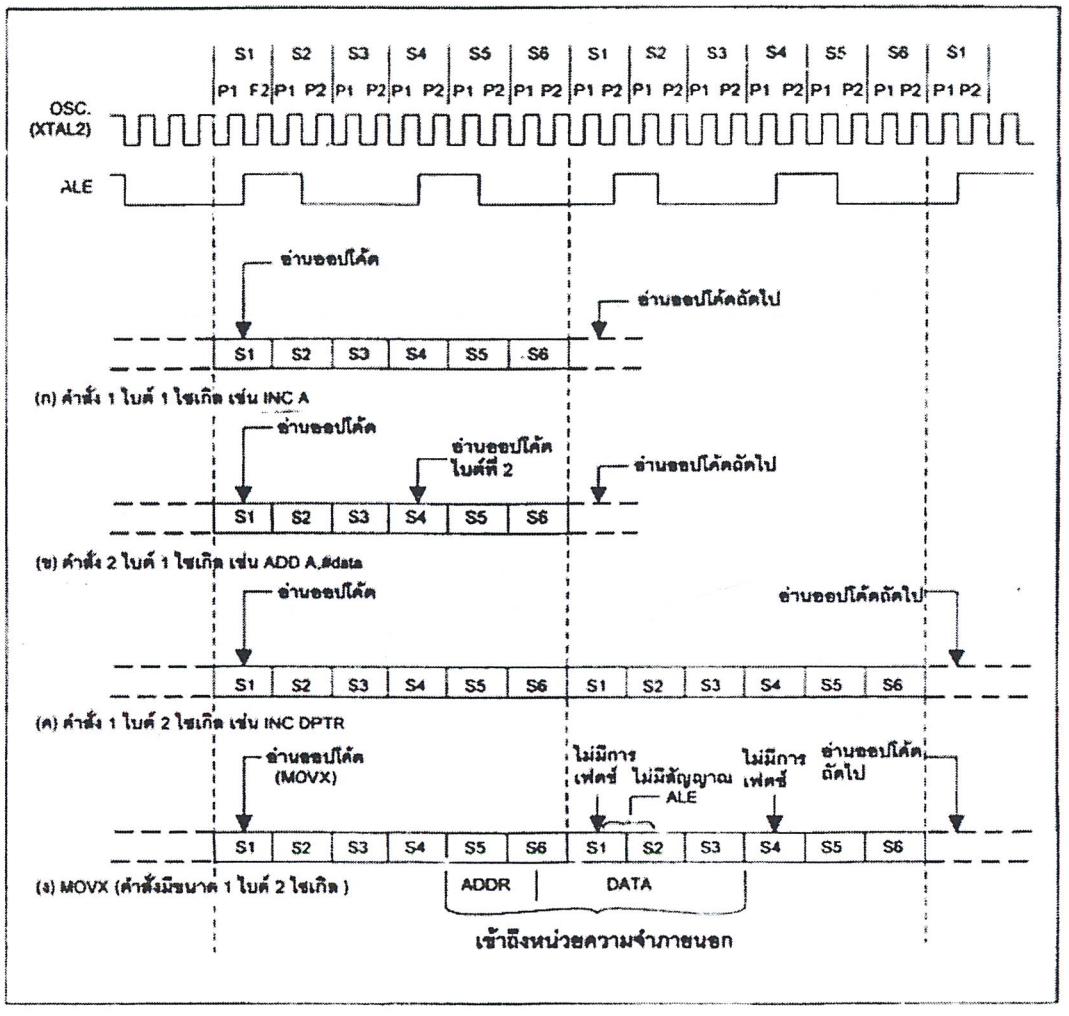
การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะ คือ อ่านจากขาพอร์ต โดยตรง และอ่านจากวงจรแลตช์ของแต่ละพอร์ต

ในกรณีที่พอร์ตต่อกับขาเบสทรานซิสเตอร์ชนิด NPN และขาอิมิตเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวด์ หากมีการส่งข้อมูล "1" ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงาน สถานะลอจิกที่ขาพอร์ตจะเป็น "0" เนื่องจากเมื่อทรานซิสเตอร์ทำงาน จะเสมือนว่าพอร์ตนั้นถูกต่อลงกราวด์ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่ถ้าหากทำงานอ่านค่าลอจิกที่วงจรแลตช์ จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริง ดังนั้น ในการอ่านค่าลอจิกพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย

จังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51

ในการใช้งานไมโครคอนโทรลเลอร์ MCS-51 จะต้องทำความเข้าใจถึงจังหวะการทำงานของซีพียูและลำดับขั้นตอนการประมวลผลคำสั่ง ในการประมวลผลคำสั่งของของซีพียูจะมีขั้นตอนหลัก ๆ 2 ขั้นตอนคือ **กระบวนการเฟตช์ (fetch)** เป็นการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วทำการแปลงรหัสคำสั่งนั้นเป็นภาษาเครื่องเพื่อเตรียมการประมวลผล ขั้นตอนต่อมาคือ **กระบวนการเอ็กซิคิวท์ (execute)** เป็นการกระทำตามคำสั่งที่กำหนดหรือตามที่เฟตช์ขึ้นมาโดยกระบวนการก่อนหน้านี้ เมื่อทำการเอ็กซิคิวท์คำสั่งเรียบร้อยแล้ว ก็จะไปเริ่มกระบวนการเฟตช์คำสั่งต่อไป



รูปที่ 2.7 ไซเกิลการทำงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

เมื่อเริ่มจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชจะเกิดการรีเซตในลักษณะที่เรียกว่า **เพาเวอร์ออนรีเซต (power-on reset)** ทำให้ซีพียูไปเริ่มต้นการทำงานที่แอดเดรส 0000H

ของหน่วยความจำโปรแกรมจังหวะการทำงานของซีพียูจะเป็นไปตามรูปแบบ โดยได้รับการกำหนดมาจากกรอบการทำงานหรือแมชชีนไซเกิล (machinecycle) ในรูปที่ 2.7 เป็นไดอะแกรมเวลาแสดงจังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51 โดยใน 1 รอบการทำงานหรือแมชชีนไซเกิลจะแบ่งย่อยออกเป็น 6 เสตท (state) กำหนดชื่อเป็น S1-S6 ในแต่ละสเตทมีค่าเวลาเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา ถ้าสัญญาณนาฬิกามีความถี่ 12 MHz จะมีคาบเวลาเท่ากับ 1 ms คาบเวลาทั้งสองภายในหนึ่งสเตตจะเรียกว่า เฟส 1 (phase1) และเฟส 2 (phase2)

ในรูปที่ 2.7 (ก) และ (ข) จะเป็นการเอ็กซิคิวต์คำสั่งที่ใช้เวลา 1 ไซเกิล เริ่มต้นที่สเตท 1 จะเป็นการอ่านค่าออปโค้ดอันเป็นกระบวนการแลตช์ค่าของออปโค้ดส่งไปให้รีจิสเตอร์คำสั่ง (Instruction Register : IR) การเฟตช์ครั้งที่สองจะเกิดขึ้นที่สเตท 4 ภายในแมชชีนไซเกิลเดียวกัน ในกรณีที่เป็นคำสั่งไบต์เดียว การเฟตช์ครั้งที่ 2 ภายในแมชชีนไซเกิลเดียวกันจะถูกตัดทิ้งไป ในคำสั่งที่มีใช้เวลา 1 ไซเกิลจะสิ้นสุดการทำงานลงในสเตท 6 ของแมชชีนไซเกิลเดียวกัน

ในกรณีที่คำสั่งใช้เวลา 2 ไซเกิล การทำงานของคำสั่งนั้นจะสิ้นสุดลงในสเตท 6 ของแมชชีนไซเกิลที่สอง ดังในดอะแกรมรูปที่ 2.7 (ค) สำหรับในการกระทำคสั่ง MOVX ซึ่งเป็นคำสั่งขนาด 1 ไบต์ 2 ไซเกิล จะไม่มีการเฟตช์เกิดขึ้นในไซเกิลที่สองของคำสั่ง MOVX นี้ เนื่องจากซีพียูจะไปทำการติดต่อกับหน่วยความจำภายนอกดังแสดงในไดอะแกรมรูปที่ 2.7 (ง) จะเห็นได้ว่า เวลาในการเอ็กซิคิวท์จะไม่ได้ขึ้นอยู่กับว่าการติดต่อหน่วยความจำโปรแกรมภายในหรือภายนอก

ในรูปที่ 2.7 แสดงสัญญาณและไดอะแกรมเวลาของการเข้าถึงหน่วยความจำโปรแกรมภายนอก โดยในรูปที่ 2.7 (ก) เป็นไดอะแกรมในขณะที่ยังไม่มีการกระทำคำสั่ง MOVX สัญญาณที่ขา ALE และ $\overline{PSEN}$ จะเกิดการแอกทีฟ 2 ครั้งภายในหนึ่งแมชชีนไซเกิล ในทุกครั้งที่ ALE เกิดการแอกตีฟที่พอร์ต 0 (P0) จะมีค่าของรีจิสเตอร์ PC ในไบต์ต่ำออกมา ในขณะที่พอร์ต 2 (P2) ก็จะมีค่าของ PC ในไบต์สูงเพื่อชี้ไปยังแอดเดรสต่อไปที่ต้องไปดำเนินการ สำหรับขา $\overline{PSEN}$ ก็จะเกิดการแอกติฟเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอกในกรณีที่กระทำคำสั่ง MOVX เพื่อเข้าถึงหน่วยความจำข้อมูลภายนอกที่ขา $\overline{PSEN}$ จะไม่เกิดการแอกตีฟ 2 ครั้งภายใน 1 แมชชีนไซเกิล เนื่องจากบัสแอดเดรสและบัสข้อมูลจะถูกใช้ในการติดต่อหน่วยความจำข้อมูลภายนอกแทน แต่สำหรับสัญญาณ ALE ยังคงแอกตีฟตามจังหวะการทำงานเหมือนเดิม

จากไดอะแกรมเวลาทั้งหมดสามารถสรุปได้ว่า ในการทำงาน 1 รอบหรือ 1 แมชชีนไซเกิล ซีพียูในไมโครคอนโทรลเลอร์ MCS-51 จะใช้เวลา 12 คาบเวลาของสัญญาณนาฬิกา นั่นคือ เวลาในการทำงาน 1 ไซเกิลมีค่าเท่ากับ 1 ms หรือมีความเร็วในการทำงานภายใน 1MHz ในกรณีที่ใช้ความถี่สัญญาณนาฬิกา 12 MHz ดังนั้นถ้าต้องการทราบความเร็วของการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถหาได้จาก ค่าความถี่สัญญาณนาฬิกาหารด้วย 12 และถ้าต้องการ

หาค่าเวลาของ 1 รอบการทำงานหรือ 1 แมชชีนไซเกิล สามารถทำได้โดยการหาส่วนกลับของความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถสรุปเป็นสูตรทางคณิตศาสตร์ได้ดังนี้

ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์เท่ากับ

ความถี่ของสัญญาณนาฬิกา (ค่าของคริสตอลที่ต่ออยู่ที่ขา XTAL1 และ XTAL2) / 12

เวลา 1 แมชชีนไซเกิล = 1 / ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์

2.1.2 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำภายในหลัก ๆ อยู่ 2 ส่วน คือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งก็มีขนาดและการจัดสรรแตกต่างกันไปในแต่ละเบอร์ซึ่งจะกล่าวถึงรายละเอียดของการจัดสรรหน่วยความจำภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช การเชื่อมต่อกับหน่วยความจำภายนอก และข้อมูลเบื้องต้นของรีจิสเตอร์ฟังก์ชันพิเศษที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

หน่วยความจำโปรแกรม (Program memory)

ในรูปที่ 2.8 แสดงการจัดหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในเบอร์ต่าง ๆ ที่นิยมใช้งาน อันประกอบด้วยเบอร์ AT89C51 และ AT89C52 จะเห็นได้ว่า ทั้งสองเบอร์สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียวหรือรวมกับภายนอกหรือเลือกใช้หน่วยความจำภายนอกอย่างเดียวก็ได้ ดังในรูปที่ 2.8 (ก) โดยภายใน AT89C51 จะมีหน่วยความจำโปรแกรมภายใน 4 กิโลไบต์ ในขณะที่ AT89C52 จะมีขนาด 8 กิโลไบต์

ในกรณีที่ใช้หน่วยความจำภายในและภายนอกรวมกัน หากใช้ AT89C51 ก็จะสามารถติดต่อกับหน่วยความจำภายนอกได้ 60 กิโลไบต์ และถ้าใช้เบอร์ AT89C52 จะสามารถติดต่อกับหน่วยความจำโปรแกรมภายนอกได้ 56 กิโลไบต์

หน่วยความจำโปรแกรมใช้เก็บข้อมูลของโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์หรือที่เรียกว่า **โปรแกรมมอนิเตอร์ (monitor program)** หากใช้หน่วยความจำภายนอกมักจะบรรจุอยู่ในหน่วยความจำชนิดอีพรอม (BPROM : Erasable Programmable Read-Only Memory) ซึ่งสามารถกระทำการอ่านได้เพียงอย่างเดียว

หน่วยความจำโปรแกรมมีแอดเดรสเริ่มต้นที่ 0000H เมื่อซีพียูได้รับการรีเซตให้เริ่มต้นการทำงาน จะต้องมาเริ่มต้นที่แอดเดรส 0000H นี้เสมอ อย่างไรก็ตาม ในพื้นที่ของหน่วยความจำ

โปรแกรมไม่ว่าจะใช้งานภายในหรือภายนอกก็ตาม ต้องมีการสงวนพื้นที่บางตำแหน่งเอาไว้สำหรับการบริการอินเตอร์รัปต์ 6 ประเภท ประเภทละ 8 ไบต์ ประกอบด้วย

พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากภายนอก กำหนดไว้ที่แอดเดรส 0003H

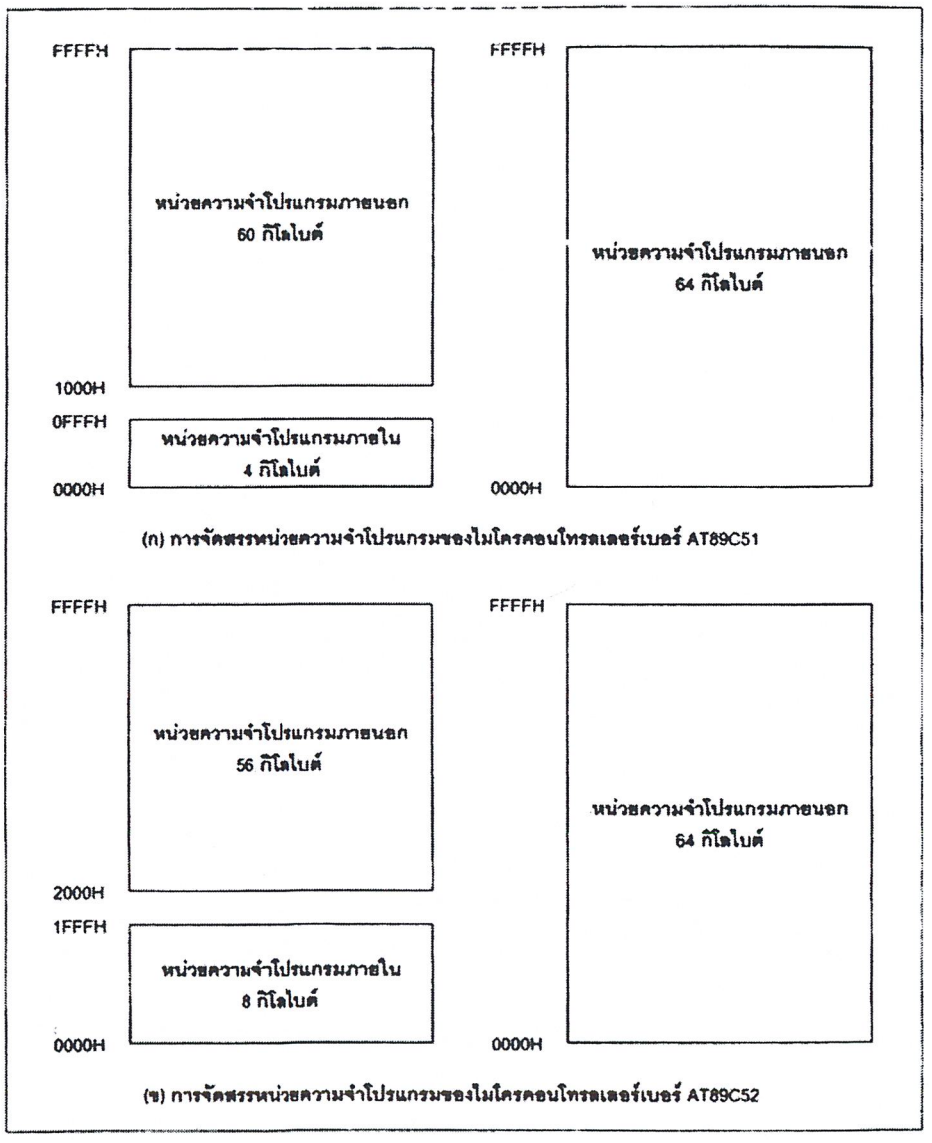
พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากไทเมอร์ 0 กำหนดไว้ที่แอดเดรส 000BH

พื้นที่สำหรับบริการอินเตอร์รัปต์ 1 จากภายนอก กำหนดไว้ที่แอดเดรส 0013H

พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากไทเมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH

พื้นที่สำหรับบริการอินเตอร์รัปต์ของการสื่อสารอนุกรม กำหนดไว้ที่แอดเดรส 0023H

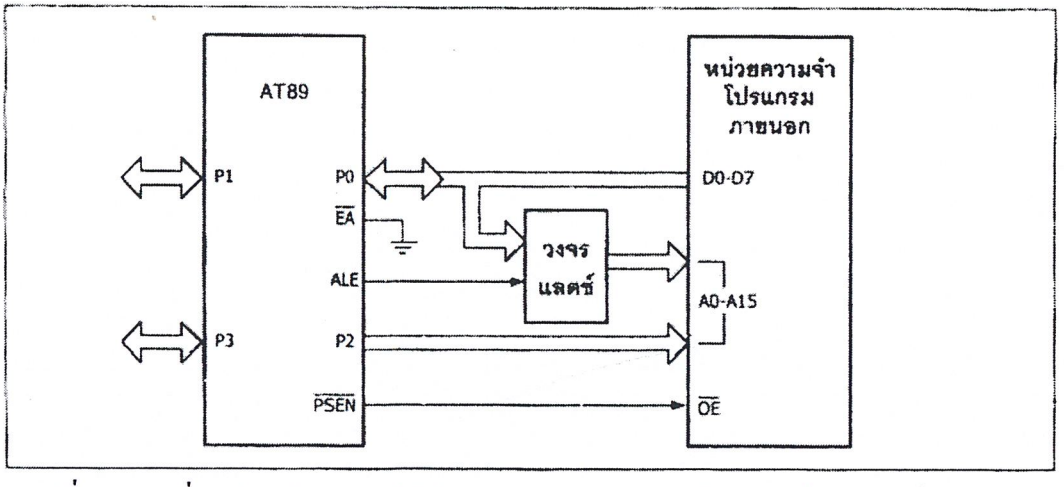
พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH



รูปที่ 2.8 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

กรณีที่ใช้ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่มีหน่วยความจำโปรแกรมภายในแต่ต้องการติดต่อกับหน่วยความจำโปรแกรมภายนอกด้วย สามารถทำได้โดยต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์ ยกตัวอย่าง ไมโครคอนโทรลเลอร์ AT89C51 มีหน่วยความจำโปรแกรมขนาด 4 กิโลไบต์มีแอดเดรสอยู่ระหว่าง 0000H – 0FFFH เมื่อต่อหน่วยความจำโปรแกรมภายนอกต้องกำหนดให้แอดเดรสอยู่ในช่วง 1000H – FFFFH

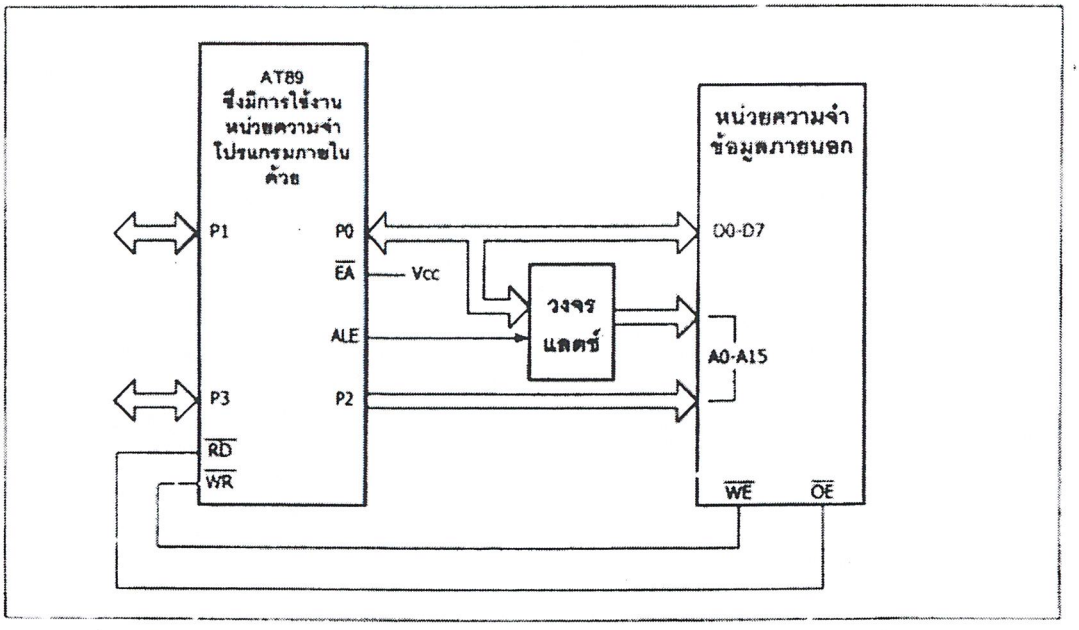
การต่อหน่วยความจำภายนอกแสดงดังในรูปที่ 2.9 จะเห็นได้ว่าขาพอร์ต P0.0-P0.7 ถูกใช้เป็นขาข้อมูล D0-D7 และขาแอดเดรสไบต์ต่ำ โดยผ่านวงจรแลทช์ ซึ่งปกติใช้ไอซีเบอร์ 74HC573 และใช้สัญญาณ ALE และ $\overline{PSEN}$ ในการเลือกว่า ต้องการใช้งานขา P0.0-P0.7 เพื่อเป็นขาข้อมูลหรือขาแอดเดรสในขณะที่ขา P2.0-P2.7 ใช้ในการเชื่อมต่อกับขาแอดเดรสไบต์สูง A8-A15 ดังนั้นเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอกไมโครคอนโทรลเลอร์จะเหลือขาพอร์ตใช้งานเพียง 16 บิต คือที่ขาพอร์ต P1.0-P1.7 และ P3.0-P3.7



รูปที่ 2.9 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์ MCS-51

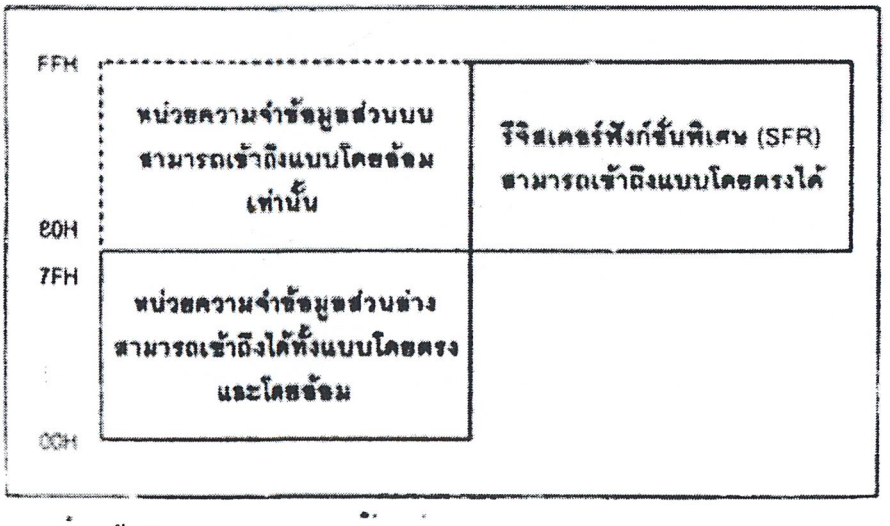
หน่วยความจำข้อมูล (Data Memory)

มีด้วยกัน 2 แบบคือ หน่วยความจำข้อมูลภายนอกและภายในโดยไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89 สามารถติดต่อกับหน่วยความจำข้อมูลภายนอกได้สูงสุด 64 กิโลไบต์โดยการใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก การติดต่อหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชแสดงดังในรูปที่ 2.10 จะเห็นได้ว่า มีลักษณะคล้ายกับการติดต่อกับหน่วยความจำโปรแกรมภายนอกแตกต่างกันที่มีสัญญาณที่ใช้สำหรับการอ่านและเขียนหน่วยความจำข้อมูลภายนอก นั่นคือ ขา $\overline{RD}$ และ $\overline{WR}$

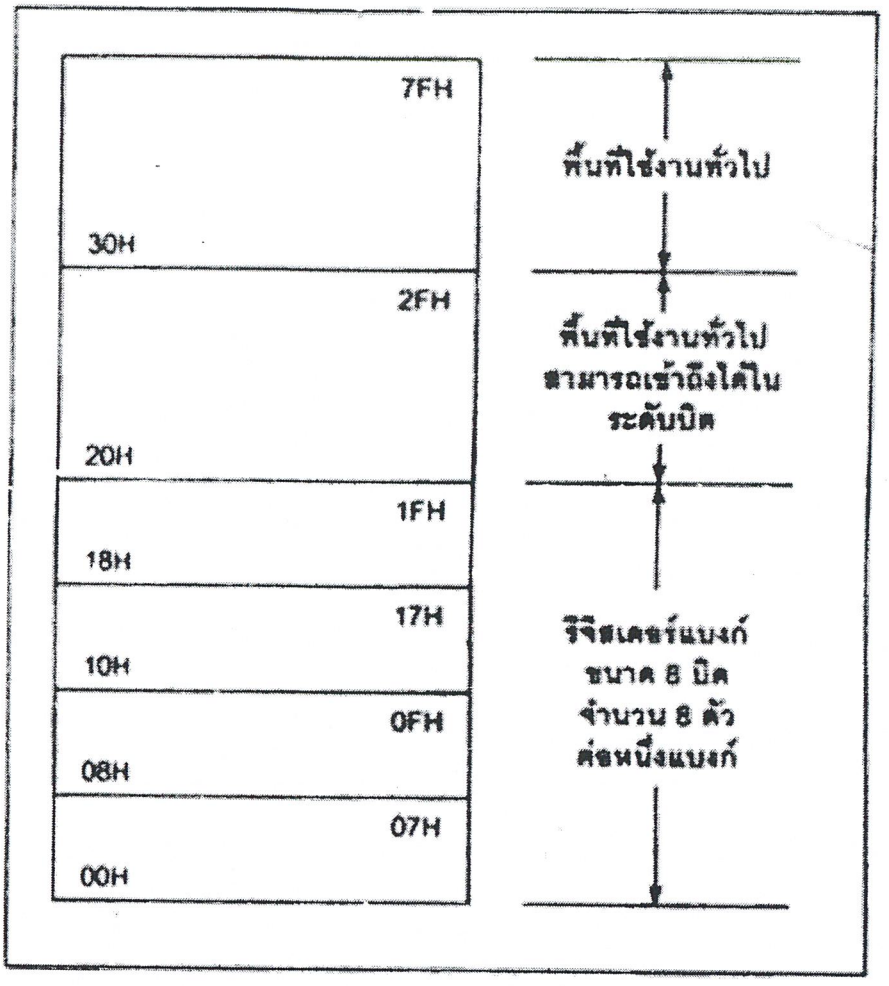


รูปที่ 2.10 การเชื่อมต่อหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในเป็นแบบแรม (RAM : Random Access Memory) โดยแต่ละเบอร์จะมีขนาดแตกต่างกันไป ในเบอร์ AT89C51 มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ในขณะที่เบอร์ AT89C52 มีขนาด 256 ไบต์ สำหรับการจัดสรรหน่วยความจำข้อมูลภายในแบ่งเป็น 3 ส่วน คือ หน่วยความจำข้อมูลส่วนกลาง (lower), ส่วนบน (upper) และรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR : Special Function Register) แต่ละส่วนมีขนาด 128 ไบต์ ดังแสดงการจัดสรรในรูปที่ 2.11



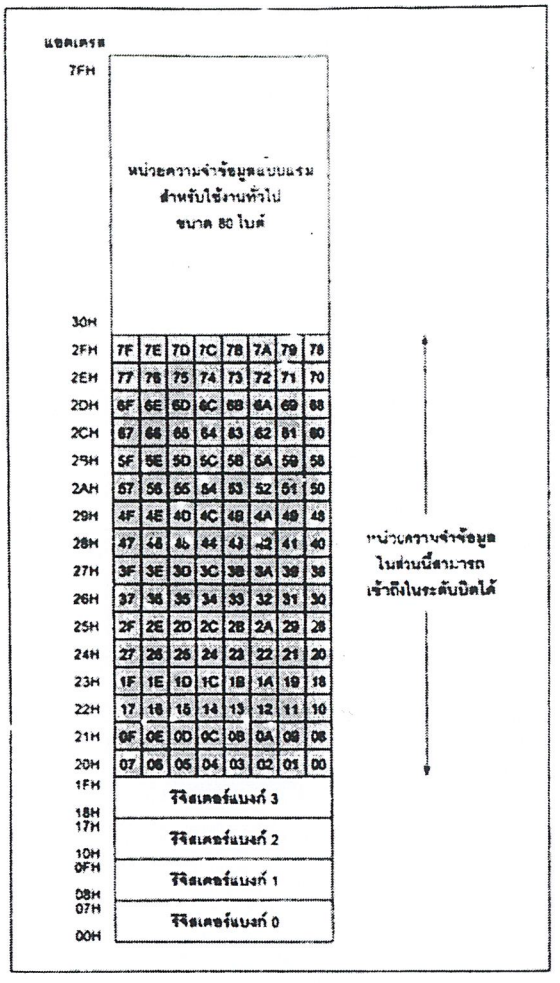
รูปที่ 2.11 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช



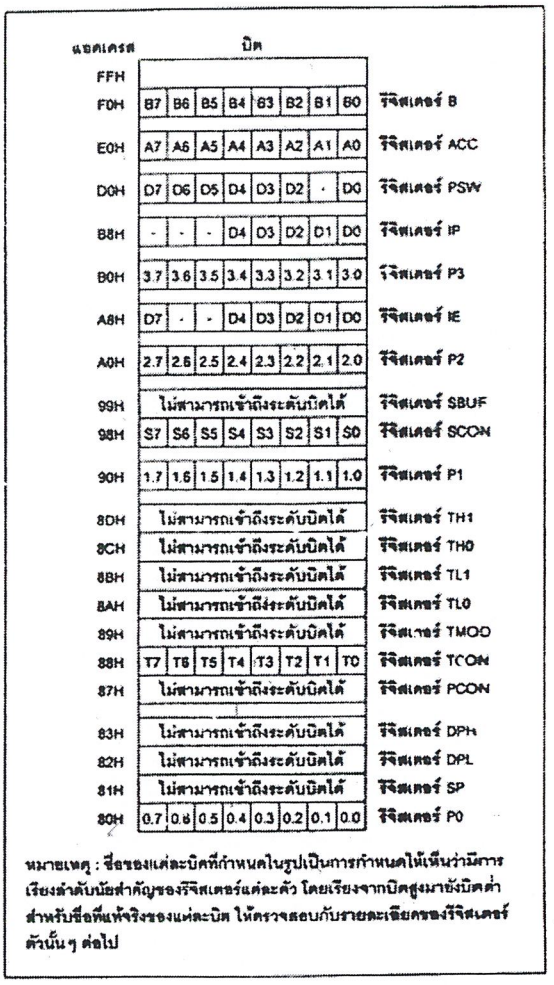
รูปที่ 2.12 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในส่วนล่างของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

จะเห็นได้ว่า หน่วยความจำข้อมูลส่วนบนและรีจิสเตอร์ฟังก์ชั่นพิเศษมีตำแหน่งทับซ้อนกัน แต่จะใช้การติดต่อที่แตกต่างกัน และในไมโครคอนโทรลเลอร์ MCS-51 บางเบอร์จะไม่มีหน่วยความจำข้อมูลส่วนบน

ขนาดของหน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดยแท้จริงแล้วมีเพียง 256 ไบต์ แต่ด้วยการจัดเข้าถึงแตกต่างกัน จึงดูเสมือนว่า ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำข้อมูลภายในสูงถึง 384 ไบต์ โดยในหน่วยความจำข้อมูลส่วนล่างขนา ด128 ไบต์ มีแอดเดรสอยู่ที่ 00H-7FH สามารถเข้าถึงได้โดยตรงและโดยอ้อม สำหรับหน่วยความจำข้อมูลส่วนบนมีขนาด 128 ไบต์เช่นกัน มีแอดเดรสอยู่ที่ 80H-FFH สามารถเข้าถึงแบบโดยอ้อมเท่านั้น ในขณะที่รีจิสเตอร์ SFR มีแอดเดรสอยู่ที่ 80H-FFH เช่นเดียวกับหน่วยความจำข้อมูลส่วนบนแต่สำหรับรีจิสเตอร์ SFR ใช้การเข้าถึงแบบโดยตรง



รูปที่ 2.13 โครงสร้างของหน่วยความจำข้อมูลภายในในส่วนบนของไมโครคอนโทรลเลอร์ MCS-51



รูปที่ 2.14 การจัดสรรพื้นที่ของรีจิสเตอร์ชั่นพิเศษ (SFR)

ดังนั้นเพื่อความสะดวกและง่ายตลอดจนป้องกันความสับสนในการเขียนโปรแกรมสำหรับผู้เริ่มต้นจึงควรใช้หน่วยความจำข้อมูลภายในเพียง 128 ไบต์จากหน่วยความจำข้อมูลส่วนล่างร่วมกับรีจิสเตอร์ SFR

ในรูปที่ 2.12 แสดงการจัดสรรหน่วยความจำข้อมูลส่วนล่าง หน่วยความจำ 32 ไบต์ต่ำสุดที่แอดเดรส 00H-1FH แบ่งเป็น 4 กลุ่ม เรียกว่า 4 แบงก์ (bank) แต่ละแบงก์มีรีจิสเตอร์ 8 ตัวคือ R0-R7 การติดต่อกับหน่วยความจำในแบงก์ใดให้กำหนดที่รีจิสเตอร์ PSW (Program Status Word register)

หน่วยความจำข้อมูล 16 ไบต์ถัดมาที่แอดเดรส 20H-2FH เป็นพื้นที่สำหรับใช้งานทั่วไปสามารถเข้าถึงได้ในระดับบิต (Bit addressable) และหน่วยความจำข้อมูลที่เหลือ 81 ไบต์จะต้องแบ่งส่วนหนึ่งสำรองไว้เป็นพื้นที่ขอสแต็ก (stack : ที่พักข้อมูลชั่วคราวในกรณีที่ซีพียูมีการกระโดดไปงานในโปรแกรมย่อย) การเข้าถึงหน่วยความจำส่วนนี้ต้องใช้การเข้าถึงในระดับไบต์

ในรูปที่ 2.13 แสดงโครงสร้างของหน่วยความจำข้อมูลส่วนบนซึ่งจะมีลักษณะที่คล้ายกับหน่วยความจำข้อมูลส่วนล่าง หากแต่ใน 80 ไบต์บนไม่จำเป็นต้องสำรองไว้สำหรับสแต็ก และต้องใช้การเข้าถึงในลักษณะโดยอ้อมเท่านั้น

รีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register : SFR)

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชทั้งหมดมีด้วยกัน 22 ตัว สำหรับในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 และ 28 ตัวในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89SXX ทั้งนี้เนื่องจากใน AT89C52 และ AT89SXX มีจำนวนไทเมอร์เคาน์เตอร์มากกว่า AT89C51

รีจิสเตอร์ SFR มีแอดเดรสอยู่ระหว่าง 80H-FFH ในพื้นที่ของหน่วยความจำข้อมูลส่วนบน สามารถเข้าถึงได้โดยตรง (direct addressing) ในรูปที่ 2.14 แสดงการจัดสรรพื้นที่ของรีจิสเตอร์ SFR แต่ละตัวในหน่วยความจำข้อมูลส่วนบน สำหรับรายละเอียดเบื้องต้นของรีจิสเตอร์ SFR มีดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**-**	**P**

CY : แฟลกทด (Carry flag) เป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้วค่าของแอกคิวมูเลเตอร์เกิน 255 (ฐานสิบ) หรือ FFH

AC : แฟลกทดเสริม (Auxiliary Carry flag) เป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์แล้วทำให้เกิดการทดข้ามจากบิต 3 มายังบิต 4 มักใช้ในการแปลงค่าเลขฐานสิบ (BCD operation)

F0 : แฟลกใช้งานทั่วไป เมื่อผู้เขียนโปรแกรมกำหนดค่าที่บิตนี้แล้ว ไม่ว่าจะกระทำคำสั่งใด ๆ ที่บิตนี้จะไม่มีการเปลี่ยนแปลง

RS1 : บิตเลือกรีจิสเตอร์แบงก์ (Register Select 1) ใช้งานร่วมกับบิต RS0 เพื่อเลือกแบงก์ของรีจิสเตอร์ R0-R7

RS0 : บิตเลือกรีจิสเตอร์แบงก์ (Register Select0) ใช้งานร่วมกับบิต RS1 เพื่อเลือกแบงก์ของรีจิสเตอร์ R0-R7

OV : บิตเกิน (Overflow) เป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ทำให้เกิดการทดเข้ามาจากบิต 6 มายังบิต 7 ของแอกคิวมูเลเตอร์ หรือแอกคิวมูเลเตอร์มีค่า เกิน 127 (ฐานสิบ) นอกจากนั้นยังใช้เป็นการแสดงค่าลบด้วย

- : บิตนี้ผู้ใช้งานสามารถกำหนดใช้งานได้อย่างอิสระ

P : บิตพาริตี้ (Parity) ใช้ในการตรวจสอบจำนวนค่า "1" ภายในแอกคิวมูเลเตอร์ ถ้าหากในแอกคิวมูเลเตอร์มีจำนวนบิตที่เป็น "1" รวมกันเป็นเลขคู่ บิตนี้จะเป็น "0" รวมกันเป็นเลขคี่ บิตนี้จะเป็น "1"

RS1	RS0	แบงก์ของรีจิสเตอร์	ช่วงแอดเดรส
0	0	แบงก์ 0	00H-07H
0	1	แบงก์ 1	08H-0FH
1	0	แบงก์ 2	10H-17H
1	1	แบงก์ 3	18H-1FH

ตารางที่ 2.3 การเลือกแบงก์ของหน่วยความจำส่วนล่างเพื่อติดต่อกับรีจิสเตอร์แบงก์ R0-R7

รีจิสเตอร์แสดงสถานะของโปรแกรม (Program Status Word : PSW)

เป็นรีจิสเตอร์ขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต นั่นหมายความว่า สามารถกระทำคำสั่งหรือกำหนดค่าในแต่ละบิตของรีจิสเตอร์ตัวนี้ได้อย่าอิสระ มีแอดเดรสอยู่ที่ D0H เป็นรีจิสเตอร์ที่เก็บสถานะของการทำงานของโปรแกรมในขณะนั้น จะเรียกสถานะต่าง ๆ ของโปรแกรมว่า แฟลก (flag) เมื่อซีพียูกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้วเกิดการเปลี่ยนแปลงสถานะขึ้น ผลของการเปลี่ยนแปลงนั้นจะมาปรากฏที่บิตต่าง ๆ ของรีจิสเตอร์ PSW ดังแสดงข้างต้น

จะเห็นได้ว่า นอกจากรีจิสเตอร์ PSW ถูกใช้ในการเก็บสถานะของโปรแกรมแล้ว ที่บิต RS0 และ RS1 ยังใช้ในการเลือกแบงก์ของหน่วยความจำส่วนล่าง ซึ่งเป็นพื้นที่ของรีจิสเตอร์ R0-R7 ด้วย ดังมีรายละเอียดแสดงในตารางที่ 2.3 โดยปกติแล้วในการใช้งานรีจิสเตอร์ R0-R7 มักนิยมเลือกใช้แบงก์ 0 เป็นลำดับแรก หากไม่เพียงพอจึงเลือกในแบงก์อื่น ๆ มาใช้ แต่ต้องระมัดระวังในการกำหนดค่าและลำดับการติดต่อให้ดี มิเช่นนั้น อาจทำให้การเขียนโปรแกรมเกิดความสับสน จึงควรเลือกใช้รีจิสเตอร์ R0-R7 ในแบงก์ 0 เพียงแบงก์เดียวให้ชำนาญเสียก่อน

การกำหนดค่าของรีจิสเตอร์ PSW เพื่อเลือกใช้งานรีจิสเตอร์ R0-R7 ควรกำหนดไว้ที่ตอนต้นของโปรแกรมเสมอ เพื่อจะได้เขียนโปรแกรมติดต่อกับรีจิสเตอร์ R0-R7 ได้อย่างสะดวกและไม่เกิดความผิดพลาด

แอกคิวมูเลเตอร์ (Accumulator : ACC)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของไมโครคอนโทรลเลอร์โดยเฉพาะอย่างยิ่งในการคำนวณทาง

คณิตศาสตร์และลอจิก ก่อนที่จะส่งข้อมูลหรือผลลัพธ์ที่ได้นั้นให้แก่ซีพียูเพื่อทำการประมวลผลต่อไป อาจเรียกรีจิสเตอร์แอกคิวมูเลเตอร์อย่างสั้น ๆ ว่า รีจิสเตอร์ A หรือ ACC

รีจิสเตอร์ A นี้สามารถเข้าถึงระดับบิตได้ นั่นหมายความว่า สามารถกระทำคำสั่งหรือกำหนดค่าในแต่ละบิตของรีจิสเตอร์ตัวนี้ได้อย่างอิสระ

รีจิสเตอร์ B

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ F0H มีหน้าที่พิเศษคือ หากมีความต้องการคูณหรือหารทางคณิตศาสตร์ จะต้องนำข้อมูลที่ต้องการหารหรือคูณนั้น มาเก็บไว้ในรีจิสเตอร์ B นี้ แล้วจึงกระทำคำสั่งการคูณหรือหารกับค่าในรีจิสเตอร์ A ต่อไป

ในกรณีที่ไม่ได้มีความต้องการคูณหรือหารข้อมูล สามารถใช้รีจิสเตอร์ B นี้โนการเก็บข้อมูลทั่วไปได้ เหมือนกับรีจิสเตอร์ปกติ และสามารถเข้าถึงในระดับบิตได้เช่นเดียวกับรีจิสเตอร์ A

โปรแกรมเคาน์เตอร์ (Program Counter : PC)

มีขนาด 16 บิต มีหน้าที่แจ้งแอดเดรสของหน่วยความจำโปรแกรมในตำแหน่งถัดไปที่ซีพียูจะต้องไปทำงาน รีจิสเตอร์ PC เป็นรีจิสเตอร์ตัวเดียวที่ไม่ได้จัดสรรไว้รวมกับรีจิสเตอร์ SFR ตัวอื่นๆ การเปลี่ยนแปลงค่าของรีจิสเตอร์ PC จะขึ้นอยู่กับผลของการกระทำคำสั่งแต่ละคำสั่งภายในหน่วยความจำโปรแกรม

รีจิสเตอร์ PC มีความสำคัญมาก โดยเฉพาะอย่างยิ่งในการตรวจสอบการทำงานของโปรแกรมว่า ดำเนินไปตามลำดับขั้นตอนตามที่กำหนดไว้หรือไม่

สแต็กพอยน์เตอร์ (Stack Pointer : SP)

หรือรีจิสเตอร์ตัวชี้สแต็ก มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 81H ใช้ในการเก็บค่าตำแหน่งของตัวชี้สแต็ก ซึ่งสามารถเปลี่ยนแปลงได้เมื่อซีพียูมีการกระโดดไปทำงานที่โปรแกรมย่อย หรือกระโดดโปรแกรมย่อยกลับมายังโปรแกรมหลัก เมื่อมีการรีเซตเกิดขึ้น (รีเซต : การกระทำที่ส่งผลให้ซีพียูต้องเริ่มต้นการทำงานใหม่ตั้งแต่ต้น) ค่าของรีจิสเตอร์ SP จะเท่ากับ 07H นั่นหมายความว่าตัวชี้สแต็กมีค่า 07H แอดเดรสแรกของพื้นที่ที่สำรองไว้ทำหน้าที่เป็นสแต็กจะเท่ากับ 08H

รีจิสเตอร์ชี้ข้อมูลหรือดาต้าพอยน์เตอร์ (Data Pointer : DTPR)

