ระบบลานจอดรถอัตโนมัติ

PARKING LOT SYSTEM

นางสาวธนัชพร สวรรยาธิปัติ
นางสาวธิดาพร ท่าหิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมระบบควบคุม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบลานจอดรถอัตโนมัติ

PARKING LOT SYSTEM

ผู้จัดทำ นางสาวธนัชพร สวรรยาธิปัติ รหัสประจำตัว 46010288

นางสาวธิดาพร ท่าหิน รหัสประจำตัว 46010307

อาจารย์ที่ปรึกษา.....

(รองศาสตราจารย์ ดร. จงกล งามวิวิทย์)

.....

(ผู้ช่วยศาสตราจารย์ ถาวร เบญจนราสุทธิ์)

ระบบลานจอดรถอัตโนมัติ

โดย
นางสาวธนัชพร สวรรยาธิปัติ
นางสาวธิดาพร ท่าหิน

อาจารย์ที่ปรึกษา
รศ.ดร.จงกล งามวิวิทย์
ผศ.ถาวร เบญจนราสุทธิ์

ปีการศึกษา 2549

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้จัดทำขึ้น เพื่ออธิบายโครงงานการออกแบบและการสร้างระบบลานจอดรถอัตโนมัติ ซึ่งระบบลานจอดรถอัตโนมัตินี้สามารถคำนวณเวลาและค่าจอดรถ และหักจากมูลค่าในบัตรจอดรถที่มี นอกจากนี้ยังสามารถบอกจำนวนรถที่สามารถจอดได้ในแต่ละชั้นเพื่อประหยัดเวลาในการหาที่จอดรถ รวมทั้งสามารถบอกจำนวนเงินคงเหลือในบัตรก่อนที่จะนำรถเข้าจอดได้

สำหรับโครงงานนี้ใช้ไมโครคอนโทรลเลอร์เข้ามาควบคุมระบบ และประยุกต์ใช้บัตรแถบแม่เหล็กเป็นบัตรจอดรถ และสามารถนำไปพัฒนาเพื่อความสะดวกในการจอดรถ

PARKING LOT SYSTEM

By

Miss Tanutchaporn Sawanyatipat

Miss Thidapron Thahin

Advisor

Assoc. Prof. Dr. Jongkol Ngamwiwit

Asst. Prof. Taworn Benjanarasuth

Academic Year 2006

ABSTRACT

This thesis describes the parking lot system which can calculate the parking time, parking fee and the balance in the parking card. Moreover, the system can show the available parking space in each floor and the balance in the card.

Microcontrollers are used to control the parking lot system and the magnetic card is applied to be a parking card. This system can also be developed further for convenience in car parking.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับความช่วยเหลือเป็นอย่างดีจาก รศ.ดร.จงกล งามวิวิทย์ และ ผศ.ถาวร เบญจนราสุทธิ์ ที่ได้กรุณาให้คำปรึกษาแนะนำที่ดีมาโดยตลอดตั้งแต่ต้น รวมทั้งเอื้อเฟื้ออุปกรณ์ที่จำเป็น และความช่วยเหลืออื่นๆ ที่เป็นประโยชน์ต่อโครงงาน ผู้จัดทำรู้สึกซาบซึ้งและขอกราบขอบพระคุณอย่างสูง

คณะผู้จัดทำขอขอบพระคุณ คุณพ่อคุณแม่ เพื่อนๆ ที่ให้ความช่วยเหลือให้กำลังใจ ให้คำปรึกษาและแนะนำข้อมูลต่างๆ เป็นอย่างดี รวมทั้งพี่เจี๋ยที่ให้คำแนะนำรวมทั้งข้อมูลในการทำโครงงานฉบับนี้จนสำเร็จลุล่วง ตลอดจนแนวทางในการแก้ไขปัญหาอันเป็นประโยชน์กับปริญญานิพนธ์ และขอขอบพระคุณอาจารย์ภาควิชาวิศวกรรมระบบควบคุมทุกท่านที่ให้ความสะดวกในการยืมวัสดุอุปกรณ์ รวมทั้งเพื่อนๆทุกคนที่ช่วยเหลือและแนะนำความรู้ให้กับคณะผู้จัดทำจนปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