มีขนาด 16 บิต โดยแบ่งเป็นรีจิสเตอร์ชี้ข้อมูลไบต์สูง (DPH) และรีจิสเตอร์ชี้ข้อมูลไบต์ต่ำ (DPL) แต่ละตัวมีขนาด 8 บิต มีแอดเดรสอยู่ที่ 82H สำหรับ DPL และ 83H สำหรับ DPH รีจิสเตอร์

DPTR นี้ใช้ในการเก็บค่าแอดเดรสของหน่วยความจำหรืออุปกรณ์ภายนอกที่ไมโครคอนโทรลเลอร์ต้องการติดต่อด้วย

รีจิสเตอร์พอร์ต (Port register)

เป็นรีจิสเตอร์ขนาด 8 บิต ที่ใช้เก็บข้อมูลของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 มีด้วยกันทั้งสิ้น 4 ตัวคือ รีจิสเตอร์พอร์ต 0 หรือ P0 มีแอดเดรสอยู่ที่ 80H, รีจิสเตอร์พอร์ต 1 หรือ P1 มีแอดเดรสอยู่ที่ 90H, รีจิสเตอร์พอร์ต 2 หรือ P2 มีแอดเดรสอยู่ที่ A0H และรีจิสเตอร์พอร์ต 3 หรือ P3 มีแอดเดรสอยู่ที B0H รีจิสเตอร์ทุกตัวสามารถเข้าถึงได้ในระดับบิต เมื่อต้องการอ่านหรือเขียนข้อมูลออกไปยังพอร์ตของไมโครคอนโทรลเลอร์ จะต้องกระทำผ่านรีจิสเตอร์นี้ทุกครั้ง

รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ทำการส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟรอ์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51

รีจิสเตอร์ไทเมอร์ (Timer register)

เป็นรีจิสเตอร์ขนาด 16 บิต แต่จะจัดแบ่งเป็นไบต์สูงและไบต์ต่ำ เช่นเดียวกับรีจิสเตอร์ DPTR รีจิสเตอร์ไทเมอร์ใช้ในการเก็บค่าของตัวนับหรือเคาน์เตอร์ (counter) ภายในไมโครคอนโทรลเลอร์เพื่อใช้ในการสร้างฐานเวลา, จับเวลา หรือนับจำนวนพัลส์สัญญาณนาฬิกาภายในบางทีเรียกรีจิสเตอร์ตัวนี้ว่า รีจิสเตอร์ไทเมอร์/เคาน์เตอร์

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 จะมีรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ 2 ตัว แบ่งเป็น T0 หรือ Timer 0 และ T1 หรือ Timer 1 ในรีจิสเตอร์ยังแบ่งเป็นรีจิสเตอร์ไทเมอร์ไบต์ต่ำ (TL) และรีจิสเตอร์ไทเมอร์ไบต์สูง (TH) เหมือนกัน โดยรีจิสเตอร์ TL0 มีแอดเดรส

อยู่ที่ 8AH รีจิสเตอร์ TH0 มีแอดเดรสอยู่ที่ 8BH ในขณะที่ TL1 และ TH1 มีแอดเดรสอยู่ที่ 8CH และ 8DH ตามลำดับ

สำหรับในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และในอนุกรม AT89SXX จะมีรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ถึง 3 ตัว โดยรีมีจิสเตอร์ TL2 และ TH2 ซึ่งมีแอดเดรสอยู่ที่ CCH และ CDH ตามลำดับเพิ่มเติมเข้ามา

รีจิสเตอร์แคปเจอร์ (Capture register)

เป็นรีจิสเตอร์ขนาด 16 บิต มีเฉพาะในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และในอนุกรม AT89SXX เท่านั้น เนื่องจากต้องใช้ร่วมกับไทเมอร์/เคาน์เตอร์2 (Timer2) ซึ่งมีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และในอนุกรม AT89SXX โดยรีจิสเตอร์แคปเจอร์นี้มีชื่อเรียกอย่างย่อว่า รีจิสเตอร์ RCAP2 ซึ่งแบ่งออกเป็นไบต์ต่ำคือ RCAP2L มีแอดเดรสอยู่ที่ CAH และไบต์สูงคือ RCAP2H มีแอดเดรสอยู่ที่ CBH

รีจิสเตอร์แคปเจอร์จะถูกใช้งานเมื่อกำหนดให้ไทเมอร์ 2 ทำงานในโหมดแคปเจอร์ ซึ่งเป็นโหมดที่กำหนดให้ไมโครคอนโทรลเลอร์ทำการตรวจจับการเปลี่ยนแปลงสถานะทางลอจิกที่ขา T2EX ทั้งนี้เพื่อใช้ประโยชน์ในการวัดคาบเวลา ความถี่ ตลอดจนการเปลี่ยนแปลงของสัญญาณพัลส์ที่ขา T2EX นี้

รีจิสเตอร์ควบคุม (Control register)

รีจิสเตอร์ SFR ที่ใช้การควบคุมการทำงานในส่วนต่าง ๆ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ยังมีอีกหลายตัวประกอบด้วย

รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของวงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ TCON และ T2CON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS51 แบบแฟลช โดย T2CON ใช้สำหรับไทเมอร์/เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT789C52

รีจิสเตอร์ TMOD และ T2MOD เป็นรีจิสเตอร์ที่ใช้กำหนดโหมดหรือลักษณะในการทำงานของไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดย T2MOD ใช้สำหรับไทเมอร์/เคาน์เตอร์2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52

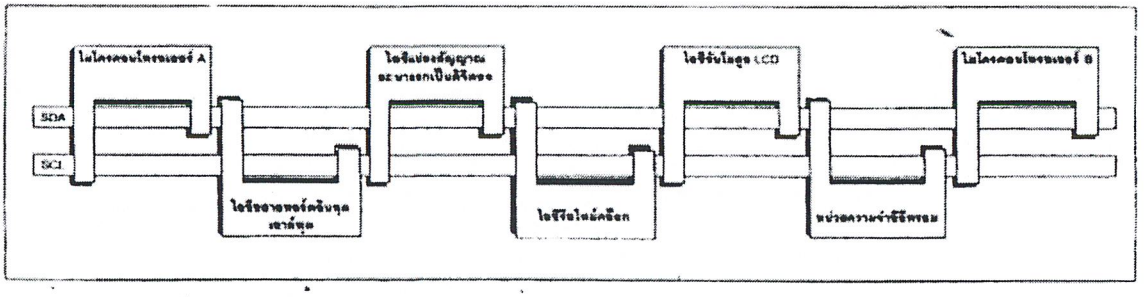
รีจิสเตอร์ IE และ IP เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการตอบสนองการอินเตอร์รัปต์ (interrupt : การขัดจังหวะการทำงานปกติของซีพียู) โดย IE เป็นรีจิสเตอร์สำหรับอีนาเบิลหรือใช้ในการกำหนดลักษณะของการตอบสนองการอินเตอร์รัปต์ ในขณะที่ IP เป็นรีจิสเตอร์สำหรับกำหนดลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่า จะให้ซีพียูตอบสนองการเกิดอินเตอร์รัปต์ในลักษณะใดก่อนหรือหลัง

2.2 หลักการของระบบบัส I^2C

2.2.1 ความรู้เบื้องต้นเกี่ยวกับ I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซีโดยบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลัก คือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สัญญาณเพียง 2 เส้น เส้นหนึ่ง คือ สายข้อมูล อีกเส้นหนึ่ง คือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรม หรือ SDA (Serial Data line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรมหรือ SCL (Serial Clock line)



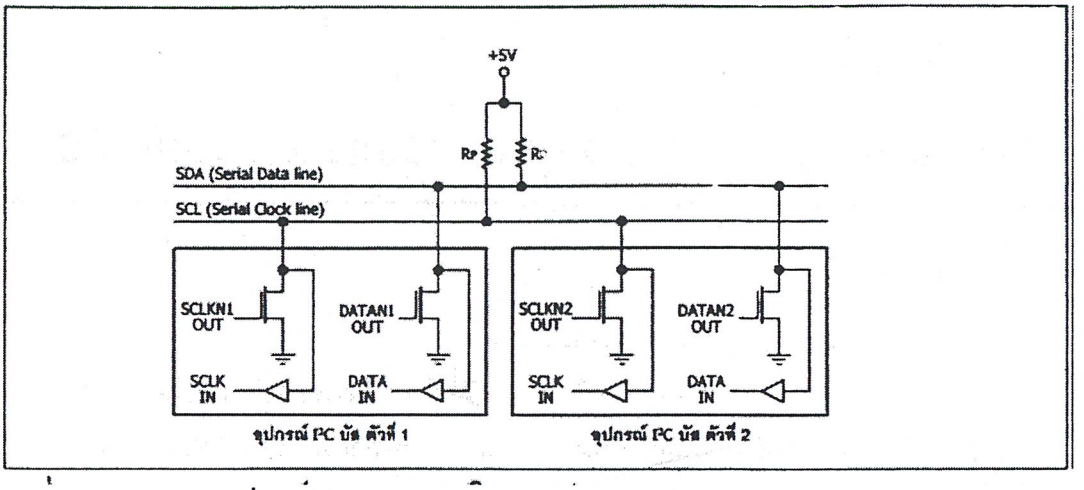
รูปที่ 2.15 ผังแสดงการเชื่อมต่อของอุปกรณ์ต่าง ๆ บนบัส I^2C

ในรูปที่ 2.15 แสดงผังของการเชื่อมต่ออุปกรณ์ต่าง ๆ บนบัส I^2C จะเห็นได้ว่า อุปกรณ์ที่ทำการเชื่อมต่อบนบัส I^2C มีหลากหลาย ไม่ว่าจะเป็นไอซีขยายพอร์ตอินพุตเอาต์พุต (I/O Expander),

ไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอล (ADC) และแปลงสัญญาณดิจิตอลเป็นอนาลอก (DAC), ไอซีรีลไทม์คล็อก (RTC), ไอซีขับโมดูล LCD, หน่วยความจำอีอีพรอม และไมโครคอนโทรลเลอร์

คุณสมบัติโดยทั่วไปของบัส I^2C

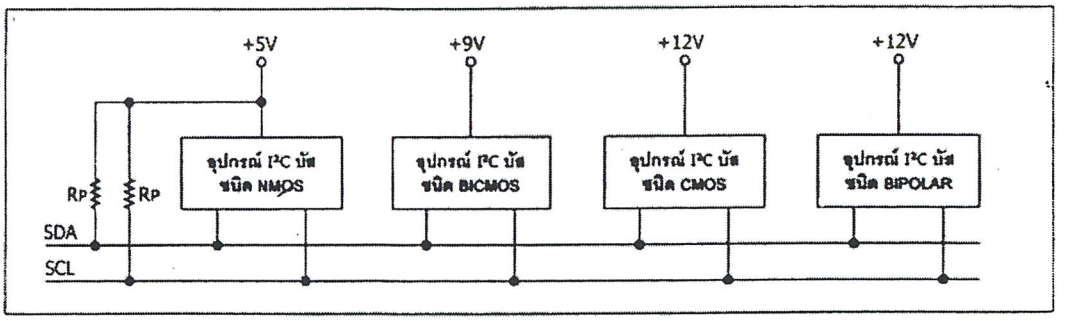
สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัพกับแรงดัน +5V ไว้ตลอดเวลาเพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งานทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (open-drain) หรือคอลเล็กเตอร์เปิด (open-collector) ดังแสดงรายละเอียดในรูปที่ 2.16



รูปที่ 2.16 แสดงวงจรเอาต์พุตของอุปกรณ์ในระบบบัส I^2C

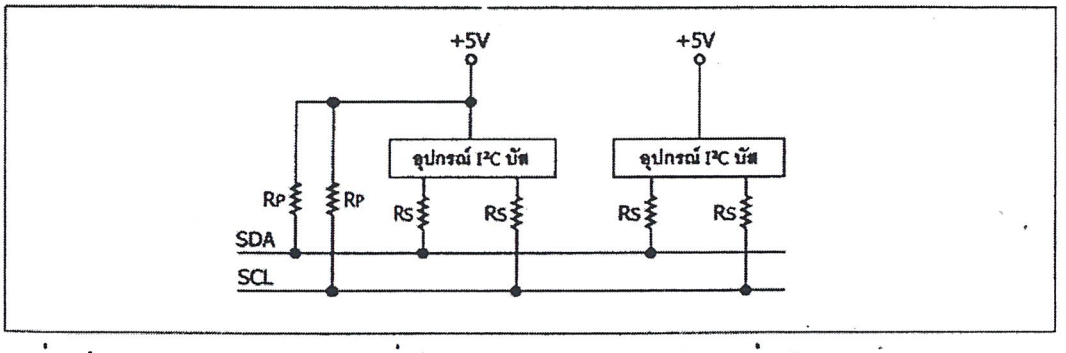
อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (fast mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400 pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่า คือ 7 บิต (7-bit addressing) หรือ 10 บิต (10-bit addressing)

ข้อเด่นอีกประการหนึ่งของบัส I^2C คือ สามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้ โดยอุปกรณ์บนบัส I^2C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12V การต่อร่วมกันบนบัส I^2C สามารถกระทำได้ในลักษณะเดียวกับกรณีที่อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือ ให้ต่อสาย SDA และ SCL ของอุปกรณ์แต่ละตัวเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัพ (R_p) เข้ากับแรงดัน +5V ไว้ด้วยเสมอ ดังแสดงในรูปที่ 2.17



รูปที่ 2.17 การต่อตัวต้านทานพูลอัพบนสายสัญญาณในระบบบัส I^2C

ในกรณีที่อาจมีแรงดันไฟกระชากขนาดใหญ่ปะปนเข้ามาในบัส I^2C ที่ขา SDA และ SCL ของอุปกรณ์แต่ละตัวต้องต่อตัวต้านทานอนุกรมกับขา SDA และ SCL เรียกว่า R_S ก่อนต่อเข้าสู่บัส I^2C ดังแสดงในรูปที่2.18



รูปที่ 2.18 การต่อตัวต้านทาน R_S เพื่อลดสัญญาณรบกวนขนาดใหญ่ที่อาจเข้ามาในบัส I^2C

หลักการของบัส I^2C

บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้น ดังที่ได้กล่าวมาแล้วคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โปรโตคอล (protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะอธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I^2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I^2C ต่อไป

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (transmitter)

อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ (receiver) ในอุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I^2C ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ (master)

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่า สเลฟ (slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

(1) การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

(2) ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

2.2.4 สภาวะที่เกิดขึ้นบนบัส I^2C

มีด้วยกัน 5 สภาวะ ดังนี้

(1) **บัสว่าง (Bus not busy)** สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

(2) **เริ่มต้นการถ่ายทอดข้อมูล (start data transfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า "สภาวะเริ่มต้น (START)"

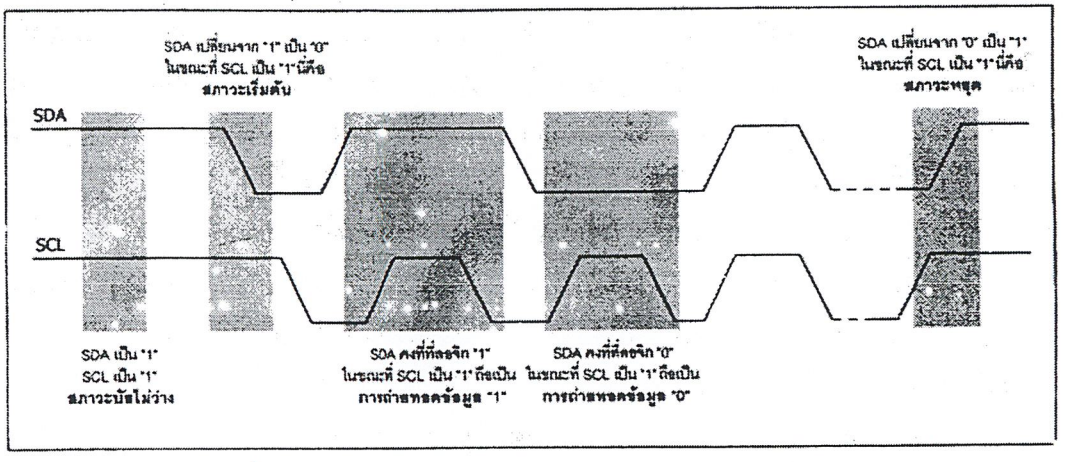
(3) **หยุดการถ่ายทอดข้อมูล (stop data transfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า "สภาวะหยุด (STOP)"

(4) **ข้อมูลดำรงอยู่บนบัส (data valid)** สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น "0" หรือ "1" ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น

(5) **รับรู้ข้อมูล (acknowledge)** เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตเรียกว่า "บิตรับรู้ (acknowledge bit)" มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตัวส่ง ทางด้านตัวรับจะส่ง

บิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

ในรูปที่ 2.19 เป็นไดอะแกรมเวลาที่แสดงถึงการเกิดสภาวะต่าง ๆ บนบัส I^2C ไม่ว่าจะเป็นสภาวะบัสว่าง, เริ่มต้น, ถ่ายทอดข้อมูล, รับรู้ และหยุดการถ่ายทอดข้อมูล



รูปที่ 2.19 ไดอะแกรมเวลาแสดงสถานะต่าง ๆ ในบัส I^2C

2.2.5 การทำงานบนบัส I^2C

ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่าง ๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงเสียก่อนโดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การอ้างถึงแบบ 7 บิต หรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้วก็จะเริ่มต้นการถ่ายทอดข้อมูลกันต่อไป

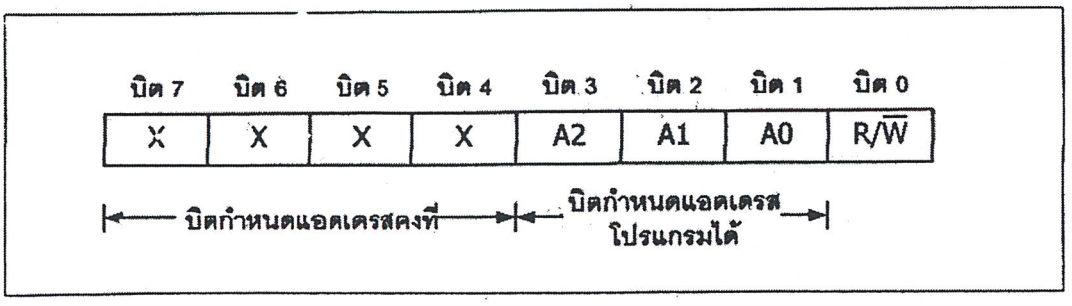
ดังนั้นหัวใจสำคัญในอันดับแรกของการทำงานบนบัส I^2C คือ การอ้างถึงอุปกรณ์แต่ละตัว ซึ่งในที่นี้จะอธิบายรายละเอียดของการอ้างถึงทั้ง 2 รูปแบบ

2.2.6 การอ้างถึงแบบ 7 บิต (7-bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ หรือ ข้อมูลกำหนดแอดเดรส โดยมีรูปแบบแสดงในรูปที่ 2.2 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็น บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกในบิต LSB เป็นบิตที่ใช้กำหนด

การอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้น ๆ หากบิต LSB เป็น “0” หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น “1” จะเป็นการอ่านข้อมูลจากอุปกรณ์เสลฟ

ข้อมูลในไบต์ต่อมา คือ ข้อมูลควบคุม (control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่างไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นเอาต์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น



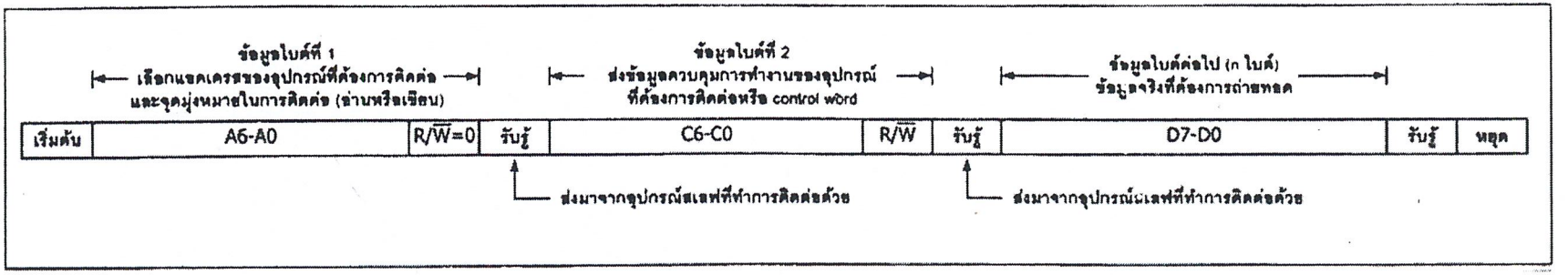
รูปที่ 2.20 รูปแบบของข้อมูลกำหนดแอดเดรสที่ใช้ในการอ้างถึงแบบ 7 บิต

ข้อมูลในไบต์ต่อมา คือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

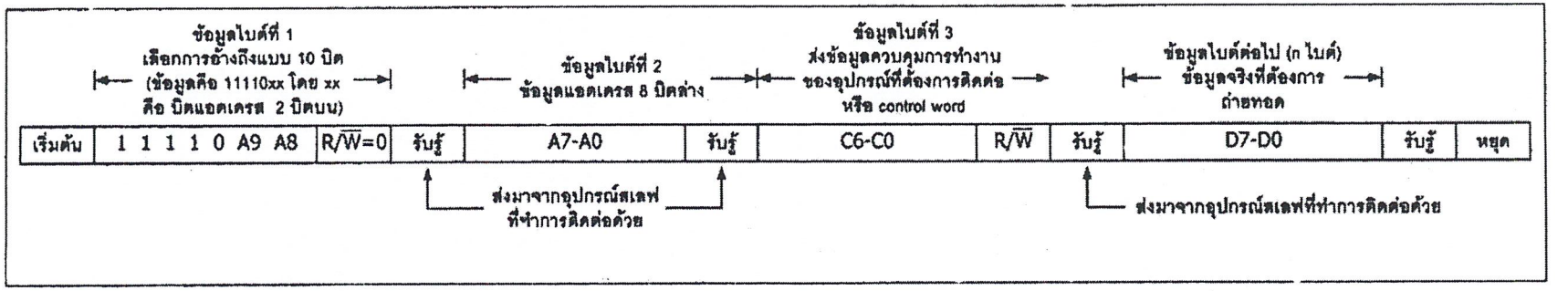
หลังจากที่มีการถ่ายทอดข้อมูลในไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ ในรูปที่2.21 แสดงรูปแบบข้อมูลอนุกรมที่เกิดขึ้นในการติดต่อบนบัส I^2C ของการอ้างถึงแบบ 7 บิต

2.2.7 การอ้างถึงแบบ 10 บิต

ในการอ้างถึงแบบนี้ยังคงใช้รูปแบบข้อมูลอนุกรมที่เหมือนกันแบบ 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับยุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อเช่นเดียวกับการอ้างถึงแบบ 7 บิต หลังจากถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ ในรูปที่ 2.22 แสดงรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ 10 บิต



รูปที่ 2.21 รูปแบบของข้อมูลอนุกรมที่ใช้ติดต่อกับอุปกรณ์บัส I^2C เมื่อใช้การอ้างถึงแบบ 7 บิต



รูปที่ 2.22 รูปแบบของข้อมูลอนุกรมที่ใช้ติดต่อกับอุปกรณ์บัส I^2C เมื่อใช้การอ้างถึงแบบ 10 บิต

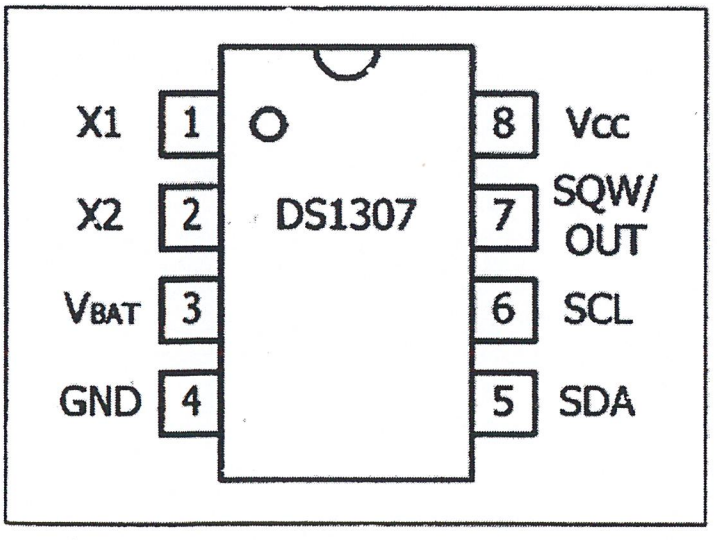
2.3 ไอซีสร้างฐานเวลาจริงเบอร์ DS1307 (RTC : Real Time Clock)

การทำงานคือจะทำหน้าที่สร้างฐานเวลาจริงให้แก่ระบบไมโครคอนโทรลเลอร์โดย DS1307 จะให้ข้อมูลเกี่ยวกับเวลาทั้งหมดไม่ว่าจะเป็นค่าของเวลาที่ละเอียดถึงหลักวินาที , นาที , ชั่วโมง , วันที่ (date) , วันในสัปดาห์ (day) , เดือน และปี โดยสามารถปรับวันเดือนปีให้ตรงตามปฏิทินได้อย่างถูกต้อง รวมถึงการกำหนดวันในปีอธิกสุรทินด้วย คุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

- เป็นไอซีรีลไทม์คล็อก (RTC) ให้ข้อมูลตั้งแต่วินาทีจนถึงปี รวมถึงการปรับวันในปีอธิกสุรทินด้วย สามารถให้ข้อมูลเวลาได้อย่างที่ยงตรงถึงปีคริสตศักราช 2100
- มีหน่วยความจำนอนโวลาไทล์แรม 56 ไบต์อยู่ภายใน สามารถใช้เก็บข้อมูลทั่วไปได้
- ใช้การเชื่อมต่อแบบระบบบัส I^2C
- มีวงจรตรวจจับไฟเลี้ยงต่ำหรือหายไปอย่างอัตโนมัติ และสามารถรักษาข้อมูลเวลาไว้ได้แม้ไม่มีไฟเลี้ยงไอซี

2.3.1 รายละเอียดการต่อใช้งานของ DS1307

ในรูปที่ 2.23 แสดงการจัดขาของ DS1307 แต่ละขามีหน้าที่และการใช้งานดังนี้



รูปที่ 2.23 การจัดขาของไอซี DS1307

V_{CC} , **GND** (ขา 8 , 4) ต่อกับไฟเลี้ยง +5 V

V_{BAT} (ขา 3) ใช้ต่อกับแบตเตอรี่ 3V เพื่อรักษาการทำงานของวงจรสร้างฐานเวลาของ DS1307 ให้คงอยู่ต่อไป แม้ว่าไม่มีไฟเลี้ยงจ่ายให้แก่ DS1307 ชนิดของแบตเตอรี่ที่เหมาะสมคือ

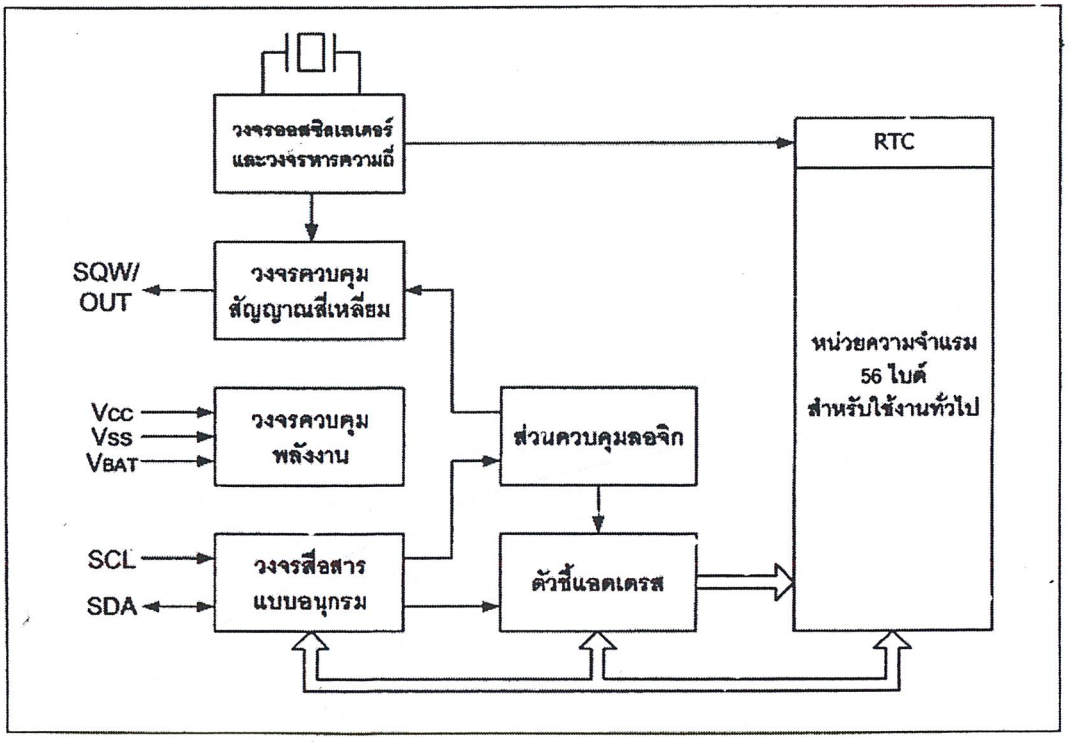
แบตเตอรี่แบบลิเธียม ซึ่งมีความจุ 40 mAhr หรือมากกว่า จะสามารถรักษาข้อมูลได้นาน 10 ปีที่อุณหภูมิ 25 องศาเซลเซียส

SDA , SCL (ขา 5 , 6) เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์ระบบบัส I^2C

SQW/OUT (ขา 7) ที่ขานี้จะมีสัญญาณรูปสี่เหลี่ยมออกมา โดยสามารถเลือกความถี่ได้ 1 Hz, 4.096 kHz, 8.192 kHz และ 32 kHz ในการใช้งานต้องต่อตัวต้านทาน 1 k Pull upที่ขานี้ด้วย

X1 , X2 (ขา 1 , 2) ใช้ต่อกับคริสตอลความถี่มาตรฐาน 32.768 kHz เพื่อใช้เป็นฐานเวลาในการสร้างค่าเวลาจริง ในการใช้งานต้องต่อคริสตอลเข้ากับขาทั้งสองนี้และที่แต่ละขาต้องต่อตัวเก็บประจุค่าต่ำ ๆ ประมาณ 15 pF คร่อมกับขากราวด์ด้วย

2.3.2 การทำงานของ DS1307



รูปที่ 2.24 โครงสร้างภายในของไอซีรีลไทม์คล็อกเบอร์ DS1307

ไอซี DS1307 จัดการเชื่อมต่อในแบบบัส I^2C โดยจะทำงานเป็นอุปกรณ์สเลฟเสมอ ดังนั้นการติดต่อเพื่อใช้งานจึงต้องกำหนดรูปแบบตามที่กำหนดไว้ในการติดต่อแบบ I^2C ในรูปที่ 2.24 แสดงส่วนประกอบหลักที่สำคัญและไดอะแกรมการทำงานของ DS1307 วงจรออสซิลเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง ในขณะที่ DS1307

ทำงานที่ขา SQW/OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมาตลอดเวลาในกรณีที่มีการอีนาเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจิสเตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ 4 ค่า คือ 1 Hz, 4.096 kHz, 8.192 kHz และ 32 kHz พร้อมกันนั้นก็จะมีการเก็บค่าของเวลาไว้ในหน่วยความจำนอนโวลาไทล์แรม ซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไปสำหรับผู้ใช้งานอีก 56 ไบต์

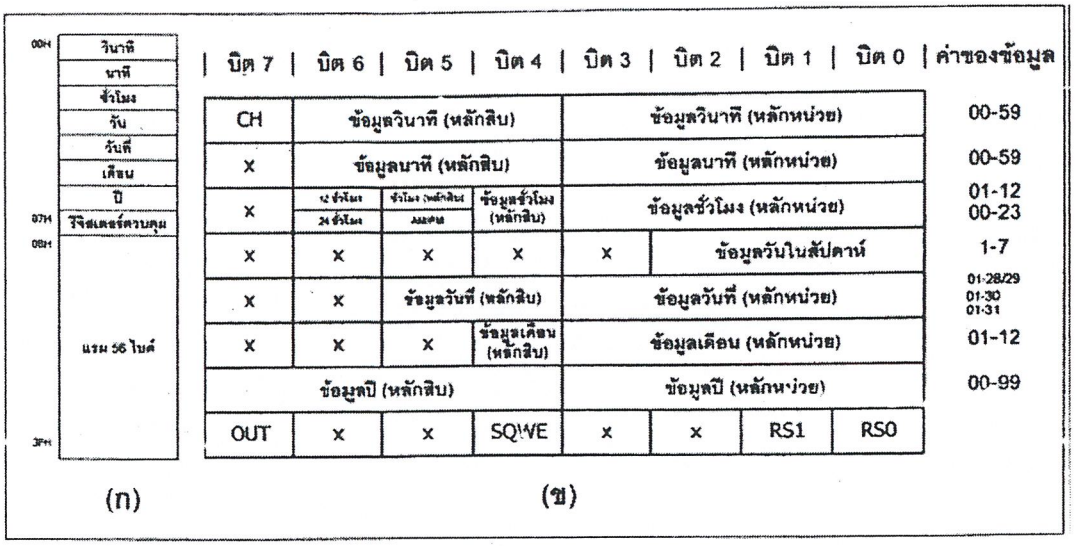
วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า $1.25 \times V_{BAT}$ ก็จะควบคุมให้ DS1307 หยุดการทำงานรีเซตค่าตัวนับแอดเดรสภายในทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นในการใช้งาน DS1307 ต้องระมัดระวังอย่าให้ไฟเลี้ยงตกต่ำกว่า $1.25 \times V_{BAT}$ หรือประมาณ 3.75 V ในกรณีที่ใช้ V_{BAT} เท่ากับ 3 V ถ้าหากไฟเลี้ยงมีค่าต่ำกว่า V_{BAT} ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที จะไม่มีการส่งสัญญาณพัลส์ออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานได้ต่อไป

วงจรสื่อสารอนุกรมภายใน DS1307 ได้รับการกำหนดให้ทำงานตามรูปแบบของบัส I^2C เป็นช่องทางการสื่อสารระหว่าง DS1307 กับอุปกรณ์มาสเตอร์ ผู้ใช้งานสามารถเข้าถึงหน่วยความจำที่ใช้เก็บค่าเวลาและหน่วยความจำใช้งานทั่วไปได้โดยการเขียนข้อมูลตามรูปแบบที่กำหนดในระบบบัส I^2C

2.3.3 การจัดสรรหน่วยความจำใน DS1307

ในรูปที่ 2.25 (ก) แสดงการจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7 ไบต์แรกตั้งแต่แอดเดรส 00H-06H เป็นพื้นที่ของรีจิสเตอร์ค่าเวลาใช้ในการเก็บข้อมูลเกี่ยวกับเวลาไบต์ต่อมาที่แอดเดรส 07H เป็นพื้นที่ของรีจิสเตอร์ควบคุมการทำงานของ DS1307 ในรูปที่ 2.25 (ข) แสดงรายละเอียดของรีจิสเตอร์ค่าเวลาและรีจิสเตอร์ควบคุมของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ ทำให้ผู้ใช้งานสามารถเรียกข้อมูลเวลาออกมาได้ตามที่ต้องการโดยไม่จำเป็นต้องอ่านออกมาทั้งหมดก็ได้ ค่าของเวลาทั้งหมดจะอยู่ในรูปของเลขฐานสิบ สำหรับการแสดงเวลาในรูปของชั่วโมง สามารถเลือกได้ว่าต้องการแบบ 12 หรือ 24 ชั่วโมง โดยกำหนดที่บิต 6 ของแอดเดรส 02H และเมื่อเลือกแบบ 12 ชั่วโมง ที่บิต 5 ในแอดเดรสเดียวกันจะใช้ในการแสดงค่า AM/PM โดยถ้าบิตนี้เป็น "1" หมายถึง ค่าชั่วโมงในขณะนี้เป็นช่วงเวลาหลังเที่ยงวัน ในกรณีที่เป็นแบบ 24 ชั่วโมง บิตนี้จะใช้ในการแสดงค่า 2 ของหลักสิบในหน่วยชั่วโมง



รูปที่ 2.25 (ก) การจัดสรรหน่วยความจำแรมภายใน DS1307

(ข) รายละเอียดของรีจิสเตอร์เก็บค่าเวลาและรีจิสเตอร์ควบคุมของ DS1307

2.3.4 รีจิสเตอร์ควบคุม

มีแอดเดรสอยู่ที่ 07H มีรายละเอียดของแต่ละบิตดังนี้

OUT (Output control) : ใช้ในการควบคุมระดับลอจิกที่ขา SQW/OUT ในกรณีที่ดิสเอเบิลการกำเนิดสัญญาณสี่เหลี่ยม โดยถ้าบิตนี้เป็น “1” ที่ขา SQW/OUT ก็จะเป็น “1” ถ้าบิตนี้เป็น “0” ที่ขา SQW/OUT ก็จะเป็น “0”

SQWE (Square Wave Enable) : ใช้ในการอีนาเบิลวงจรกำเนิดสัญญาณสี่เหลี่ยมที่ขา SQW/OUT ถ้าต้องการให้มีสัญญาณสี่เหลี่ยมออกให้กำหนดบิตนี้เป็น “1”

RS1 , RS0 (Rate Select) : ใช้ในการเลือกความถี่สัญญาณสี่เหลี่ยมที่ออกจากขา SQW/OUT ดังมีรายละเอียดต่อไปนี้

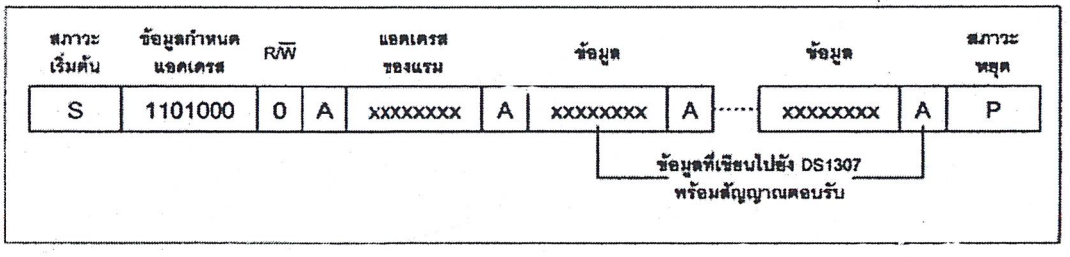
RS1	RS0	ค่าความถี่ของสัญญาณสี่เหลี่ยม
0	0	1 Hz
0	1	4.096 kHz
1	0	8.192 kHz
1	1	32.768 kHz

2.3.5 โหมดการทำงานของ DS1307

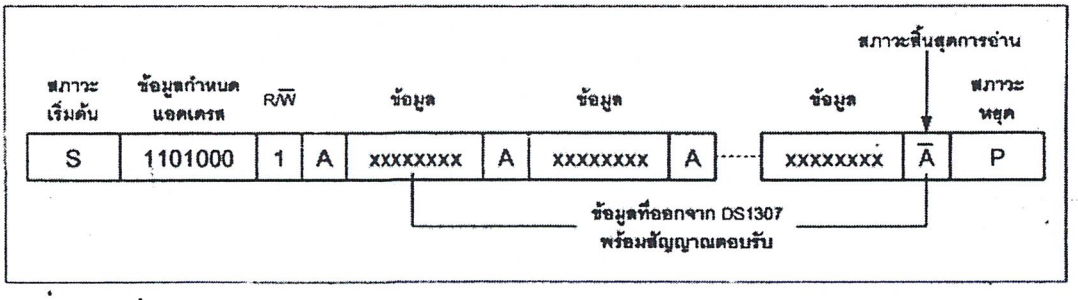
มีด้วยกัน 2 โหมด คือ โหมดเขียนข้อมูลและโหมดอ่านข้อมูล ในการใช้งาน DS1307 ตามปกติจะใช้งานเฉพาะโหมดอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1307 เพื่ออ่านข้อมูลของเวลาไปใช้งาน โหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาใหม่และต้องการเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไปอย่างไรก็ตามเมื่อเริ่มต้นติดต่อกับ DS1307 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นโหมดการอ่านข้อมูลต่อไป

(1) โหมดการเขียนข้อมูล

มีรูปแบบดังในรูปที่ 2.26 เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ทำการกำหนดสภาวะเริ่มต้น (START : S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียน นั่นคือค่า 0 จากนั้นจะรอการตอบรับจาก DS1307 ขั้นตอนต่อมา คือ ส่งข้อมูลเพื่อเลือกแอดเดรสที่ต้องการเขียน จากนั้นรอการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มทยอยเขียนข้อมูลลงไปครั้งละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรส จะต้องหยุดรอการตอบรับจาก DS1307 ทุกครั้ง จึงจะสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วให้ส่งสภาวะหยุด (STOP : P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล



รูปที่ 2.26 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการเขียนข้อมูล



รูปที่ 2.27 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการอ่านข้อมูล

(2) โหมดการอ่านข้อมูล

มีรูปแบบแสดงในรูปที่ 2.27 เริ่มต้นการทำงานเหมือนกับโหมดการเขียนข้อมูล คือ ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้นแล้วส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่านซึ่งเท่ากับ 1 จากนั้นรอการตอบรับจาก DS1307 เมื่อตอบรับเรียบร้อย DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์คราวละ 1 แอดเดรสหรือ 1 ไบต์ โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดมาก่อนล่วงหน้าด้วยโหมดการเขียนข้อมูล วิธีการง่าย ๆ คือ เข้าสู่โหมดการเขียนข้อมูลก่อน เมื่อถึงจังหวะที่ต้องเขียนข้อมูลให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูลกำหนดแอดเดรสใหม่อีกครั้ง ตามด้วยเลือกโหมดการอ่านข้อมูล ข้อมูลที่ออกมาจาก DS1307 ก็จะเป็นข้อมูลจากแอดเดรสที่กำหนดไว้ก่อนหน้านี้

2.4 LCD (Liquid Crystal Displays)

2.4.1 รายละเอียดเกี่ยวกับโมดูล LCD

ในโมดูล LCD จะมีส่วนประกอบหลัก ๆ 3 ส่วน ดังนี้

ตัวแสดงผล (display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

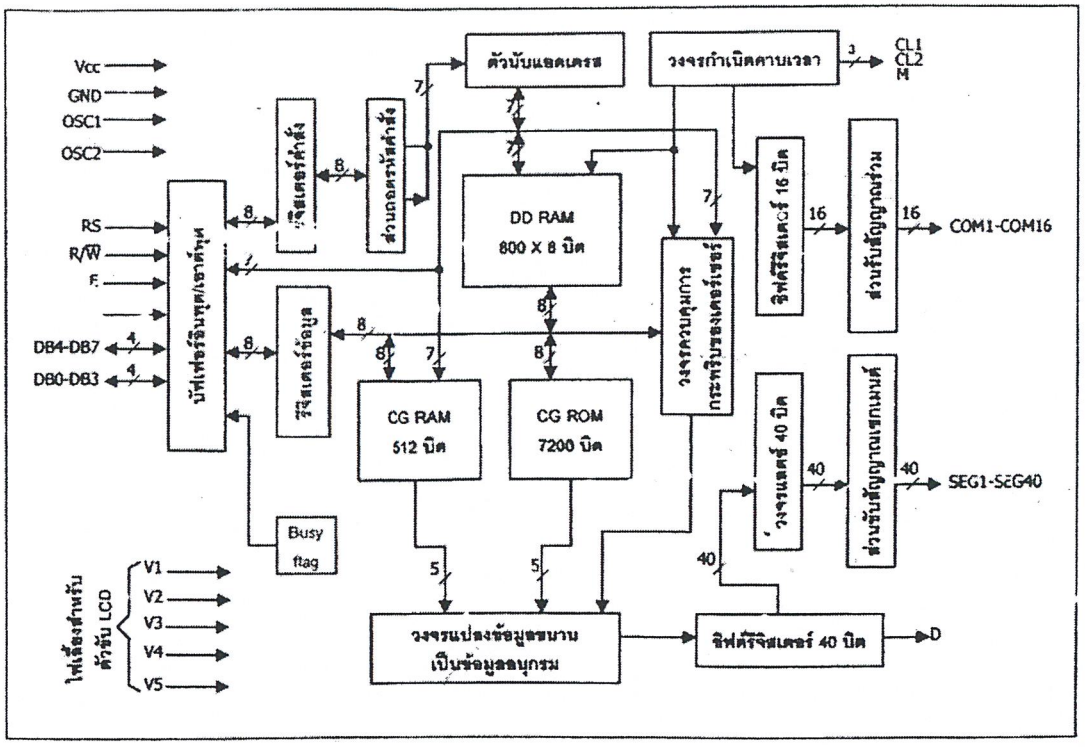
ตัวควบคุม (controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะชิปที่นิยมใช้ คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

2.4.2 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในที่นี้ขอยกตัวอย่างโมดูล LCD แบบอักขระ ซึ่งเป็น LCD ที่ใช้ในตัว project จริง ๆ เพราะสามารถเข้าใจได้ง่าย ให้รูปที่ 2.28 เป็นบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย

บัฟเฟอร์อินพุทเอาท์พุท เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม



รูปที่ 2.28 ไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

รีจิสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register : DR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอกเพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data RAM : DDRAM) เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up-lable) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษร เพื่อนำไปแสดงที่ตัวแสดงผล

รอมเก็บตัวอักษร (Character Generator ROM : CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงผลที่ตัวแสดงผลได้ มีขนาด 7,200 บิต โดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ มีขนาด 512 บิต

การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

2.4.3 โมดูล LCD ขนาด 16 ตัวอักษร 2 บรรทัด (LCD 16×2)

โมดูล LCD ขนาด 16×2 มีขาต่อใช้งานทั้งสิ้น 14 ขา สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

V_{SS} (ขา 1) : ต่อกราวด์

V_{DD} (ขา 2) : ต่อไฟเลี้ยง +5 โวลต์

V_O (ขา 3) : เป็นขาอินพุทรับแรงดันเพื่อปรับความเข้มการแสดงผล

RS (ขา 4) : เป็นขาอินพุทใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น "0" ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น "1" ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/$\overline{w}$ (ขา 5) : เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น "0" เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาอีนาเบิล LCD ให้ทำงาน

D0-D7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต

2.4.4 คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ R/W เป็น "0" แล้วเขียนคำสั่งตามไป คำสั่งควบคุมโมดูล LCD ของชิปควบคุม HD44780 ที่สำคัญมี 10 คำสั่งดังนี้

1. คำสั่งเคลียร์ตัวแสดงผล (clear display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซิคิวท์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D (ซึ่งจะกล่าวถึงภายหลัง) ให้เป็น "1"

2. คำสั่ง return home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลงนั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode Set)

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	0	1	I/D	S

บิต **S** เป็นบิตที่ใช้ในการกำหนดลักษณะของการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิต S เป็น "1" เมื่อเกิดข้อมูลใหม่บนจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย แต่ถ้าหากบิตนี้เป็น "0" เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต **I/D** เป็นบิตที่ใช้ในการกำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว ทำให้แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0" แอดเดรสจะลดลง

ดังนั้น ข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H – 07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดให้เมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรสของ DDRAM เพิ่มขึ้น

4. คำสั่งควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	1	D	C	B

บิต **D** ใช้ควบคุมการเปิดปิดจอแสดงผล ถ้าบิตนี้เป็น "1" จะเป็นการเปิดจอแสดงผล ถ้าเป็น "0" จะเป็นการปิดจอแสดงผล

บิต **C** ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปิดเคอร์เซอร์ หรือไม่แสดงเคอร์เซอร์

บิต **B** ใช้ควบคุมการกะพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกะพริบ

ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H – 0FH (8 รูปแบบคำสั่ง) ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์ และ 0FH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์ และสั่งให้เคอร์เซอร์กะพริบ

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	1	S/C	R/L	*	*

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผลขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

S/C	R/L	ลักษณะการเลื่อน	ข้อมูลคำสั่ง
0	0	เลื่อนเคอร์เซอร์ไปทางซ้าย	10H – 13H
0	1	เลื่อนเคอร์เซอร์ไปทางขวา	14H – 17H
1	0	เลื่อนตัวอักษรใหม่ไปทางซ้าย	18H – 1BH
1	1	เลื่อนตัวอักษรใหม่ไปทางขวา	1CH – 1FH

6. คำสั่งกำหนดฟังก์ชันการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	1	DL	N	F	*	*

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น “0” จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าเป็น “1” จะเป็นแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น “0” จะแสดงผล 1 บรรทัด ถ้าเป็น “1” จะแสดงผล 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัด ก็กำหนดบิต N นี้ให้เป็น “1”

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น “0” จะเป็นการแสดงผลแบบ 5×7 จุด และถ้าเป็น “1” จะแสดงผลเป็นแบบ 5×10 จุด

ข้อมูลคำสั่งที่ใช้บ่อยคือ 38H เป็นการกำหนดให้โมดูล LCD ทำงานในแบบ 8 บิต แสดงผล 2 บรรทัด และเลือกความละเอียดเป็น 5×7 จุด

7. คำสั่งเลือกแอดเดรสของ CGRAM

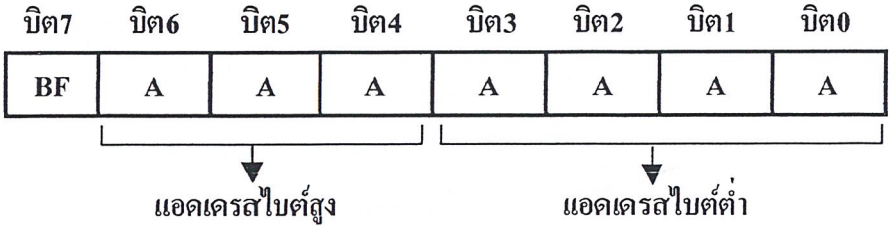
เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น "0" บิต 6 เป็น "1" ส่วนอีก 6 บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ก่อนที่จะอ่านหรือเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H-3FH

8. คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดยบิต 7 ต้องเป็น "1" และข้อมูลอีก 7 บิตที่เหลือจะเป็นค่าแอดเดรสของ DDRAM ซึ่งแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH-0FFH ทั้งนี้จำนวนแอดเดรสยังขึ้นกับการกำหนดสถานะที่บิต N ด้วย หากบิต N เป็น "0" แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H-0CFH และถ้าบิต N เป็น "1" แอดเดรสของ DDRAM จะมี 2 ช่วงคือ 8CH-87H และ 0C0H-0C7H

9. คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้



เป็นคำสั่งที่ใช้อ่านแฟลก BUSY (BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ ถ้าหากบิต BF เป็น "0" แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น "1" แสดงว่า ขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง

เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา $R/\overline{w}$ เป็น "1" ด้วย แต่สัญญาณที่ RS ยังต้องเป็น "0" อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง

นอกจากนี้ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต0–บิต6 เป็นค่าข้อมูลของ แอดเดรสที่ต้องการอ่าน

2.4.5 การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้งานต้องการ ต้องส่งคำสั่ง (instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (data) ที่ต้องการแสดงผลเนื่องจากบัสข้อมูลของโมดูล LCD มี 8 เส้น คือ D0-D7 และใช้เป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ถ้าหากที่ขา RS ได้ลอจิก "0" หมายความว่า ข้อมูลที่ป้อนให้แก่โมดูล LCD ขณะนั้นเป็นคำสั่งในทางตรงข้าม หากขา RS ได้รับลอจิก "1" ข้อมูลที่ป้อนให้ขณะนั้นเป็นข้อมูลที่ใช้ในการแสดงผล

เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อน โดยใช้คำสั่งเลือกแอดเดรส จากนั้นกำหนดให้ขา RS เป็น "1" เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลบ่กติไม่ใช่คำสั่ง

ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา $R/\overline{w}$ เป็น "1" ข้อมูลขนาด 8 บิต (หรือ 4 บิต) ก็จะปรากฏบนบัสข้อมูล โดยข้อมูลที่อ่านออกมาได้จะเป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ

ในกรณีที่ต้องการเขียนข้อมูล เมื่อกำหนดแอดเดรสและป้อนลอจิก "1" ให้ขา RS แล้ว แล้วต้องกำหนดให้ขา $R/\overline{w}$ เป็น "0" ข้อมูลที่อยู่บนบัสข้อมูลจะถูกเขียนลงในรีจิสเตอร์ DR จากนั้นจึงถ่ายทอดลงใน DDRAM ต่อไป

2.4.6 จังหวะการทำงานของ LCD โมดูล

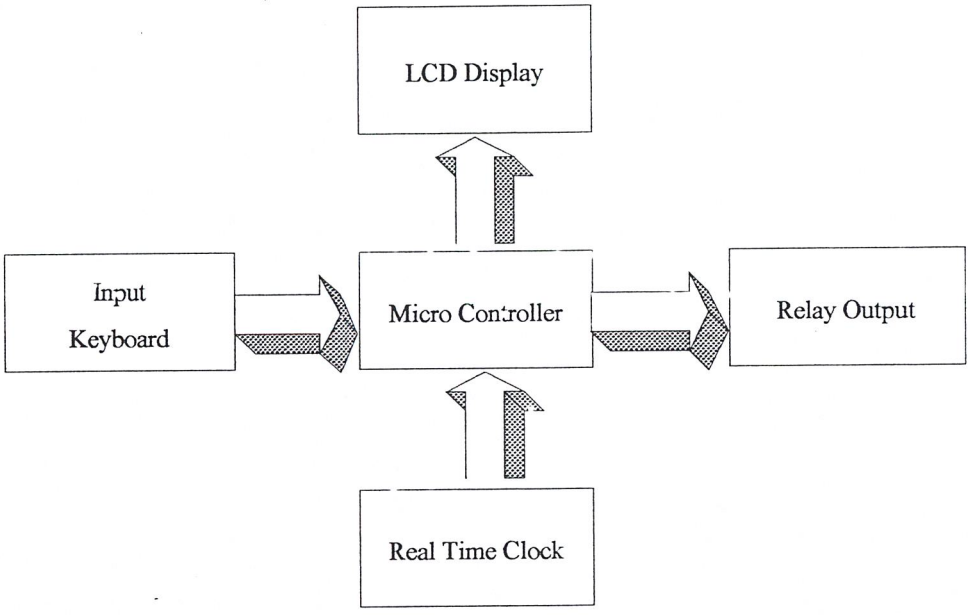
ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูลเนื่องจากต้องรอให้คอนโทรลเลอร์ภายใน LCD โมดูล แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อน จากนั้นจึงจะรับข้อมูลหรือดำเนินการต่อไป

ดังนั้น ในการใช้งานโมดูล LCD ผู้เขียนโปรแกรมต้องมีโปรแกรมเพื่อหน่วงเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้แก่โมดูล LCD ต้องรอประมาณ 10 มิลลิวินาที เพื่อให้โมดูล LCD ทำการเตรียมความพร้อมหรืออินิเชียล (initial) หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขา RS ของโมดูล LCD แล้วต้องหน่วงเวลาอีกประมาณ 2 มิลลิวินาทีเพื่อให้คอนโทรลเลอร์ใน LCD โมดูลแปลความหมายของลอจิกที่ขา RS ว่าข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่ออีนาเบิลโมดูล LCD ให้รับข้อมูลจากบัสข้อมูลเข้าไป โดยพัลส์ที่ป้อนเข้าที่ขา E ของโมดูล LCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาที

ทั้งหมดที่กล่าวมาคือขั้นตอนและจังหวะในการทำงาน 1 รอบของโมดูล LCD จะเห็นได้ว่ามีโปรแกรมย่อยที่สำคัญอยู่ 3 โปรแกรมย่อยคือ โปรแกรมอินนิเชียล LCD , โปรแกรมหน่วงเวลา และ โปรแกรมย่อยการส่งพัลส์เพื่ออีนาเบิลโมดูล LCD

บทที่ 3
การคำนวณและการสร้าง

3.1 แนวคิดและหลักการทำงาน



รูปที่ 3.1 บล็อกไดอะแกรมของเครื่องควบคุมอุปกรณ์ไฟฟ้าแบบโปรแกรมได้

จากรูปที่3.1แสดงบล็อกไดอะแกรมของเครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้ ซึ่งจะเห็นได้ว่ามีไมโครคอนโทรลเลอร์เป็นหน่วยประมวลผลกลาง (CPU) ซึ่งจะทำหน้าที่ติดต่อกับอุปกรณ์ต่าง ๆ ที่นำมาต่อร่วม คือ คีย์แพด (Key Pad), ส่วนแสดงผล (LCD), วงจรฐานเวลาอ้างอิง (Real Time Clock), เอาท์พุท (Relay Output) ซึ่งในการออกแบบและหลักการทำงานของวงจรต่าง ๆ สามารถอธิบายได้ดังนี้

ในระบบจะมีวงจรฐานเวลาอ้างอิง (Real Time Clock : RTC) เป็นตัวสร้างเวลาจริงให้กับระบบซึ่งเป็นหัวใจหลักของระบบ โดยที่วงจรนี้จะทำการผลิตสัญญาณนาฬิกาไปประมวลผลที่ไมโครคอนโทรลเลอร์ เพื่อไปอ้างอิงในการ เปิด-ปิด (ON-OFF) เอาท์พุทที่เป็นอุปกรณ์ไฟฟ้าภายในบ้าน ในส่วนของวงจรนี้จะต้องทำงานตลอดเวลาเพื่อให้สามารถสร้างเวลาที่แท้จริงอยู่ตลอด ทำให้การอ้างอิงเพื่อไป เปิด-ปิด (ON-OFF) สามารถทำได้อย่างถูกต้องตามเวลาที่แท้จริง ในส่วนของอินพุท (Key Pad) โดยจะเป็นคีย์แพดแบบ 4 x 4 จะเป็นส่วนที่ทำการป้อนค่าเข้าไปตามที่ต้องการ โดยที่เมื่อทำการป้อนค่าเข้าไปจะไปทำการกำหนดค่าต่าง ๆ แล้วดึงเอาโปรแกรมต่าง ๆ ที่ถูกเขียนอยู่ในไมโครคอนโทรลเลอร์มาใช้งาน กล่าวคือ การกำหนดให้เอาท์พุท เปิด-ปิด จะต้องกำหนดที่คีย์แพด

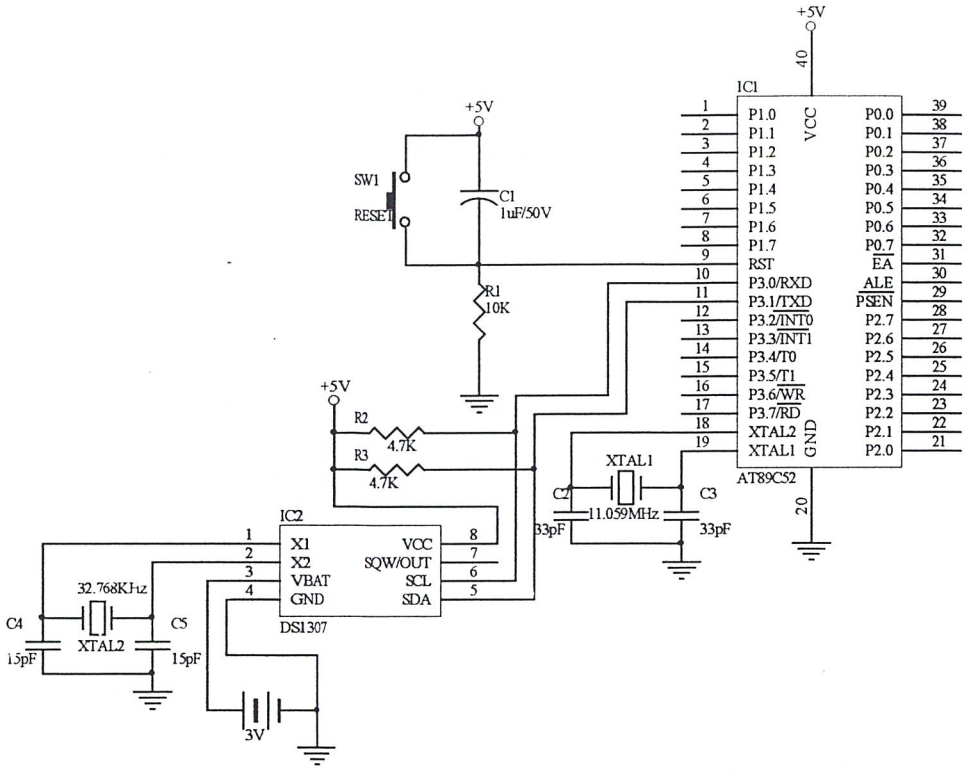
โดยมีฐานเวลาอ้างอิงอยู่ และในส่วนของส่วนแสดงผล (LCD) ก็เป็นเพียงการแสดงค่าต่าง ๆ โดยหลัก ๆ แล้วจะทำการแสดงเวลาอยู่ตลอดเวลาและแสดงค่าการกำหนดเอาท์พุทต่าง ๆ นั่นเอง และในบล็อกไดอะแกรมสุดท้าย คือ เอาท์พุท (Relay Output) เป็นวงจรรีเลย์ที่ทำหน้าที่เป็นสวิตช์เปิด-ปิด (ON-OFF) อุปกรณ์ไฟฟ้า โดยอาศัยการกำหนดค่าจากคีย์แพดแล้วสัญญาณจะถูกประมวลผลที่ไมโครคอนโทรลเลอร์ซึ่งสัญญาณที่ถูกประมวลผลจะถูกส่งมาควบคุมให้รีเลย์ทำการเปิด-ปิดได้ตามเวลาที่ต้องการ โดยอาศัยโปรแกรมในไมโครคอนโทรลเลอร์เป็นตัวอ้างอิง

จากหลักการทั้งหมดจะเห็นว่า แนวคิดในการสร้างเครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้นี้จะใช้การเปิด-ปิด (ON-OFF) ให้ได้ตามเวลาที่ต้องการโดยให้ระบบมีฐานเวลาอ้างอิงเท่านั้น ซึ่งในการสร้างก็จะเป็นการเขียนโปรแกรมรูปแบบต่าง ๆ เพื่อไปกำหนดเอาท์พุทให้ทำงานได้ตามเวลาที่กำหนด ซึ่งในแต่ละเอาท์พุทนั้นจะสามารถกำหนดให้ทำงานได้ในรอบสัปดาห์หรือรอบวัน โดยแต่ละเอาท์พุทสามารถตั้งโปรแกรมให้ทำงานได้ทั้งหมด 4 รอบ

3.2 วงจรฐานเวลาอ้างอิง (Real Time Clock : RTC)

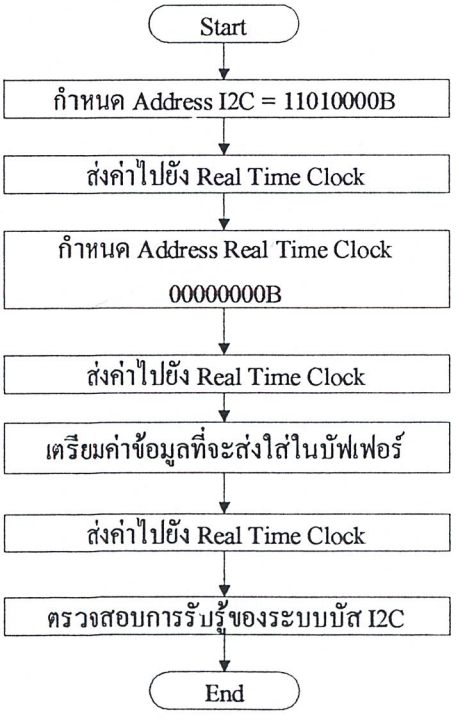
ในรูปที่ 3.2 แสดงวงจรการต่อใช้งานไอซีสร้างฐานเวลาจริง (Real Time Clock : RTC) เบอร์ DS1307 ซึ่งจะต้องมี crystal ขนาด 32.768 KHz และตัวเก็บประจุค่าต่ำ ๆ ประมาณ 15 pF ต่ออยู่ที่ขา X1 และ X2 เพื่อใช้เป็นตัวกำเนิดสัญญาณนาฬิกาในการสร้างค่าเวลาจริง สำหรับการต่อร่วมกับไมโครคอนโทรลเลอร์มีลักษณะการต่อใช้งานในลักษณะอุปกรณ์ระบบบัส I^2C (ดูรายละเอียดอุปกรณ์ระบบบัส I^2C ได้ในหัวข้อที่ 2.2) ซึ่งรายละเอียดการต่อใช้งานนั้นมีการต่อขา SDA และ SCL เพื่อทำการติดต่อกับไมโครคอนโทรลเลอร์เพียง 2 เส้นเท่านั้น ซึ่งเป็นความสามารถพิเศษของอุปกรณ์ระบบบัส I^2C และจะต้องต่อความต้านทานพูลอัพที่ 2 ขานี้ด้วยเพื่อคงสภาวะเริ่มต้นของ DS1307 ให้เป็น high โดยหากไม่มีการติดต่อกันระหว่าง DS1307 กับ ไมโครคอนโทรลเลอร์ที่พอร์ต 3.0 และ 3.1 จะเป็น high แต่ถ้ามีการติดต่อกันระหว่าง DS1307 กับไมโครคอนโทรลเลอร์ที่พอร์ต 3.0 และ 3.1 จะเป็น low

จากวงจรมีการต่อแบตเตอรีที่ขา V_{BAT} ไว้ตลอดเวลาไม่ว่าจะใช้งานหรือไม่ ทั้งนี้เพื่อรักษาการทำงานของวงจรภายในDS1307ไว้ให้ยังคงทำงานต่อเนื่องไปได้ เมื่อใดที่ไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลก็จะได้ข้อมูลเวลาที่เป็นจริงอยู่ตลอดเวลา



รูปที่ 3.2 วงจรการต่อใช้งานไอซีสร้างฐานเวลาจริง (Real Time Clock : RTC)

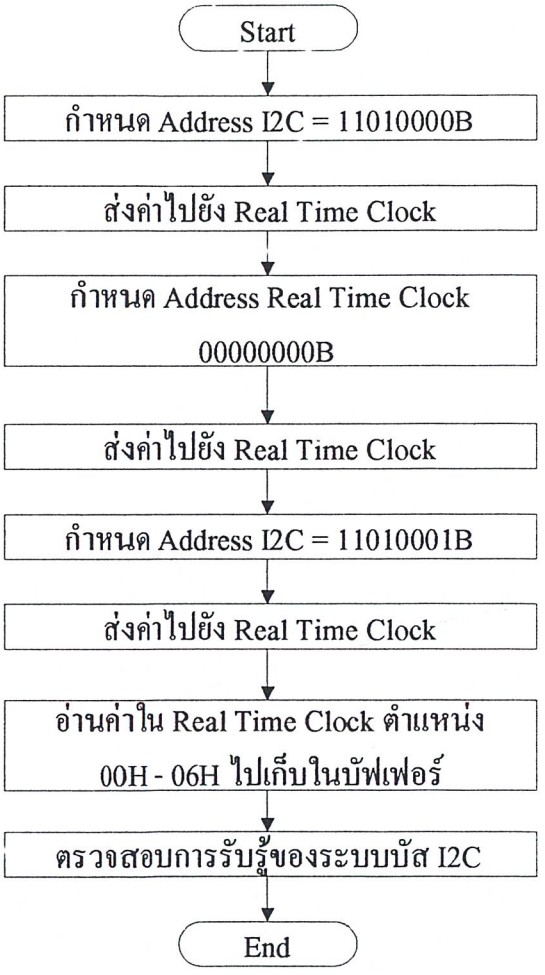
3.2.1 การเขียนข้อมูลลงไอซีสร้างฐานเวลาจริง (Real Time Clock) DS1307



รูปที่ 3.3 Flowchart การเขียนข้อมูลลง Real Time Clock

การทำงานของ Folwchart ดังรูปที่ 3.3 เป็นการติดต่อกับไอซีสร้างฐานเวลาจริง (Real Time Clock) ในโหมดการเขียนค่า ต้องกำหนด Address ในการเขียน (Address = 11010000B) จากนั้นส่งค่าไปให้กับไอซีสร้างฐานเวลาจริง เมื่อส่งค่าข้อมูลเสร็จสิ้น ก็กำหนดแอดเดรส Real Time Clock ที่ต้องการจะเขียนข้อมูลลงไป ในที่นี้เราต้องการเขียนข้อมูลลง ณ ตำแหน่ง 00H – 06H ดังนั้นเราจึงนำค่าข้อมูลในบัฟเฟอร์เขียนลงในตำแหน่ง 00H – 06H โดยการเขียนข้อมูลลงแต่ละครั้งจะทำการตรวจสอบการรับรู้ของระบบบัส I^2C ว่าพร้อมทำงานต่อไปหรือยัง

3.2.2 การอ่านค่าจากไอซีสร้างฐานเวลาจริง (Real Time Clock) DS1307



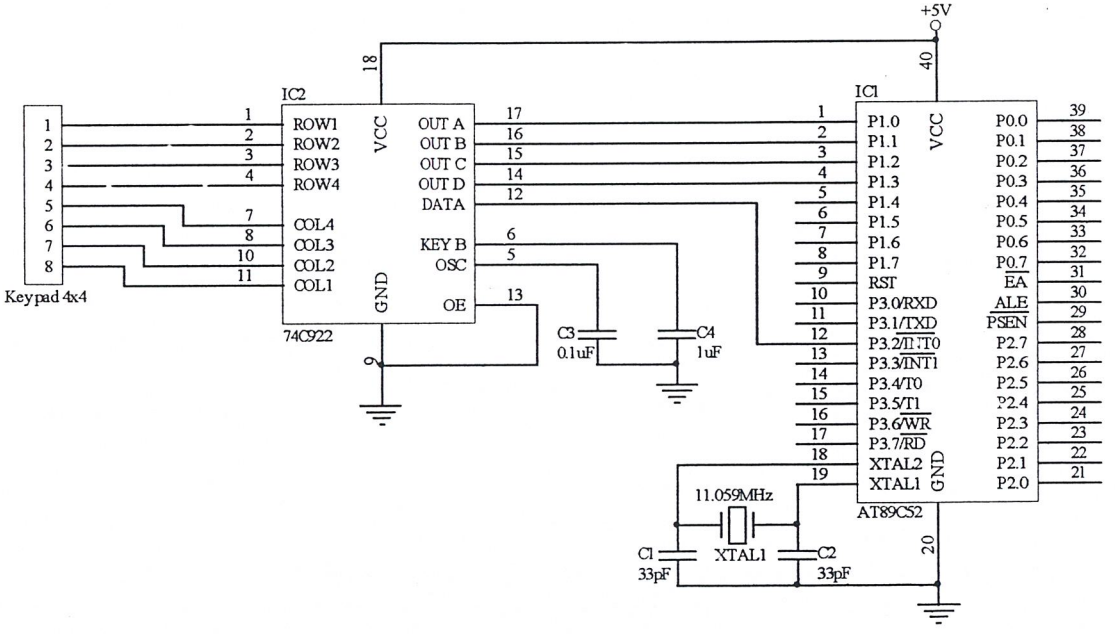
รูปที่ 3.4 Flowchart การอ่านข้อมูลจากไอซี Real Time Clock

การทำงานของ Flowchart ดังรูปที่ 3.4 เป็นการติดต่อกับไอซีสร้างฐานเวลาจริง ในโหมดการอ่านค่าต้องกำหนดแอดเดรสในการเขียน (Address = 11010000B) จากนั้นส่งค่าไปให้ไอซีสร้างฐานเวลาจริง ซึ่งการส่งค่าจะต้องส่งค่า start bit ตามด้วยข้อมูลจากนั้นต้องส่ง stop bit เมื่อการ

ส่งค่าข้อมูลเสร็จสิ้น จากนั้นทำการกำหนดแอดเดรสในไอซีสร้างฐานเวลาจริง ที่จะอ่านค่าออกมา ซึ่งในที่นี้เราต้องการอานค่าตั้งแต่ตำแหน่ง 00H – 06H ซึ่งเป็นตำแหน่งที่เก็บค่า วินาที, นาที, ชั่วโมง, วัน, วันที่, เดือน, ปี ตามลำดับ

การอ่านค่าข้อมูลแต่ละครั้งจะนำมาเก็บในบัฟเฟอร์เพื่อนำไปใช้ต่อไป และการอ่านค่าแต่ละครั้งก็จะทำการตรวจสอบการรับรู้ของระบบบัส I^2C ด้วยว่าพร้อมทำงานหรือยัง การที่จะอ่านค่าต้องกำหนดแอดเดรสในการอ่านก่อน

3.3 วงจร Scan Keyboard



รูปที่ 3.5 วงจรการเชื่อมต่อ Key Pad โดยใช้ IC Scankey 74C922

คีย์บอร์ดแบบเมตริกซ์ 4 x 4 หรือเรียกอีกอย่างว่า คีย์แพด (Key Pad) เป็นเทคนิคการนำสวิตช์จำนวน 16 ตัวมาต่อกันแบบแนวแกนตั้งและแนวแกนนอน จะเรียกแนวแกนตั้งว่า คอลัมน์ (Column) และแนวแกนนอนว่าโรว์ (Row) ซึ่งเมื่อต่อแล้วจะได้สายสัญญาณทั้งหมด 8 เส้น แบ่งเป็นแนวแกนตั้ง 4 เส้น และแนวแกนนอน 4 เส้น จะเห็นว่าถ้าต่อสวิตช์โดยตรงจะต้องมีสายสัญญาณถึง 16 เส้น ดังนั้น การใช้คีย์แพด 4 x 4 นี้ มีข้อดี คือ ช่วยลดสายสัญญาณที่จะไปเชื่อมต่อกับไมโครคอนโทรลเลอร์ให้น้อยลงได้ทำให้สามารถนำพอร์ตที่เหลือของไมโครคอนโทรลเลอร์ไปใช้งานอย่างอื่นได้

ในรูปที่ 3.5 แสดงการเชื่อมต่อคีย์แพด 4 x 4 กับไมโครคอนโทรลเลอร์โดยจะมี IC Scankey เบอร์ 74C922 ซึ่งเป็น 16 key encoder เป็นตัวเชื่อมต่อระหว่างคีย์แพด 4 x 4 กับไมโครคอนโทรลเลอร์ โดยที่เอาท์พุทของไอซีนี้จะมี 4 เส้น เพื่อต่อกับไมโครคอนโทรลเลอร์ดังรูป ข้อดีของ IC Scankey นี้ คือ จะช่วยลดความยุ่งยากในการเขียนโปรแกรมทำให้โปรแกรมมีขนาดสั้นลง ไม่เปลือง Flash Memory ของไมโครคอนโทรลเลอร์ ทั้งนี้เพราะว่าเราไม่จำเป็นต้องไปสร้าง Table เพื่อทำการจดจำค่าคีย์ของแต่ละคีย์นั่นเอง จากวงจรในรูปที่ 3.5 ถ้าทำการกดคีย์ต่าง ๆ ที่คีย์แพดจะได้ค่าเอาท์พุท A-D ของ IC Scankey ดังในตารางที่ 3.1

Key	1	2	3	↑	4	5	6	↓	7	8	9	2nd	clear	0	Help	Ent
A	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
B	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
C	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
D	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

ตารางที่ 3.1 แสดงค่าเอาท์พุทที่ได้จาก IC Scankey 74C922

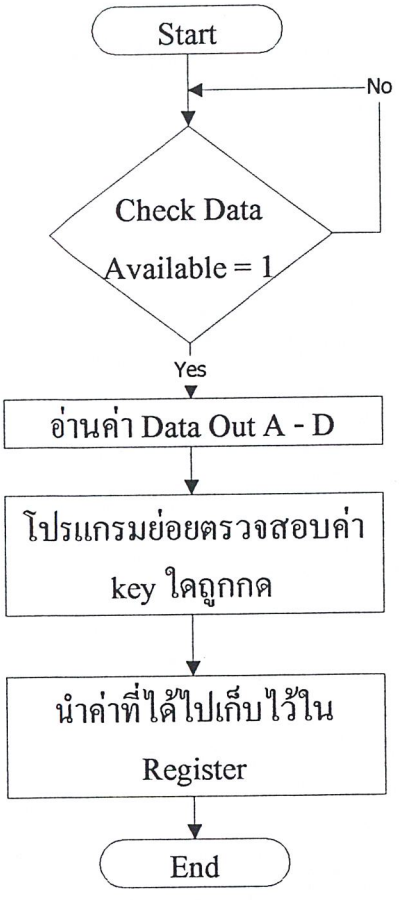
จากค่าเอาท์พุทของ IC Scankey ในตารางที่ 3.1 จะเป็นตัวกำหนดว่า คีย์ใดถูกกด เช่นถ้าทำการกดเลข 1 จะได้ลอจิกเป็น 0000 ดังนั้นเมื่อลอจิก 0000 ถูกส่งเข้าไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์ก็จะทำการประมวลผลว่า คีย์ หมายเลข 1 ถูกกด จากนั้นตัวไมโครคอนโทรลเลอร์ก็จะทำงานตามที่โปรแกรมไว้

จากวงจรในรูปที่ 3.5 มีวิธีการกำหนดค่าตัวเก็บประจุที่ต่ออยู่ที่ขา OSC และ KEY B กล่าวคือ ตัวเก็บประจุที่ขา OSC เป็นตัวกำหนดค่าความถี่ในการสแกนคีย์ (F_{scan} (Hz)) และตัวเก็บประจุที่ขา KEY B จะเป็นตัวกำหนดค่าเวลาในการกดว่าให้ระยะเวลาในการกดคีย์แรกและคีย์ถัดไปมีระยะเวลาห่างกันเท่าใด มีหน่วยเป็น second โดยตัวเก็บประจุที่ขา KEY B จะต้องเป็น 10 เท่าของตัวเก็บประจุที่ขา OSC จากรูปเลือกใช้ค่าตัวเก็บประจุดังนี้

C_{osc} = 0.1 uF $\Rightarrow$ F_{scan} = 600 Hz

C_{KBM} = 1 uF $\Rightarrow$ Debounce Period = 0.01 sec

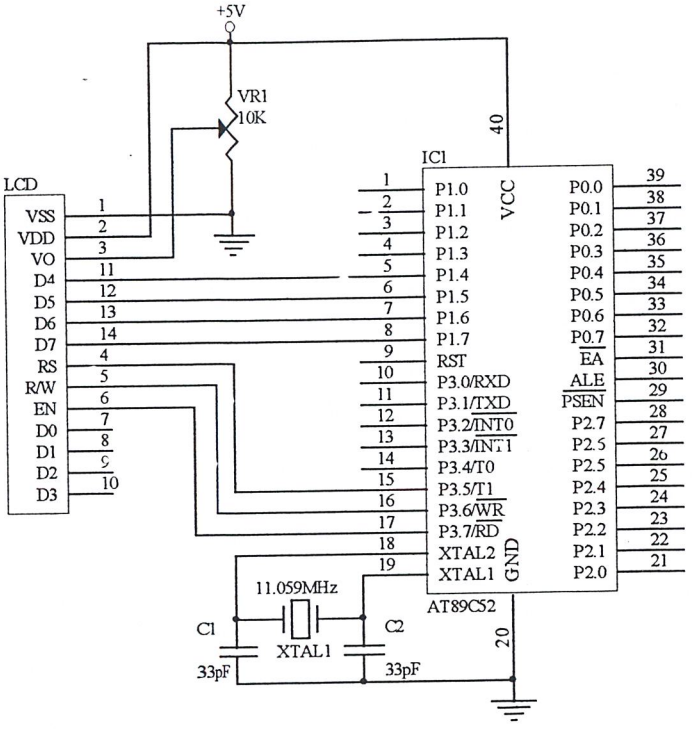
(รายละเอียดการเลือกค่าตัวเก็บประจุดูได้จากภาคผนวก)



รูปที่ 3.6 Flowchart การติดต่อกับ IC Scankey 16 Key Encoder

การทำงานของ Flowchart ดังรูปที่ 3.6 เป็นการติดต่อกับ IC Scankey 16 Key Encoder โปรแกรมจะทำการตรวจสอบค่า Key Data ว่ามีค่าเป็น "1" หรือไม่ ถ้าเป็น "1" แสดงว่ามีการกดคีย์ใด ๆ เกิดขึ้น แต่ถ้า Key Data มีค่าเป็น "0" แสดงว่าไม่มีการกดคีย์ใด ๆ เมื่อมีการกดค่าคีย์ใด ๆ ก็จะอ่านค่าจาก Data Out A-D เข้ามาเก็บไว้ เพื่อนำไปตรวจสอบว่าได้ค่าคีย์ที่กดมีค่าเท่าใด เพื่อนำไปใช้ต่อไป

3.4 วงจร LCD



รูปที่ 3.7 วงจรการเชื่อมต่อ LCD กับไมโครคอนโทรลเลอร์

ในรูปที่ 3.7 แสดงการต่อใช้งาน LCD ขนาด 16 ตัวอักษร 2 บรรทัด แบบ 4 บิต (รายละเอียด LCD ขนาด 16 x2 ดูในหัวข้อ 2.4) ในการเลือกการต่อร่วมกับไมโครคอนโทรลเลอร์มีให้เลือกทั้งแบบ 8 บิต และ 4 บิต ซึ่งจะมีข้อดีข้อเสียต่างกันคือ

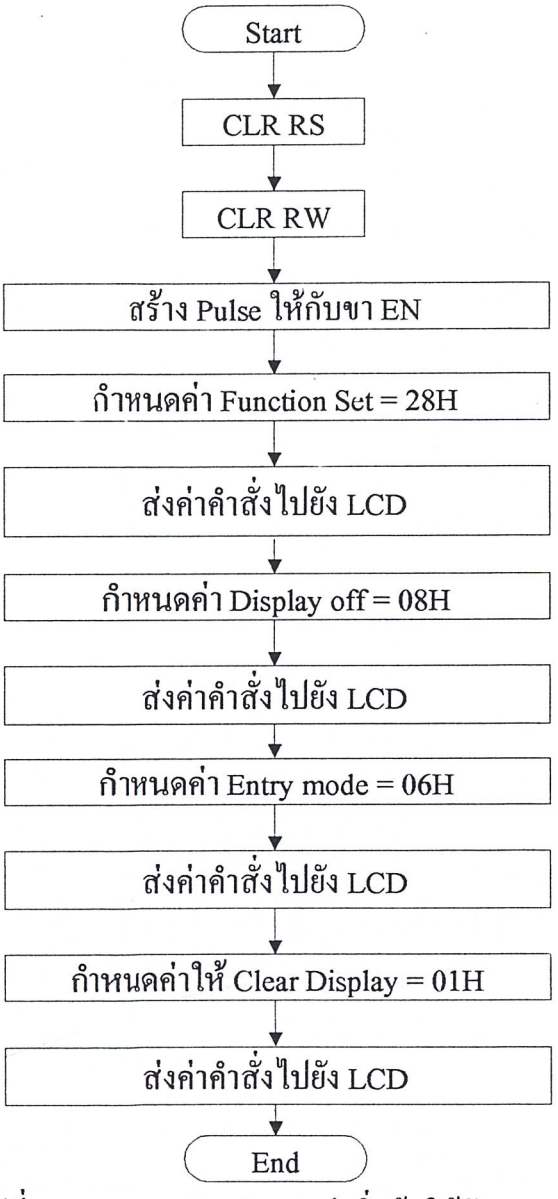
1. ความเร็วในการแสดงผลแบบ 8 บิตจะเร็วกว่าแบบ 4 บิต

2. การใช้สายสัญญาณติดต่อกับไมโครคอนโทรลเลอร์แบบ 4 บิต จะใช้น้อยกว่าแบบ 8 บิต

จากข้อดีข้อเสียดังกล่าวถึงแม้ว่าแบบ 4 บิตจะมีอัตราการแสดงผลที่ช้ากว่าแบบ 8 บิต แต่ข้อดีที่ใช้สายสัญญาณน้อยกว่าเป็นข้อได้เปรียบที่มีประโยชน์มากเพราะการเหลือพอร์ตไว้มาก ๆ ของไมโครคอนโทรลเลอร์จะช่วยให้สามารถทำการต่ออุปกรณ์อื่น ๆ เพิ่มขึ้นมาได้อีก เป็นการขยายประสิทธิภาพของระบบให้มีการใช้งานได้หลากหลายขึ้นด้วย

3.4.1 การติดต่อกับ LCD

- การกำหนดค่าเริ่มต้นให้กับ LCD



รูปที่ 3.8 Flowchart การกำหนดค่าเริ่มต้นให้กับ LCD

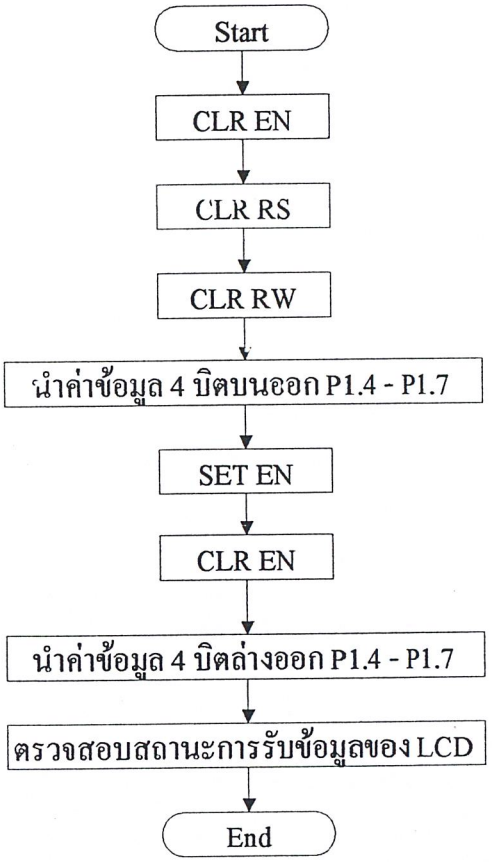
การทำงานของ Flowchart ดังรูปที่ 3.8 เราจะต้องกำหนดค่าต่าง ๆ ก่อนเพื่อให้ LCD พร้อมที่จะทำงานโดยการเขียนกำหนดดังนี้

RS = 0 เพื่อบอกว่าข้อมูลที่จะส่งเป็นคำสั่ง

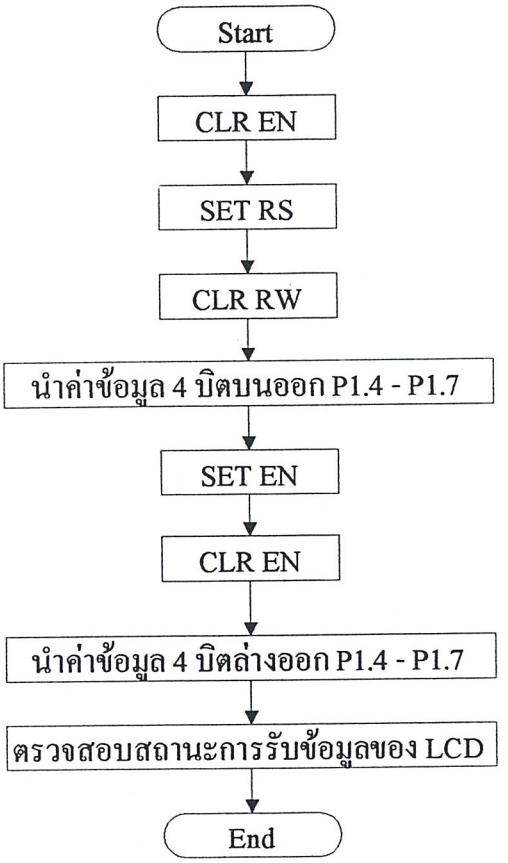
RW = 0 เพื่อบอกว่าเป็นการเขียนข้อมูล

จากนั้นสร้างสัญญาณพัลส์ให้กับขา EN เพื่อที่จะนำค่าสั่งต่าง ๆ ส่งให้กับส่วนควบคุมของ LCD ซึ่งคำสั่งควบคุมที่ส่งก็มี function set, Display off, Entry mode, Clear display