ผู้จัดทำ
นางสาวธนัชพร สวรรยาธิปัติ
นางสาวธิดาพร ท่าหิน

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

สัญลักษณ์และคำย่อ

k	kilo
m	milli
ACC	accumulator
ADDR	address
ALU	arithmetic and logic unit
CGRAM	character generater random access memory
CGROM	character generater read only memory
CLK	clock
CPU	central processing unit
DDRAM	display data random access memory
DR	data register
GND	ground
IE	interrupt enable register
INT	interrupt
IR	instruction register
LASER	light amplification by stimulated of radiation
LCD	liquid crystal display
LED	light emitting diode
R	register
RAM	random access memory
ROM	read only memory

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบันระบบควบคุมถูกนำมาประยุกต์ใช้ในการควบคุมอุปกรณ์อิเล็กทรอนิกส์ เพื่อนำไปประยุกต์ใช้ในการควบคุมระบบต่างๆอย่างมากมาย ไมโครคอนโทรลเลอร์จึงเข้ามามีบทบาทเป็นอย่างมากในการควบคุมระบบ เนื่องจากมีความสะดวกและง่ายในการพัฒนาเพื่อใช้ในการควบคุม

สำหรับโครงงานฉบับนี้ได้ใช้ไมโครคอนโทรลเลอร์เข้ามาควบคุมระบบ เพื่อใช้ในการเก็บเงินค่าจอดรถแบบอัตโนมัติ โดยผู้ที่นำรถเข้ามาจอดจะรูดบัตรผ่านเครื่องอ่านบัตรแถบแม่เหล็ก จอLCD จะแสดงจำนวนเงินคงเหลือในบัตร ประตูทางเข้าจึงเปิดให้นำรถเข้าไปจอด เมื่อต้องการจะนำรถออกจากที่จอดรถจะต้องทำการรูดบัตรผ่าน จากนั้นเครื่องจะทำการคิดคำนวณเวลาและค่าจอดรถ และหักเงินออกจากมูลค่าในบัตรที่มีอยู่ ประตูทางออกจึงจะเปิดให้นำรถออกไปได้ ระบบลานจอดรถอัตโนมัตินี้ยังออกแบบเพื่อทำการบอกจำนวนรถที่สามารถจอดได้ในแต่ละชั้น ทำให้ผู้ที่นำรถเข้ามาจอดไม่ต้องเสียเวลาในการเข้าไปหาที่จอดรถ ในกรณีที่ที่จอดรถในชั้นนั้นเต็มจะได้นำรถเข้าไปจอดในชั้นอื่นๆ ที่มีที่ว่างได้ เพื่อประหยัดเวลาในการหาที่จอดรถ

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาหลักการออกแบบและสร้างระบบลานจอดรถอัตโนมัติ
2. เพื่อศึกษาวงจรอิเล็กทรอนิกส์แบบต่างๆ รวมถึงการประยุกต์ใช้งาน
3. เพื่อศึกษาการใช้งานไมโครคอนโทรลเลอร์ในระบบควบคุม

1.3 ขอบเขตของโครงงาน

1. ศึกษาและพัฒนาวงจรอิเล็กทรอนิกส์ที่เกี่ยวข้อง
2. ศึกษาและพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์ เพื่อใช้ในการทำงานของระบบ
3. สร้างระบบลานจอดรถอัตโนมัติ ซึ่งมีคุณสมบัติดังนี้
   - สามารถคิดเงินค่าจอดรถได้อัตโนมัติและเก็บเงินค่าจอดรถได้จากมูลค่าในบัตร
   - มีบัตรผ่านที่มีข้อมูลเจ้าของรถและเลขทะเบียนรถ
   - สามารถแสดงจำนวนรถที่จอดได้ในแต่ละชั้นและบอกจำนวนเงินก่อนเข้าจอด
   - บัตรผ่านเข้าและออกต้องเป็นบัตรเดียวกันที่มีข้อมูลตรงกันจึงจะนำรถออกได้
   - สามารถเปิดปิดประตูอัตโนมัติ