- การส่งค่าคำสั่งและการส่งค่าข้อมูลแบบ 4 บิต



รูปที่ 3.9 Flowchart การส่งค่าคำสั่งแบบ 4 บิต

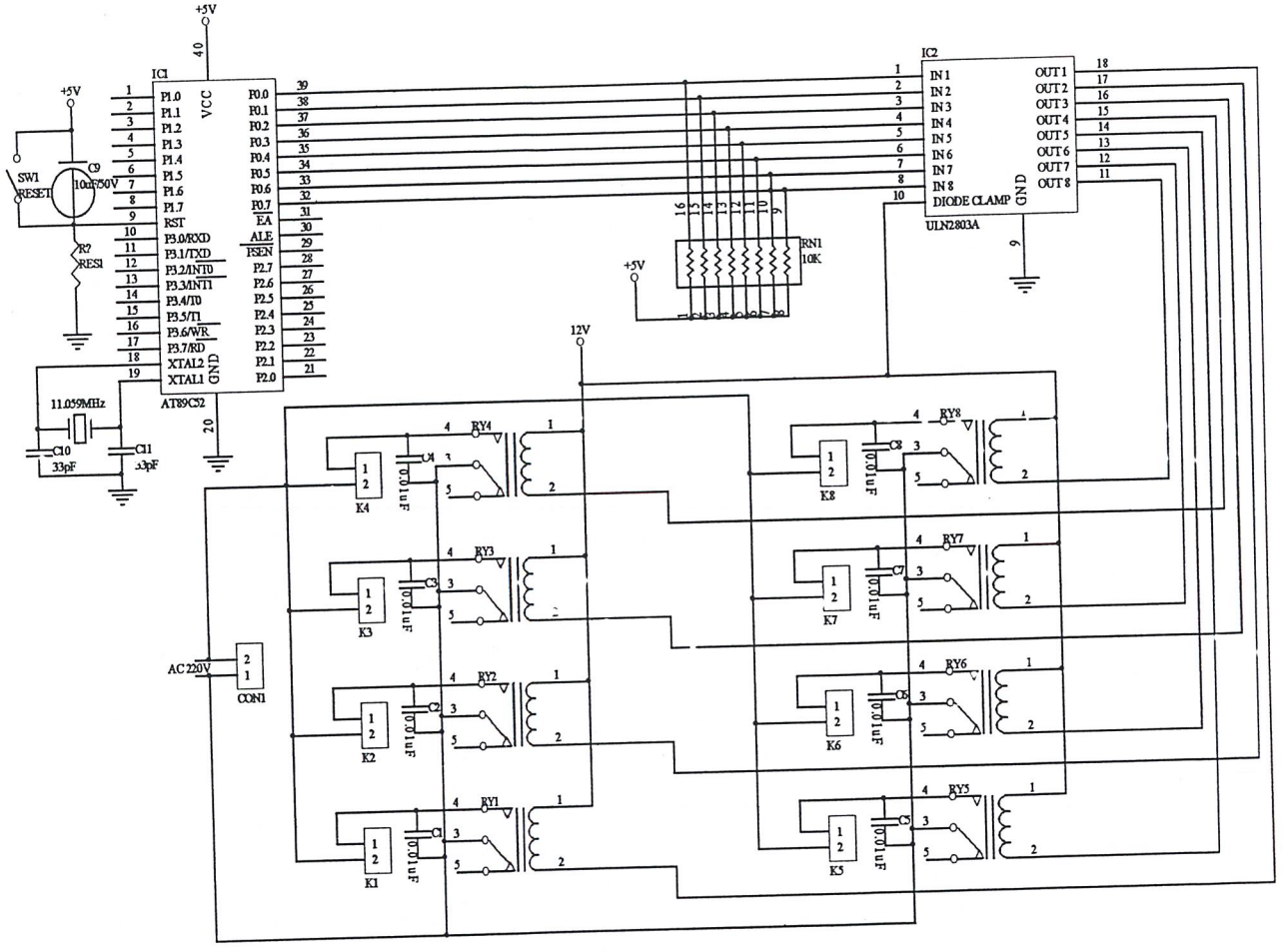


รูปที่ 3.10 Flowchart การส่งค่าข้อมูลแบบ 4 บิต

การทำงานของ Flowchart ดังรูปที่ 3.9 เป็นการส่งค่าคำสั่งแบบ 4 บิต จะต้องทำให้ขา EN = 0 และขา RS = 0 เพื่อเป็นการบอกว่าข้อมูลที่จะส่งไปให้ส่วนควบคุมของ LCD เข้าใจว่าเป็นคำสั่ง จากนั้นให้ขา RW = 0 เพื่อเขียนค่าลงในส่วนของ LCD โดยการส่งข้อมูลจะส่งแบบ 4 บิต ซึ่งจะส่ง 4 บิตแรกก่อนจากนั้นค่อยส่งข้อมูล 4 บิตล่างตามมา จากนั้นจะทำการตรวจสถานะการรับข้อมูลเมื่อได้รับข้อมูลแล้วก็จะสามารถรับข้อมูลใหม่ได้

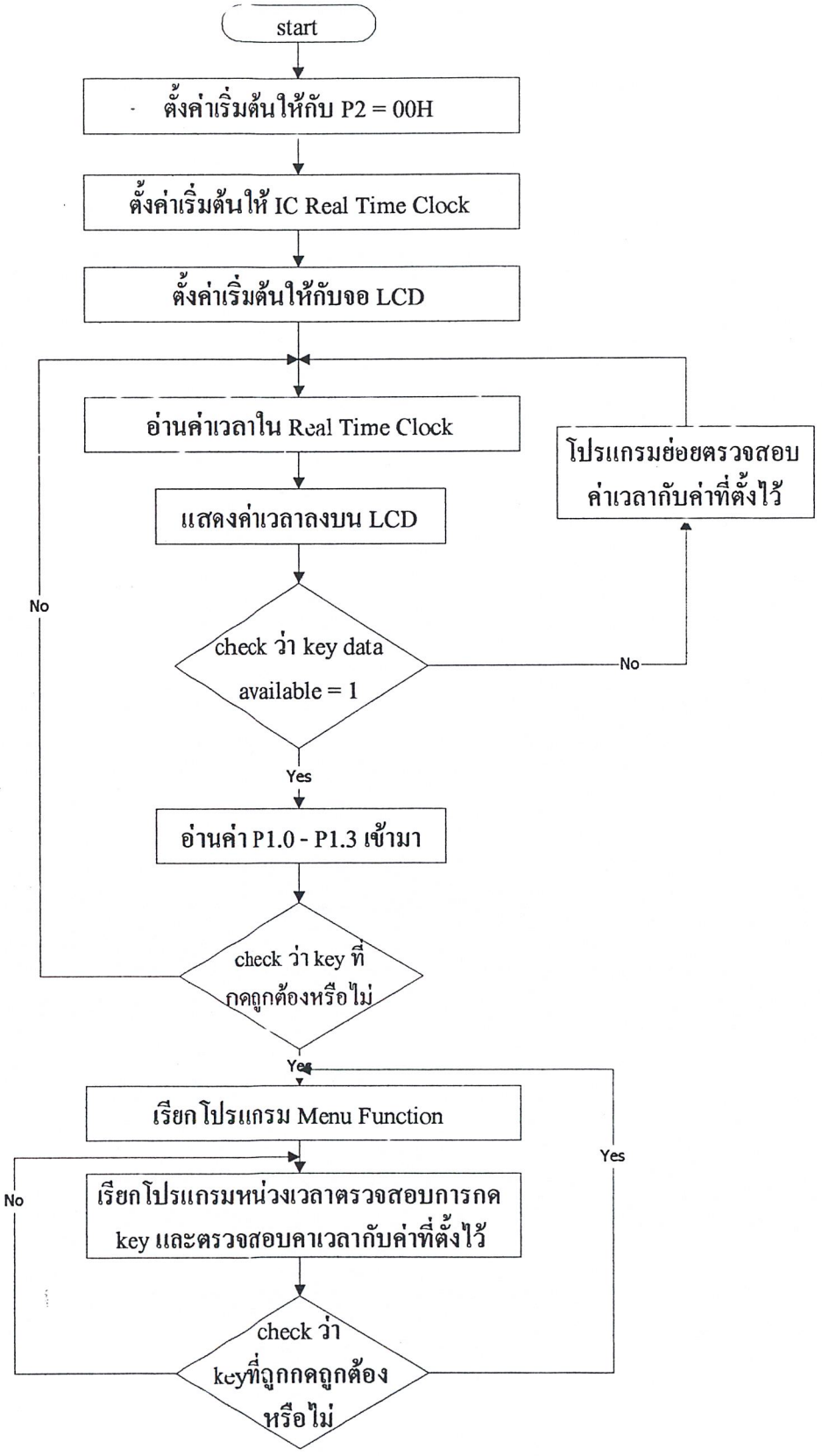
การทำงานของ Flowchart ดังรูปที่ 3.10 เป็นการส่งค่าข้อมูลแบบ 4 บิตจะมีลักษณะเหมือนกับการส่งค่าคำสั่งแต่จะต่างกันตรงที่ขา RS จะต้องทำการกำหนดให้เป็น 1 เพื่อให้ตัวควบคุมของ LCD ได้เข้าใจว่าเป็นข้อมูล

3.5 วงจร Relay Output



รูปที่ 3.11 วงจรการต่อใช้งาน Relay ร่วมกับไมโครคอนโทรลเลอร์

ในรูปที่ 3.11 แสดงการต่อใช้งานรีเลย์เป็นสวิตช์เปิด-ปิด แรงดันไฟ AC 220 V ซึ่งมีการทำงาน คือ สัญญาณที่ถูกประมวลผลจากไมโครคอนโทรลเลอร์เพื่อไปควบคุมการเปิด-ปิดรีเลย์จะออกจากพอร์ต P0.0 – P0.7 จากนั้นสัญญาณทั้ง 8 เส้นจะถูกส่งเข้าไปที่อินพุทของ ULN2803A ซึ่งเป็นไอซีไดรเวอร์แบบอินเวอร์เตอร์ที่สามารถจ่ายกระแสได้สูงสุด 500 mA และมีไดโอดป้องกันแรงดันย้อนกลับจากการยุบตัวของสนามแม่เหล็กในขดลวดรีเลย์ อยู่ภายใน และสามารถนำสัญญาณเอาท์พุททั้ง 8 เส้นไปใช้ขับรีเลย์ 12 V ทั้ง 8 ตัวได้โดยตรง ส่วนอีกขั้วของรีเลย์จะถูกต่ออยู่กับไฟบวก 12 V รอไว้อยู่แล้ว โดยรีเลย์ที่ใช้จะเป็นแบบ 1 หน้าสัมผัส ซึ่งหน้าสัมผัสแบบปกติเปิด (Normal Open : NO) จะถูกต่ออยู่กับไฟ AC 220 V เพื่อนำไปต่อกับอุปกรณ์ไฟฟ้า จะเห็นว่าที่หน้าสัมผัสของรีเลย์จะมีตัวเก็บประจุ 0.01 uF ต่อคร่อมไว้เพื่อป้องกันความถี่สูงจากหน้าสัมผัสรีเลย์เข้าไปรบกวนการทำงานของไมโครคอนโทรลเลอร์



รูปที่3.12 Flowchart ของระบบทั้งหมด

การทำงานของ Flowchart ดังรูปที่ 3.12 เริ่มแรกจะทำการตั้งค่าเริ่มต้นให้กับเอาท์พุทมีค่าเท่ากับ 00H เพื่อให้สภาวะเริ่มแรกไม่ทำงาน จากนั้นจะทำการตั้งค่าเริ่มต้นให้กับ Real Time Clock และจอ LCD และจะเข้าสู่โปรแกรมส่วนของหลักการทำงาน คือ จะทำการอ่านค่าเวลาจริงใน Real Time Clock ไปแสดงผลบนจอ LCD จากนั้นจะทำการตรวจสอบว่ามีการกดคีย์ใด ๆ หรือไม่จากขา Data Available ถ้าไม่มีการกดก็จะไปทำการเรียกโปรแกรมย่อยตรวจสอบค่าเวลาจริงกับค่าเวลาที่ตั้งไว้ โดยในส่วนของโปรแกรมย่อยการตรวจสอบนี้จะตรวจสอบทั้ง 8 เอาท์พุท โดยแต่ละเอาท์พุทจะมีทั้งหมด 8 โปรแกรมแบ่งเป็น โปรแกรมเปิด (ON) 4 โปรแกรม และโปรแกรมปิด (OFF) 4 โปรแกรม

ถ้ามีการกดคีย์ใด ๆ ก็จะทำการอ่านค่าจาก P1.0 – P1.3 เข้ามาเก็บไว้ จากนั้นก็จะนำไปตรวจสอบว่าคีย์ใดถูกกด ถ้าเป็นคีย์ที่ไม่ได้กำหนดให้ใช้งานก็จะกระโดดกลับไปอ่านค่าจาก Real Time Clock ใหม่แล้วเริ่มทำการตรวจสอบใหม่ แต่ถ้าคีย์ที่กดตรงกับค่าคีย์ที่ได้ตั้งไว้ ก็จะไปเรียกโปรแกรมย่อย Menu Function ซึ่ง Menu Function จะมีอยู่หลาย ๆ เมนูด้วยกัน เช่น Set Clock, Set Output, Set Password และดูค่าที่ทำการ Set Output

โดยในแต่ละ Menu จะมีโปรแกรมย่อยในส่วนของการหน่วงเวลาตรวจสอบการกดคีย์และตรวจสอบค่าเวลาจริงกับค่าเวลาที่ตั้งไว้ คือ ถ้าไม่มีการกดคีย์ใด ๆ ภายในเวลาประมาณ 10 วินาที โปรแกรมจะทำการกระโดดออกไปยังส่วนการอ่านค่าจาก Real Time Clock ใหม่อีกครั้ง พร้อมทั้งตรวจสอบค่าเวลาจริงกับค่าเวลาที่ตั้งไว้ เพื่อไปควบคุม Output แต่ถ้ามีการกดคีย์ ก็จะนำไปตรวจสอบว่าตรงกับค่าคีย์ที่ตั้งไว้หรือไม่ถ้าตรงก็ให้ทำงานต่อไป แต่ถ้าไม่ตรงก็ให้กระโดดไปยังโปรแกรมย่อยของการหน่วงเวลาตรวจสอบการกดคีย์ และตรวจสอบค่าเวลาจริงกับค่าเวลาที่ตั้งไว้

บทที่ 4

การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองการใช้งานเครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้ว่ามีการใช้งานอย่างไร ผู้ใช้สามารถกำหนดการใช้งานอย่างไร และจะได้ผลเป็นอย่างไร ดังต่อไปนี้

Day DD/MM/YY
*** hh:mm:ss ***

รูปที่ 4.1 แสดงหน้าจอการแสดงผลที่ LCD Display

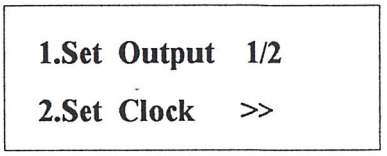
Day	=	Sun – Sat	(วัน)
DD	=	01 – 31	(วันที่)
MM	=	01 – 12	(เดือน)
YY	=	00 – 99	(ปี)
hh	=	00 – 23	(ชั่วโมง)
mm	=	00 – 59	(นาที)
ss	=	00 – 59	(วินาที)

Sat 16/03/02
*** 09:45:00 ***

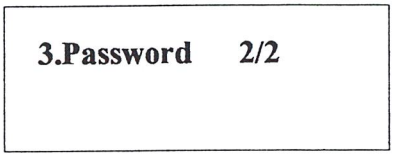
รูปที่ 4.2 ตัวอย่างการแสดงค่าที่ LCD Display

จากรูปที่ 4.2 เป็นตัวอย่างการแสดงผลที่ LCD Display โดยที่หน้าจอจะแสดงค่าวันและเวลาต่าง ๆ เป็นค่าจริงอยู่ตลอดเวลา ซึ่งจะเป็นการบอกสภาวะว่าเครื่องพร้อมใช้งานเมื่อทำการกดคีย์ต่าง ๆ ก็จะทำการเข้าไปสู่เมนูการตั้งเวลา, การตั้งรหัสผ่าน และการตั้งค่าให้เครื่องใช้ไฟฟ้าที่ต่ออยู่ที่เอาท์พุทสามารถทำงานได้ตามเวลาที่กำหนดดังจะอธิบายดังต่อไปนี้

เมื่อทำการกดคีย์ 2^{nd} 1 ครั้ง ที่ LCD Display จะแสดงผลดังรูปที่ 4.3



รูปที่ 4.3 แสดงหน้าจอเมื่อทำการกดคีย์ 2nd

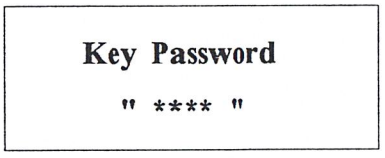


รูปที่ 4.4 หน้าที่สองเมื่อทำการกดคีย์ ↓

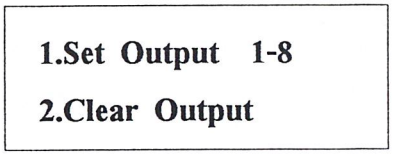
จากรูปที่ 4.3 ที่บรรทัดที่ 1 จะเห็นว่ามีตัวเลข 1/2 แสดงอยู่ซึ่งแสดงว่าในเมนูนี้มีทั้งหมด 2 หน้าโดยในรูปที่ 4.3 เป็นหน้าแรก ถ้าจะทำการดูหน้าที่สองก็ให้ทำการกดคีย์ ↓ หนึ่ง ครั้งหน้าจอก็จะทำการแสดงดังในรูปที่ 4.4 ดังนั้นในการเลื่อนหน้าที่ 1 และหน้าที่ 2 จะใช้คีย์ ↓ และคีย์ ↑ เป็นตัวเลื่อนหน้าจอ

ในเมนู Set Output เป็นเมนูการเข้าไปกำหนดค่าเวลาเปิดหรือปิดอุปกรณ์ไฟฟ้าให้ทำงานได้ตามเวลาที่ต้องการซึ่งจะมีการแสดงผลและการทำงานดังนี้

จากหน้าจอ LCD ในรูปที่ 4.3 ให้ทำการ กด 1 เพื่อทำการเข้าไปกำหนดค่าเวลาเปิดหรือปิดอุปกรณ์ไฟฟ้าโดยเมื่อทำการกดเลข 1 แล้วหน้าจอจะแสดงดังในรูปที่ 4.5



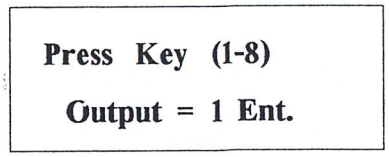
รูปที่ 4.5 แสดงการกำหนดรหัสผ่าน



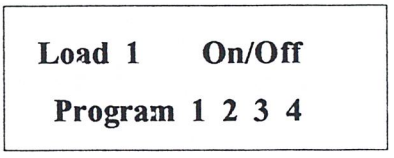
รูปที่ 4.6 แสดงเมื่อทำการกดรหัสผ่านถูกต้อง

ถ้าทำการกดรหัสผ่าน 4 ตัวถูกต้องหน้าจอก็จะแสดงดังในรูปที่ 4.6 เพื่อเข้าสู่การตั้งค่าเวลาเปิดหรือปิดอุปกรณ์ไฟฟ้า แต่ถ้ากดรหัสผ่านไม่ถูกต้องหน้าจอก็จะกลับไปแสดงดังในรูปที่ 4.3

จากรูปที่ 4.6 ถ้าทำการกดเลข 1 จะเข้าสู่การกำหนดค่าเวลาให้กับเอาท์พุท 1 – 8 หน้าจอจะแสดงดังในรูปที่ 4.7 แต่ถ้ากดเลข 2 จะเข้าสู่การเคลียร์ค่าเอาท์พุท

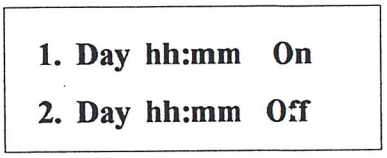


รูปที่ 4.7แสดงการกำหนดหมายเลขเอาท์พุท

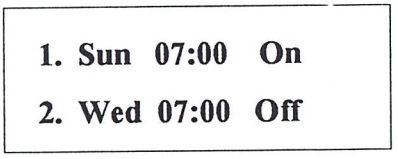


รูปที่ 4.8 แสดงการกำหนดค่าเวลาเอาท์พุท

จากรูปที่ 4.8 เป็นการเข้าไปกำหนดค่าเอาท์พุทที่ 1 จะเห็นว่ามีการให้กำหนดโปรแกรม 1, 2, 3, 4 โดยใน 1 โปรแกรมจะสามารถตั้งค่าได้ 2 คำสั่ง คือ On และ Off เช่นถ้าทำการกด 1 ก็จะเป็นการเข้าไปกำหนดโปรแกรมครั้งแรก ซึ่งหน้าจอจะแสดงดังในรูปที่ 4.9



รูปที่ 4.9แสดงการกำหนดค่าเวลาให้เอาท์พุท



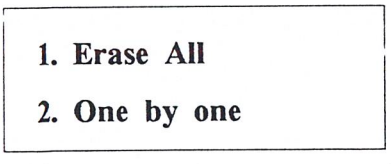
รูปที่ 4.10 แสดงเมื่อกำหนดค่าเวลาแล้ว

จากรูปที่ 4.9 เป็นการเข้าสู่การกำหนดค่าเวลาให้กับเอาท์พุท โดย Day ให้ทำการ กด 1 – 7 เพื่อตั้งค่าวัน (Sun – Sat) แต่ถ้ากด Enter จะไม่มีการตั้งค่าวัน ซึ่งถ้าไม่มีการตั้งค่าวันโปรแกรมก็จะไปกำหนดให้เอาท์พุททำงานในวันปัจจุบันที่แสดงอยู่ จากนั้นทำการตั้งค่า ชั่วโมงและนาที แล้วทำการกด Enter เพื่อเป็นการเก็บค่า แต่ถ้ากด 2nd จะเป็นการเข้าไปเคลียร์ค่าเอาท์พุทในแต่ละคำสั่ง

เมื่อทำการตั้งค่าเสร็จแล้วหน้าจอจะแสดงดังในรูปที่ 4.10 ซึ่งผลที่ได้จะสามารถอธิบายได้ดังนี้ โปรแกรมจะไปกำหนดให้ เอาท์พุทที่ 1 เปิดในวันอาทิตย์ เวลา 07.00 น. แล้วไปทำการปิดในวันพุธ เวลา 07.00 น.

ในการตั้งค่าเอาท์พุทอื่น ๆ กับโปรแกรมตัวต่อ ๆ ไป ก็จะทำงานเหมือนกันโดยการตรวจสอบจะเริ่มตรวจสอบจากคำสั่ง 1 – 8 ถ้าคำสั่งใดไม่มีการตั้งค่าก็จะกระโดดข้ามไปตรวจสอบคำสั่งต่อไป

ส่วนในการเคลียร์ค่าเอาท์พุทหน้าจอจะแสดงดังในรูปที่ 4.11

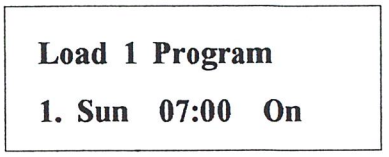


รูปที่ 4.11 แสดงหน้าจอการเคลียร์ค่าเอาท์พุทที่ได้ตั้งไว้

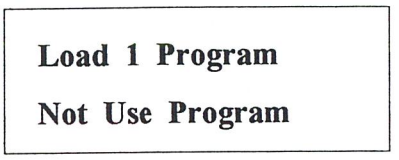
ถ้าทำการกด 1 จะทำการลบโปรแกรมทั้งหมดที่ตั้งไว้ให้กับเอาท์พุททุกตัว

ถ้าทำการกด 2 จะทำการลบโปรแกรมทั้งหมดของเอาท์พุทที่เราต้องการเลือก

ในหน้าจอหลักดังแสดงในรูปที่ 4.2 ถ้าทำการกดคีย์ Help จะเป็นการดูค่าที่เราตั้งโปรแกรมไว้ในเอาท์พุทแต่ละตัว เช่น ถ้าเราทำการตั้งค่าเอาท์พุทที่ 1 ไว้ หน้าจอจะแสดงดังในรูปที่ 4.12 (ก)



(ก)



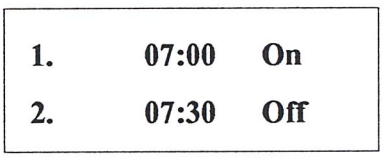
(ข)

รูปที่ 4.12 แสดงการดูค่าที่เอาท์พุท

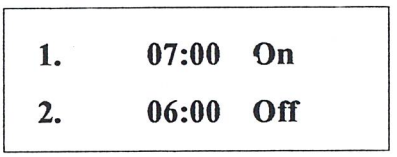
ในรูปที่4.12 (ข) เป็นการแสดงว่าที่เอาท์พุทไม่ได้มีการตั้งค่าเวลาเอาไว้ ส่วนในการดูค่าโปรแกรมอื่น ๆ ว่ามีการตั้งค่าไว้หรือไม่ก็ให้ทำการกดคีย์ ↓ และ ↑ ไปเรื่อย ๆ จนกว่าโปรแกรมที่ได้ตั้งค่าไว้จะไม่มี

ในทุก ๆ คำสั่งถ้าไม่มีการกดค่าคีย์ใด ๆ เกิน 20 วินาที โปรแกรมจะทำการตัดเข้าสู่หน้าจอหลักดังแสดงในรูปที่ 4.2 ทันทีไม่ว่าจะอยู่ที่หน้าจอใดก็ตาม

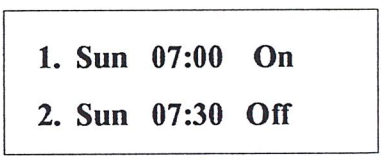
รูปแบบการกำหนดค่าเวลาให้กับเอาท์พุท



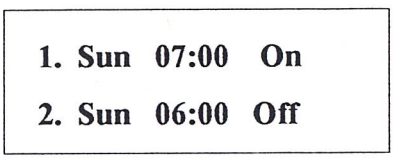
(ก)



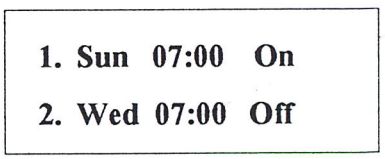
(ข)



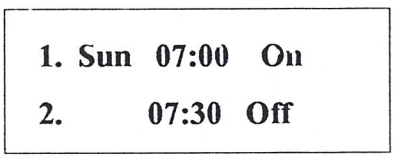
(ค)



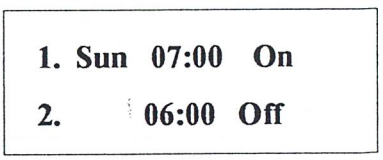
(ง)



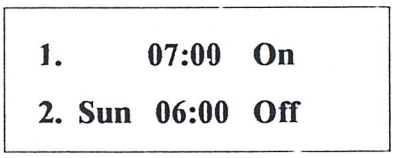
(จ)



(ฉ)



(ช)



(ซ)

รูปที่ 4.13 แสดงรูปแบบการกำหนดค่าเวลาให้กับเอาท์พุทในแบบต่าง ๆ

รูป (ก) เป็นการโปรแกรมให้เปิดและปิดในวันเดียวกันโดยอ้างเวลาปัจจุบันที่หน้าจอหลัก

รูป (ข) เป็นการโปรแกรมให้เปิดและปิดโดยให้ทำงานข้ามวันเพียง 1 วัน

รูป (ค) เป็นการโปรแกรมให้ทำงานเหมือนในรูป (ก) แต่ทำการอ้างวันด้วย

รูป (ง) เป็นการโปรแกรมให้เปิดและปิดโดยทำข้ามสัปดาห์

รูป (จ) เป็นการโปรแกรมให้เปิดและปิดตามวันและเวลาที่กำหนด

รูป (ฉ) เป็นการโปรแกรมให้ทำงานเหมือนในรูป (ค) แต่ทำการกำหนดวันเปิดด้วย

รูป (ช) เป็นการโปรแกรมให้ทำงานเหมือนในรูป (ข) แต่ทำการกำหนดวันเปิดด้วย

รูป (ซ) เป็นการโปรแกรมให้เปิดตามวันโดยอ้างเวลาปัจจุบันและกำหนดวันปิดเอง

ถ้าทำการกำหนดค่าเวลาการเปิดให้ตรงกับค่าเวลาการปิดโปรแกรมก็จะไม่สามารถทำงานได้ดังแสดงในรูปที่ 4.14

```
1. Sun  07:00  On
2. Sun  07:00  Off
```

รูปที่ 4.14 แสดงการตั้งค่าเวลาที่โปรแกรมไม่สามารถทำงานได้

บทที่ 5
บทวิจารณ์และสรุป

5.1 สรุปผลการทดลอง

จากการทดลองเครื่องควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบโปรแกรมได้จะสามารถสรุปผลการทดลองได้ว่า ในการทำงานจะต้องอาศัยฐานเวลาอ้างอิง(Real Time Clock) โดยฐานเวลาอ้างอิงนี้จะต้องมีความเที่ยงตรงอย่างมาก เพราะเป็นตัวกำหนดการทำงานของเอาท์พุทให้ทำงานได้ตรงตามค่าเวลาที่แท้จริง ซึ่งถ้าหากเราใช้ไทม์เมอร์ที่มีอยู่ในตัวของไมโครคอนโทรลเลอร์นั้นอาจจะไม่ได้ค่าเวลาที่แท้จริงเท่ากับที่ใช้ Real Time Clock แต่ในการใช้ Real Time Clock ก็มีข้อจำกัดที่ว่าจะต้องทำการใส่แบตเตอร์รี่แบ็กอัพให้ฐานเวลาสามารถผลิตสัญญาณนาฬิกาออกมาได้ตรงตามความเป็นจริงมากที่สุด ส่วนในการเขียนโปรแกรมให้ทำงานนั้นก็ใช้ความรู้ในการประยุกต์ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มาเป็นส่วนประมวลผลให้โปรแกรมสามารถไปควบคุมวงจรที่นำมาต่อร่วมกับไมโครคอนโทรลเลอร์ได้อย่างมีประสิทธิภาพมากที่สุด

ส่วนในการตั้งโปรแกรมให้เอาท์พุททำงานนั้นสามารถทำให้เอาท์พุทเปิดหรือปิดได้สูงสุดเป็นในรอบสัปดาห์เท่านั้น ถ้าต้องการให้สามารถตั้งค่าการเปิดหรือปิดได้มากกว่านี้จะต้องทำการเพิ่มหน่วยความจำให้กับระบบมากขึ้นเพื่อเป็นการเพิ่มประสิทธิภาพให้มีฟังก์ชั่นการทำงานที่หลากหลายมากยิ่งขึ้น ซึ่งถ้าทำการเพิ่มหน่วยความจำเข้าไปแล้วก็จะสามารถทำการเพิ่มจำนวนเอาท์พุทให้มากกว่า 8 ได้ซึ่งอาจจะเป็น 16 หรือมากกว่านี้ก็ได้

ส่วนในวงจรภาครีเลย์เอาท์พุทถ้าต้องการให้มีการใช้งานกับเครื่องใช้ไฟฟ้าที่มีกำลังไฟเพิ่มขึ้น เช่น เครื่องปรับอากาศ จะต้องทำการเปลี่ยนรีเลย์ใหม่ให้หน้าสัมผัสสามารถทนได้มากกว่านี้ โดยในโครงงานนี้จะใช้รีเลย์ที่มีหน้าสัมผัสทนได้ 250 VAC 5A ซึ่งเป็นค่าแรงดันของเครื่องใช้ไฟฟ้าปกติทั่ว ๆ ไป

5.2 ปัญหาที่เกิดขึ้นจากการทดลอง

จากการทดลองในส่วนของโปรแกรมนั้น จากการศึกษาพบว่าหน่วยความจำภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่มีอยู่ 256 ไบท์ นั้นเราสามารถเขียนให้สามารถตั้งค่าโปรแกรมได้ทั้งหมด 64 โปรแกรม โดยในแต่ละโปรแกรมจะตั้งค่าได้เฉพาะวัน (อาทิตย์ – เสาร์), ชั่วโมง และนาที เท่านั้น หากเราต้องการที่จะตั้งค่าโปรแกรมให้ได้มากกว่านี้ และการตั้งเวลาสามารถอ้างอิงกับวันที่ เดือน ปี ด้วย ก็จะต้องทำการเพิ่มหน่วยความจำภายนอกเข้ามาเพื่อเป็นที่เก็บในส่วนนี้ด้วย

จากการเขียนโปรแกรมได้พบว่า เมื่อเราใช้หน่วยความจำภายในของไมโครคอนโทรลเลอร์ เราต้องตรวจสอบดูว่ามีการทับกันของข้อมูลของสแตคหรือไม่ ซึ่งสแตคนี้จะทำการเก็บค่าเมื่อมีการกระโดดไม่เรียกโปรแกรมย่อยและการเก็บค่าข้อมูลบางส่วนที่เราต้องการใช้ ถ้าหากในโปรแกรมย่อยเรามีการเรียกคำสั่ง push และ call หลาย ๆ ครั้ง ก็จะทำให้พื้นที่ที่สแตคชี้เก็บค่าข้อมูลไว้ไปทับกับหน่วยความจำตรงส่วนที่เราใช้งานทำให้โปรแกรมการทำงานเกิดการผิดพลาดได้ เพราะฉะนั้นในส่วนนี้จึงเป็นข้อจำกัดและปัญหาในการเขียนโปรแกรมนั่นเอง

ภาคผนวก

DALLAS SEMICONDUCTOR

DS1307/DS1308
64 X 8 Serial Real Time Clock

FEATURES

- Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation valid up to 2100
- 56 byte nonvolatile RAM for data storage
- 2–wire serial interface
- Programmable squarewave output signal
- Automatic power fail detect and switch circuitry
- Consumes less than 500 nA in battery backup mode with oscillator running
- Optional industrial temperature range –40°C to +85°C (IND) available for DS1307 and DS1308
- DS1307 available in 8–pin DIP or SOIC
- DS1308 available in 36–pin SMD BGA (Ball Grid Array)
- DS1308 accuracy is better than ±2 minute/month at 25°C

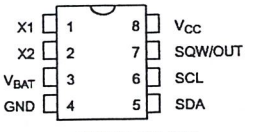
ORDERING INFORMATION

DS1307	Serial Timekeeping Chip; 8–pin DIP
DS1307Z	Serial Timekeeping Chip; 8–pin SOIC (150 mil)
DS1307N	8–pin DIP (IND)
DS1307ZN	8–pin SOIC (IND)
DS1308	36–pin BGA
DS1308N	36–pin BGA(IND)

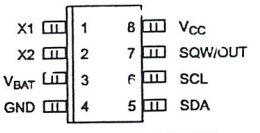
DESCRIPTION

The DS1307 Serial Real Time Clock is a low power, full BCD clock/calendar plus 56 bytes of nonvolatile SRAM. Address and data are transferred serially via a 2–wire bi–directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24–hour or 12–hour format with AM/PM indicator. The DS1307 has a built–in power sense circuit which detects power failures and automatically switches to the battery supply.

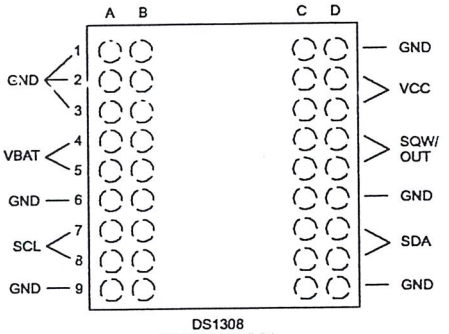
PIN ASSIGNMENT



DS1307 8–PIN DIP (300 MIL)



DS1307Z 8–PIN SOIC (150 MIL)



DS1308
36–PIN SMD BGA
(TOP VIEW)

PIN DESCRIPTION DS1307/DS1308

V_{CC}	– Primary Power Supply
X1, X2	– 32.768 kHz Crystal Connection
V_{BAT}	– +3 Volt Battery Input
GND	– Ground
SDA	– Serial Data
SCL	– Serial Clock
SQW/OUT	– Square wave/Output Driver

DS1308 PIN IDENTIFIER

V_{CC}	– C2, C3, D2, D3
V_{BAT}	– A4, A5, B4, B5
SDA	– C7, C8, D7, D8
SCL	– A7, A8, B7, B8
SQW/OUT	– C4, C5, D4, D5
GND	– All Remaining Balls

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.5V to +7.0V
Operating Temperature	0°C to 70°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds (See NOTE 12)

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

The Dallas Semiconductor DS1307/DS1308 is built to the highest quality standards and manufactured for long term reliability. All Dallas Semiconductor devices are made using the same quality materials and manufacturing methods. However, standard versions of the DS1307/DS1308 are not exposed to environmental stresses, such as burn–in, that some industrial applications require. Products which have successfully passed through this series of environmental stresses are marked IND or N, denoting their extended operating temperature and reliability rating. For specific reliability information on this product, please contact the factory at (972) 371–4448.

RECOMMENDED DC OPERATING CONDITIONS

(0°C to 70°C)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}	4.5	5.0	5.5	V	1
Logic 1	V_{IH}	2.2		V_{CC}+0.3	V	1
Logic 0	V_{IL}	–0.3		+0.8	V	1
V_{BAT} Battery Voltage	V_{BAT}	2.0		3.5	V	1

DC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC}=4.5V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Leakage	I_{LI}			1	µA	10
I/O Leakage	I_{LO}			1	µA	11
Logic 0 Output	V_{OL}			0.4	V	2
Active Supply Current	I_{CCA}			1.5	mA	9
Standby Current	I_{CCS}			200	µA	3
Battery Current (OSC ON); SQW/OUT OFF	I_{BAT1}		300	500	nA	4
Battery Current (OSC ON); SQW/OUT ON (32 KHz)	I_{BAT2}		480	800	nA	4

AC ELECTRICAL CHARACTERISTICS

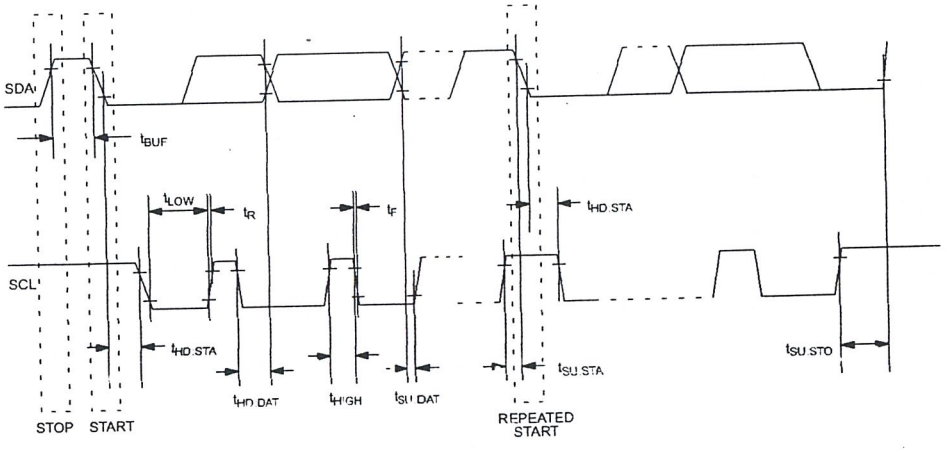
(0°C to 70°C; V_{CC}=4.5V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
SCL Clock Frequency	f_{SCL}	0		100	kHz	
Bus Free Time Between a STOP and START Condition	t_{BUF}	4.7			µs	
Hold Time (Repeated) START Condition	$t_{HD:STA}$	4.0			µs	5
LOW Period of SCL Clock	t_{LOW}	4.7			µs	
HIGH Period of SCL Clock	t_{HIGH}	4.0			µs	
Set–up Time for a Repeated START Condition	$t_{SU:STA}$	4.7			µs	
Data Hold Time	$t_{HD:DAT}$	0			µs	6, 7
Data Set–up Time	$t_{SU:DAT}$	250			ns	
Rise Time of Both SDA and SCL Signals	t_R			1000	ns	
Fall Time of Both SDA and SCL Signals	t_F			300	ns	
Set–up Time for STOP Condition	$t_{SU:STO}$	4.7			µs	
Capacitive Load for each Bus Line	C_B			400	pF	8
I/O Capacitance	$C_{I/O}$		10		pF	
Crystal Specified Load Capacitance			12.5		pF	

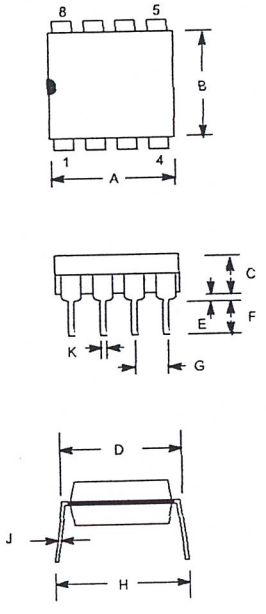
NOTES:

1. All voltages are referenced to ground.
2. Logic zero voltages are specified at a sink current of 5 mA at V_{CC}=4.5V, V_{OL}=GND for capacitive loads.
3. I_{CCS} specified with V_{CC}=5.0V and SDA, SCL=5.0V.
4. V_{CC}=0V, V_{BAT}=3V.
5. After this period, the first clock pulse is generated.
6. A device must internally provide a hold time of at least 300 ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.
7. The maximum $t_{HD:DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.
8. C_B – total capacitance of one bus line in pF.
9. I_{CCA} – SCL clocking at max frequency = 100 KHz.
10. SCL only.
11. SDA and SQW/OUT
12. The DS1308 is designed to be subjected to no more than two (2) passes through a solder reflow process to limit premature crystal aging effects and maintain a reasonable accuracy of ±2 minutes/month at 25 degrees C (Worst case).

TIMING DIAGRAM Figure 8

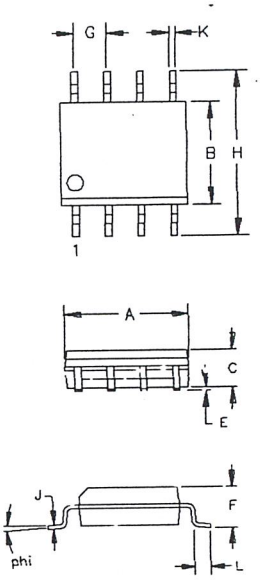


DS1307 64 X 8 SERIAL REAL TIME CLOCK 8–PIN DIP MECHANICAL DIMENSIONS



PKC	8–PIN	
DIM	MIN	MAX
A IN.	0.360	0.400
MM	9.14	10.16
B IN.	0.240	0.260
MM	6.10	6.60
C IN.	0.120	0.140
MM	3.05	3.56
D IN.	0.300	0.325
MM	7.62	8.26
E IN.	0.015	0.040
MM	0.38	1.02
F IN.	0.120	0.140
MM	3.04	3.56
G IN.	0.090	0.110
MM	2.29	2.79
H IN.	0.320	0.370
MM	8.13	9.40
J IN.	0.008	0.012
MM	0.20	0.30
K IN.	0.015	0.021
MM	0.38	0.53

DS1307Z 64 X 8 SERIAL REAL TIME CLOCK 8–PIN SOIC (150 MIL) MECHANICAL DIMENSIONS

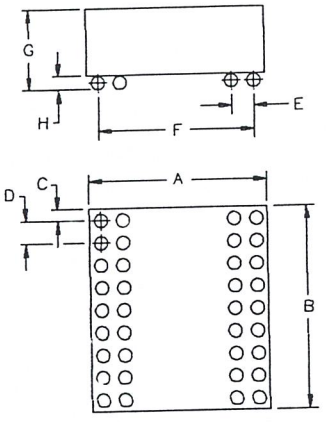


PKG	8–PIN (150 MIL)	
DIM	MIN	MAX
A IN.	0.188	0.196
MM	4.78	4.98
B IN.	0.150	0.158
MM	3.81	4.01
C IN.	0.048	0.062
MM	1.22	1.57
E IN.	0.004	0.010
MM	0.10	0.25
F IN.	0.053	0.069
MM	1.35	1.75
G IN.	0.050 BSC	
MM	1.27 BSC	
H IN.	0.230	0.244
MM	5.84	6.20
J IN.	0.007	0.011
MM	0.18	0.28
K IN.	0.012	0.020
MM	0.30	0.51
L IN.	0.016	0.050
MM	0.41	1.27
phi	0°	8°

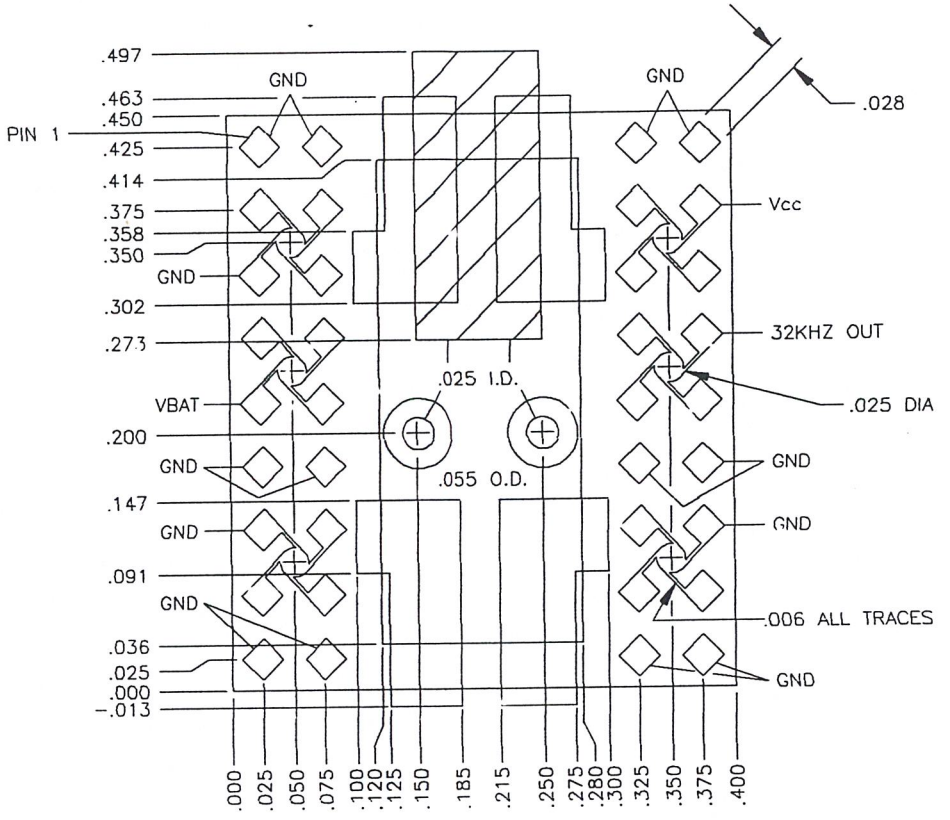
56–G2008–001

DS1308 MECHANICAL DIMENSIONS

PKG	36–PIN BALL GRID	
DIM	MIN	MAX
A IN. MM	0.395	0.405
B IN. MM	0.445	0.455
C IN. MM	0.022	0.028
D IN. MM	0.047	0.053
E IN. MM	0.047	0.053
F IN. MM	0.347	0.353
G IN. MM	0.170	0.190
H IN. MM	0.025	0.030



DS1308 RECOMMENDED LAYOUT LAND PATTERN



National Semiconductor

July 1993

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

General Description

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

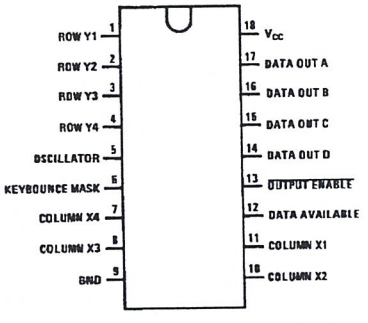
An internal register remembers the last key pressed even after the key is released. The TRI-STATE® outputs provide for easy expansion and bus operation and are LPTTL compatible.

Features

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

Connection Diagrams

Pin Assignment for Dual-In-Line Package

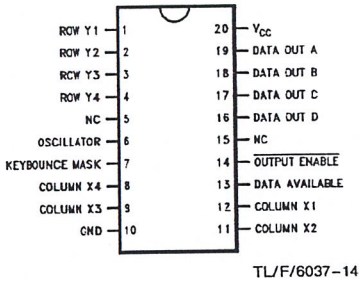


TL/F/6037–1

Top View

Order Number MM54C922 or MM74C922

Pin Assignment for SOIC

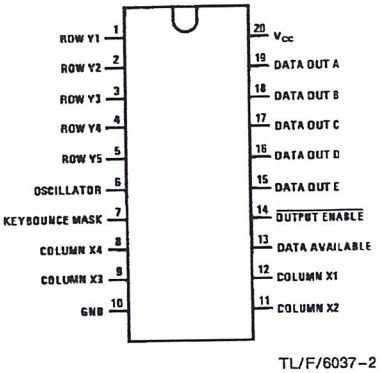


TL/F/6037–14

Top View

Order Number MM74C922

Pin Assignment for DIP and SOIC Package



TL/F/6037–2

Top View

Order Number MM54C923 or MM74C923

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

RRD-B30M105/Printed in U. S. A.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin	$V_{CC} - 0.3V$ to $V_{CC} + 0.3V$
Operating Temperature Range	
MM54C922, MM54C923	−55°C to +125°C
MM74C922, MM74C923	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Power Dissipation (P_D)	
Dual-In-Line	700 mW
Small Outline	500 mW
Operating V_{CC} Range	3V to 15V
V_{CC}	18V
Lead Temperature (Soldering, 10 seconds)	260°C

DC Electrical Characteristics Min/Max limits apply across temperature range unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Units
CMOS TO CMOS						
V_{T+}	Positive-Going Threshold Voltage at Osc and KBM Inputs	$V_{CC} = 5V$, $I_{IN} \geq 0.7$ mA $V_{CC} = 10V$, $I_{IN} \geq 1.4$ mA $V_{CC} = 15V$, $I_{IN} \geq 2.1$ mA	3.0 6.0 9.0	3.6 6.8 10	4.3 8.6 12.9	V V V
V_{T-}	Negative-Going Threshold Voltage at Osc and KBM Inputs	$V_{CC} = 5V$, $I_{IN} \geq 0.7$ mA $V_{CC} = 10V$, $I_{IN} \geq 1.4$ mA $V_{CC} = 15V$, $I_{IN} \geq 2.1$ mA	0.7 1.4 2.1	1.4 3.2 5	2.0 4.0 6.0	V V V
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	$V_{CC} = 5V$ $V_{CC} = 10V$ $V_{CC} = 15V$	3.5 8.0 12.5	4.5 9 13.5		V V V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	$V_{CC} = 5V$ $V_{CC} = 10V$ $V_{CC} = 15V$		0.5 1 1.5	1.5 2 2.5	V V V
I_{rp}	Row Pull-Up Current at Y1, Y2, Y3, Y4 and Y5 Inputs	$V_{CC} = 5V$, $V_{IN} = 0.1\ V_{CC}$ $V_{CC} = 10V$ $V_{CC} = 15V$		−2 −10 −22	−5 −20 −45	μA μA μA
$V_{OUT(1)}$	Logical "1" Output Voltage	$V_{CC} = 5V$, $I_O = -10\ \mu A$ $V_{CC} = 10V$, $I_O = -10\ \mu A$ $V_{CC} = 15V$, $I_O = -10\ \mu A$	4.5 9 13.5			V V V
$V_{OUT(0)}$	Logical "0" Output Voltage	$V_{CC} = 5V$, $I_O = 10\ \mu A$ $V_{CC} = 10V$, $I_O = 10\ \mu A$ $V_{CC} = 15V$, $I_O = 10\ \mu A$			0.5 1 1.5	V V V
R_{on}	Column "ON" Resistance at X1, X2, X3 and X4 Outputs	$V_{CC} = 5V$, $V_O = 0.5V$ $V_{CC} = 10V$, $V_O = 1V$ $V_{CC} = 15V$, $V_O = 1.5V$		500 300 200	1400 700 500	Ω Ω Ω
I_{CC}	Supply Current Osc at 0V, (one Y low)	$V_{CC} = 5V$ $V_{CC} = 10V$ $V_{CC} = 15V$		0.55 1.1 1.7	1.1 1.9 2.6	mA mA mA
$I_{IN(1)}$	Logical "1" Input Current at $\overline{\text{Output Enable}}$	$V_{CC} = 15V$, $V_{IN} = 15V$		0.005	1.0	μA
$I_{IN(0)}$	Logical "0" Input Current at $\overline{\text{Output Enable}}$	$V_{CC} = 15V$, $V_{IN} = 0V$	−1.0	−0.005		μA
CMOS/LPTTL INTERFACE						
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	54C, $V_{CC} = 4.5V$ 74C, $V_{CC} = 4.75V$	$V_{CC} - 1.5$ $V_{CC} - 1.5$			V V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	54C, $V_{CC} = 4.5V$ 74C, $V_{CC} = 4.75V$			0.8 0.8	V V
$V_{OUT(1)}$	Logical "1" Output Voltage	54C, $V_{CC} = 4.5V$ $I_O = -360\ \mu A$ 74C, $V_{CC} = 4.75V$ $I_O = -360\ \mu A$	2.4 2.4			V V
$V_{OUT(0)}$	Logical "0" Output Voltage	54C, $V_{CC} = 4.5V$ $I_O = -360\ \mu A$ 74C, $V_{CC} = 4.75V$ $I_O = -360\ \mu A$			0.4 0.4	V V

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

DC Electrical Characteristics

Min/Max limits apply across temperature range unless otherwise specified (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
OUTPUT DRIVE (See 54C/74C Family Characteristics Data Sheet) (Short Circuit Current)						
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 5V, V_{OUT} = 0V, T_A = 25°C	−1.75	−3.3		mA
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 10V, V_{OUT} = 0V, T_A = 25°C	−8	−15		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 5V, V_{OUT} = V_{CC}, T_A = 25°C	1.75	3.6		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 10V, V_{OUT} = V_{CC}, T_A = 25°C	8	16		mA

AC Electrical Characteristics*

T_A = 25°C, C_L = 50 pF, unless otherwise noted

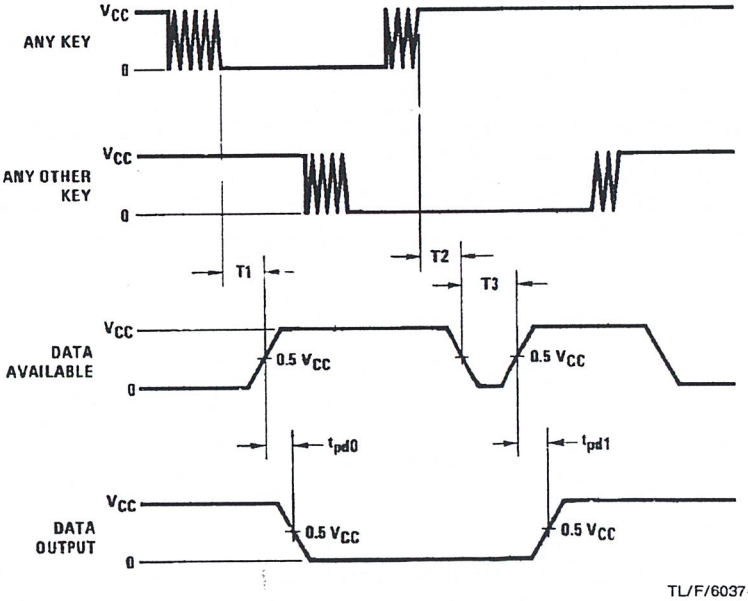
Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd0}, t_{pd1}	Propagation Delay Time to Logical "0" or Logical "1" from D.A.	C_L = 50 pF *(Figure 1)* V_{CC} = 5V V_{CC} = 10V V_{CC} = 15V		 60 35 25	 150 80 60	 ns ns ns
t_{0H}, t_{1H}	Propagation Delay Time from Logical "0" or Logical "1" into High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

*AC Parameters are guaranteed by DC correlated testing.

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

Note 2: Capacitance is guaranteed by periodic testing.

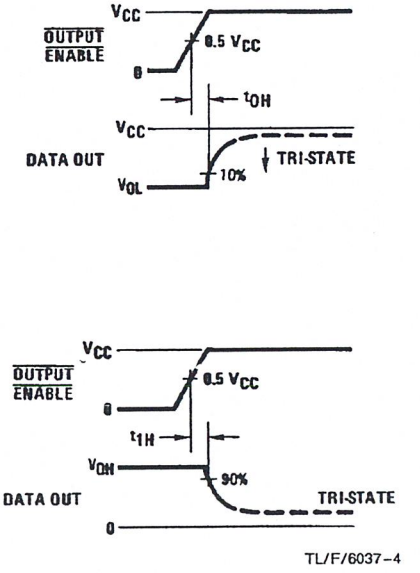
Switching Time Waveforms



TL/F/6037–3

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

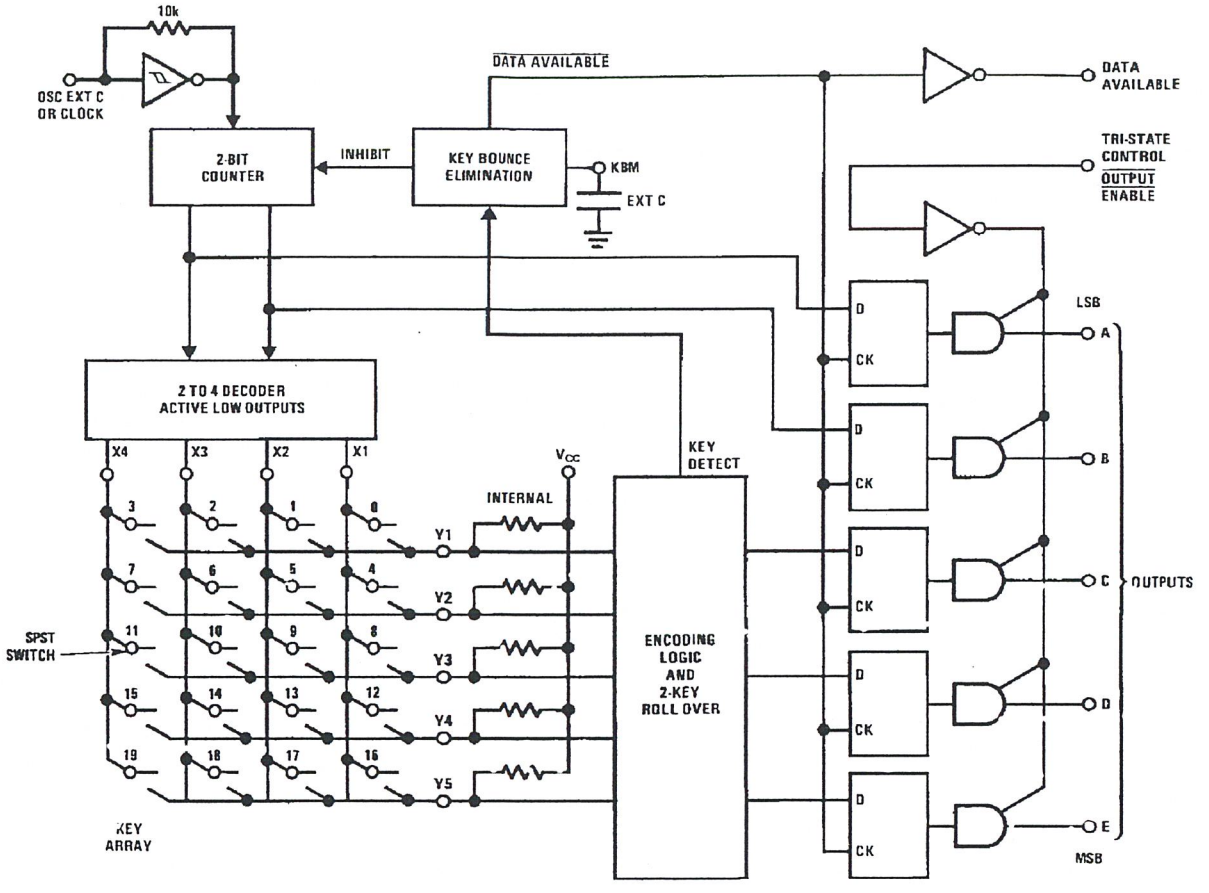
FIGURE 1



TL/F/6037–4

FIGURE 2

Block Diagram



TL/F/6037–5

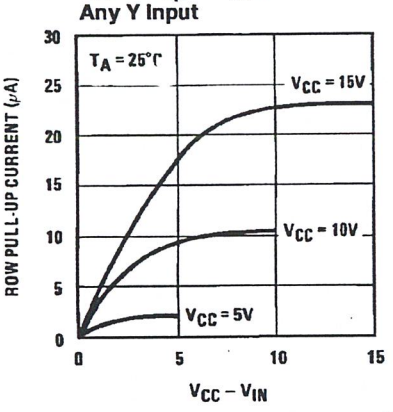
Truth Table

Switch Position	0 Y1,X1	1 Y1,X2	2 Y1,X3	3 Y1,X4	4 Y2,X1	5 Y2,X2	6 Y2,X3	7 Y2,X4	8 Y3,X1	9 Y3,X2	10 Y3,X3	11 Y3,X4	12 Y4,X1	13 Y4,X2	14 Y4,X3	15 Y4,X4	16 Y5*,X1	17 Y5*,X2	18 Y5*,X3	19 Y5*,X4
DATA OUT A	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
DATA OUT B	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
DATA OUT C	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
DATA OUT D	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
DATA OUT E*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

*Omit for MM54C922/MM74C922

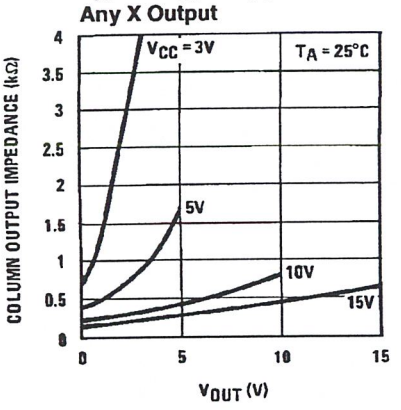
Typical Performance Characteristics

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



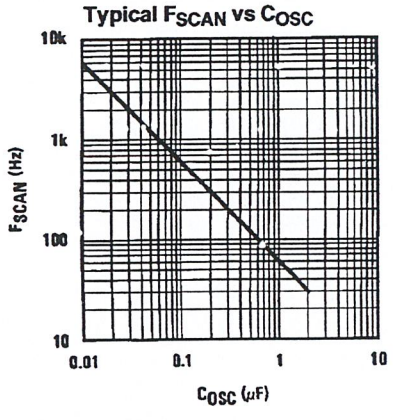
TL/F/6037–6

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



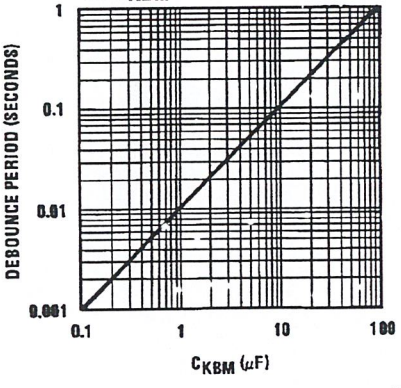
TL/F/6037–7

Typical F_{SCAN} vs C_{OSC}



TL/F/6037–8

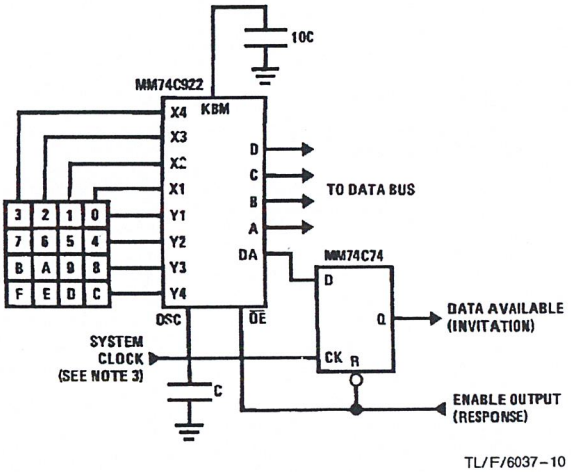
Typical Debounce Period vs C_{KBM}



TL/F/6037–9

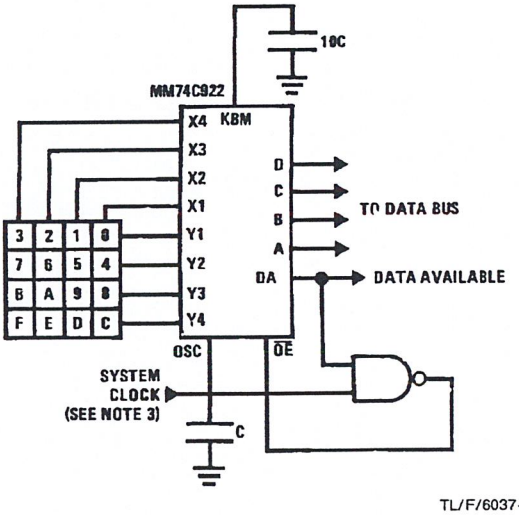
Typical Applications

Synchronous Handshake (MM74C922)



TL/F/6037–10

Synchronous Data Entry Onto Bus (MM74C922)



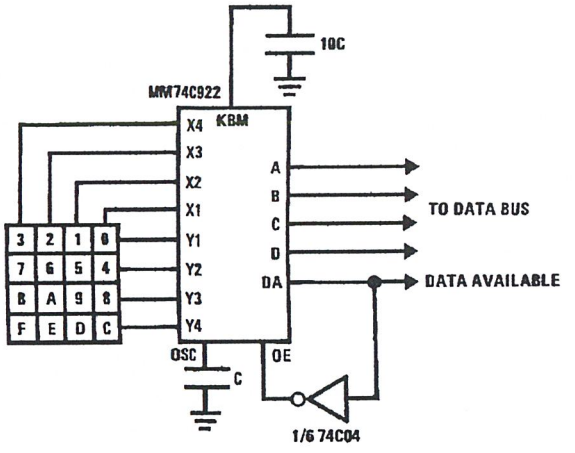
TL/F/6037–11

Outputs are enabled when valid entry is made and go into TRI-STATE when key is released.

Note 3: The keyboard may be synchronously scanned by omitting the capacitor at osc. and driving osc. directly if the system clock rate is lower than 10 kHz.

Typical Applications (Continued)

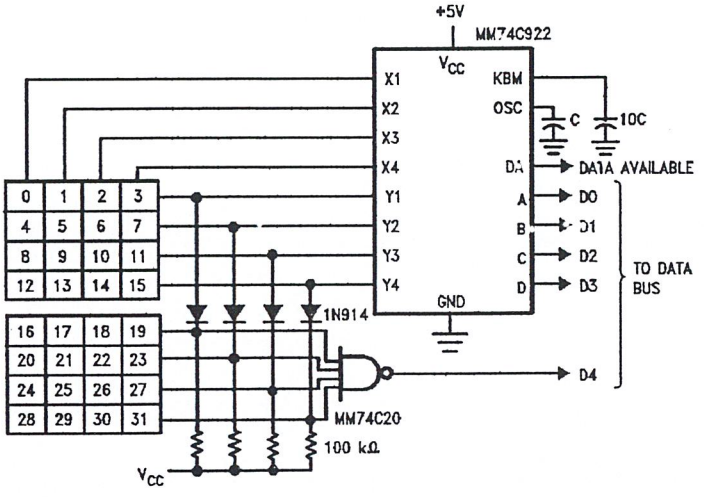
Asynchronous Data Entry Onto Bus (MM74C922)



TL/F/6037–12

Outputs are in TRI-STATE until key is pressed, then data is placed on bus. When key is released, outputs return to TRI-STATE.

Expansion to 32 Key Encoder (MM74C922)



TL/F/6037–13

Theory of Operation

The MM74C922/MM74C923 Keyboard Encoders implement all the logic necessary to interface a 16 or 20 SPST key switch matrix to a digital system. The encoder will convert a key switch closer to a 4(MM74C922) or 5(MM74C923) bit nibble. The designer can control both the keyboard scan rate and the key debounce period by altering the oscillator capacitor, C_{OSE}, and the key bounce mask capacitor, C_{MSK}. Thus, the MM74C922/MM74C923's performance can be optimized for many keyboards.

The keyboard encoders connect to a switch matrix that is 4 rows by 4 columns (MM74C922) or 5 rows by 4 columns (MM74C923). When no keys are depressed, the row inputs are pulled high by internal pull-ups and the column outputs sequentially output a logic "0". These outputs are open drain and are therefore low for 25% of the time and otherwise off. The column scan rate is controlled by the oscillator input, which consists of a Schmitt trigger oscillator, a 2-bit counter, and a 2–4-bit decoder.

When a key is depressed, key 0, for example, nothing will happen when the X1 input is off, since Y1 will remain high. When the X1 column is scanned, X1 goes low and Y1 will go low. This disables the counter and keeps X1 low. Y1 going low also initiates the key bounce circuit timing and locks out the other Y inputs. The key code to be output is a combination of the frozen counter value and the decoded Y inputs. Once the key bounce circuit times out, the data is latched, and the Data Available (DAV) output goes high.

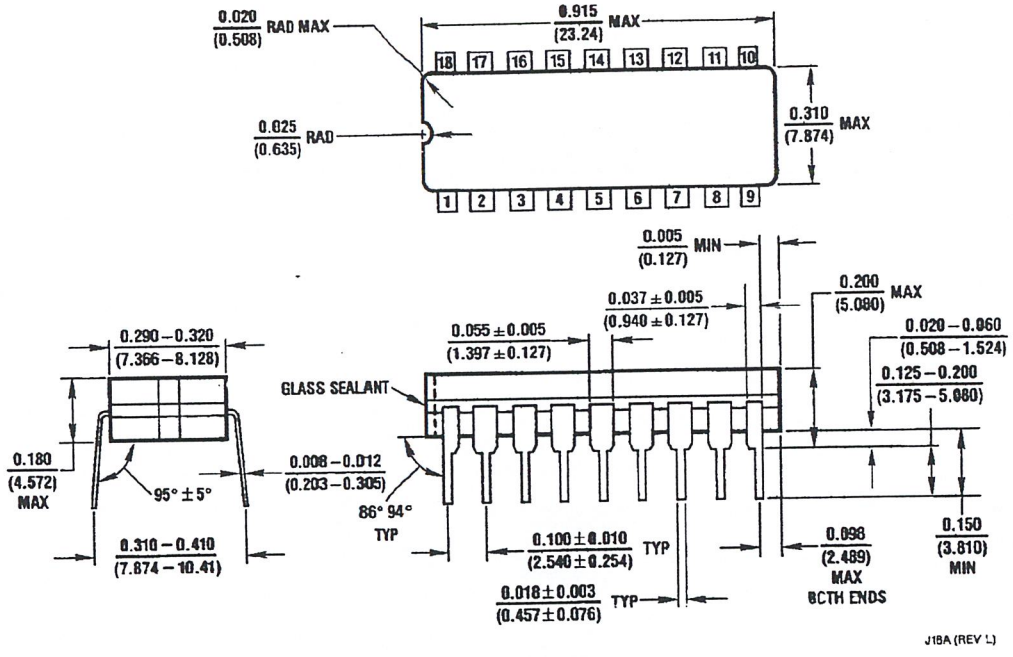
If, during the key closure the switch bounces, Y1 input will go high again, restarting the scan and resetting the key bounce circuitry. The key may bounce several times, but as soon as the switch stays low for a debounce period, the closure is assumed valid and the data is latched.

A key may also bounce when it is released. To ensure that the encoder does not recognize this bounce as another key closure, the debounce circuit must time out before another closure is recognized.

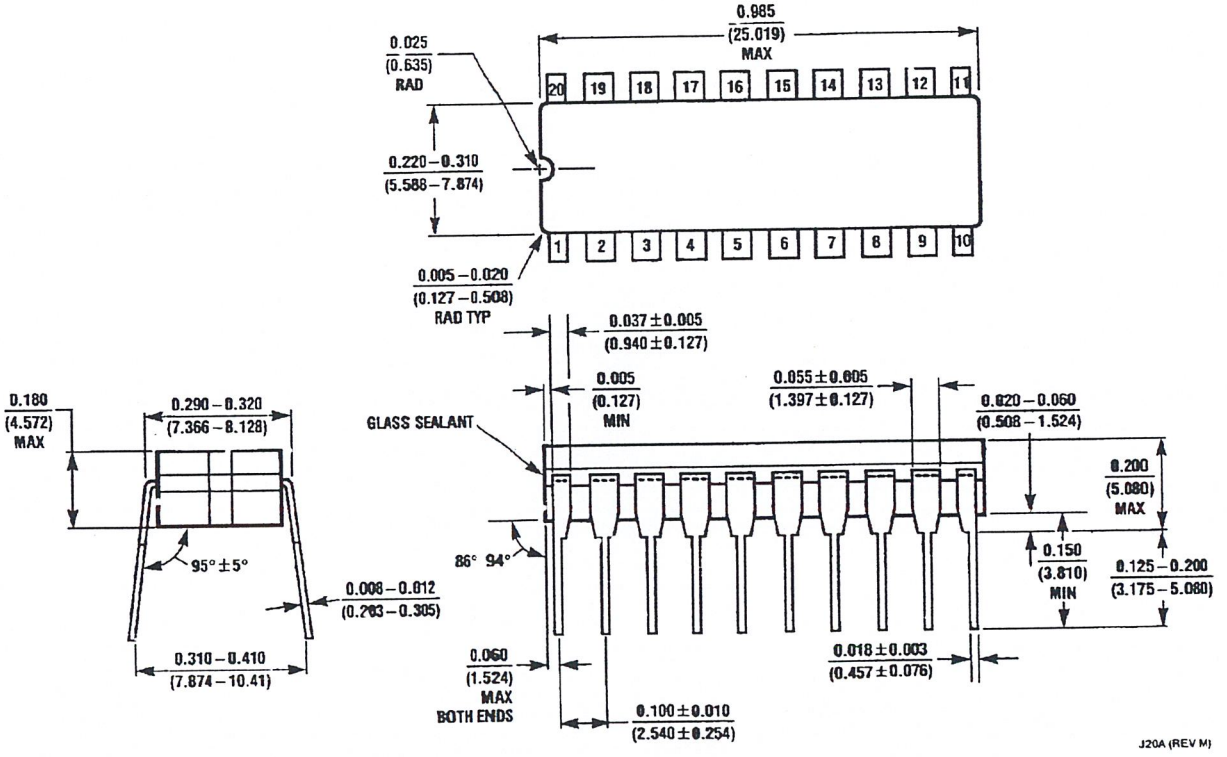
The two-key roll-over feature can be illustrated by assuming a key is depressed, and then a second key is depressed. Since all scanning has stopped, and all other Y inputs are disabled, the second key is not recognized until the first key is lifted and the key bounce circuitry has reset.

The output latches feed TRI-STATE, which is enabled when the Output Enable ($\overline{OE}$) input is taken low.

Physical Dimensions inches (millimeters)

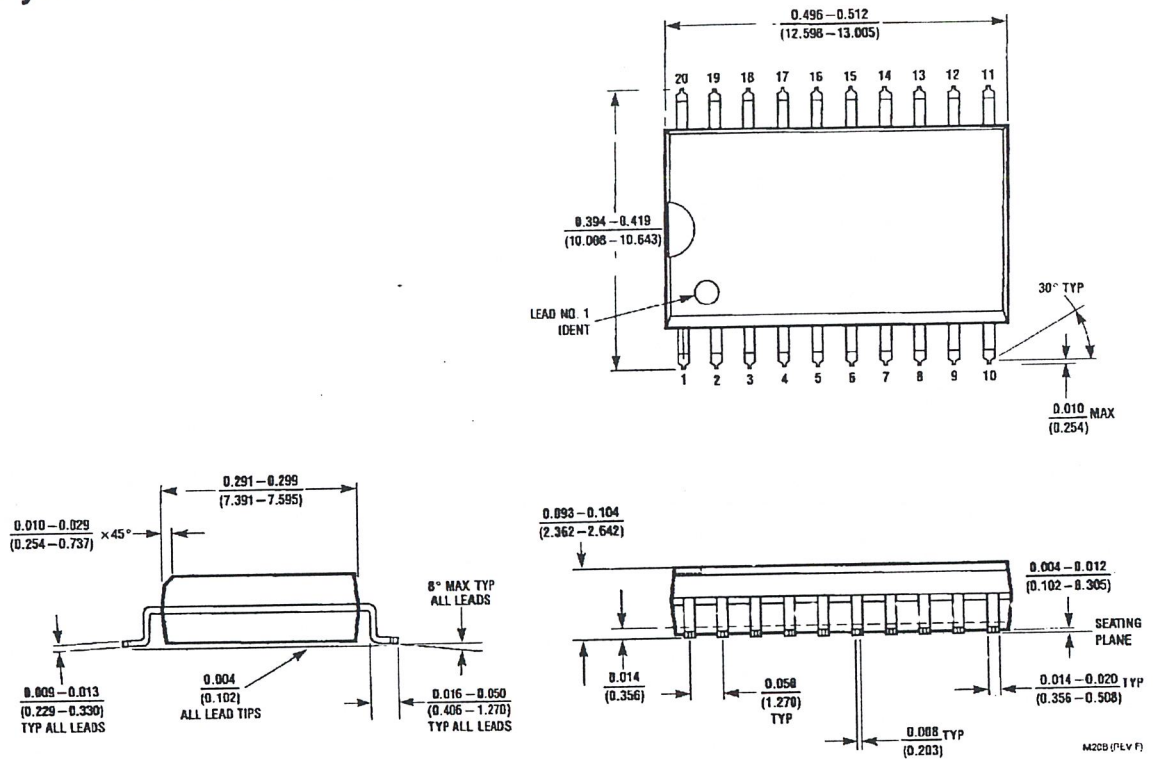


Ceramic Dual-In-Line Package (J)
Order Number MM54C922J or MM74C922J
NS Package Number J18A



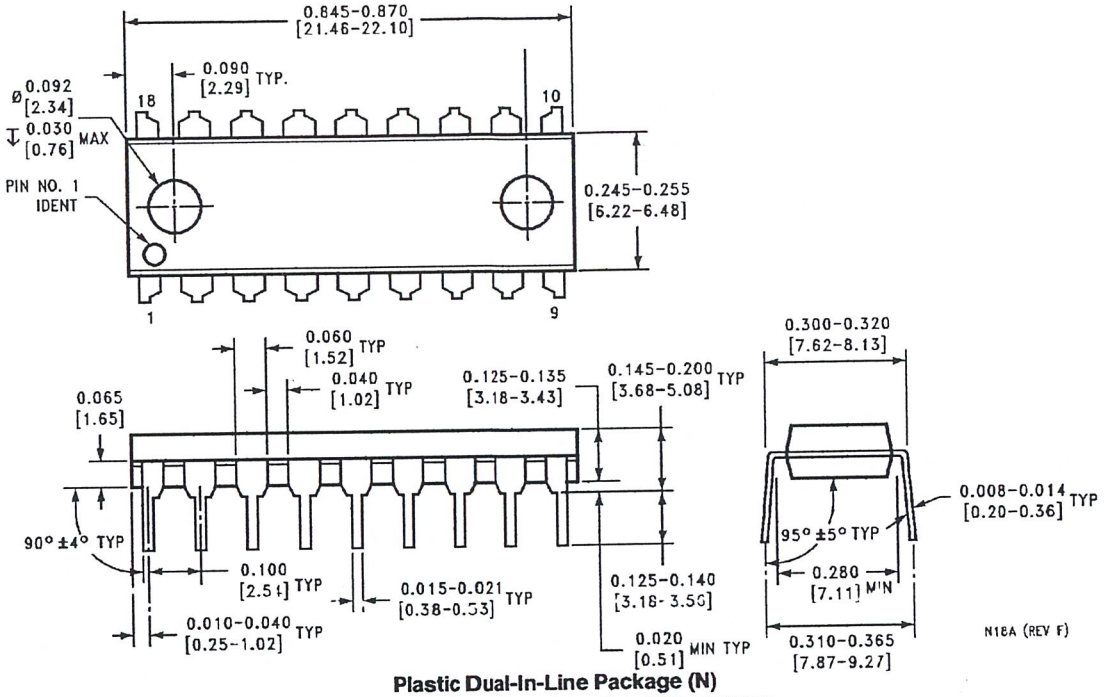
Ceramic Dual-In-Line Package (J)
Order Number MM54C923J or MM74C923J
NS Package Number J20A

Physical Dimensions inches (millimeters) (Continued)



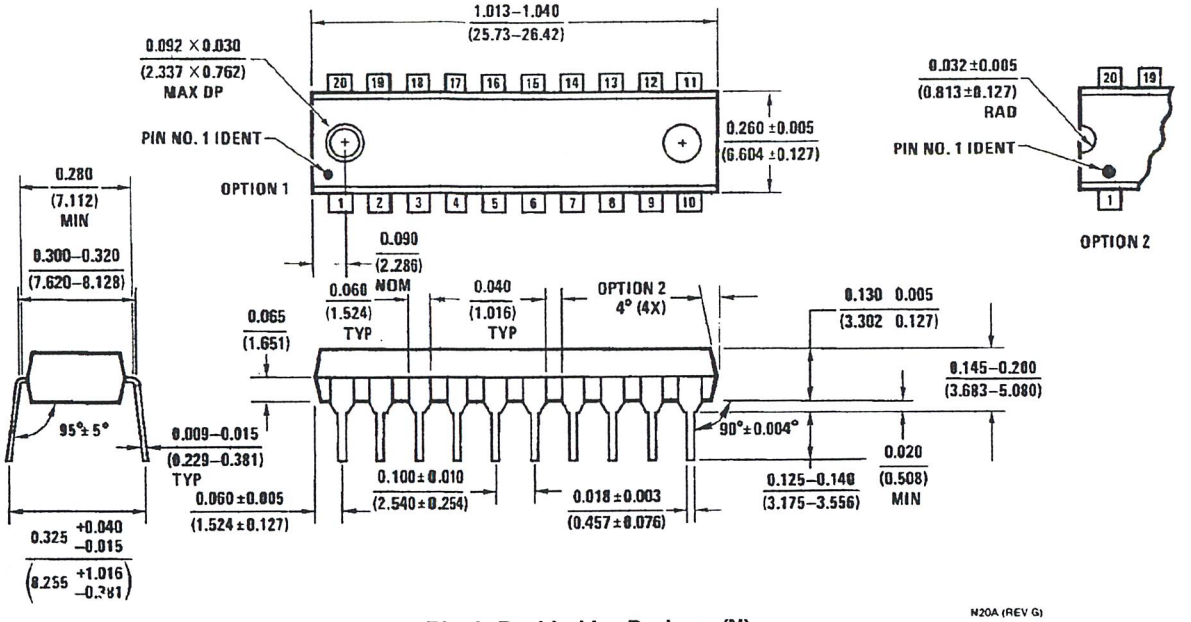
Plastic Small Outline I.C. Package (M)
Order Number MM74C922M or MM74C923M
NS Package Number M20B

Physical Dimensions inches (millimeters) (Continued)



Plastic Dual-In-Line Package (N)
Order Number MM54C922N or MM74C922N
NS Package Number N18A

Physical Dimensions inches (millimeters) (Continued)



Plastic Dual-In-Line Package (N)
Order Number MM54C923N or MM74C923N
NS Package Number N20A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

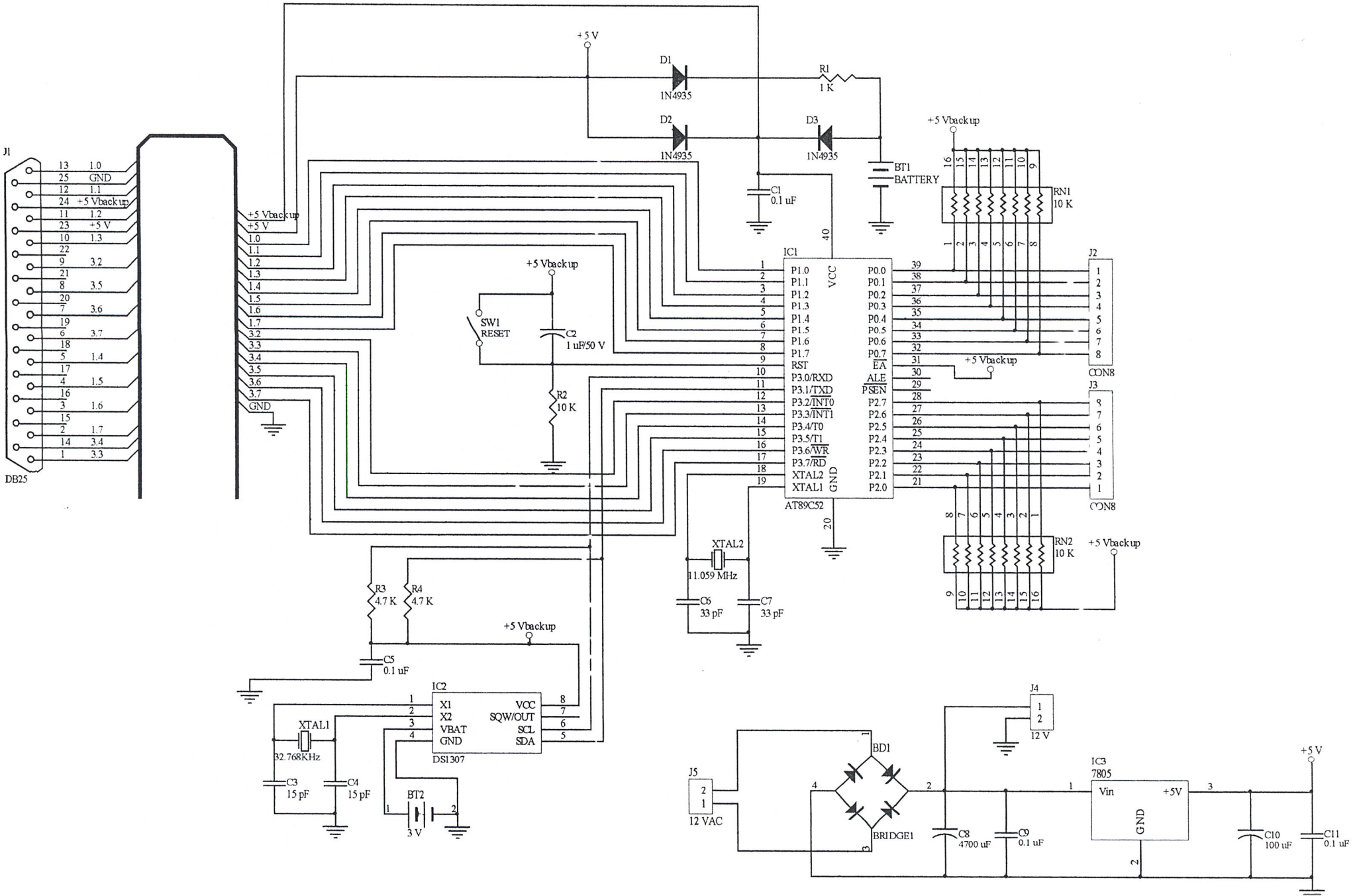
1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

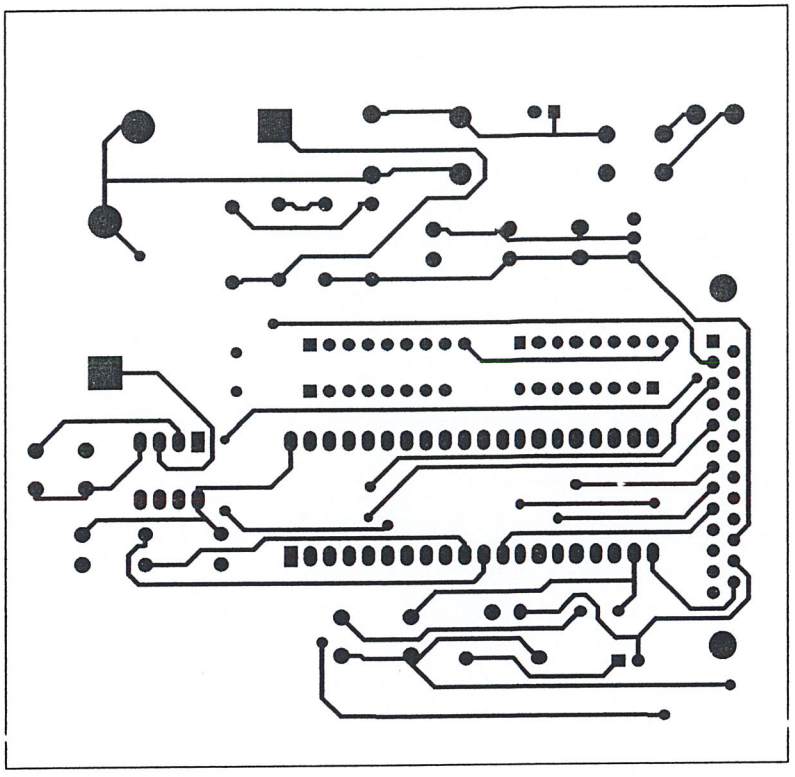
National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

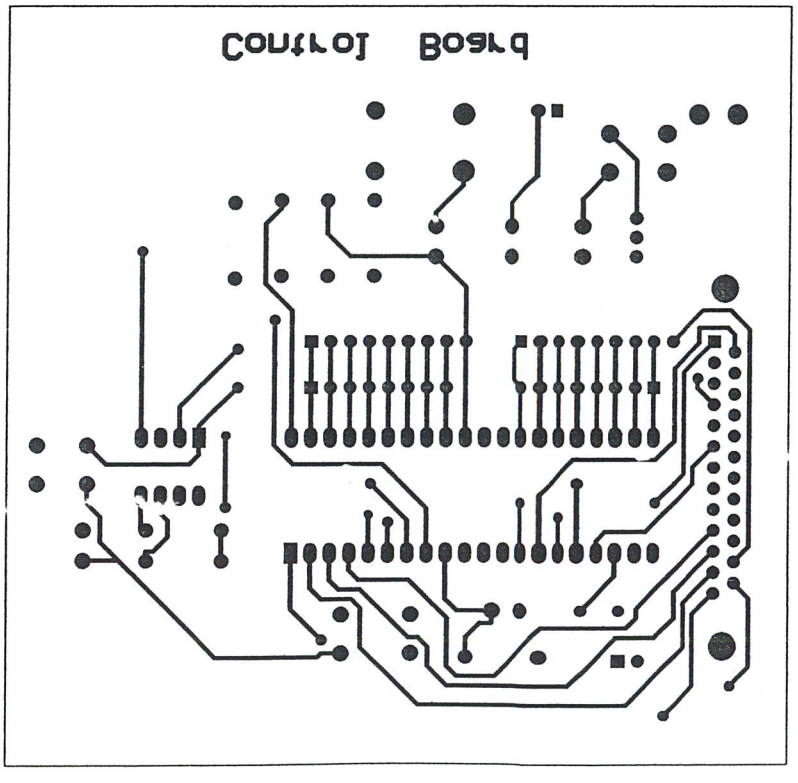
National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

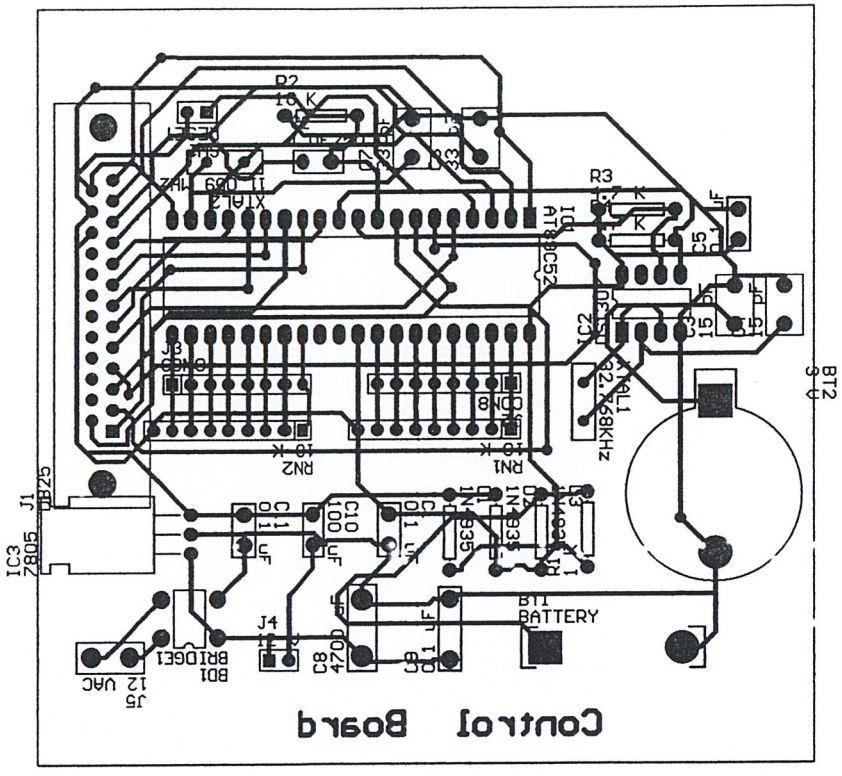
National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