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาทฤษฎีและการทำงานของวงจรอิเล็กทรอนิกส์ต่างๆที่ใช้และศึกษาการใช้งานของไมโครคอนโทรลเลอร์
2. ทำการออกแบบวงจรอิเล็กทรอนิกส์ต่างๆ
3. ทำการทดสอบวงจรอิเล็กทรอนิกส์ที่ออกแบบได้
4. ทำการเขียนโปรแกรมเพื่อคำนวณและควบคุมอุปกรณ์อิเล็กทรอนิกส์ที่ออกแบบไว้
5. นำวงจรอิเล็กทรอนิกส์มาประกอบเพื่อใช้งานจริงในระบบลานจอดรถจำลอง
6. สรุปผลการทดลองและจัดทำรายงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ต้นแบบระบบลานจอดรถอัตโนมัติ ซึ่งมีคุณสมบัติตามที่ระบุในขอบเขตของโครงงาน ซึ่งมีข้อดีคือ
   - ความสะดวกและความปลอดภัยในการจอดรถ
   - สามารถลดปัญหาการถูกโจรกรรมรถได้
   - ลดแรงงานคนและช่วยประหยัดค่าใช้จ่ายได้
2. เป็นแนวทางในการศึกษาโดยการนำอุปกรณ์อิเล็กทรอนิกส์มาประยุกต์ใช้งานร่วมกับไมโครคอนโทรลเลอร์เพื่อใช้ควบคุมระบบต่างๆได้

บทที่ 2
ทฤษฎีและหลักการ

ในการออกแบบและการสร้างระบบลานจอดรถอัตโนมัติต้องใช้อุปกรณ์อิเล็กทรอนิกส์ต่างๆ ประกอบไปด้วย ไมโครคอนโทรลเลอร์ MCS-51 อุปกรณ์รับแสง ไอซีสร้างฐานเวลา DS1307 บัตรแถบแม่เหล็ก เครื่องอ่านบัตรแถบแม่เหล็กและโมดูล LCD ดังนั้นในบทนี้จะกล่าวถึงทฤษฎีและหลักการในการใช้งานอุปกรณ์อิเล็กทรอนิกส์ที่เกี่ยวข้อง

2.1 การใช้งานไมโครคอนโทรลเลอร์ MCS-51

MCS-51 เป็นไมโครคอนโทรลเลอร์ที่ออกแบบมาเพื่อสนองความต้องการของผู้ใช้แบบสำเร็จในไอซีตัวเดียว คือมีอุปกรณ์สำเร็จรูปภายในมากมาย ไม่ว่าจะเป็นพอร์ตของอินพุตและเอาต์พุต บัฟเฟอร์ที่ใช้เชื่อมต่อกับวงจรภายนอก (Interface Buffer) และสายสัญญาณควบคุมอื่นๆที่ใช้สำหรับแยกสายสัญญาณข้อมูลกับสายสัญญาณกำหนดตำแหน่งหน่วยความจำ นอกจากนี้ยังมีชุดคำสั่งพิเศษเพื่อจัดการข้อมูล วงจรนับเวลา และวงจรตั้งเวลาอีกด้วย

การพัฒนาภายในไมโครคอนโทรลเลอร์ให้มีหน่วยความจำเป็นแบบแฟลช (Flash Memory) ทำให้สามารถโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกจากวงจร เรียกว่า การโปรแกรมจากภายในวงจร (In-System Programming) และมีการติดต่อแบบ SPI (Serial Peripheral Interface) ทำให้สามารถพัฒนาและการแก้ไขปรับปรุงโปรแกรมได้สะดวก

2.1.1 คุณสมบัติของไมโครคอนโทรลเลอร์ MCS-51

โครงสร้างภายในพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แสดงได้ดังรูปที่ 2.1 ซึ่งมีความสามารถและอุปกรณ์ภายในต่างๆ ดังนี้

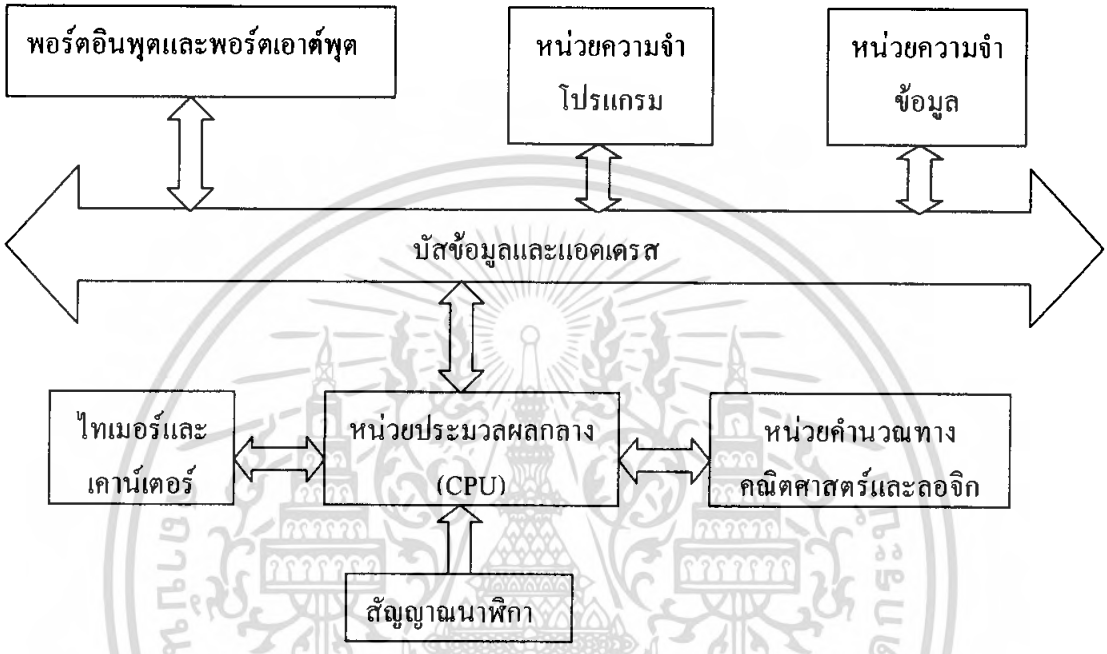
1) หน่วยประมวลผลกลางขนาด 8 บิต
2) หน่วยประมวลผลสำหรับข้อมูลแบบบูลีน (Boolean Processor)
3) หน่วยความจำโปรแกรมภายในแบบอีพรอมขนาด 4 กิโลไบต์
4) หน่วยความจำแบบแรมภายในจำนวน 128 ไบต์
5) พอร์ตอินพุตและเอาท์พุตแบบขนานจำนวน 32 พอร์ตซึ่งทำงานได้อย่างอิสระ
6) ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรมขนาด 64 กิโลไบต์
7) ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูลขนาด 64 กิโลไบต์
8) วงจรนับและจับเวลาขนาด 16 บิต จำนวน 2 วงจร

9) วงจรสื่อสารแบบอนุกรมชนิดดูเพล็กเต็ม (Full Duplex)

10) วงจรควบคุมการอินเตอร์รัปจากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนดลำดับ