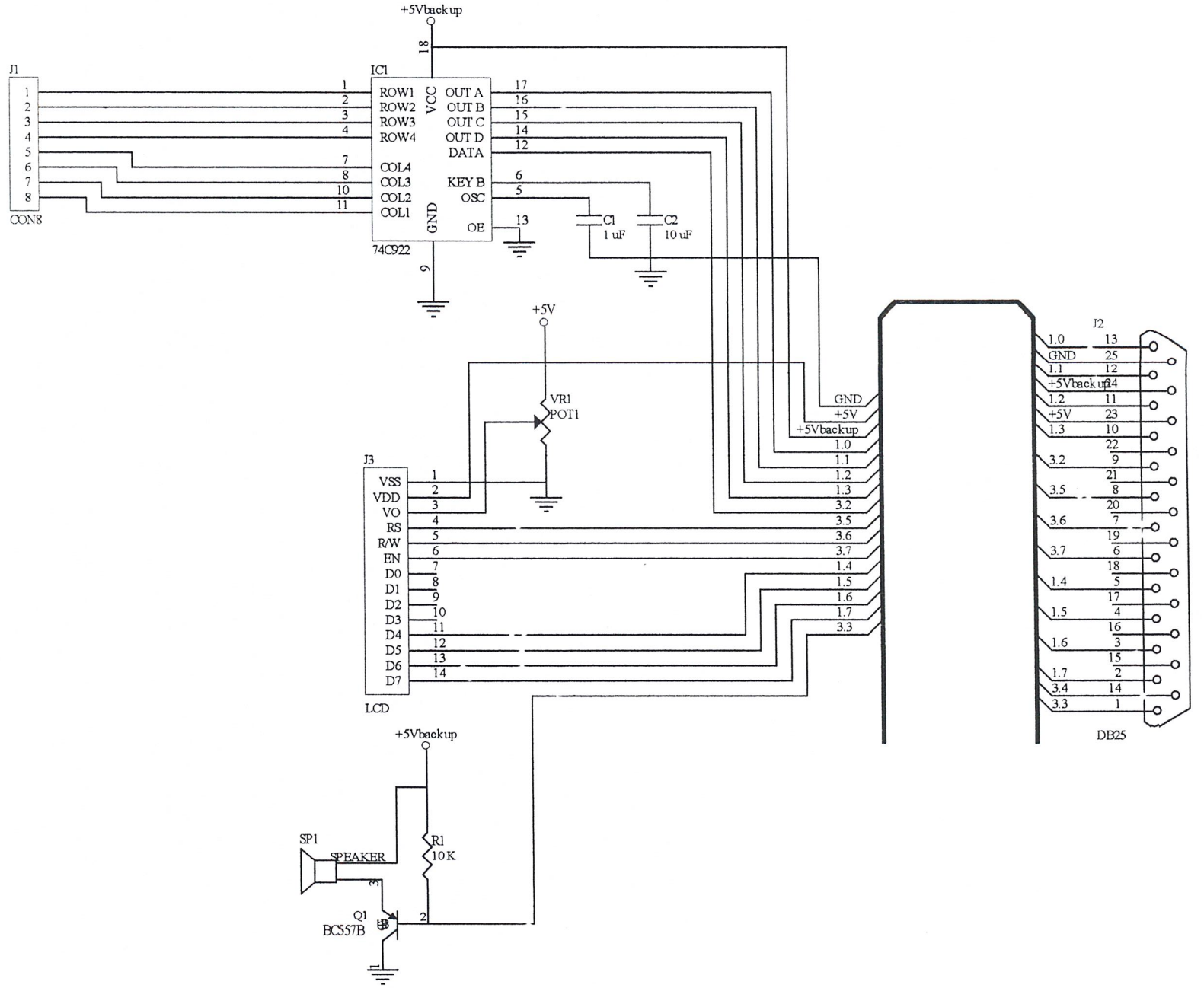
National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

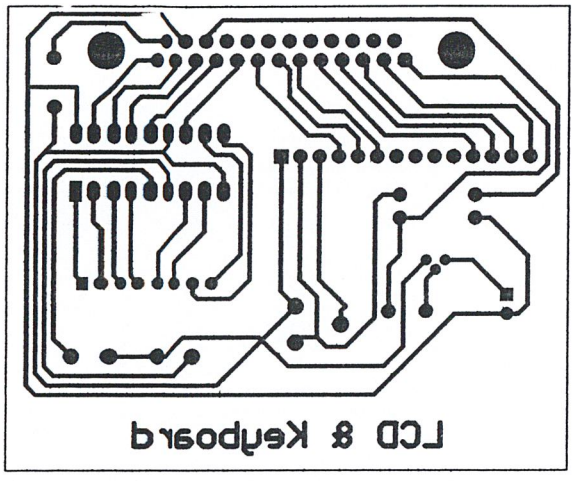

J1
DB25
13 1.0
25 GND
12 1.1
24 +5 Vbackup
11 1.2
23 +5 V
10 1.3
22
9 3.2
21
8 3.5
20
7 3.6
19
6 3.7
18
5 1.4
17
4 1.5
16
3 1.6
15
2 1.7
14 3.4
1 3.3
+5 Vbackup
+5 V
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
3.2
3.3
3.4
3.5
3.6
3.7
GND
+5 V
D1
1N4935
D2
1N4935
D3
1N4935
R1
1 K
BT1
BATTERY
C1
0.1 uF
+5 Vbackup
RN1
10 K
IC1
AT89C52
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
RST
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
XTAL2
XTAL1
VCC
GND
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
EA
ALE
PSEN
P2.7
P2.6
P2.5
P2.4
P2.3
P2.2
P2.1
P2.0
J2
CON8
J3
CON8
+5 Vbackup
SW1
RESET
C2
1 uF/50 V
R2
10 K
XTAL2
11.059 MHz
C6
33 pF
C7
33 pF
RN2
10 K
+5 Vbackup
R3
4.7 K
R4
4.7 K
+5 Vbackup
C5
0.1 uF
IC2
DS1307
X1
X2
VBAT
GND
VCC
SQW/OUT
SCL
SDA
XTAL1
32.768KHz
C3
15 pF
C4
15 pF
BT2
3 V
J4
12 V
J5
12 VAC
BD1
BRIDGE1
C8
4700 uF
C9
0.1 uF
IC3
7805
Vin
GND
+5V
+5 V
C10
100 uF
C11
0.1 uF

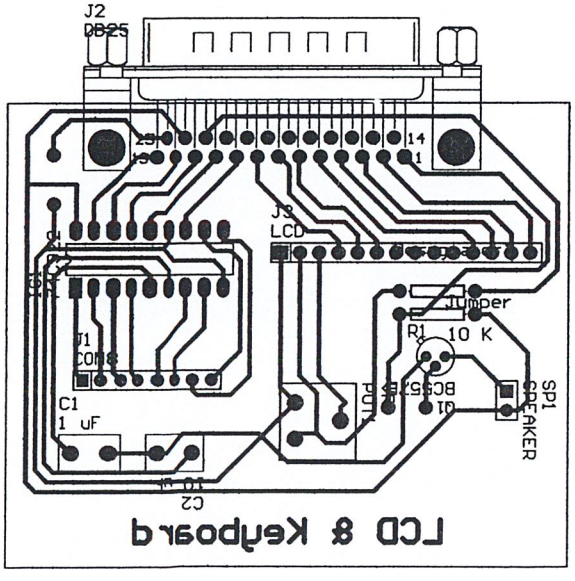


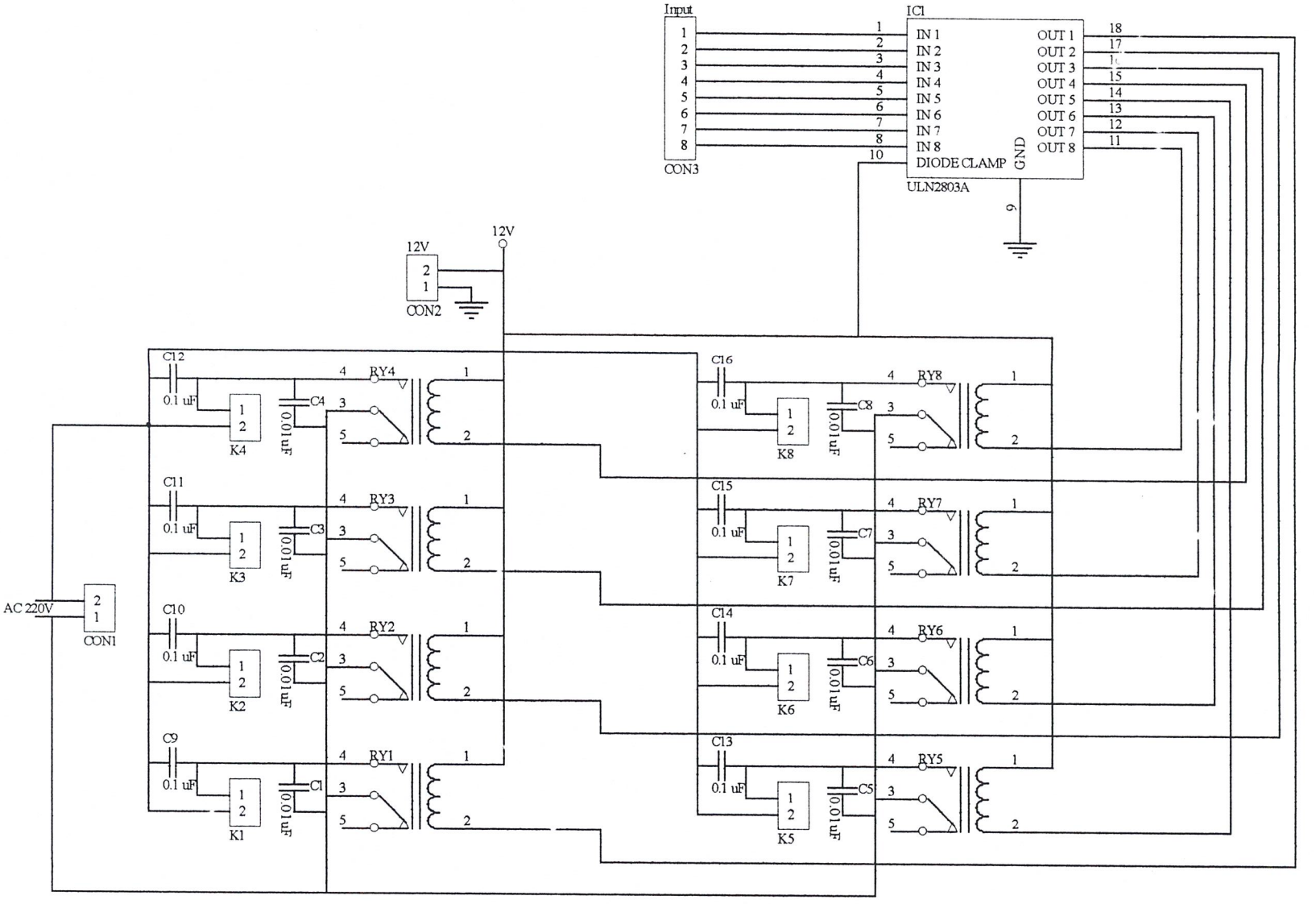

Control Board


Control Board
R2
10 K
R3
10 K
C5
.1 UF
C30
15 pF
C33
15 pF
IC2
XTAL1
12.288KHz
IC1
AT89C52
RN1
10 K
RN2
10 K
J3
CON8
J2
CON8
J1
PB25
IC3
7805
C11
.1 UF
C10
100 UF
C9
.1 UF
D1
IN4935
D2
IN4935
C7
.1 UF
C8
470 UF
C6
.1 UF
BT1
BATTERY
BT2
J4
BD1
BRIDGE1
J5
12 VAC


+5Vbackup
J1
CON8
IC1
74C922
ROW1
ROW2
ROW3
ROW4
COL4
COL3
COL2
COL1
VCC
GND
OUT A
OUT B
OUT C
OUT D
DATA
KEY B
OSC
OE
C1
1 uF
C2
10 uF
+5V
VR1
POT1
J3
LCD
VSS
VDD
VO
RS
R/W
EN
D0
D1
D2
D3
D4
D5
D6
D7
GND
+5V
+5Vbackup
J2
DB25
SP1
SPEAKER
R1
10 K
Q1
BC557B


LCD & Keyboard


J2
DB25
J3
LCD
Jumper
R1 10 K
J1
COM8
C1
1 uF
SP1
SPEAKER
LCD & Keyboard


Input
1
2
3
4
5
6
7
8
CON3
IC1
IN 1
IN 2
IN 3
IN 4
IN 5
IN 6
IN 7
IN 8
DIODE CLAMP
GND
OUT 1
OUT 2
OUT 3
OUT 4
OUT 5
OUT 6
OUT 7
OUT 8
ULN2803A
12V
CON2
AC 220V
CON1
C12
C11
C10
C9
C16
C15
C14
C13
0.1 uF
0.01uF
K1
K2
K3
K4
K5
K6
K7
K8
RY1
RY2
RY3
RY4
RY5
RY6
RY7
RY8