11) วงจรผลิตสัญญาณนาฬิกาภายใน

การทำงานของโมโครคอนโทรเลอร์จะอาศัยหลักการทำงานที่เกี่ยวข้องกันโดยอาศัยหลักการทำงานที่เป็นไปตามโครงสร้างเสมอ



รูปที่ 2.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

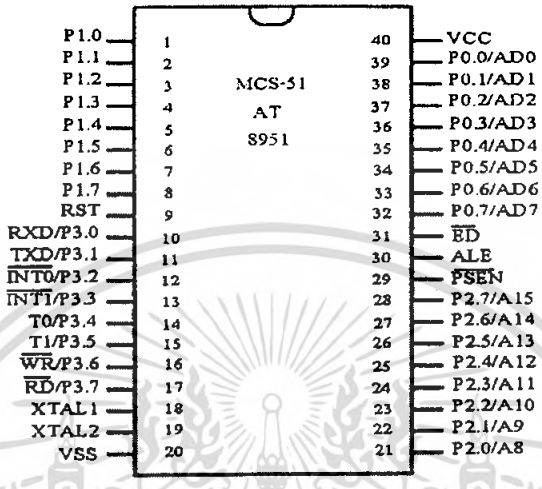
2.1.2 การจัดตำแหน่งขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและพอร์ตใช้งานพื้นฐานเหมือนกัน ได้แบ่งการใช้งานออกเป็นพอร์ตอินพุตและเอาต์พุต (Input/Output Port) พอร์ตสัญญาณควบคุม พอร์ตสัญญาณกำหนดตำแหน่งหน่วยความจำ และพอร์ตสัญญาณข้อมูล แสดงดังรูปที่ 2.2 ตำแหน่งขาของไมโครคอนโทรลเลอร์ MCS-51 และหน้าที่การทำงาน ดังนี้

- **พอร์ต 0** (P0.0 - P0.7 : ขาที่ 32-39) ทำหน้าที่เป็นขารับและขาส่งสัญญาณภายนอก ซึ่งสามารถรับข้อมูลอินพุตและส่งข้อมูลเอาต์พุตขนาด 8 บิตได้ การตั้งค่าให้พอร์ต 0 รับข้อมูลอินพุตทำได้โดยการส่งค่าสถานะ 1 ไปยังบิตที่ต้องการให้รับข้อมูลอินพุต วงจรภายในจะทำให้บิตนั้นมีค่าความต้านทานสูงและสามารถรับข้อมูลอินพุตได้ และยังใช้เป็นขาสัญญาณกำหนดตำแหน่งหน่วยความจำ (A0-A7) และสัญญาณข้อมูล (D0-D7) โดยมีการใช้ตัวแยกสัญญาณชนิด

D-latch มาทำหน้าที่เป็นมัลติเพล็กซ์ (Multiplex) โดยเลือกช่วงเวลาของสัญญาณที่ใช้กำหนดตำแหน่งหน่วยความจำและสัญญาณข้อมูลออกจากกัน

ในขณะที่ใช้เป็นพอร์ตอินพุตและพอร์ตเอาต์พุต วงจรภายในจะไม่มีวงจรเพิ่มกระแสไฟฟ้า (Pull up) จึงจำเป็นต้องต่อวงจรเพิ่มกระแสไฟฟ้าจากภายนอกเข้าไป



รูปที่ 2.2 รูปร่างและการจัดวางขาต่าง ๆ ของไมโครคอนโทรลเลอร์ MCS-51

- **พอร์ต 1** (P1.0 –P1.7 : ขาที่ 1-8) ทำหน้าที่เป็นขารับและขาส่งสัญญาณภายนอก ซึ่งสามารถรับข้อมูลอินพุตและส่งข้อมูลเอาต์พุตขนาด 8 บิตได้ สามารถอ้างถึงการทำงานได้ทีละบิต และวงจรภายในมีตัวต้านทานเพิ่มกระแสไฟฟ้า (Pull Up) ในกรณีที่ต้องการให้รับข้อมูลอินพุตก็สามารถทำได้เหมือนพอร์ต 0

- **พอร์ต 2** (P2.0 –P2.7 : ขาที่ 21-28) ทำหน้าที่เป็นขารับและส่งสัญญาณภายนอก ซึ่งสามารถรับข้อมูลอินพุตและส่งข้อมูลเอาต์พุตขนาด 8 บิตได้ สามารถใช้เป็นขาสัญญาณกำหนดตำแหน่งหน่วยความจำ (A8-A15) และมีวงจรเพิ่มกระแสไฟฟ้าภายใน การกำหนดให้เป็นขาอินพุตทำได้โดยการส่งค่าข้อมูลสถานะ 1 ไปยังบิตที่ต้องการให้เป็นอินพุต ก็จะสามารถรับข้อมูลอินพุตได้

- **พอร์ต 3** (P3.0-P3.7 : ขาที่ 10-17) ทำหน้าที่เป็นขารับและส่งสัญญาณภายนอก ซึ่งสามารถรับข้อมูลอินพุตและส่งข้อมูลเอาต์พุตขนาด 8 บิตได้ คุณสมบัติทั่วไปจะเหมือนกับพอร์ตอื่นๆ แต่จะมีคุณสมบัติที่ต่างออกไป คือ ใช้ทำหน้าที่พิเศษเป็นสัญญาณควบคุมการทำงานต่างๆของไมโครคอนโทรลเลอร์ ดังนี้

P3.0 หรือขา RxD ใช้เป็นขาอินพุตรับข้อมูลจากการสื่อสารแบบอนุกรม

P3.1 หรือขา TxD ใช้เป็นขาอินพุตส่งข้อมูลจากการสื่อสารแบบอนุกรม

P3.2 หรือขา $\overline{\text{INT0}}$ ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0

P3.3 หรือขา $\overline{\text{INT1}}$ ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1

P3.4 หรือขา T0 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0

P3.5 หรือขา T1 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 1

P3.6 หรือขา $\overline{\text{WR}}$ เป็นขาสัญญาณเขียนข้อมูลหน่วยความจำหรืออุปกรณ์ภายนอก

P3.7 หรือขา $\overline{\text{RD}}$ เป็นขาสัญญาณอ่านข้อมูลหน่วยความจำหรืออุปกรณ์ภายนอก

- **$\overline{\text{PSEN}}$ (Program Store Enable ขาที่ 29)** ขานี้จะทำงานที่สภาวะลอจิกเป็น 0 ไมโครคอนโทรลเลอร์ต้องอ่านค่าจากหน่วยความจำภายนอกที่เป็นข้อมูลโดยโปรแกรมจะเก็บในหน่วยความจำถาวร (ROM EPROM EEPROM) ส่วนมากใช้ต่อเป็นขาเลือกทำงาน (Enable ; OE) แต่ถ้าไมโครคอนโทรลเลอร์ใช้หน่วยความจำภายใน ขานี้ก็จะไม่ได้ใช้งานและมีค่าลอจิกเป็น 1

- **ขา ALE (Address Latch Enable ขาที่ 30)** ทำหน้าที่ควบคุมการทำงานของสัญญาณกำหนดตำแหน่งกับสัญญาณข้อมูล โดยใช้การเลือกเส้นทาง (Data Select หรือ Multiplex) โดยปกติเมื่อไมโครคอนโทรลเลอร์ทำงานจะส่งสัญญาณกำหนดตำแหน่งออกมาก่อน พร้อมกับส่งสัญญาณให้ขา ALE ทำงาน เพื่อเลือกให้สัญญาณกำหนดตำแหน่ง (A0-A7) ผ่านไอซี 74LS373 ที่ทำหน้าที่เลือกเส้นทาง ถ้าส่งสัญญาณข้อมูลออกมา ไอซี 74LS373 จะไม่ทำงาน ข้อมูลก็จะถูกส่งไปที่สายสัญญาณข้อมูล