Relay 8 Channel

AC 220V
CON1
RY1
RELAY
RY2
RELAY
RY3
RELAY
RY4
RELAY
RY5
RELAY
RY6
RELAY
RY7
RELAY
RY8
RELAY
RN1
2k
IC1
ULN2803A
12V
CON2
Relay 8 Channel

```
;*******************************************************************************
; FILENAME        PEAC.ASM
; ASSEMBLER          SXA51
; ADVISER         Kiattisak  Kumwachara , D.Eng
; SOFTWARE ENG.      Mr.Thossaporn Srisuk (PICK) and Mr.Piphat Chepjareanrat (LEANG)
;                 King Mongkut's Institute of Technology Ladkrabang
;*******************************************************************************

; ***REAL TIME CLOCK CONTROL***
SCL             BIT         P3.0
SDA             BIT         P3.1
DS1307W         EQU         0D0H        ; SLAVE ADDRESS 1101 000 + 0 TO WRITE
DS1307R         EQU         0D1H        ; SLAVE ADDRESS 1101 000 + 1 TO READ
FLAGS           EQU         20H
LASTREAD        BIT         FLAGS.0
ACK             BIT         FLAGS.5
BUS_FLT         BIT         FLAGS.6
_2W_BUSY        BIT         FLAGS.7
BITCNT          EQU         21H
BYTECNT         EQU         22H
SECS            EQU         24H         ;    '   SECONDS STORAGE RAM
MINS            EQU         25H         ;    '   MINUTES     '      '
HRS             EQU         26H         ;    '   HOURS       '      '
DAY             EQU         27H         ;    '   DAY         '      '
DATE            EQU         28H         ;    '   DATE        '      '
MONTH           EQU         29H         ;    '   MONTH       '      '
YEAR            EQU         2AH         ;    '   YEAR        '      '
; ***LCD CONTROL***
LCD_DATA        EQU         090H
LCD_DB4         EQU         094H        ; P1.4 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_DB5         EQU         095H        ; P1.5 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_DB6         EQU         096H        ; P1.6 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_DB7         EQU         097H        ; P1.7 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_RS          EQU         0B5H        ; P3.3 LCD REGISTER SELECT LINE
LCD_RW          EQU         0B6H        ; P3.4 LCD READ / WRITE LINE
LCD_E           EQU         0B7H        ; P3.5 LCD ENABLE LINE
CONFIG          EQU         28H         ; 4-BIT DATA,2 LINES,5X7 MATRIX LCD
ENTRYMODE       EQU         6           ; INCREMENT CURSOR DON'T SHIFT DISPLAY
; ***CURSOR CONTROL INSTRUCTIONS***
OFFCUR          EQU         0CH
BLINKCUR        EQU         0DH
; ***DISPLAY CONTROL INSTRUCTIONS***
CLRDSP          EQU         01H
ONDSP           EQU         08H
;***SCAN KEY CONTROL***
KEY_IN0         BIT         P1.0
KEY_IN1         BIT         P1.1
KEY_IN2         BIT         P1.2
KEY_IN3         BIT         P1.3
KEY_DA          BIT         P3.2
BEEB            BIT         P3.3
;***BUFFER TIME DATE AND OUTPUT CONTROL***
BUF             EQU         23H             ; BUFFER
BUFOUT          EQU         2BH             ; BUFFER OUTPUT
STATUS          EQU         2CH             ; STATUS ON OFF
ONOFF           EQU         2DH             ; CHECK ON OFF
BUFDAY          EQU         2EH
BUFHOUR         EQU         2FH
BUFMIN          EQU         30H
BUFOR           EQU         31H
BUFORDER        EQU         32H
OLDPASS         EQU         1CH
BUFPAS0         EQU         33H
BUFPAS1         EQU         34H
BUFPAS2         EQU         35H
BUFPAS3         EQU         36H
CHKOUT1         EQU         38H
CHKOUT2         EQU         39H
CHKOUT3         EQU         3AH
CHKOUT4         EQU         3BH
CHKOUT5         EQU         3CH
CHKOUT6         EQU         3DH
CHKOUT7         EQU         3EH
```

```
CHKOUT8         EQU         3FH
BUFON           EQU         40H                 ; START BUFFER ON
BUFOFF          EQU         0A0H                ; START BUFFER OFF

                ORG 0000H
INITIAL:
                MOV     P2,#0
                SETB        SDA                 ; ENSURE SDA HIGH
                SETB        SCL                 ; ENSURE SCL HIGH
                JNB     SCL,$
                CLR     ACK                     ; CLEAR STATUS FLAGS
                CLR     BUS_FLT
                CLR     _2W_BUSY
                MOV     BUFPAS0,#1
                MOV     BUFPAS1,#2
                MOV     BUFPAS2,#3
                MOV     BUFPAS3,#4
                MOV     R0,#BUFON
                MOV     R2,#193
INI_1:          DJNZ        R2,INI_2
                AJMP        INI_3
INI_2:          MOV     @R0,#0FFH
                INC     R0
                AJMP        INI_1
INI_3:          LCALL       PULSEEWAIT4
                LCALL       OSC_CONTROL
                LCALL       RESETLCD4
;**************** TEST  PROGRAM***************
                MOV         R2,#8
                MOV         R0,#CHKOUT1
TEST:           MOV         @R0,#01H
                INC         R0
                DJNZ        R2,TEST
MAIN:
                LCALL       READ_CLOCK
MAIN_0:
                ACALL       DIS_TIME
MAIN_1:         JB          KEY_DA,GET_KEY
                LCALL       CHK_OUTPUT
                LCALL       READ_CLOCK
                MOV         A,SECS
                CJNE        A,#00H,MAIN_2
                AJMP        MAIN_0
MAIN_2:         ACALL       DIS_SEC
                AJMP        MAIN_1
GET_KEY:
                LCALL       BEEB_KEY
                JB          KEY_DA,$
                MOV         A,P1
                ANL         A,#0FH
                CJNE        A,#0BH,SCAN
                ACALL       SET_MENU                ;SET FUNCTION
                AJMP        MAIN
SCAN:           CJNE        A,#0EH,SCAN1
                LCALL       SET_HELP                ;DISPLAY OUTPUT
SCAN1:          CJNE        A,#0CH,MAIN
                MOV         R4,#CLRDSP
                LCALL       WRLCDCOM4
               AJMP     MAIN
;******************* DISPLAY LCD ********************
DIS_TIME:
                LCALL       RESETLCD4
                LCALL       INITLCD4
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '          /  /    ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '*     :  :       *',0
```

```
                MOV         R0,DAY
                MOV         A,#1
                MOV         B,#1
                ACALL           PRINTDAY

                MOV         R0,DATE
                MOV         A,#1
                MOV         B,#7
                ACALL       PRINTNUM

                MOV         R0,MONTH
                MOV         A,#1
                MOV         B,#10
                ACALL       PRINTNUM

                MOV         R0,YEAR
                MOV         A,#1
                MOV         B,#13
                ACALL       PRINTNUM

                MOV         R0,HRS
                MOV         A,#2
                MOV         B,#3
                ACALL       PRINTNUM

                MOV         R0,MINS
                MOV         A,#2
                MOV         B,#6
                ACALL       PRINTNUM

                MOV         R0,SECS
                MOV         A,#2
                MOV         B,#9
                ACALL       PRINTNUM
                ACALL       OFF_CUR
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                LCALL       DELAY_10MS
                RET
DIS_SEC:
                MOV         R0,SECS
                MOV         A,#2
                MOV         B,#9
                ACALL     PRINTNUM
                ACALL       OFF_CUR
                RET
;*************************************************************************
;   FUNCTION MENU EDIT
;*************************************************************************
SET_MENU:
                MOV         R4,#CLRDSP
                LCALL       WRLCDCOM4
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '1.Set Output 1/2',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '2.Set Clock    >>',0
CHK_MENU:       LCALL       KEY_DEE
                CJNE        R3,#01H,MENU
                RET
MENU:           CJNE        A,#00H,MENU_1           ;NUMBER KEY(1) SET OUTPUT
```

```
                LCALL     CHECKPASS
                CJNE      A,#0FFH,SET_ME
                RET
SET_ME:         CJNE      A,#0FH,SET_MENU
                ACALL     SET_OUTPUT
                CJNE      A,#0FFH,SET_MENU
                RET

MENU_1:         CJNE      A,#01H,MENU_2          ;NUMBERKEY(2) SET CLOCK
                LCALL     CHECKPASS
                CJNE      A,#0FFH,SET_ME1
                RET
SET_ME1:        CJNE      A,#0FH,SET_MENU
                ACALL     SET_CLK
                CJNE      A,#0FFH,SET_MENU
                RET
MENU_2:         CJNE      A,#07H,MENU_3          ;NUMBER KEY(NEXT)
                AJMP      NEXT_MENU
MENU_3:         CJNE      A,#0CH,SET_MENU        ;NUMBER KEY(CANCLE)
                RET
;***FUNCTION KEY NEXT***
NEXT_MENU:
                MOV       R4,#CLRDSP
                LCALL     WRLCDCOM4
                MOV       A,#1
                MOV       B,#0
                LCALL     PLACECUR4
                LCALL     PRTLCD4
                DB        '3.Password   2/2',0
                MOV       A,#2
                MOV       B,#0
                LCALL     PLACECUR4
                LCALL     PRTLCD4
                DB        '              <<',0
CHK_NEXT:       LCALL     KEY_DEE
                CJNE      R3,#01H,NEXT
                RET
NEXT:           CJNE      A,#02H,NEXT_1          ;NUMBER KEY(3) SET PASSWORD
                LCALL     PASSWORD
                CJNE      A,#0FFH,NEXT_MENU
                RET
NEXT_1:         CJNE      A,#03H,NEXT_2          ;NUMBERKEY(PREVIEW)
                AJMP      SET_MENU
NEXT_2:         CJNE      A,#0CH,CHK_NEXT        ;NUMBER KEY(CANCLE)
                RET
;********************* SUB. FUNCTION MEUN ***********************************************

;********************************************************************************************
;***FUNCTION SET CLOCK***
;********************************************************************************************
SET_CLK:
                MOV       R4,#CLRDSP
                LCALL     WRLCDCOM4
                MOV       A,#1
                MOV       B,#0
                LCALL     PLACECUR4
                LCALL     PRTLCD4
                DB        '1.Set Time   2-1',0
                MOV       A,#2
                MOV       B,#0
                LCALL     PLACECUR4
                LCALL     PRTLCD4
                DB        '2.Set Date       ',0
CHK_CLK:        LCALL     KEY_DEE
                CJNE      R3,#01H,CLK
                RET
CLK:            CJNE      A,#00H,CLK_1           ;NUMBER KEY(1) SET PASSWORD
                ACALL     SET_TIME
                CJNE      A,#0FFH,SET_CLK
                RET
CLK_1:          CJNE      A,#01H,CLK_2           ;NUMBERKEY(2)
                ACALL     SET_DATE
                CJNE      A,#0FFH,SET_CLK
```

```
                RET
CLK_2:          CJNE        A,#0CH,SET_CLK          ;NUMBER KEY(CANCLE)
                RET
;****************** SET DATE *****************************
SET_DATE:
                LCALL       READ_CLOCK
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '    Set Date     ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '         /  /    ',0
                MOV         R0,DAY
                MOV         A,#2
                MOV         B,#1
                ACALL       PRINTDAY

                MOV         R0,DATE
                MOV         A,#2
                MOV         B,#7
                ACALL       PRINTNUM

                MOV         R0,MONTH
                MOV         A,#2
                MOV         B,#10
                ACALL       PRINTNUM

                MOV         R0,YEAR
                MOV         A,#2
                MOV         B,#13
                ACALL       PRINTNUM
                MOV         R2,#0FFH
DAYS:
                MOV         R6,#1
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
                ACALL       BLINK_CUR

DAY_HI:         MOV         R1,#8
                LCALL       KEY_DE
                CJNE        R3,#01H,DY_0
                RET
DY_0:           MOV         BUF,DAY
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,DY_1
                AJMP        DAYS
DY_1:           CJNE        R0,#00H,DY_2
                AJMP        DAYS
DY_2:           MOV         A,DAY
                ACALL       LO
                MOV         DAY,A
                ACALL       YES_NODAT
                CJNE        R2,#0FH,DAY_0
                RET
DAY_0:          CJNE        R0,#0AH,DAY_1       ;NEXT
                AJMP        DATES
DAY_1:          CJNE        R0,#0BH,DAY_2       ;PREVIEW
                MOV         R6,#13
                AJMP        YEAR_LOW
DAY_2:
                ACALL       DIS_DAY
DATES:
                MOV         R6,#7
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
DATE_HI:
                MOV         R0,MONTH
```

```
                CJNE    R0,#02H,DATE_0
                MOV     R1,#3
                AJMP    DAT_H
DATE_0:         MOV     R1,#4
DAT_H:          LCALL   KEY_DE
                CJNE    R3,#01H,DATE_1
                RET
DATE_1:         MOV     BUF,DATE
                ACALL   KEY_NUM
                CJNE    R0,#0FFH,DAT
                AJMP    DATES
DAT:
                MOV     A,DATE
                ACALL   HI
                MOV     DATE,A
                ACALL   YES_NODAT
                CJNE    R2,#0FH,DAT_0
                RET
DAT_0:          CJNE    R0,#0AH,DAT_1     ;NEXT
                AJMP    DATE_LOW
DAT_1:          CJNE    R0,#0BH,DAT_2     ;PREVIEW
                MOV     R6,#1
                AJMP    DAYS
DAT_2:
                ACALL   DIS
DATE_LOW:
                INC     R6
DATE_LW:        MOV     A,DATE
                ANL     A,#0F0H
                MOV     R0,A
                MOV     A,#2
                MOV     B,R6
                LCALL   PLACECUR4
DA_L0:
                CJNE    R0,#20H,DAT_L1
                AJMP    DAT_CK
DAT_L1:         CJNE    R0,#30H,DAT_L
                MOV     R2,MONTH
                AJMP    CHK_DAT
DAT_CK:         MOV     R2,MONTH
                CJNE    R2,#2,DAT_L
                MOV     A,YEAR
                MOV     B,#4
                DIV     AB
                MOV     R2,B
                CJNE    R2,#0,CHK_DA
                AJMP    DAT_L
CHK_DA:         MOV     R1,#9
                AJMP    DA_L1
CHK_DAT:        CJNE    R2,#4,CHK_DAT_1
                MOV     R1,#1
                AJMP    DA_L1
CHK_DAT_1:      CJNE    R2,#6,CHK_DAT_2
                MOV     R1,#1
                AJMP    DA_L1
CHK_DAT_2:      CJNE    R2,#9,CHK_DAT_3
                MOV     R1,#1
                AJMP    DA_L1
CHK_DAT_3:      CJNE    R2,#11H,CHK_DAT_4
                MOV     R1,#1
                AJMP    DA_L1
CHK_DAT_4:      MOV     R1,#2
                AJMP    DA_L1
DAT_L:          MOV     R1,#10
DA_L1:          LCALL   KEY_DE
                CJNE    R3,#01H,DA_L2
                RET
DA_L2:          MOV     BUF,DATE
                ACALL   KEY_NUM
                CJNE    R0,#0FFH,DA_L3
                MOV     R2,#0FFH
                AJMP    DATE_LW
DA_L3:          CJNE    R0,#00H,DA_L4
```

```
                MOV         R2,#0FFH
                AJMP        DATE_LW
DA_LL:          MOV         A,DATE
                ACALL       LO
                MOV         DATE,A
                ACALL       YES_NODAT
                CJNE        R2,#0FH,DA_L4
                RET
DA_L4:          CJNE        R0,#0AH,DA_L5       ;NEXT
                AJMP        MONTHS
DA_L5:          CJNE        R0,#0BH,DA_L6       ;PREVIEW
                AJMP        DATES
DA_L6:          ACALL       DIS
MONTHS:
                MOV         R6,#10
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
MON_HI:         MOV         R1,#2
                LCALL       KEY_DE
                CJNE        R3,#01H,MON_0
                RET
MON_0:          MOV         BUF,MONTH
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,MON
                AJMP        MONTHS
MON:
                MOV         A,MONTH
                ACALL       HI
                MOV         MONTH,A
                ACALL       YES_NODAT
                CJNE        R2,#0FH,MON_1
                RET
MON_1:          CJNE        R0,#0AH,MON_2       ;NEXT
                AJMP        MON_LOW
MON_2:          CJNE        R0,#0BH,MON_3       ;PREVIEW
                MOV         R6,#7
                AJMP        DATE_LOW
MON_3:          ACALL       DIS
                CJNE        R0,#01H,MON_4
                MOV         R2,#01H
                AJMP        MON_LOW
MON_4:          MOV         R2,#03H
MON_LOW:
                INC         R6
MON_LW:         MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
MN_L0:
                CJNE        R2,#01H,MN_L1
                MOV         R1,#3
                AJMP        MN_L2
MN_L1:          MOV         R1,#10
MN_L2:          LCALL       KEY_DE
                CJNE        R3,#01H,MN_L3
                RET
MN_L3:          MOV         BUF,MONTH
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,MN_1
                AJMP        MON_LW
MN_1:           MOV         A,MONTH
                ACALL       LO
                MOV         MONTH,A
                ACALL       YES_NODAT
                CJNE        R2,#0FH,MN_4
                RET
MN_4:           CJNE        R0,#0AH,MN_5        ;NEXT
                AJMP        YEARS
MN_5:           CJNE        R0,#0BH,MN_6        ;PREVIEW
                AJMP        MONTHS
MN_6:           ACALL       DIS
YEARS:
                MOV         R6,#13
```

```
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
YRS_HI:         MOV         R1,#10
                LCALL       KEY_DE
                CJNE        R3,#01H,YS_0
                RET
YS_0:           MOV         BUF,YEAR
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,YRS
                AJMP        YEARS
YRS:
                MOV         A,YEAR
                ACALL       HI
                MOV         YEAR,A
                ACALL       YES_NODAT
                CJNE        R2,#0FH,YR_0
                RET
YR_0:           CJNE        R0,#0AH,YR_1        ;NEXT
                AJMP        YEAR_LOW
YR_1:           CJNE        R0,#0BH,YR_2        ;PREVIEW
                MOV         R6,#10
                AJMP        MON_LOW
YR_2:
                ACALL       DIS
YEAR_LOW:
                INC         R6
YR_LW:          MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
YE_L0:
                MOV         R1,#10
YE_L1:          LCALL       KEY_DE
                CJNE        R3,#01H,YE_L2
                RET
YE_L2:          MOV         BUF,YEAR
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,YE_L3
                JMP     YEAR_LOW
YE_L3.          MOV         A.YEAR
                ACALL       LO
                MOV         YEAR,A
                ACALL       YES_NODAT
                CJNE        R2,#0FH,YE_L4
                RET
YE_L4:          CJNE        R0,#0AH,YE_L5       ;NEXT
                AJMP        DAYS
YE_L5:          CJNE        R0,#0BH,YE_L6       ;PREVIEW
                AJMP        YEARS
YE_L6:          ACALL       DIS
                AJMP        DAYS
;****************** SET TIME *****************************
SET_TIME:       LCALL       READ_CLOCK
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '    Set Time     ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '      :  :       ',0
                MOV         R0,HRS
                MOV         A,#2
                MOV         B,#4
                ACALL         PRINTNUM

                MOV         R0,MINS
                MOV         A,#2
                MOV         B,#7
                ACALL         PRINTNUM
```

```
                MOV         R0,SECS
                MOV         A,#2
                MOV         B,#10
                ACALL        PRINTNUM
                MOV         R2,#0FFH
HOUR:
                MOV         R6,#4
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
                ACALL       BLINK_CUR
HR_HI:          MOV         R1,#3
                LCALL       KEY_DE
                CJNE        R3,#01H,HR_0
                RET
HR_0:           MOV         BUF,HRS
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,HR
                AJMP        HOUR
HR:
                MOV         A,HRS
                ACALL       HI
                MOV         HRS,A
                ACALL       YES_NOCLK
                CJNE        R2,#0FH,CHK_0
                RET
CHK_0:          CJNE        R0,#0AH,CHK             ;NEXT
                AJMP        HOUR_LOW
CHK:            CJNE        R0,#0BH,CHK_1       ;PREVIEW
                MOV         R6,#10
                AJMP        SEC_LOW
CHK_1:          ACALL       DIS
                CJNE        R0,#02H,CHK_2
                MOV         R2,#02H
                AJMP        HOUR_LOW
CHK_2:          MOV         R2,#01H
HOUR_LOW:
                INC         R6
LW:             MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
HR_L0:
                CJNE        R2,#02H,HR_L1
                MOV         R1,#5
                AJMP        HR_L2
HR_L1:          MOV         R1,#10
HR_L2:          LCALL       KEY_DE
                CJNE        R3,#01H,HR_L3
                RET
HR_L3:          MOV         BUF,HRS
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,CH_L1
                AJMP        LW
CH_L1:          MOV         A,HRS
                ACALL       LO
                MOV         HRS,A
                ACALL       YES_NOCLK
                CJNE        R2,#0FH,CHK_4
                RET
CHK_4:          CJNE        R0,#0AH,CHK_5       ;NEXT
                AJMP        MIN
CHK_5:          CJNE        R0,#0BH,CHK_6       ;PREVIEW
                AJMP        HOUR
CHK_6:          ACALL       DIS
MIN:
                MOV         R6,#7
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
MIN_HI:         MOV         R1,#6
                LCALL       KEY_DE
                CJNE        R3,#01H,MIN_0
                RET
```

```
MIN_0:      MOV     BUF,MINS
            ACALL   KEY_NUM
            CJNE    R0,#0FFH,MI
            AJMP    MIN
MI:
            MOV     A,MINS
            ACALL   HI
            MOV     MINS,A
            ACALL   YES_NOCLK
            CJNE    R2,#0FH,MI_0
            RET
MI_0:       CJNE    R0,#0AH,MI_1        ;NEXT
            AJMP    MIN_LOW
MI_1:       CJNE    R0,#0BH,MI_2        ;PREVIEW
            MOV     R6,#4
            AJMP    HOUR_LOW
MI_2:
            ACALL   DIS
MIN_LOW:
            INC     R6
MIN_LW:     MOV     A,#2
            MOV     B,R6
            LCALL   PLACECUR4
MI_L0:
            MOV     R1,#10
MI_L1:      LCALL   KEY_DE
            CJNE    R3,#01H,MI_L2
            RET
MI_L2:      MOV     BUF,MINS
            ACALL   KEY_NUM
            CJNE    R0,#0FFH,MI_L3
            AJMP    MIN_LOW
MI_L3:      MOV     A,MINS
            ACALL   LO
            MOV     MINS,A
            ACALL   YES_NOCLK
            CJNE    R2,#0FH,MI_L4
            RET
MI_L4:      CJNE    R0,#0AH,MI_L5       ;NEXT
            AJMP    SEC
MI_L5:      CJNE    R0,#0BH,MI_L6       ;PREVIEW
            AJMP    MIN
MI_L6:      ACALL   DIS
SEC:
            MOV     R6,#10
            MOV     A,#2
            MOV     B,R6
            LCALL   PLACECUR4
SEC_HI:     MOV     R1,#6
            LCALL   KEY_DE
            CJNE    R3,#01H,SEC_0
            RET
SEC_0:      MOV     BUF,SECS
            ACALL   KEY_NUM
            CJNE    R0,#0FFH,SE
            AJMP    SEC
SE:
            MOV     A,SECS
            ACALL   HI
            MOV     SECS,A
            ACALL   YES_NOCLK
            CJNE    R2,#0FH,SE_0
            RET
SE_0:       CJNE    R0,#0AH,SE_1        ;NEXT
            AJMP    SEC_LOW
SE_1:       CJNE    R0,#0BH,SE_2        ;PREVIEW
            MOV     R6,#7
            AJMP    MIN_LOW
SE_2:
            ACALL   DIS
SEC_LOW:
            INC     R6
SEC_LW:     MOV     A,#2
```

```
                MOV         B,R6
                LCALL       PLACECUR4
SE_L0:
                MOV         R1,#10
SE_L1:          LCALL       KEY_DE
                CJNE        R3,#01H,SE_L2
                RET
SE_L2:          MOV         BUF,SECS
                ACALL       KEY_NUM
                CJNE        R0,#0FFH,SE_L3
                AJMP        SEC_LOW
SE_L3:          MOV         A,SECS
                ACALL       LO
                MOV         SECS,A
                ACALL       YES_NOCLK
                CJNE        R2,#0FH,SE_L4
                RET
SE_L4:          CJNE        R0,#0AH,SE_L5       ;NEXT
                AJMP        HOUR
SE_L5:          CJNE        R0,#0BH,SE_L6       ;PREVIEW
                AJMP        SEC
SE_L6:          ACALL       DIS
                AJMP        HOUR
;************************ SUB SET CLOCK ************************************
YES_NOCLK:
                CJNE        R0,#0CH,YN          ;CANCLE
                MOV         R2,#0FH
                RET
YN:             CJNE        R0,#0DH,YN_1        ;SAVE
                LCALL       SET_CLOCK
                MOV         R2,#0FH
YN_1:           RET
YES_NODAT:
                CJNE        R0,#0CH,YN_2        ;CANCLE
                MOV         R2,#0FH
                RET
YN_2:           CJNE        R0,#0DH,YN_3        ;SAVE
                LCALL       SET_CLKDAT
                MOV         R2,#0FH
YN_3:           RET
DIS:
                MOV         A,#2
                MOV         B,R6
                LCALL         PLACECUR4
                MOV         A,R0
                LCALL       FIND
                RET
DIS_DAY:
                MOV         A,#2
                MOV         B,R6
                LCALL         PLACECUR4
                MOV         A,R0
                LCALL       FIND_DAY
                RET
HI:
                PUSH        01H
                ANL         A,#0FH
                MOV         R1,A
                MOV         A,BUF
                ANL         A,#70H
                ORL         A,R1
                POP         01H
                RET
LO:
                PUSH        01H
                ANL         A,#70H
                MOV         R1,A
                MOV         A,BUF
                ANL         A,#0FH
                ORL         A,R1
                POP         01H
                RET
KEY_NUM:
```

```
                MOV     R0,#0FFH
                MOV     R7,#0FFH
NUM:
                CJNE    A,#0DH,N_0      ; 0
                MOV     BUF,#00H
                MOV     R0,#00H
                RET
N_0:            DJNZ    R1,NUM_0
                AJMP    NUM_9
NUM_0:          CJNE    A,#00H,N_1      ; 1
                MOV     BUF,#11H
                MOV     R0,#01H
                RET
N_1:            DJNZ    R1,NUM_1
                AJMP    NUM_9
NUM_1:          CJNE    A,#01H,N_2      ; 2
                MOV     BUF,#22H
                MOV     R0,#02H
                RET
N_2:            DJNZ    R1,NUM_2
                AJMP    NUM_9
NUM_2:          CJNE    A,#02H,N_3      ; 3
                MOV     BUF,#33H
                MOV     R0,#03H
                RET
N_3:            DJNZ    R1,NUM_3
                AJMP    NUM_9
NUM_3:          CJNE    A,#04H,N_4      ; 4
                MOV     BUF,#44H
                MOV     R0,#04H
                RET
N_4:            DJNZ    R1,NUM_4
                AJMP    NUM_9
NUM_4:          CJNE    A,#05H,N_5      ; 5
                MOV     BUF,#55H
                MOV     R0,#05H
                RET
N_5:            DJNZ    R1,NUM_5
                AJMP    NUM_9
NUM_5:          CJNE    A,#06H,N_6      ; 6
                MOV     BUF,#66H
                MOV     R0,#06H
                RET
N_6:            DJNZ    R1,NUM_6
                AJMP    NUM_9
NUM_6:          CJNE    A,#08H,N_7      ; 7
                MOV     BUF,#77H
                MOV     R0,#07H
                RET
N_7:            DJNZ    R1,NUM_7
                AJMP    NUM_9
NUM_7:          CJNE    A,#09H,N_8      ; 8
                MOV     BUF,#88H
                MOV     R0,#08H
                RET
N_8:            DJNZ    R1,NUM_8
                AJMP    NUM_9
NUM_8:          CJNE    A,#0AH,N_9      ; 9
                MOV     BUF,#99H
                MOV     R0,#09H
                RET
N_9:            DJNZ    R1,NUM_9
NUM_9:          CJNE    A,#07H,NUM_10   ;NUMBER KEY(NEXT) SHIFT CURSOR AND BUFFER
                MOV     R0,#0AH
                MOV     R7,#0FH
                RET
NUM_10:         CJNE    A,#03H,NUM_11   ;NUMBERKEY(PREVIEW) BACK CURSOR AND BUFFER
                MOV     R0,#0BH
                MOV     R7,#0FH
                RET
NUM_11:         CJNE    A,#0CH,NUM_12   ;NUMBER KEY(CANCLE)
                MOV     R0,#0CH
                RET
```

```
NUM_12:         CJNE        A,#0FH,NUM_13     ;NUMBER KEY(SAVE)
                MOV         R0,#0DH
                MOV         R7,#0FH
                RET
NUM_13:         CJNE        A,#0BH,NUM_14     ;NUMBER KEY(2ND)
                MOV         R7,#0FH
                MOV         R0,#0EH
NUM_14:         RET
;*******************************************************************************
;*** END SET CLOCK ***
;*******************************************************************************
;*****************************************************
;   SUB. FUNCTION CHECK KEY AND DELAY 5 SECOND
;*****************************************************
KEY_DE:
                PUSH        00H
                PUSH        01H
                PUSH        02H
                MOV         R2,#45
                MOV         R0,#100
DE_100MS_1:
                MOV         R1,#0FFH              ; 1 MS
DE_100MS_2:     JB          KEY_DA,GET
                DJNZ        R1,DE_100MS_2
                DJNZ        R0,DE_100MS_1
                DJNZ        R2,DE_100MS_1
                MOV         A,#0FFH
                MOV         R3,#01H
                POP         02H
                POP         01H
                POP         00H
                RET
GET:            LCALL       BEEB_KEY
                JB          KEY_DA,$
                MOV         A,P1
                ANL         A,#0FH
                MOV         R3,#00H
                POP         02H
                POP         01H
                POP         00H
                RET
;*****************************************************
;            SUB. FUNCTION DISPLAY LCD
;*****************************************************
PRINTNUM:
                LCALL         PLACECUR4
                LCALL       WRITE_BCD
                RET
PRINTNUM_1:
                LCALL         PLACECUR4
                MOV         A,R0
                LCALL       FIND
                RET
PRINTDAY:
                LCALL         PLACECUR4
                LCALL       WRITE_DAY
                RET
OFF_CUR:
                MOV          R4,#OFFCUR           ; TURN OFF CURSOR
                LCALL       WRLCDCOM4
                RET
BLINK_CUR:
                MOV          R4,#BLINKCUR
                LCALL       WRLCDCOM4
                RET
OUT_NUM:
                MOV         A,STATUS
                MOV         B,R6
                LCALL         PLACECUR4
                MOV         A,R0
                LCALL       FIND
                RET
```

```
;************************************************
;  FUNCTION  OUTPUT
;************************************************
SET_OUTPUT:
                MOV         R4,#CLRDSP
                LCALL       WRLCDCOM4
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '1.Set Output 1-1',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '2.Clear Output  ',0
S_CLK:          LCALL       KEY_DEE
                CJNE        R3,#01H,S_CLK_0
                RET
S_CLK_0:        CJNE        A,#00H,S_CLK_1          ;NUMBER KEY(1) SET OUTPUT
                ACALL       SET_OUT
                CJNE        A,#0FFH,SET_OUTPUT
                RET
S_CLK_1:        CJNE        A,#01H,S_CLK_2          ;NUMBERKEY(2) CLEAR OUTPUT
                LCALL       SET_CLR
                CJNE        A,#0FFH,SET_OUTPUT
                RET
S_CLK_2:        CJNE        A,#0CH,S_CLK            ;NUMBER KEY(CANCLE)
S_CLK_3:        RET
;************************************************
;  FUNCTION SET OUTPUT
;************************************************
SET_OUT:
                MOV         R4,#CLRDSP
               LCALL        WRLCDCOM4
                MOV         BUFOUT,#0FFH
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' Press Key(1-8) ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '  Output= 1 Ent.',0
                MOV         R7,#0FFH
                MOV         BUFOUT,#01H
SETOUT:
                MOV         R6,#10
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
                LCALL       BLINK_CUR
SETOUT1:
                MOV         R1,#9
                LCALL       KEY_DEE
                CJNE        R3,#01H,ST_OUT1
                RET
ST_OUT1:
                LCALL       KEY_NUM
                CJNE        R0,#0CH,ST_OUT2         ;CANCLE
SETOUT2:        MOV         A,#55H
                MOV         BUFOUT,#0FFH
                RET
ST_OUT2:        CJNE        R0,#0DH,ST_OUT3         ;SET FUNCTION OUTPUT
                MOV         A,#0F0H
                LCALL       SET_ORDER
                CJNE        A,#0FFH,SET_OUT
                RET
ST_OUT3:        CJNE        R7,#0FH,ST_OUT4
                LJMP        SETOUT1
ST_OUT4:        CJNE        R0,#0FFH,ST_OUT5
```

```
                LJMP        SETOUT1
ST_OUT5:        CJNE        R0,#00H,ST_OUT6
                LJMP        SETOUT1
ST_OUT6:        MOV         A,R0
                ANL         A,#0FH
                MOV         BUFOUT,A
                LCALL       DIS
                LJMP        SETOUT
;***********************************************
;  FUNCTION SET PROGRAM ORDER
;***********************************************
SET_ORDER:
                MOV          R4,#CLRDSP
               LCALL        WRLCDCOM4
                MOV         BUFOR,#0FFH
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          'Load  .   On/Off',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' Program 1 2 3 4',0
                MOV         R7,#0FFH
                MOV         A,#1
                MOV         B,#5
                MOV         R0,BUFOUT
                LCALL         PRINTNUM_1
                LCALL       OFF_CUR
SETOR:
                MOV         R1,#5
                LCALL       KEY_DEE
                CJNE        R3,#01H,ST_OR
                RET
ST_OR:
                LCALL       KEY_NUM
                CJNE        R0,#0CH,ST_OR1          ;CANCLE
                MCV         A,#55h
                RET
ST_OR1:
                CJNE        R7,#0FH,ST_OR2          ;DON'T  CARE  KEY  NEXT  ,  PREVIEW   AND
ENTER
                AJMP        SETOR
ST_OR2:
                CJNE        R0,#0FFH,ST_OR3
                AJMP        SETOR
ST_OR3:
                CJNE        R0,#00H,ST_OR4          ;DON'T CARE KEY "0"
                AJMP        SETOR
ST_OR4:
                MOV         A,R0
                MOV         BUFORDER,A
                MOV         B,#2
                MUL         AB
                MOV         BUFOR,A
                ACALL       OUT_FUN
                CJNE        A,#0FFH,SET_ORDER
                RET
;***********************************************************************************************
OUT_FUN:
                MOV         STATUS,#1
                MOV         R1,#BUFON
                ACALL       CHK_STATUS
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' . Day hh:mm  On',0
                ACALL       PRT_FUN
                MOV         STATUS,#2
                MOV         R1,#BUFOFF
```

```
                ACALL       CHK_STATUS
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' . Day hh:mm Off',0
                ACALL       PRT_FUN
                MOV         A,#2
                MOV         B,#0
                MOV         R0,BUFOR
                LCALL         PRINTNUM_1
                MOV         A,#1
                MOV         B,#0
                DEC         R0
                LCALL         PRINTNUM_1
                ACALL       OUT_ON
                RET
;*******************************************************************************
;     SUB. PRINT ON DISPLAY
;*******************************************************************************
PRT_FUN:
                MOV         A,STATUS
                MOV         B,#3
                MOV         R0,BUFDAY
                LCALL       PRINTDAY
                MOV         A,STATUS
                MOV         B,#7
                MOV         R0,BUFHOUR
                LCALL       PRINTNUM
                MOV         A,STATUS
                MOV         B,#10
                MOV         R0,BUFMIN
                LCALL       PRINTNUM
                RET
;*******************************************************************************
;     CHECK STATUS OF BUFFER OUTPUT
;*******************************************************************************
CHK_STATUS:
                MOV         R2,BUFORDER
CHK_ST.         DJNZ        R2,CHK_ST1
                MOV         R2,BUFOUT
CHK_S:          DJNZ        R2,CHK_S1
                ACALL       KEEP_OUT
                RET
CHK_S1:
                MOV         A,R1
                ADD         A,#3
                MOV         R1,A
                AJMP        CHK_S
CHK_ST1:        MOV         A,R1
                ADD         A,#24
                MOV         R1,A
                AJMP        CHK_ST
KEEP_OUT:       MOV         R2,#3
                MOV         R0,#BUFDAY
KEEP:           MOV         A,@R1
                MOV         @R0,A
KEEP_CHK:       DJNZ        R2,KEEP_CHK1
                RET
KEEP_CHK1:      INC         R1
                INC         R0
                AJMP        KEEP
;*******************************************************************************
;     SET OUTPUT ON
;*******************************************************************************
OUT_ON:
                MOV         STATUS,#1
                MOV         A,#1
                MOV         B,#3
                LCALL       PLACECUR4
                LCALL       BLINK_CUR
                MOV         R1,#8
                LCALL       KEY_DE
```

```
                CJNE    R3,#01H,ON_DA0
                RET
ON_DA0:
                LCALL   KEY_NUM
                CJNE    R0,#0AH,ON_DA1          ;KEY NEXT
                AJMP    OUT_OFF
                RET
ON_DA1:
                CJNE    R0,#0CH,ON_DA2          ;CANCLE    EXIT
                RET
ON_DA2:
                CJNE    R0,#0DH,ON_DA3          ;KEY SAVE
                MOV     BUFDAY,#0FFH
                MOV     A,#1
                MOV     B,#3
                LCALL   PLACECUR4
                LCALL   PRTLCD4
                DB      '   ',0
                AJMP    ON_TIME
ON_DA3:
                CJNE    R0,#0EH,ON_DA4
                ACALL   CLRPRO                  ;CLEAR ONE PROGRAM
                AJMP    OUT_ON
ON_DA4:         CJNE    R7,#0FH,ON_DA5
                AJMP    OUT_ON
ON_DA5:
                CJNE    R0,#0FFH,ON_DA6             ;NOT PRESS KEY NUMBER
                AJMP    OUT_ON
ON_DA6:
                CJNE    R0,#00H,ON_DA7          ;DON'T CARE KEY NUMBER 0
                AJMP    OUT_ON
ON_DA7:
                MOV     A,BUF                   ;DISPLAY DAY ON LCD
                LCALL   LO
                ANL     A,#0FH
                MOV     BUFDAY,A
                MOV     A,STATUS
                MOV     B,#3
                LCALL     PLACECUR4
                MOV     A,R0
                ACALL   FIND_DAY
ON_TIME:
                ACALL   WE_HOUR
                CJNE    R1,#0FH,ON_TIME1
                AJMP    OUT_ON
ON_TIME1:
                CJNE    A,#0FFH,ON_TIME2
                RET
ON_TIME2:
                MOV     A,#1
                MOV     B,#15
                LCALL   PLACECUR4
                LCALL   BLINK_CUR
                MOV     R1,#0
                LCALL   KEY_DE
                CJNE    R3,#01H,ON_TIME3
                RET
ON_TIME3:
                LCALL   KEY_NUM
                CJNE    R0,#0DH,ON_TIME4        ;SAVE
                ACALL   SAVE_OUT
                AJMP    OUT_OFF
ON_TIME4:
                CJNE    R0,#0CH,ON_TIME5        ;CANCLE
                AJMP    OUT_ON
ON_TIME5:
                CJNE    R0,#0AH,ON_TIME6        ;KEY NEXT GO TO SET FUNCTION OFF
                AJMP    OUT_OFF
ON_TIME6:
                CJNE    R0,#0EH,ON_TIME7
                ACALL   CLRPRO                  ;CLEAR ONE PROGRAM
                AJMP    OUT_ON
ON_TIME7:
```

```
                CJNE        R7,#0FH,ON_TIME8
                AJMP        OUT_ON
ON_TIME8:
                CJNE        R0,#0FFH,ON_TIME9
ON_TIME9:       AJMP        OUT_ON
;*****************************************************************************
;     SET OUTPUT OFF
;*****************************************************************************
OUT_OFF:
                MOV         STATUS,#2
                MOV         A,#2
                MOV         B,#3
                LCALL       PLACECUR4
                LCALL       BLINK_CUR
                MOV         R1,#8
                LCALL       KEY_DE
                CJNE        R3,#01H,OFF_DA0
                RET
OFF_DA0:
                LCALL       KEY_NUM
                CJNE        R0,#0BH,OFF_DA1             ;KEY PREVIEW
                AJMP        OUT_ON
                RET
OFF_DA1:
                CJNE        R0,#0CH,OFF_DA2             ;CANCLE    EXIT
                RET
OFF_DA2:
                CJNE        R0,#0DH,OFF_DA3             ;KEY SAVE
                MOV         BUFDAY,#0FFH
                MOV         A,#2
                MOV         B,#3
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '    ',0
                AJMP        OFF_TIME
OFF_DA3:
                CJNE        R0,#0EH,OFF_DA4             ;CLEAR ONE PROGRAM
                ACALL       CLRPRO
                AJMP        OUT_OFF
OFF_DA4:        CJNE        R7,#0FH,OFF_DA5
                AJMP        OUT_OFF
OFF_DA5:
                CJNE        R0,#0FFH,OFF_DA6        ;NOT PRESS KEY NUMBER
                AJMP        OUT_OFF
OFF_DA6:
                CJNE        R0,#00H,OFF_DA7             ;DON'T CARE KEY NUMBER 0
                AJMP        OUT_OFF
OFF_DA7:
                MOV         A,BUF                   ;DISPLAY DAY ON LCD
                LCALL       LO
                ANL         A,#0FH
                MOV         BUFDAY,A
                MOV         A,STATUS
                MOV         B,#3
                LCALL         PLACECUR4
                MOV         A,R0
                ACALL       FIND_DAY
OFF_TIME:
                ACALL       WE_HOUR
                CJNE        R1,#0FH,OFF_TIME1
                AJMP        OUT_OFF
OFF_TIME1:
                CJNE        A,#0FFH,OFF_TIME2
                RET
OFF_TIME2:
                MOV         A,#2
                MOV         B,#15
                LCALL       PLACECUR4
                LCALL       BLINK_CUR
                MOV         R1,#0
                LCALL       KEY_DE
                CJNE        R3,#01H,OFF_TIME3
                RET
```

```
OFF_TIME3:
                LCALL       KEY_NUM
                CJNE        R0,#0DH,OFF_TIME4       ;SAVE
                ACALL       SAVE_OUT
                AJMP        OUT_ON
OFF_TIME4:
                CJNE        R0,#0CH,OFF_TIME5       ;CANCLE
                AJMP        OUT_OFF
OFF_TIME5:
                CJNE        R0,#0BH,OFF_TIME6           ;KEY PREVIEW GO TO SET FUNCTION
OFF
                AJMP        OUT_ON
OFF_TIME6:
                CJNE        R0,#0EH,OFF_TIME7
                ACALL       CLRPRO                  ;CLEAR ONE PROGRAM
                AJMP        OUT_OFF
OFF_TIME7:
                CJNE        R7,#0FH,OFF_TIME8
                AJMP        OUT_OFF
OFF_TIME8:
                CJNE        R0,#0FFH,OFF_TIME9
OFF_TIME9:      AJMP        OUT_OFF
;*************************HIGH HOUR*********************************
WE_HOUR:
                MOV         R7,#0FFH
                MOV         R6,#7
                MOV         A,STATUS
                MOV         B,R6
                LCALL       PLACECUR4
                MOV         R1,#3
                LCALL       KEY_DE
                CJNE        R3,#01H,WE_HR0
                RET
WE_HR0:         MOV         BUF,BUFHOUR
                LCALL       KEY_NUM
                CJNE        R0,#0FFH,WE_HR1
                AJMP        WE_HOUR
WE_HR1:         CJNE        R7,#0FH,WE_HR2
                AJMP        WE_HOUR
WE_HR2:         CJNE        R0,#0CH,WE_HR3          ;CANCLE
                MOV         R1,#0FH
                RET
WE_HR3:         MOV         A,BUFHOUR
                LCALL       HI
                MOV         BUFHOUR,A
                LCALL       OUT_NUM
                CJNE        R0,#02H,WE_CHK_2
                MOV         R2,#02H
                AJMP        WE_HR_LOW
WE_CHK_2:       MOV         R2,#01H
;*************************LOW HOUR**********************************
WE_HR_LOW:
                INC         R6
WE_LW:          MOV         A,STATUS
                MOV         B,R6
                LCALL       PLACECUR4
WE_L0:
                CJNE        R2,#02H,WE_L1
                MOV         R1,#4
                AJMP        WE_L2
WE_L1:          MOV         R1,#10
WE_L2:          LCALL       KEY_DE
                CJNE        R3,#01H,WE_L3
                RET
WE_L3:          MOV         BUF,BUFHOUR
                LCALL       KEY_NUM
                CJNE        R0,#0FFH,WE_CH_L1
                AJMP        WE_LW
WE_CH_L1:       CJNE        R7,#0FH,WE_CHK_4
                MOV         R6,#7
                AJMP        WE_HR_LOW
WE_CHK_4:       CJNE        R0,#0CH,WE_CHK_5        ;CANCLE
                AJMP        WE_HOUR
```

```
WE_CHK_5:       MOV         A,BUFHOUR
                LCALL       LO
                MOV         BUFHOUR,A
                LCALL       OUT_NUM
;*************************HIGH MINUTE*************************************
WE_MIN:
                MOV         R6,#10
                MOV         A,STATUS
                MOV         B,R6
                LCALL       PLACECUR4
WE_MIN_HI:      MOV         R1,#6
                LCALL       KEY_DE
                CJNE        R3,#01H,WE_MIN_0
                RET
WE_MIN_0:       MOV         BUF,BUFMIN
                LCALL       KEY_NUM
                CJNE        R0,#0FFH,WE_MI
                AJMP        WE_MIN
WE_MI:          CJNE        R7,#0FH,WE_MI_1
                AJMP        WE_MIN
WE_MI_1:        CJNE        R0,#0CH,WE_MI_2             ;CANCLE
                MOV         R6,#7
                AJMP        WE_HR_LOW
WE_MI_2:        MOV         A,BUFMIN
                LCALL       HI
                MOV         BUFMIN,A
                LCALL       OUT_NUM
;*************************LOW MINUTE*************************************
WE_MIN_LOW:
                INC         R6
WE_MIN_LW:      MOV         A,STATUS
                MOV         B,R6
                LCALL       PLACECUR4
WE_MI_L0:
                MOV         R1,#10
WE_MI_L1:       LCALL       KEY_DE
                CJNE        R3,#01H,WE_MI_L2
                RET
WE_MI_L2:       MOV         BUF,BUFMIN
                LCALL       KEY_NUM
                CJNE        R0,#0FFH,WE_MI_L3
                AJMP        WE_MIN_LOW
WE_MI_L3:       CJNE        R0,#0CH,WE_MI_L4          ; CANCLE
                AJMP        WE_MIN
WE_MI_L4:       CJNE        R7,#0FH,WE_MI_L5
                MOV         R6,#10
                AJMP        WE_MIN_LOW
WE_MI_L5:       MOV         A,BUFMIN
                LCALL       LO
                MOV         BUFMIN,A
                LCALL       OUT_NUM
                RET
;*************************************************************
; SUB SAVES THE OUTPUT CHECKING
;*************************************************************
SAVE_OUT:       MOV         A,STATUS
                CJNE        A,#01H,SA_OUT
                MOV         R0,#BUFON
                ACALL       SAVE
                RET
SA_OUT:         CJNE        A,#02H,SAVE_OUT
                MOV         R0,#BUFOFF
                ACALL       SAVE
                RET
SAVE:
                MOV         R2,BUFORDER
SAVE1:          DJNZ        R2,SAVE2
                MOV         R2,BUFOUT
SAVE_S:         DJNZ        R2,SAVE_S1
                ACALL       SAV_OUT
                RET
SAVE_S1:        MOV         A,R0
                ADD         A,#3
```

```
                MOV         R0,A
                AJMP        SAVE_S
SAVE2:          MOV         A,R0
                ADD         A,#24
                MOV         R0,A
                AJMP        SAVE1
SAV_OUT:        MOV         R2,#3
                MOV         R1,#BUFDAY
SAVE3:          MOV         A,@R1
                MOV         @R0,A
SAVE_CHK:       DJNZ        R2,SAVE_CHK1
                RET
SAVE_CHK1:      INC         R1
                INC         R0
                AJMP        SAVE3
;*************************************************************
;  CLEAR OUTPUT FUNCTION
;*************************************************************
CLRPRO:
                PUSH        00H
                MOV         BUFDAY,#0FFH
                MOV         BUFHOUR,#0FFH
                MOV         BUFMIN,#0FFH
                ACALL       SAVE_OUT
                MOV         R0,STATUS
                MOV         A,STATUS
                MOV         B,#3
                LCALL       PLACECUR4
                CJNE        R0,#1,PRO
                LCALL       PRTLCD4
                DB          'Day hh:mm  On',0
                POP         00H
                RET
PRO:            LCALL       PRTLCD4
                DB          'Day hh:mm Off',0
                POP         00H
                RET
;*************************************************
;  FUNCTION  CLEAR OUTPUT
;*************************************************
SET_CLR:
                MOV         R4,#CLRDSP
               LCALL        WRLCDCOM4
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '1.Erase All      ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '2.One by one     ',0
S_CLR:          LCALL       KEY_DEE
                CJNE        R3,#01H,S_CLR_0
                RET
S_CLR_0:        CJNE        A,#00H,S_CLR_1          ;NUMBER KEY(1) CLEAR OUTPUT ALL
                ACALL       ERASE_ALL
                RET
S_CLR_1:        CJNE        A,#01H,S_CLR_2      ;NUMBERKEY(2) CLEAR OUTPUT ONE BY ONE
                ACALL       ERASE_ONE
                CJNE        A,#0FFH,SET_CLR
                RET
S_CLR_2:        CJNE        A,#0CH,S_CLR            ;NUMBER KEY(CANCLE)
                RET
;*********************************************
;  SUB. CLEAR OUTPUT ALL
;*********************************************
ERASE_ALL:
                PUSH        00H
                ACALL       SUB_DEL
                CJNE        A,#0FFH,E_ALL1
E_ALL:          RET
```

```
E_ALL1:
                CJNE        R0,#22H,E_ALL2          ;CANCLE
                POP         00H
                RET
E_ALL2:         CJNE        R0,#11H,ERASE_ALL       ;SET FUNCTION OUTPUT
                MOV         R0,#BUFOUT
                MOV         R2,#193
E_1:            DJNZ        R2,E_3
                MOV         R2,#8
                MOV         R0,#CHKOUT1
E_2:            MOV         @R0,#01H
                INC         R0
                DJNZ        R2,E_2
                MOV         P2,#00H
                POP         00H
                RET
E_3:            MOV         @R0,#0FFH
                INC         R0
                AJMP        E_1
;************************************************
;  SUB. CLEAR OUTPUT ONE BY ONE
;************************************************
ERASE_ONE:
                MOV         R4,#CLRDSP
               LCALL        WRLCDCOM4
                MOV         BUFOUT,#0FFH
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' Press Key(1-8) ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '  Delete= 1 Ent.',0
                MOV         BUFOUT,#01H
CLR_OUT:
                MOV         R6,#10
                MOV         A,#2
                MOV         B,R6
                LCALL       PLACECUR4
                LCALL       BLINK_CUR
CLR_OUT1:
                MOV         R7,#0FFH
                MOV         R1,#9
                LCALL       KEY_DEE
                CJNE        R3,#01H,CL_OUT1
                RET
CL_OUT1:
                LCALL       KEY_NUM
                CJNE        R0,#0CH,CL_OUT2         ;CANCLE
                MOV         A,#55H
                RET
CL_OUT2:        CJNE        R0,#0DH,CL_OUT3         ;SET FUNCTION OUTPUT
                ACALL       SUB_DEL
                CJNE        R0,#22H,CL0
                AJMP        ERASE_ONE
CL0:            CJNE        R0,#11H,ERASE_ONE
                MOV         R0,ONOFF
                SETB        ONOFF.4
                LCALL       ON_OFF
                MOV         ONOFF,R0
                LCALL       CLEAR
                CJNE        A,#0FFH,ERASE_ONE
                RET
CL_OUT3:        CJNE        R7,#0FH,CL_OUT4
                AJMP        CLR_OUT1
CL_OUT4:        CJNE        R0,#0FFH,CL_OUT5
                AJMP        CLR_OUT1
CL_OUT5:        CJNE        R0,#00H,CL_OUT6
                AJMP        CLR_OUT1
CL_OUT6:        MOV         A,R0
```

```
                ANL         A,#0FH
                MOV         BUFOUT,A
                LCALL       DIS
                LJMP        CLR_OUT
; ************************************************************
; SUB DELETE YES NO FUNCTION
; ************************************************************
SUB_DEL:
                MOV         R4,#CLRDSP
                LCALL       WRLCDCOM4
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' Do you delete? ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          'Yes=Ent. No=Clr.',0
SU_DEL1:        MOV         R0,#0FH
                MOV         R1,#0
                LCALL       KEY_DE
                CJNE        R3,#01H,S_DEL1
                RET
S_DEL1:
                LCALL       KEY_NUM
                CJNE        R0,#0CH,S_DEL2        ;CANCLE
                MOV         R0,#22H
                RET
S_DEL2:         CJNE        R0,#0DH,SU_DEL1       ;SET FUNCTION OUTPUT
                MOV         R0,#11H
                RET
; ************************************************************
; SUB SETS THE DS1307 OSCILLATOR
; ************************************************************
OSC_CONTROL:
                ACALL       SEND_START      ; GENERATE START CONDITION
                MOV         A,#DS1307W      ; 1101 0000 ADDRESS + WRITE-BIT
                ACALL       SEND_BYTE       ; SEND BYTE TO 1307
                MOV         A,#00H          ; ADDRESS BYTE TO REGISTER 00H
                ACALL       SEND_BYTE       ; ECONDS REGISTER, ALWAYS LEAVE
                SETB        LASTREAD        ; REG 00H-BIT #7 = 0 (LOW)
                ACALL       SEND_STOP       ; IF REG 00H-BIT #7 = 1 CLOCK
                ACALL       SEND_START      ; OSCILLATOR IS OFF.
                MOV         A,#DS1307R      ; 1101 0001 ADDRESS + READ-BIT
                ACALL       SEND_BYTE
                ACALL       READ_BYTE       ; READ A BYTE FROM THE 1307
                CLR         ACC.7           ; CLEAR REG 00H-BIT #7 TO ENABLE
OSC_SET:                                    ; OSCILLATOR.
                PUSH        ACC             ; SAVE ON STACK
                ACALL       SEND_STOP
                ACALL       SEND_START
                MOV         A,#DS1307W      ; SETUP TO WRITE
                ACALL       SEND_BYTE
                MOV         A,#00H          ; REGISTER 00H ADDRESS
                ACALL       SEND_BYTE
                POP         ACC             ; GET DATA TO START OSCILLATOR
                ACALL       SEND_BYTE       ; SEND IT
                ACALL       SEND_STOP
                RET
; ************************************************************
; SUB SETS THE CLOCK
; ************************************************************
SET_CLOCK:
                ACALL       SEND_START      ; SEND 2-WIRE START CONDITION
                MOV         A,#DS1307W      ; SEND 1307 WRITE COMMAND
                ACALL       SEND_BYTE
                MOV         A,#00H          ; SET DATA POINTER TO REGISTER
                                            ; 00H ON THE DS1307
                ACALL       SEND_BYTE
                MOV         R1,#24H         ; START WITH SECONDS REGISTER
                                            ; IN INTERNAL RAM
```

```
SEND_LOOP:
                MOV         A,@R1               ; MOVE FIRST BYTE OF DATA TO ACC
                ACALL       SEND_BYTE           ; SEND DATA ON 2-WIRE BUT
                INC         R1                  ; LOOP UNTIL CLOCK DATA SENT
                CJNE        R1,#27H,SEND_LOOP
                ACALL       SEND_STOP           ; SEND 2-WIRE STOP CONDITION
                RET
; ***************************************************
SET_CLKDAT:
                ACALL       SEND_START          ; SEND 2-WIRE START CONDITION
                MOV         A,#DS1307W          ; SEND 1307 WRITE COMMAND
                ACALL       SEND_BYTE
                MOV         A,#03H              ; SET DATA POINTER TO REGISTER
                                                ; 00H ON THE DS1307
                ACALL       SEND_BYTE
                MOV         R1,#27H             ; START WITH SECONDS REGISTER
                                                ; IN INTERNAL RAM
SEND_LOOP1:
                MOV         A,@R1               ; MOVE FIRST BYTE OF DATA TO ACC
                ACALL       SEND_BYTE           ; SEND DATA ON 2-WIRE BUT
                INC         R1                  ; LOOP UNTIL CLOCK DATA SENT
                CJNE        R1,#2BH,SEND_LOOP1
                ACALL       SEND_STOP           ; SEND 2-WIRE STOP CONDITION
                RET
; *********************************************************
; SUB SENDS START CONDITION
; *********************************************************
SEND_START:
                SETB        _2W_BUSY            ; INDICATE THAT 2-WIRE
                CLR         ACK                 ; OPERATION IS IN PROGRESS
                CLR         BUS_FLT             ; CLEAR STATUS FLAGS
                JNB         SCL,FAULT
                JNB         SDA,FAULT
                SETB        SDA                 ; BEGIN START CODITION
                SETB        SCL
                JNB         SCL,$
                CLR         SDA
                ACALL       DELAY
                CLR         SCL
                RET
FAULT:
                SETB        BUS_FLT
                RET
; *********************************************************
; SUB SENDS STOP CONDITION
; *********************************************************
SEND_STOP:
                CLR         SDA
                SETB        SCL
                JNB         SCL,$
                SETB        SDA
                CLR         _2W_BUSY
                RET
; *********************************************************
; THIS SUB READS ONE BYTE OF DATA FROM THE DS1307
; *********************************************************
READ_BYTE:
                MOV         BITCNT,#08H         ; SET COUNTER FOR 8-BITS DATA
                MOV         A,#00H
                SETB        SDA                 ; SET SDA HIGH TO ENSURE LINE
                                                ; FREE
READ_BITS:
                SETB        SCL                 ; TRANSITION SCL LOW-TO-HIGH
                JNB         SCL,$
                MOV         C,SDA               ; MOVE DATA BIT INTO CARRY
                RLC         A                   ; ROTATE CARRY-BIT INTO ACC.0
                CLR         SCL                 ; TRANSITION SCL HIGH-TO-LOW
                DJNZ        BITCNT,READ_BITS    ; LOOP FOR 8-BITS
                JB          LASTREAD,ACKN       ; CHECK TO SEE IF THIS IS
                                                ; THE LAST READ

                CLR         SDA                 ; IF NOT LAST READ SEND ACK-BIT
                SETB        SCL                 ; PULSE SCL TO TRANSMIT ACKNOWLEDGE
```

```
ACKN:           JNB         SCL,$
                CLR         SCL                 ; OR NOT ACKNOWLEDGE BIT
                RET
; ***********************************************************
; THIS SUB SENDS 1 BYTE OF DATA TO THE DS1307
; CALL THIS FOR EACH REGISTER SECONDS TO YEAR
; ACC MUST CONTAIN DATA TO BE SENT TO CLOCK
; ***********************************************************
SEND_BYTE:
                MOV         BITCNT,#08H         ; SET COUNTER FOR 8-BITS
SB_LOOP:
                JNB         ACC.7,NOTONE        ; CHECK TO SEE IF BIT-7 OF
                SETB        SDA                 ; ACC IS A 1, AND SET SDA HIGH
                JMP         ONE
NOTONE:
                CLR         SDA                 ; CLR SDA LOW
ONE:
                SETB        SCL                 ; TRANSITION SCL LOW-TO-HIGH
                JNB         SCL,$
                RL          A                   ; ROTATE ACC LEFT 1-BIT
                CLR         SCL                 ; TRANSITION SCL LOW-TO-HIGH
                DJNZ        BITCNT,SB_LOOP      ; LOOP FOR 8-BITS
                SETB        SDA                 ; SET SDA HIGH TO LOOK FOR
                SETB        SCL                 ; ACKNOWLEDGE PULSE
                JNB         SCL,$
                CLR         ACK
                JNB         SDA,SB_EX           ; CHECK FOR ACK OR NOT ACK
                SETB        ACK                 ; SET ACKNOWLEDGE FLAG FOR
                                                ; NOT ACK
SB_EX:
                ACALL       DELAY               ; DELAY FOR AN OPERATION
                CLR         SCL                 ; TRANSITION SCL HIGH-TO-LOW
                ACALL       DELAY               ; DELAY FOR AN OPERATION
                RET
; ***********************************************************
; SUB DELAYS THE BUS
; ***********************************************************
DELAY:
                NOP                             ; DELAY FOR BUS TIMING
                RET
; ***********************************************************
; SUB READS THE CLOCK AND WRITES IT TO THE SCRATCHPAD MEMORY
; ON RETURN FROM HERE DATE & TIME DATA WILL BE STORED IN THE
; DATE & TIME REGISTERS FROM 24H (SECS) TO 2AH (YEAR)
; ALARM SETTINGS IN REGISTERS 2CH(HRS) AND 2DH(MINUTES).
; ***********************************************************
READ_CLOCK:
                MOV         R1,#24H             ; SECONDS STORAGE LOCATION
                MOV         BYTECNT,#00H
                CLR         LASTREAD
                ACALL       SEND_START
                MOV         A,#DS1307W
                ACALL       SEND_BYTE
                MOV         A,#00H
                ACALL       SEND_BYTE
                ACALL       SEND_STOP
                ACALL       SEND_START
                MOV         A,#DS1307R
                ACALL       SEND_BYTE
READ_LOOP:
                MOV         A,BYTECNT
                CJNE        A,#09H,NOT_LAST
                SETB        LASTREAD
NOT_LAST:
                ACALL       READ_BYTE
                MOV         @R1,A
                MOV         A,BYTECNT
                CJNE        A,#00H,NOT_FIRST
                MOV         A,@R1
                CLR         ACC.7               ; ENSURE OSC BIT=0 (ENABLED)
                MOV         @R1,A
NOT_FIRST:
                INC         R1
```

```
                INC         BYTECNT
                MOV         A,BYTECNT
                CJNE        A,#0AH,READ_LOOP
                ACALL       SEND_STOP
                RET
; ************************************************************
; WRITE NUMBER ON LCD DISPLAY
; ************************************************************
WRITE_BCD:
                PUSH        ACC
                PUSH        00H
                MOV         A,R0
                SWAP        A
                ANL         A,#0FH
                ACALL       FIND            ; LOOK IT UP & THEN PRINT IT.
                MOV         A,R0
                ANL         A,#0FH
                ACALL       FIND
                POP         00H
                POP         ACC
                RET
FIND:
                SJMP        LOOK_UP
                RET
LOOK_UP:
                CJNE        A,#00000000B,M1
                LCALL       PRTLCD4
                DB          '0',0
                RET
M1:
                CJNE        A,#00000001B,M2
                LCALL       PRTLCD4
                DB          '1',0
                RET
M2:
                CJNE        A,#00000010B,M3
                LCALL       PRTLCD4
                DB          '2',0
                RET
M3:
                CJNE        A,#00000011B,M4
                LCALL       PRTLCD4
                DB          '3',0
                RET
M4:
                CJNE        A,#00000100B,M5
                LCALL       PRTLCD4
                DB          '4',0
                RET
M5:
                CJNE        A,#00000101B,M6
                LCALL       PRTLCD4
                DB          '5',0
                RET
M6:
                CJNE        A,#00000110B,M7
                LCALL       PRTLCD4
                DB          '6',0
                RET
M7:
                CJNE        A,#00000111B,M8
                LCALL       PRTLCD4
                DB          '7',0
                RET
M8:
                CJNE        A,#00001000B,M9
                LCALL       PRTLCD4
                DB          '8',0
                RET
M9:
                CJNE        A,#00001001B,M10
                LCALL       PRTLCD4
                DB          '9',0
```

```
M10:            RET
; ********************************************************
; WRITE DAY ON LCD DISPLAY
; ********************************************************
WRITE_DAY:
                PUSH        ACC                 ; DECODE HIGH DIGIT
                PUSH        00H
                MOV         A,R0
                ANL          A,#0FH
                ACALL        FIND_DAY           ; LOOK IT UP & THEN PRINT IT.
                POP         00H
                POP          ACC
                RET
FIND_DAY:
                SJMP         GET_DAY
                RET
GET_DAY:
                CJNE         A,#00000001B,D1
                LCALL        PRTLCD4
                DB           'Sun',0
                RET
D1:
                CJNE         A,#00000010B,D2
                LCALL        PRTLCD4
                DB           'Mon',0
                RET
D2:
                CJNE         A,#00000011B,D3
                LCALL        PRTLCD4
                DB           'Tue',0
                RET
D3:
                CJNE         A,#00000100B,D4
                LCALL        PRTLCD4
                DB           'Wed',0
                RET
D4:
                CJNE         A,#00000101B,D5
                LCALL        PRTLCD4
                DB           'Thu',0
                RET
D5:
                CJNE         A,#00000110B,D6
                LCALL        PRTLCD4
                DB           'Fri',0
                RET
D6:
                CJNE         A,#00000111B,D7
                LCALL        PRTLCD4
                DB           'Sat',0
D7:             RET
; ********************************************************
; SOFTWARE VERSION OF THE POWER ON RESET
; ********************************************************
RESETLCD4:
                CLR          LCD_RS             ; LCD REGISTER SELECT LINE
                CLR          LCD_RW             ; READ / WRITE LINE
                CLR          LCD_E              ; ENABLE LINE
                SETB        LCD_E               ; START ENABLE PULSE
                CLR          LCD_E              ; END ENABLE PULSE
                MOV          R4, #CONFIG        ; FUNCTION SET
                ACALL        WRLCDCOM4
                MOV          R4, #08H           ; DISPLAY OFF
                ACALL        WRLCDCOM4
                MOV          R4, #1              ; CLEAR DISPLAY, HOME CURSOR
                ACALL        WRLCDCOM4
                MOV          R4, #ENTRYMODE     ; SET ENTRY MODE
                ACALL        WRLCDCOM4
INITLCD4:
                CLR          LCD_RS             ; LCD REGISTER SELECT LINE
                CLR          LCD_RW             ; READ / WRITE LINE
                CLR          LCD_E              ; ENABLE LINE
                MOV          R4, #CONFIG        ; FUNCTION SET - DATA BITS,
```

```
                                        ; LINES, FONTS
        ACALL   WRLCDCOM4
        MOV     R4, #ONDSP              ; DISPLAY ON
        ACALL   WRLCDCOM4
        MOV     R4, #ENTRYMODE          ; SET ENTRY MODE
        ACALL   WRLCDCOM4               ; INCREMENT CURSOR RIGHT, NO SHIFT
        MOV     R4, #CLRDSP             ; CLEAR DISPLAY, HOME CURSOR
        ACALL   WRLCDCOM4
        RET
; ********************************************************
; SUB WRITES A COMMAND WORD TO THE LCD
; COMMAND MUST BE PLACED IN R4 BY CALLING PROGRAM
; ********************************************************
WRLCDCOM4:
        CLR     LCD_E
        CLR     LCD_RS                  ; SELECT SEND COMMAND
        CLR     LCD_RW                  ; SELECT WRITE OPERATION
        PUSH    ACC                     ; SAVE ACCUMULATOR
        MOV     A, R4                   ; PUT DATA BYTE IN ACC
        MOV     C, ACC.4                ; LOAD HIGH NIBBLE ON DATA BUS
        MOV     LCD_DB4, C              ; ONE BIT AT A TIME USING...
        MOV     C, ACC.5                ; BIT MOVE OPERATOINS
        MOV     LCD_DB5, C
        MOV     C, ACC.6
        MOV     LCD_DB6, C
        MOV     C, ACC.7
        MOV     LCD_DB7, C
        SETB    LCD_E                   ; PULSE THE ENABLE LINE
        CLR     LCD_E
        MOV     A,R4
        MOV     C, ACC.0                ; SIMILARLY, LOAD LOW NIBBLE
        MOV     LCD_DB4, C
        MOV     C, ACC.1
        MOV     LCD_DB5, C
        MOV     C, ACC.2
        MOV     LCD_DB6, C
        MOV     C, ACC.3
        MOV     LCD_DB7, C
        LCALL   PULSEEWAIT4             ; PULSE THE ENABLE LINE...
                                        ; AND WAIT FOR BUSY FLAG TO CLEAR
        POP     ACC
        RET
; ********************************************************
; SUB TO WRITE A DATA WORD TO THE LCD
; DATA MUST BE PLACED IN R4 BY CALLING PROGRAM
; ********************************************************
WRLCDDATA:
        CLR     LCD_E
        SETB    LCD_RS                  ; SELECT SEND DATA
        CLR     LCD_RW                  ; SELECT WRITE OPERATION
        PUSH    ACC                     ; SAVE ACCUMULATOR
        MOV     A, R4                   ; PUT DATA BYTE IN ACC
        MOV     C, ACC.4                ; LOAD HIGH NIBBLE ON DATA BUS
        MOV     LCD_DB4, C              ; ONE BIT AT A TIME USING...
        MOV     C, ACC.5                ; BIT MOVE OPERATOINS
        MOV     LCD_DB5, C
        MOV     C, ACC.6
        MOV     LCD_DB6, C
        MOV     C, ACC.7
        MOV     LCD_DB7, C
        SETB    LCD_E                   ; PULSE THE ENABLE LINE
        CLR     LCD_E
        MOV     A,R4
        MOV     C, ACC.0                ; SIMILARLY, LOAD LOW NIBBLE
        MOV     LCD_DB4, C
        MOV     C, ACC.1
        MOV     LCD_DB5, C
        MOV     C, ACC.2
        MOV     LCD_DB6, C
        MOV     C, ACC.3
        MOV     LCD_DB7, C
        ACALL   PULSEEWAIT4             ; PULSE THE ENABLE LINE...
                                        ; AND WAIT FOR BUSY FLAG TO CLEAR
```

```
                POP         ACC
                RET
; ********************************************************
; GENERATES A POSITIVE PULSE ON THE LCD ENABLE LINE.
; WAITS FOR THE BUSY FLAG TO CLEAR BEFORE RETURNING.
; ********************************************************
PULSEEWAIT4:
                CLR         LCD_E
                SETB        LCD_E                PULSE THE ENABLE LINE
                CLR         LCD_E
                MOV         LCD_DATA, #0FFH      ; PREPARE PORT FOR INPUT
                SETB        LCD_RW               ; PREPARE R/W FOR READ OPERATION
                PUSH        ACC                  ; SAVE ACCUMULATOR CONTENTS
PEW:
                SETB        LCD_E                START THE ENABLE PULSE
                MOV         A, LCD_DATA          ; READ STATUS NIBBLE
                CLR         LCD_E                END ENABLE PULSE
                SETB        LCD_E                PRETEND READING STATUS LOW NIBBLE
                CLR         LCD_E
                JB          ACC.7, PEW           ; LOOP WHILE BUSY FLAG IS SET
                POP         ACC                  ; RESTORE ACCUMULATOR
                RET
; ********************************************************
; SUB TAKES THE STRING IMMEDIATELY FOLLOWING THE CALL AND
; DISPLAYS ON THE LCD. STRING MUST BE TERMINATED WITH A
; NULL (0).
; ********************************************************
PRTLCD4:
                POP         DPH                  ; POP RETURN ADDRESS INTO DPTR
                POP         DPL
PRTNEXT:
                CLR         A                    SET OFFSET = 0
                MOVC        A, @A+DPTR           ; GET CHR FROM CODE MEMORY
                CJNE        A, #0, CHROK         IF CHR = 0 THEN RETURN
                SJMP        RETPRTLCD
CHROK:
                MOV         R4, A
                LCALL       WRLCDDATA            SEND CHARACTER
                INC         DPTR                 POINT AT NEXT CHARACTER
                LJMP        PRTNEXT              ; LOOP TILL END OF STRING
RETPRTLCD:
                MOV         A,#1H                POINT TO INSTRUCTION AFTER STRING
                JMP         @A+DPTR              ; RETURN WITH A JUMP INSTRUCTION
; ********************************************************
; SUB SETS THE CURSOR POSITION.
; ********************************************************
PLACECUR4:
                DEC         ACC                  ; ACC=0 FOR LINE=1
                JNZ         LINE2                IF ACC=0 THEN FIRST LINE
                MOV         A, B
                ADD         A, #080H             CONSTRUCT CONTROL WORD FOR LINE 1
                SJMP        SETCUR
LINE2:
                MOV         A, B
                ADD         A, #0C0H             CONSTRUCT CONTROL WORD FOR LINE 2
SETCUR:
                MOV         R4, A                PLACE CONTROL WORD
                LCALL       WRLCDCOM4            ISSUE COMMAND
                RET
;************************************************************
;  CLEAR OUTPUT FUNCTION
;************************************************************
CLEAR:
                MOV         R6,#BUFON
                MOV         R7,#BUFOFF
                MOV         R4,#5
                MOV         R2,BUFOUT
                MOV         R0,07H
                MOV         R1,06H
DEL:            DJNZ        R2,DEL_1             ; POINTER
DEL1:           DJNZ        R4,DEL0
                RET
DEL0:           MOV         R3,#4
```

```
                MOV         R0,07H
                MOV         R1,06H
DEL_0:          DJNZ        R3,DEL_2
                MOV         A,R6
                ADD         A,#24
                MOV         R6,A
                MOV         R1,A
                MOV         A,R7
                ADD         A,#24
                MOV         R7,A
                MOV         R0,A
                LJMP        DEL1
DEL_2:          MOV         @R1,#0FFH
                MOV         @R0,#0FFH
                INC         R1
                INC         R0
                LJMP        DEL_0
DEL_1:
                MOV         A,R1
                ADD         A,#3
                MOV         R1,A
                MOV         R6,A
                MOV         A,R0
                ADD         A,#3
                MOV         R0,A
                MOV         R7,A
                LJMP        DEL
;*************************************************************
;   ON AND OFF OUTPUT 16 OUTLET  (USE P2&P0)
;*************************************************************
ON_OFF:
                MOV         A,BUFOUT
                CJNE        A,#1,ON_OFF_0
                JB          ONOFF.4,OFF
                SETB        P2.0                        ;ON OUTPUT 1
                RET
OFF:            CLR         P2.0                        ;OFF OUTPUT 1
                RET
ON_OFF_0:       CJNE        A,#2,ON_OFF_1
                JB          ONOFF.4,OFF_0
                SETB        P2.1                        ;ON OUTPUT 2
                RET
OFF_0:          CLR         P2.1                        ;OFF OUTPUT 2
                RET
ON_OFF_1:       CJNE        A,#3,ON_OFF_2
                JB          ONOFF.4,OFF_1
                SETB        P2.2                        ;ON OUTPUT 3
                RET
OFF_1:          CLR         P2.2                        ;OFF OUTPUT 3
                RET
ON_OFF_2:       CJNE        A,#4,ON_OFF_3
                JB          ONOFF.4,OFF_2
                SETB        P2.3                        ;ON OUTPUT 4
                RET
OFF_2:          CLR         P2.3                        ;OFF OUTPUT 4
                RET
ON_OFF_3:       CJNE        A,#5,ON_OFF_4
                JB          ONOFF.4,OFF_3
                SETB        P2.4                        ;ON OUTPUT 5
                RET
OFF_3:          CLR         P2.4                        ;OFF OUTPUT 5
                RET
ON_OFF_4:       CJNE        A,#6,ON_OFF_5
                JB          ONOFF.4,OFF_4
                SETB        P2.5                        ;ON OUTPUT 6
                RET
OFF_4:          CLR         P2.5                        ;OFF OUTPUT 6
                RET
ON_OFF_5:       CJNE        A,#7,ON_OFF_6
                JB          ONOFF.4,OFF_5
                SETB        P2.6                        ;ON OUTPUT 7
                RET
OFF_5:          CLR         P2.6                        ;OFF OUTPUT 7
```

```
                RET
ON_OFF_6:       CJNE        A,#8,ON_OFF_7
                JB          ONOFF.4,OFF_6
                SETB        P2.7                        ;ON OUTPUT 8
                RET
OFF_6:          CLR         P2.7                        ;OFF OUTPUT 8
ON_OFF_7:       RET
;*************************************************************************
;  POINTER CHECK DISPLAY OUTPUT
;*************************************************************************
OUT_FUNDIS:
                MOV         R4,#CLRDSP
                LCALL       WRLCDCOM4
                MOV         A,#1
                MOV         3,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          'Load  . Program ',0
                MOV         A,#1
                MOV         B,#5
                MOV         R0,BUFOUT
                LCALL       PRINTNUM_1
                MOV         BUFORDER,#1
                MOV         R6,#04H
OUT_DIS:
                MOV         R1,#BUFON
                LCALL       CHK_STATUSDIS
                MOV         R2,BUFMIN
                CJNE        R2,#0FFH,FUNDI0
                MOV         R1,#BUFOFF
                LCALL       CHK_STATUSDIS
                MOV         R2,BUFMIN
                CJNE        R2,#0FFH,FUNDI1
                INC         BUFORDER
                DJNZ        R6,OUT_DIS
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          'Not Use Program.',0
OUT_DIS0:       LCALL       KEY_DEE
                CJNE        R3,#01H,OUT_DIS1
                RET
OUT_DIS1:       CJNE        R3,#0CH,OUT_DIS0            ;NUMBER KEY(CANCLE)
                RET
FUNDI0:         AJMP        OUT_DISON
FUNDI1:         AJMP        OUT_DISOFF
OUT_DISON:      MOV         A,BUFORDER
                MOV         B,#2
                MUL         AB
                DEC         A
                MOV         BUFOR,A
                MOV         R1,#BUFON
                LCALL       CHK_STATUSDIS
                MOV         R2,BUFMIN
                CJNE        R2,#0FFH,FUNDIS_0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' . Not Set.    On',0
                ACALL       PRT_FUNDIS1
                AJMP        FUNDIS_1
FUNDIS_0:       MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' .         :     On',0
                ACALL       PRT_FUNDIS1
                ACALL       PRT_FUNDIS
FUNDIS_1:
                LCALL       KEY_DEE
                CJNE        R3,#01H,FUNDIS_2
```

```
                RET
FUNDIS_2:       CJNE        A,#07H,FUNDIS_3                 ;NUMBER KEY(NEXT)
                AJMP        OUT_DISOFF
FUNDIS_3:       CJNE        A,#03H,FUNDIS_4                 ;NUMBER KEY(PREVIEW)
                DEC         BUFORDER
                MOV         A,BUFORDER
                CJNE        A,#0,OUT_DISOFF
                INC         BUFORDER
                AJMP        FUNDIS_1
FUNDIS_4:       CJNE        A,#0CH,FUNDIS_1                 ;NUMBER KEY(CANCLE)
                RET
OUT_DISOFF:     MOV         A,BUFORDER
                MOV         B,#2
                MUL         AB
                MOV         BUFOR,A
                MOV         R1,#BUFOFF
                ACALL       CHK_STATUSDIS
                MOV         R2,BUFMIN
                CJNE        R2,#0FFH,FUNDIS_5
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' . Not Set.  Off',0
                ACALL       PRT_FUNDIS1
                AJMP        FUNDIS_6
FUNDIS_5:       MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          ' .       :   Off',0
                ACALL       PRT_FUNDIS1
                ACALL       PRT_FUNDIS
FUNDIS_6:
                LCALL       KEY_DEE
                CJNE        R3,#01H,FUNDIS_7
                RET
FUNDIS_7:       CJNE        A,#07H,FUNDIS_8                 ;NUMBER KEY(NEXT)
                INC         BUFORDER
                MOV         A,BUFORDER
                CJNE        A,#5,FUNDI_1
                DEC         BUFORDER
                AJMP        FUNDIS_6
FUNDIS_8:       CJNE        A,#03H,FUNDIS_9                 ;NUMBER KEY(PREVIEW)
FUNDI_1:        LJMP        OUT_DISON
FUNDIS_9:       CJNE        A,#0CH,FUNDIS_6                 ;NUMBER KEY(CANCLE)
                RET
;*******************************************************************************
;    SUB. PRINT ON DISPLAY
;*******************************************************************************
PRT_FUNDIS:
                MOV         A,#2
                MOV         B,#3
                MOV         R0,BUFDAY
                LCALL       PRINTDAY
                MOV         A,#2
                MOV         B,#7
                MOV         R0,BUFHOUR
                LCALL       PRINTNUM
                MOV         A,#2
                MOV         B,#10
                MOV         R0,BUFMIN
                LCALL       PRINTNUM
                LCALL       OFF_CUR
                RET
PRT_FUNDIS1:
                MOV         A,#2
                MOV         B,#0
                MOV         R0,BUFOR
                LCALL       PRINTNUM_1
                LCALL       OFF_CUR
                RET
```

```
;*****************************************************************************
;    CHECK STATUS OF BUFFER OUTPUT
;*****************************************************************************
CHK_STATUSDIS:
              MOV       R2,BUFORDER
CHK_STDIS:    DJNZ      R2,CHK_STDIS1
              MOV       R2,BUFOUT
CHK_SDIS:     DJNZ      R2,CHK_SDIS1
              ACALL     KEEP_OUTDIS
              RET
CHK_SDIS1:
              MOV       A,R1
              ADD       A,#3
              MOV       R1,A
              AJMP      CHK_SDIS
CHK_STDIS1:   MOV       A,R1
              ADD       A,#24
              MOV       R1,A
              AJMP      CHK_STDIS
KEEP_OUTDIS:  MOV       R2,#3
              MOV       R0,#BUFDAY
KEEPDIS:      MOV       A,@R1
              MOV       @R0,A
KEEP_CHKDIS:  DJNZ      R2,KEEP_CHKDIS1
              RET
KEEP_CHKDIS1: INC       R1
              INC       R0
              AJMP      KEEPDIS
;*****************************************************************************
;   DISPLAY OUTPUT
;*****************************************************************************
SET_HELP:
              ACALL     SET_OUTDIS
              RET
;***********************************************
;  FUNCTION SET DISPLAY OUTPUT
;***********************************************
SET_OUTDIS:
              MOV       R4,#CLRDSP
              LCALL     WRLCDCOM4
              MOV       BUFOUT,#0FFH
              MOV       A,#1
              MOV       B,#0
              LCALL     PLACECUR4
              LCALL     PRTLCD4
              DB        ' Press Key(1-8) ',0
              MOV       A,#2
              MOV       B,#0
              LCALL     PLACECUR4
              LCALL     PRTLCD4
              DB        '  Output= 1 Ent.',0
              MOV       R7,#0FFH
              MOV       BUFOUT,#01H
SETOUTDIS:
              MOV       R6,#10
              MOV       A,#2
              MOV       B,R6
              LCALL     PLACECUR4
              LCALL     BLINK_CUR
SETOUTDIS1:
              MOV       R1,#9
              LCALL     KEY_DEE
              CJNE      R3,#01H,ST_OUTDIS1
              RET
ST_OUTDIS1:
              LCALL     KEY_NUM
              CJNE      R0,#0CH,ST_OUTDIS2          ;CANCLE
SETOUTDIS2:   MOV       A,#55H
              MOV       BUFOUT,#0FFH
              RET
ST_OUTDIS2:   CJNE      R0,#0DH,ST_OUTDIS3          ;FUNCTION SET DISPLAY OUTPUT
              LCALL     OUT_FUNDIS
              CJNE      A,#0FFH,SET_OUTDIS
```

```
                RET
ST_OUTDIS3:     CJNE        R7,#0FH,ST_OUTDIS4
                AJMP        SETOUTDIS1
ST_OUTDIS4:     CJNE        R0,#0FFH,ST_OUTDIS5
                AJMP        SETOUTDIS1
ST_OUTDIS5:     CJNE        R0,#00H,ST_OUTDIS6
                AJMP        SETOUTDIS1
ST_OUTDIS6:     MOV         A,R0
                ANL         A,#07H
                MOV         BUFOUT,A
                LCALL       DIS
                AJMP        SETOUTDIS
;*********************************************
;  CHECK OUTPUT FUNCTION
;*********************************************
KEY_DEE:
                PUSH        00H
                PUSH        01H
                PUSH        05H
                PUSH        06H
                PUSH        07H
                PUSH        BUFOUT
                PUSH        BUFORDER
                MOV         R5,#7
                MOV         37H,#0FFH
DELOAD:         JB          KEY_DA,GETDE
                LCALL       READ_CLOCK
                MOV         R4,#9                   ;POINTER COUNTER LOAD
                MOV         R6,#5                   ;POINTER COUNTER PROGRAM
                MOV         BUFOUT,#1
                MOV         R0,#CHKOUT1
DMAIN_LOAD:
                DJNZ        R4,DSUB_LOAD0           ;COUNTER LOAD 1-8
                DJNZ        37H,DELOAD
                DJNZ        R5,DELOAD
                MOV         A,#0FFH
                MOV         R3,#01H
                POP         BUFORDER
                POP         BUFOUT
                POP         07H
                POP         06H
                POP         05H
                POP         01H
                POP         00H
                RET
GETDE:
                LCALL       BEEB_KEY
                JB          KEY_DA,$
                MOV         A,P1
                ANL         A,#0FH
                MOV         R3,#00H
                POP         BUFORDER
                POP         BUFOUT
                POP         07H
                POP         06H
                POP         05H
                POP         01H
                POP         00H
                RET
DSUB_LOAD0:     DJNZ        R6,DLOAD_0              ;COUNTER PROGRAM 1-8 OF LOAD
DSUB_LOAD:
                INC         BUFOUT
                MOV         A,BUFOUT
                MOV         R0,#CHKOUT1
DSUB0:          DJNZ        BUFOUT,DSUB
                AJMP        DSUB1
DSUB:           INC         R0
                AJMP        DSUB0
DSUB1:          MOV         BUFOUT,A
                MOV         R6,#5
                AJMP        DMAIN_LOAD
DLOAD_0:
                MOV         A,@R0
```

```
                MOV         ONOFF,A
                ANL         A,#0FH
                MOV         BUFORDER,A
                JB          ONOFF.4,DLOAD_OFF           ;CHECK STATE OF LOAD (ON AND OFF)
DLOAD_ON:
                MOV         R1,#BUFON
                ACALL       CHKS
                MOV         R1,BUFMIN
                CJNE        R1,#0FFH,DON1
DLOAD_ON1:
DLOAD_OFF:
                MOV         R1,#BUFOFF
                ACALL       CHKS
                MOV         R1,BUFMIN
                CJNE        R1,#0FFH,DOFF1
DLOAD_OFF1:
                MOV         A,BUFORDER
                CJNE        A,#4,DLOAD_1                ;CHECK END OF LOOP PROGRAM
                AJMP        DSUB_LOAD
DLOAD_1:
                INC         ONOFF
                MOV         A,CNOFF
                MOV         @R0,A
                AJMP        DSUB_LOAD0
DON1:
                ACALL       CHK_OUT
                CJNE        R3,#0,DEXIT_1               ;CHECK ON AND OFF LOAD
                CLR         ONOFF.4
                LCALL       ON_OFF
DEXIT_1:        AJMP        DLOAD_ON1
DOFF1:
                ACALL       CHK_OUT
                CJNE        R3,#0,DEXIT_2               ;CHECK ON AND OFF LOAD
                SETB        ONOFF.4
                LCALL       ON_OFF
DEXIT_2:        AJMP        DLOAD_OFF1                  ;NOT COMPLETE
;*******************************************
;  CHECK OUTPUT FUNCTION
;*******************************************
CHK_OUTPUT:
                PUSH        00H
                PUSH        01H
                PUSH        BUFOUT
                PUSH        BUFORDER
                LCALL       READ_CLOCK
                MOV         R7,#9                       ;POINTER COUNTER LOAD
                MOV         R6,#5                       ;POINTER COUNTER PROGRAM
                MOV         BUFOUT,#1
                MOV         R0,#CHKOUT1
MAIN_LOAD:      DJNZ        R7,SUB_LOAD0                ;COUNTER LOAD 1-8
                POP         BUFORDER
                POP         BUFOUT
                POP         01H
                POP         00H
                RET
SUB_LOAD0:      DJNZ        R6,LOAD_0                   ;COUNTER PROGRAM 1-8 OF LOAD
SUB_LOAD:
                INC         BUFOUT
                MOV         A,BUFOUT
                MOV         R0,#CHKOUT1
SUB0:           DJNZ        BUFOUT,SUB
                AJMP        SUB1
SUB:            INC         R0
                AJMP        SUB0
SUB1:           MOV         BUFOUT,A
                MOV         R6,#5
                AJMP        MAIN_LOAD
LOAD_0:
                MOV         A,@R0
                MOV         ONOFF,A
                ANL         A,#0FH
                MOV         BUFORDER,A
                JB          ONOFF.4,LOAD_OFF            ;CHECK STATE OF LOAD (ON AND OFF)
```