- **ขา $\overline{\text{EA}}$ (External Access ขาที่ 31)** ทำหน้าที่เลือกการทำงานของหน่วยความจำ ถ้ามีค่าลอจิกเป็น 1 หมายถึง ใช้ข้อมูลจากหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ แต่ถ้ามีค่าลอจิกเป็น 0 หมายถึง ใช้ข้อมูลจากหน่วยความจำภายนอก

- **ขา RST (Reset ขาที่ 9)** ทำหน้าที่สั่งให้ไมโครคอนโทรลเลอร์เริ่มต้นการทำงานใหม่ การทำงานที่ค่าลอจิก 1 นี้จะทำให้ไมโครคอนโทรลเลอร์เริ่มต้นทำงานที่ตำแหน่ง 0000 เพื่ออ่านข้อมูลโปรแกรมและจัดระบบการทำงาน

- **ขาสัญญาณนาฬิกา (ขาที่ 18-19)** ทำหน้าที่เป็นตัวสร้างสัญญาณนาฬิกาให้กับไมโครคอนโทรลเลอร์ใช้เป็นฐานเวลาในการทำงาน โดยใช้แผ่นผลึก (Crystal) ที่มีความถี่ตั้งแต่ 0-24 เมกกะเฮิร์ต ร่วมกับตัวเก็บประจุขนาด 20-33 pF

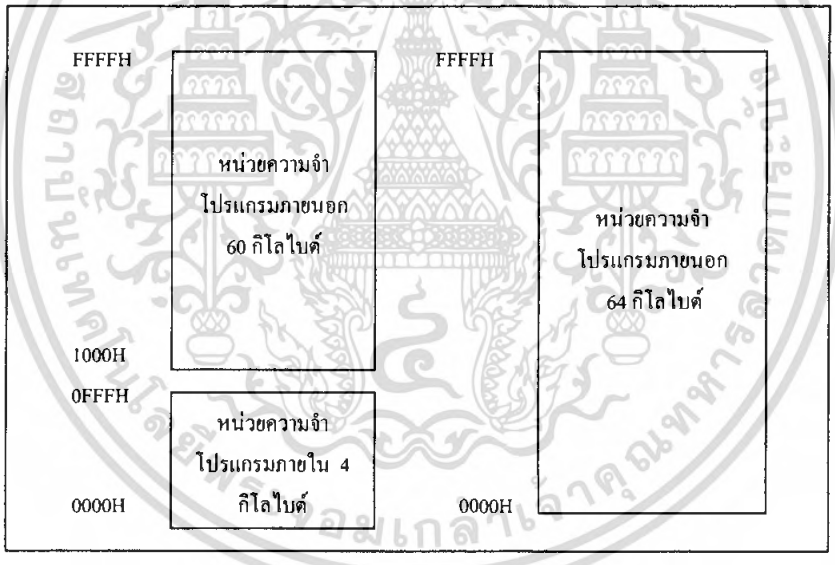
- **แหล่งจ่ายไฟ (Power Supply)** ขาที่ 20 จะเป็นขากราวด์ (Ground) และขาที่ 40 จะเป็นแหล่งจ่ายไฟบวกให้กับไมโครคอนโทรลเลอร์ ซึ่งใช้แหล่งจ่ายไฟขนาดไม่เกิน 5 โวลต์

2.1.3 หน่วยความจำโปรแกรม

ในรูปที่ 2.3 แสดงการจัดหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในเบอร์ AT89C51 และ AT89C52 จะเห็นได้ว่า ทั้งสองเบอร์สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียว หรือรวมกับภายนอก หรือเลือกใช้หน่วยความจำภายนอกอย่างเดียวก็ได้ ดังในรูปที่ 2.3 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 ซึ่งสามารถทำการอ่านได้เพียงอย่างเดียว