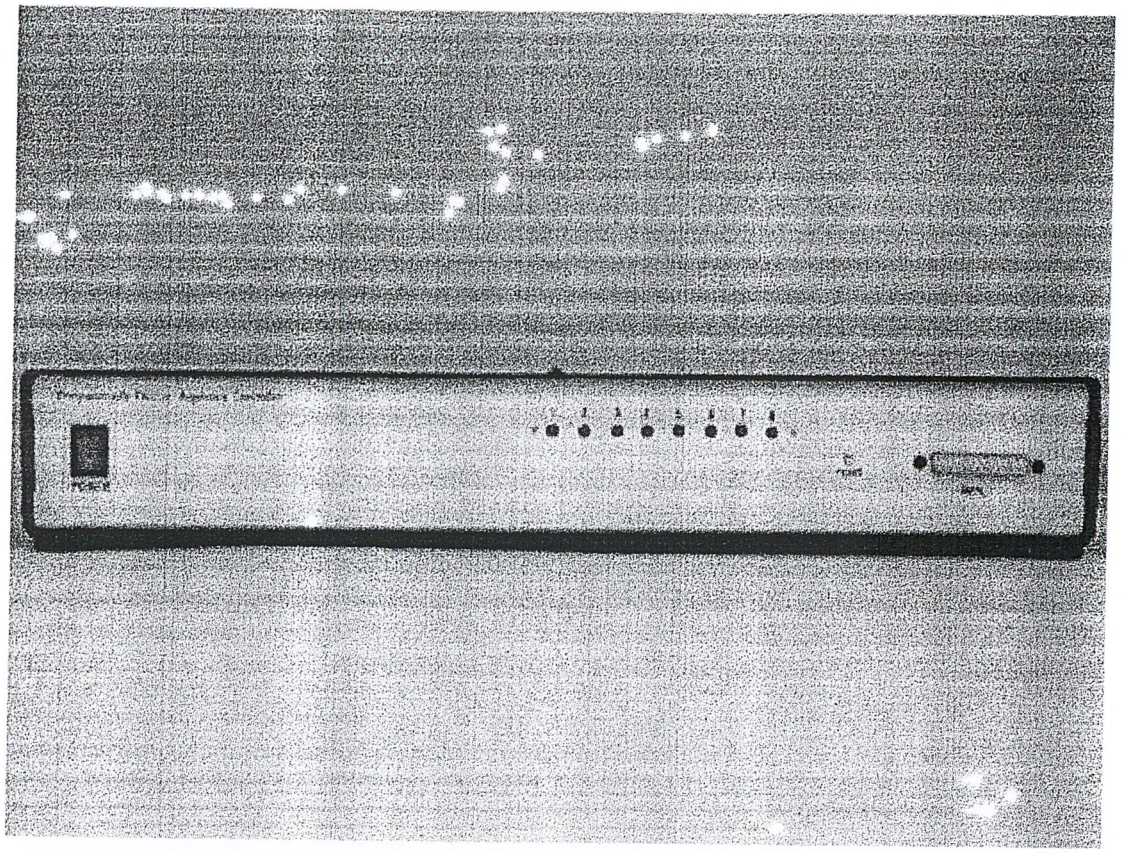
```
LOAD_ON:
                MOV         R1,#BUFON
                ACALL       CHKS
                MOV         R1,BUFMIN
                CJNE        R1,#0FFH,ON1
LOAD_ON1:
LOAD_OFF:
                MOV         R1,#BUFOFF
                ACALL       CHKS
                MOV         R1,BUFMIN
                CJNE        R1,#0FFH,OFF1
LOAD_OFF1:
                MOV         A,BUFORDER
                CJNE        A,#4,LOAD_1                 ;CHECK END OF LOOP PROGRAM
                AJMP        SUB_LOAD
LOAD_1:
                INC         ONOFF
                MOV         A,ONOFF
                MOV         @R0,A
                AJMP        SUB_LOAD0
ON1:
                ACALL       CHK_OUT
                CJNE        R3,#0,EXIT_1                ;CHECK ON AND OFF LOAD
                CLR         ONOFF.4
                LCALL       ON_OFF
EXIT_1:         AJMP        LOAD_ON1
OFF1:
                ACALL       CHK_OUT
                CJNE        R3,#0,EXIT_2                ;CHECK ON AND OFF LOAD
                SETB        ONOFF.4
                LCALL       ON_OFF
EXIT_2:         AJMP        LOAD_OFF1                   ;NOT COMPLETE
;*****************************************
;  SUB. CHECK OUTPUT FUNCTION
;*****************************************
CHK_OUT:
                MOV         R3,#0FFH
                MOV         A,BUFDAY
                CJNE        A,#0FFH,OUT_WEEK
                MOV         A,BUFHOUR
                CJNE        A,HRS,EXIT_0
                MOV         A,BUFMIN
                CJNE        A,MINS,EXIT_0
                MOV         R3,#0
EXIT_0:         RET
OUT_WEEK:
                CJNE        A,DAY,EXIT_0
                MOV         A,BUFHOUR
                CJNE        A,HRS,EXIT_0
                MOV         A,BUFMIN
                CJNE        A,MINS,EXIT_0
                MOV         R3,#0
                RET
;*************************************************************************
;  POINTER CHECK OUTPUT
;*************************************************************************
CHKS:
                MOV         R3,BUFORDER
CHKS_ST:        DJNZ        R3,CHKS_ST1
                MOV         R3,BUFOUT
CHKS_S:         DJNZ        R3,CHKS_S1
                LCALL       KEEP_OUT
                RET
CHKS_S1:
                MOV         A,R1
                ADD         A,#3
                MOV         R1,A
                AJMP        CHKS_S
CHKS_ST1:       MOV         A,R1
                ADD         A,#24
                MOV         R1,A
                AJMP        CHKS_ST
KEEPS_OUT:      MOV         R3,#3
```

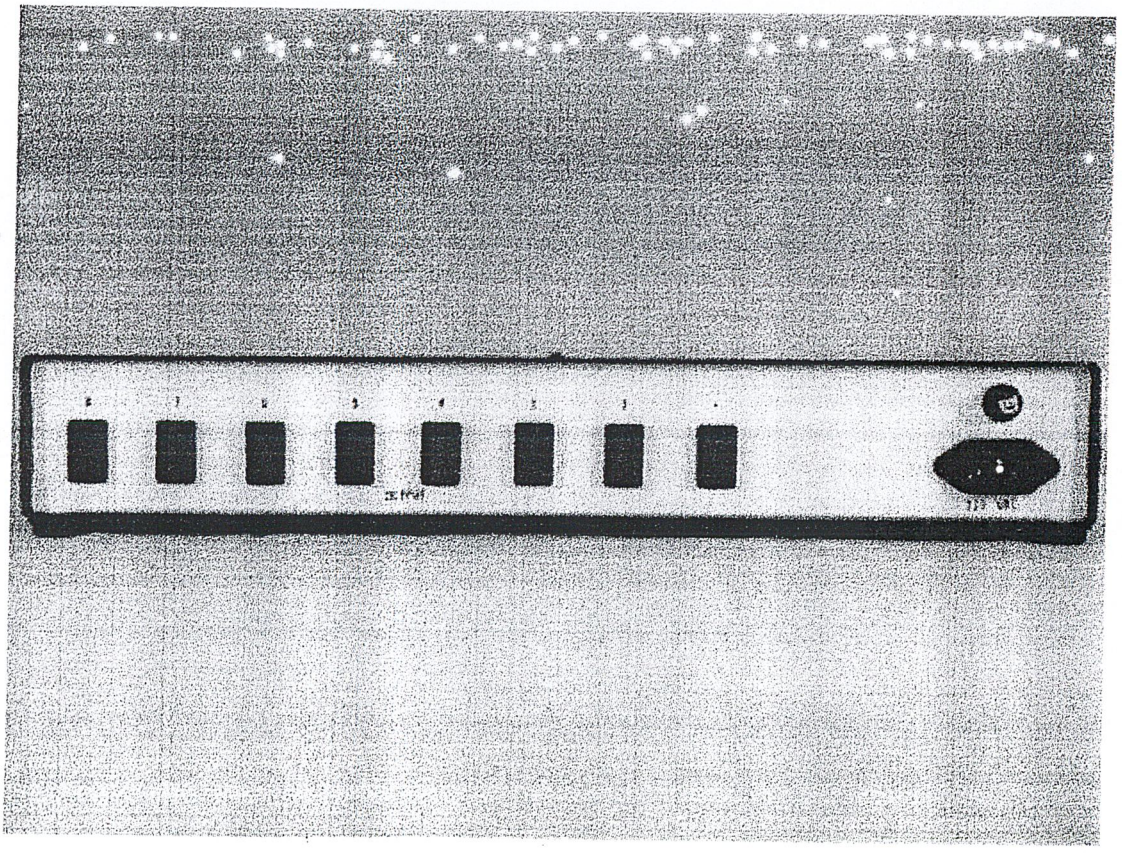
```
                MOV         R0,#BUFDAY
KEEPS:          MOV         A,@R1
                MOV         @R0,A
KEEPS_CHK:      DJNZ        R3,KEEPS_CHK1
                RET
KEEPS_CHK1:
                INC         R1
                INC         R0
                AJMP        KEEPS
;*********************************************************
;   FUNCTION SET PASSWORD
;*********************************************************
CHECKPASS:
                MOV         R4,#CLRDSP
                LCALL       WRLCDCOM4
                MOV         1CH,#0FFH
                MOV         1DH,#0FFH
                MOV         1EH,#0FFH
                MOV         1FH,#0FFH
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '  Key Password  ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          '    " **** "    ',0
CHKPASS1:       MOV         R5,#6
                MOV         R6,#OLDPASS
CHKPASS:
                MOV         A,#2
                MOV         B,R5
                LCALL       PLACECUR4
                LCALL       BLINK_CUR
CHKPASS_0:
                MOV         R1,#10
                LCALL       KEY_DEE
                CJNE        R3,#01H,CHKPASS_1
                RET
CHKPASS_1:
                LCALL       KEY_NUM
                CJNE        R0,#0FFH,CHKPASS_2
                AJMP        CHKPASS
CHKPASS_2:
                CJNE        R0,#0CH,CHKPASS_3           ;CANCLE
                CJNE        R5,#6,CKPASS1
                RET
CKPASS1:        DEC         R5
                DEC         R6
                AJMP        CHKPASS
CHKPASS_3:
                CJNE        R0,#0DH,CHKPASS_4           ;ENTER
                CJNE        R5,#6,CHKPASS
                ACALL       CHKPAS
                RET
CHKPASS_4:
                CJNE        R7,#0FH,CHKPASS_5           ;DON'T CARE ANY KEY
                AJMP        CHKPASS
CHKPASS_5:
                MOV         R1,06H
                MOV         @R1,00H
                INC         R1
                INC         R5
                MOV         R6,01H
                CJNE        R1,#20H,CHKPASS
                MOV         R6,#OLDPASS
                MOV         R5,#6
                AJMP        CHKPASS
```

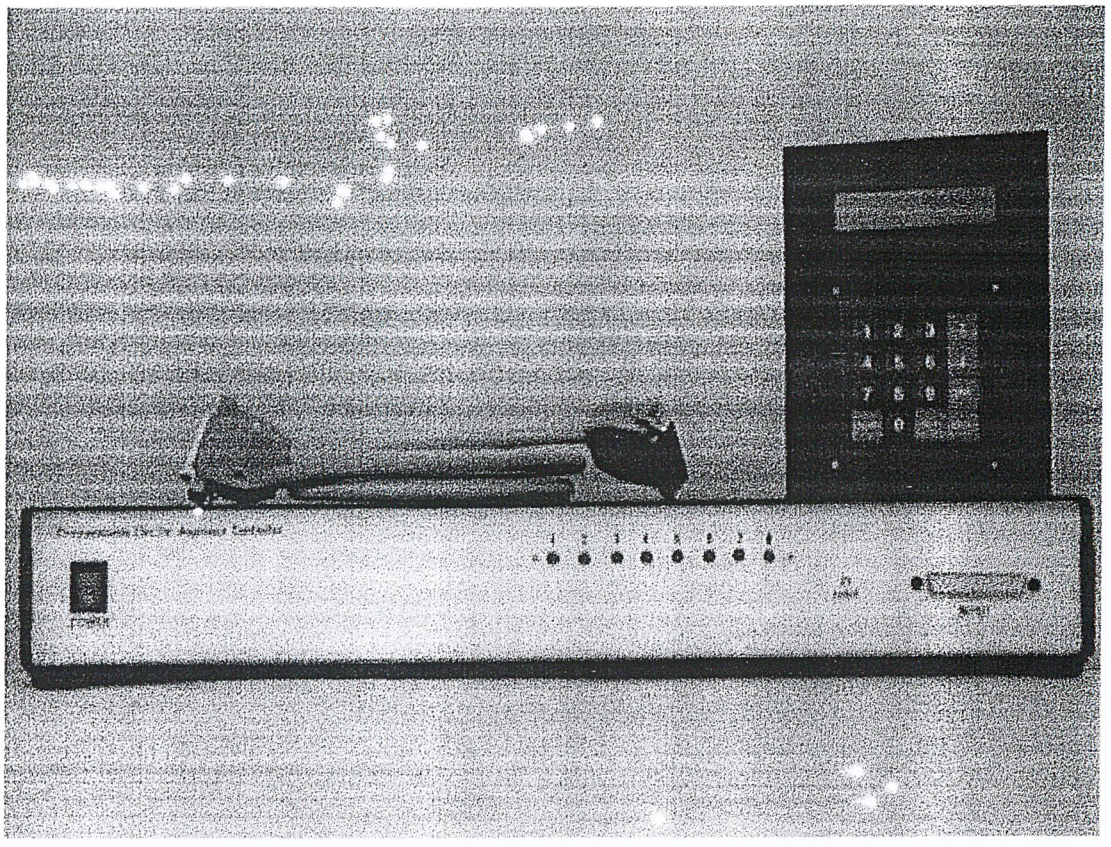
```
;*********************************************************
;   FUNCTION SET PASSWORD
;*********************************************************
PASSWORD:
                MOV         R4,#CLRDSP
                LCALL       WRICDCOM4
                MOV         1CH,#0FFH
                MOV         1DH,#0FFH
                MOV         1EH,#0FFH
                MOV         1FH,#0FFH
                MOV         A,#1
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          'Old Pass.= **** ',0
                MOV         A,#2
                MOV         B,#0
                LCALL       PLACECUR4
                LCALL       PRTLCD4
                DB          'New Pass.= **** ',0
PASSWORD1:
                MOV         R5,#11
                MOV         R6,#OLDPASS
PASS:
                MOV         A,#1
                MOV         B,R5
                LCALL       PLACECUR4
                LCALL       BLINK_CUR
PASS_0:
                MOV         R1,#10
                LCALL       KEY_DEE
                CJNE        R3,#01H,PASS_1
                RET
PASS_1:
                LCALL       KEY_NUM
                CJNE        R0,#0FFH,PASS_2
                AJMP        PASS
PASS_2:
                CJNE        R0,#0CH,PASS_3              ;CANCLE
                CJNE        R5,#11,PASS1
                RET
PASS1:          DEC         R5
                DEC         R6
                AJMP        PASS
PASS_3:
                CJNE        R0,#0DH,PASS_4              ;ENTER
                CJNE        R5,#11,PASS
                ACALL       CHKPAS
                CJNE        A,#0FH,PASSWORD1
                AJMP        NEWPASS1
PASS_4:
                CJNE        R7,#0FH,PASS_5              ;DON'T CARE ANY KEY
                AJMP        PASS
PASS_5:
                MOV         R1,06H
                MOV         @R1,00H
                INC         R1
                INC         R5
                MOV         R6,01H
                CJNE        R1,#20H,PASS
                MOV         R6,#OLDPASS
                MOV         R5,#11
                AJMP        PASS
NEWPASS1:
                MOV         R5,#11
                MOV         R6,#OLDPASS
NEWPASS:
                MOV         A,#2
                MOV         B,R5
                LCALL       PLACECUR4
NPASS_0:
                MOV         R1,#10
                LCALL       KEY_DEE
```

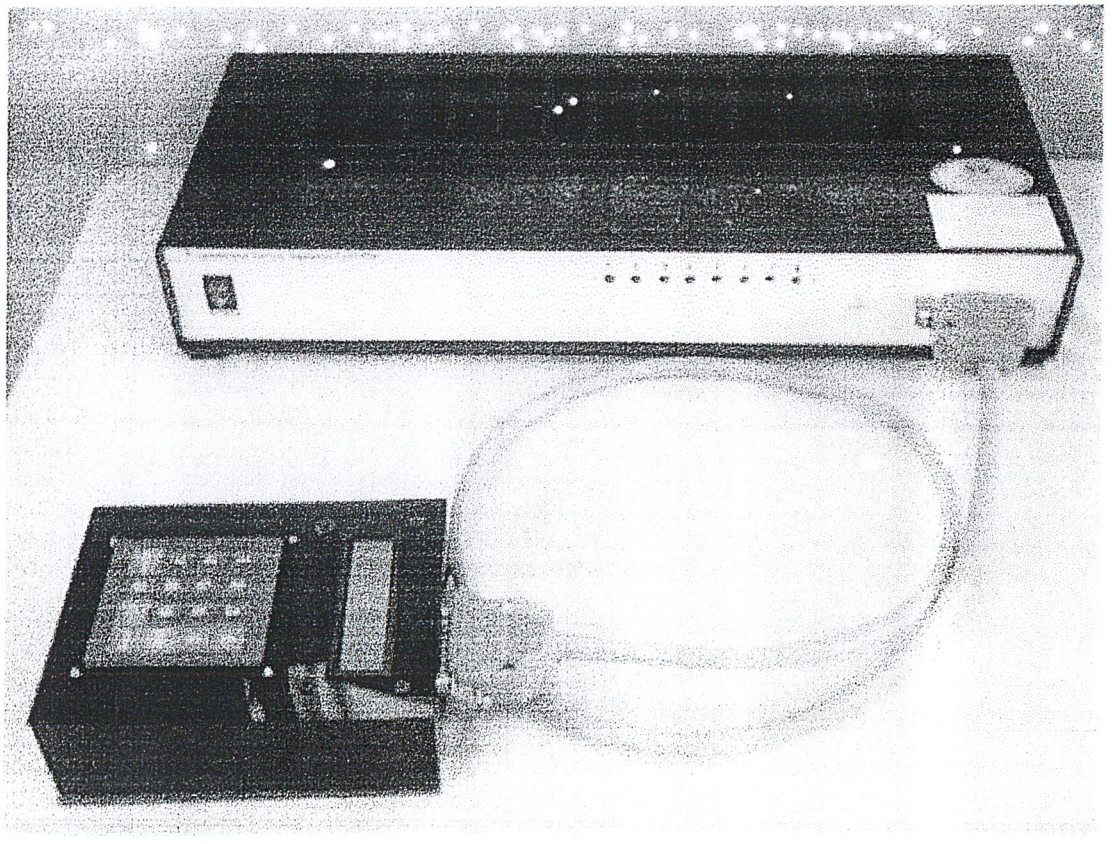
```
                CJNE        R3,#01H,NPASS_1
                RET
NPASS_1:
                LCALL       KEY_NUM
                CJNE        R0,#0FFH,NPASS_2
                AJMP        NEWPASS
NPASS_2:
                CJNE        R0,#0CH,NPASS_3                 ;CANCLE
                CJNE        R5,#11,NPASS1
                RET
NPASS1:         DEC         R5
                DEC         R6
                AJMP        NEWPASS
NPASS_3:
                CJNE        R0,#0DH,NPASS_4                 ;ENTER
                CJNE        R5,#11,NEWPASS
                ACALL       EDITPAS
                MOV         A,#0F0H
                RET
NPASS_4:
                CJNE        R0,#0FH,NPASS_5                 ;DON'T CARE ANY KEY
                AJMP        NEWPASS
NPASS_5:
                MOV         R1,06H
                MOV         @R1,00H
                INC         R1
                INC         R5
                MOV         R6,01H
                CJNE        R1,#20H,NEWPASS
                MOV         R6,#OLDPASS
                MOV         R5,#11
                AJMP        NEWPASS
;************************************************************
CHKPAS:
                MOV         R3,#4
                MOV         R1,#OLDPASS
                MOV         R0,#BUFPAS0
CHKPAS1:        MOV         02H,@R0
                MOV         A,@R1
                CJNE        A,02H,MISS
                DJNZ        R3,CHKPA1
                MOV         A,#0FH                          ;CHECK PASSWORD IS PASS
                RET
CHKPA1:         INC         R1
                INC         R0
                AJMP        CHKPAS1
MISS:           MOV         A,#0F0H                         ;CHECK PASSWORD IS FAIL
                RET
;************************************************************
EDITPAS:
                MOV         R3,#4
                MOV         R1,#OLDPASS
                MOV         R0,#BUFPAS0
CHKNPAS1:       MOV         A,@R1
                MOV         @R0,A
                DJNZ        R3,CHKNPA1
                RET
CHKNPA1:        INC         R1
                INC         R0
                AJMP        CHKNPAS1
; ***********************************************************
; DELAY ROUTINE
; ***********************************************************
BEEB_KEY:
                CLR         BEEB
                ACALL       DELAY_10MS
                ACALL       DELAY_10MS
                ACALL       DELAY_10MS
                SETB        BEEB
                RET
DELAY_10MS:
                MOV         R0,#10                          ; 10 MS
DE_10MS_1:      MOV         R1,#0E6H                        ; 1 MS
```

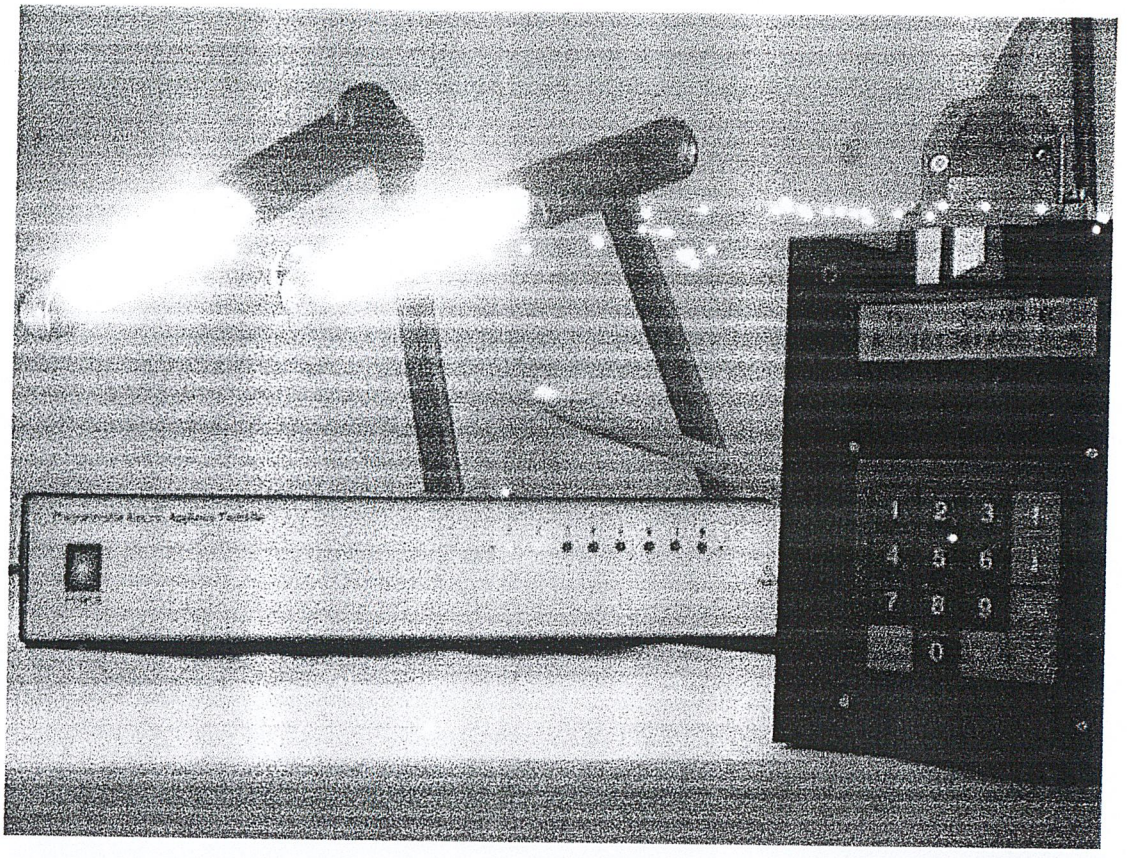
```
DE_10MS_2:      NOP
                NOP
                DJNZ        R1,DE_10MS_2
                DJNZ        R0,DE_10MS_1
               RET
               END
; ***************; ********** ****************************
; END OF PROGRAM
; ***************************************************
```

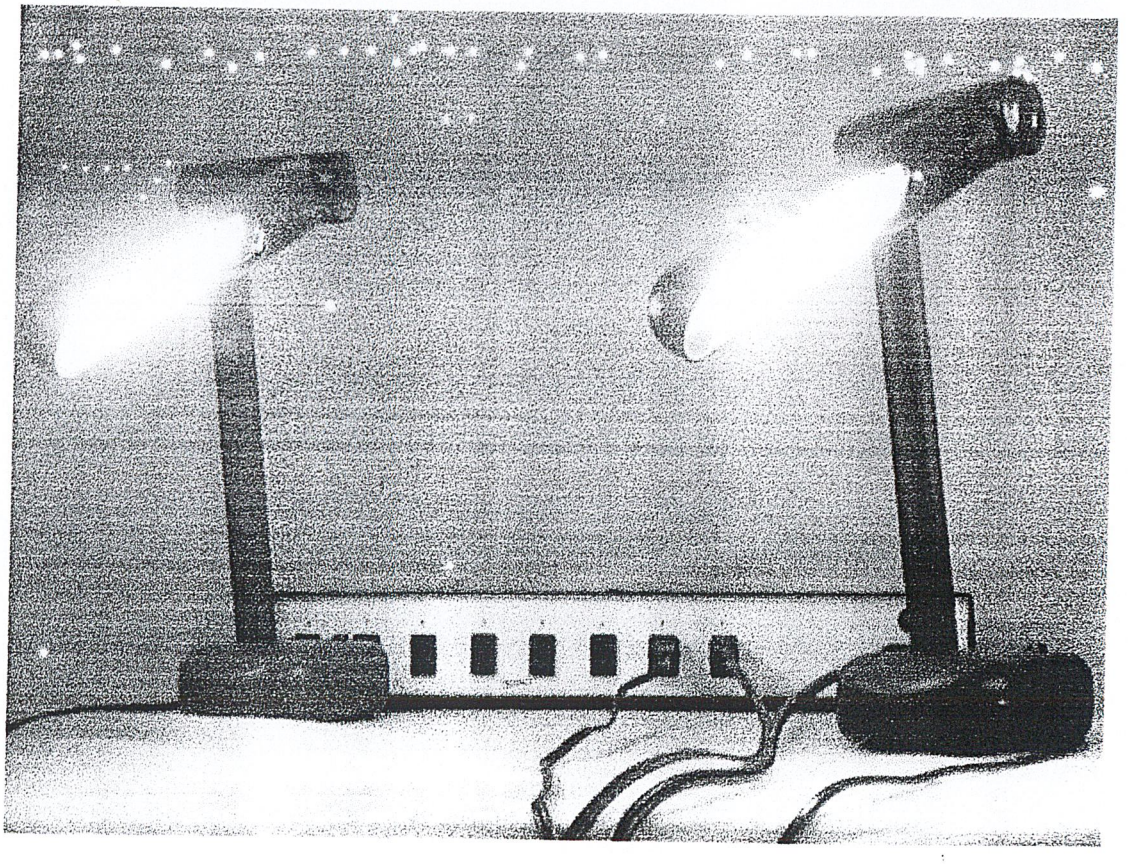


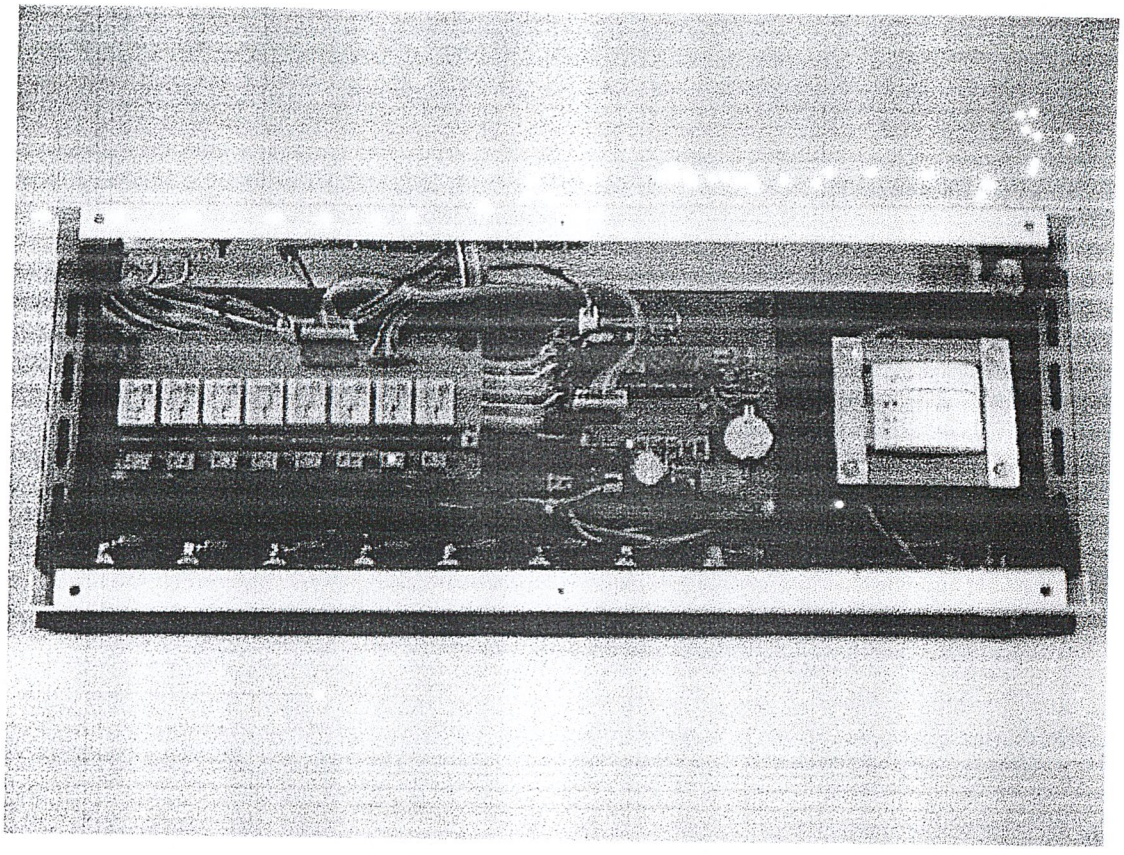







1 2 3
4 5 6
7 8 9
0





กิตติกรรมประกาศ

ขอขอบคุณ ผศ.ดร.เกียรติศักดิ์ คมวัชระ อาจารย์ที่ปรึกษา และ อาจารย์ทุกท่านในภาควิชาวิศวกรรมระบบควบคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้คำแนะนำปรึกษา เสนอแนวทางการศึกษา แนวทางแก้ไขปัญหา และข้อบกพร่องต่าง ๆ ในการทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณบิดา มารดา ที่ได้สนับสนุนด้านทุนทรัพย์ และให้กำลังใจทางการศึกษาตลอดมาจนถึงปัจจุบัน

ขอขอบคุณเพื่อน ๆ ทุกคนที่คอยให้คำปรึกษาและให้หยิบยืมอุปกรณ์ในการทำโครงงาน และคอยให้ความช่วยเหลือ ให้คำแนะนำที่ดี เป็นกำลังใจ กำลังกายที่ดีเสมอมา

หนังสืออ้างอิง

1. ชัยวัฒน์ ลิ้มพรจิตรวิไล, วรพจน์ กรแก้ววัฒนกุล, "เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์MCS-51", บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด, 475 หน้า
2. ธีรวัฒน์ ประกอบผล, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์", สมาคมส่งเสริมเทคโนโลยี (ไทย – ญี่ปุ่น), 155 หน้า, 2540
3. สุนทร วิทูสุรพจน์, "การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 179 หน้า, 2537
4. ชวลิต บุนราม, "บอร์ดตั้งเวลาเครื่องใช้ไฟฟ้าผ่านคู่สายโทรศัพท์รุ่นใหม่", วารสารเซมิคอนดักเตอร์อิเล็กทรอนิกส์, ฉบับที่ 2.09, 2543, หน้า 145 – 151.