รูปที่ 2.3 การจัดสรรหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51

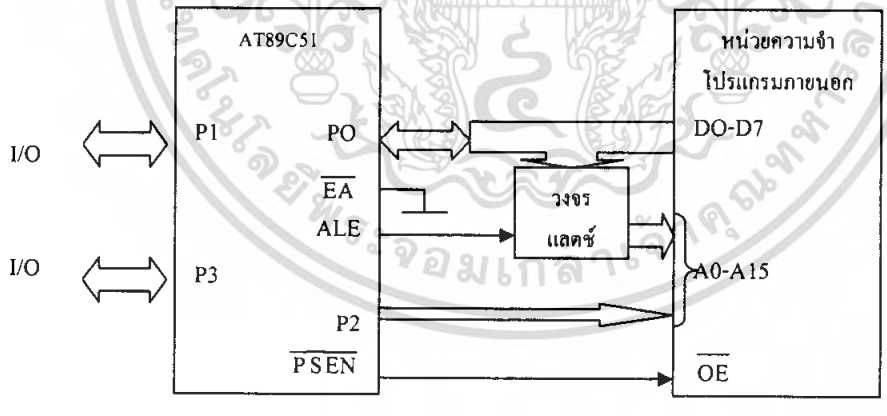
หน่วยความจำโปรแกรมมีแอดเดรสเริ่มต้นที่ 0000H เมื่อซีพียูได้รับการรีเซตให้เริ่มต้นการทำงาน จะต้องมาเริ่มต้นที่แอดเดรส 0000H เสมอ อย่างไรก็ตาม ในพื้นที่หน่วยความจำโปรแกรมไม่ว่าจะใช้งานจากภายในหรือภายนอกก็ตามต้องมีการสงวนพื้นที่บางตำแหน่งเอาไว้สำหรับการบริการอินเตอร์รัปต์ 6 ประเภท ประเภทละ 8 ไบต์ ประกอบด้วย

- พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากภายนอก กำหนดไว้ที่แอดเดรส 0003H

- พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 0 กำหนดไว้ที่แอดเดรส 000BH
- พื้นที่สำหรับบริการอินเตอร์รัปต์ 1 จากภายนอก กำหนดไว้ที่แอดเดรส 0013H
- พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH
- พื้นที่สำหรับบริการอินเตอร์รัปต์ของการสื่อสารอนุกรม กำหนดไว้ที่แอดเดรส 0023H
- พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH

กรณีที่ใช้ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่มีหน่วยความจำโปรแกรมภายในแต่ต้องการติดต่อกับหน่วยความจำภายนอกด้วย สามารถทำได้โดยต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์ ยกตัวอย่าง ไมโครคอนโทรลเลอร์ AT89C51 มีหน่วยความจำโปรแกรมขนาด 4 กิโลไบต์ มีแอดเดรสอยู่ระหว่าง 0000H-0FFFH เมื่อต่อหน่วยความจำโปรแกรมภายนอกต้องกำหนดให้แอดเดรสอยู่ในช่วง 1000H-FFFFH

การต่อหน่วยความจำภายนอกแสดงดังรูปที่ 2.4 ขาพอร์ต P0.0-P0.7 ใช้เป็นขาข้อมูล D0-D7 และขาแอดเดรสไบต์ต่ำ โดยผ่านวงจรแลตช์ ซึ่งปกติใช้ไอซีเบอร์ 74HC573 และใช้สัญญาณ ALE และ PSEN ในการเลือกใช้งานขา P0.0-P0.7 เพื่อเป็นขาข้อมูลหรือขาแอดเดรส ในขณะที่ขา P2.0-P2.7 ใช้ในการเชื่อมต่อกับขาแอดเดรสไบต์สูง A8-A15 ดังนั้นเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ไมโครคอนโทลเลอร์จะเหลือขาพอร์ตเพียง 16 บิต คือ ที่ขาพอร์ต P1.0-P1.7 และ P3.0-P3.7

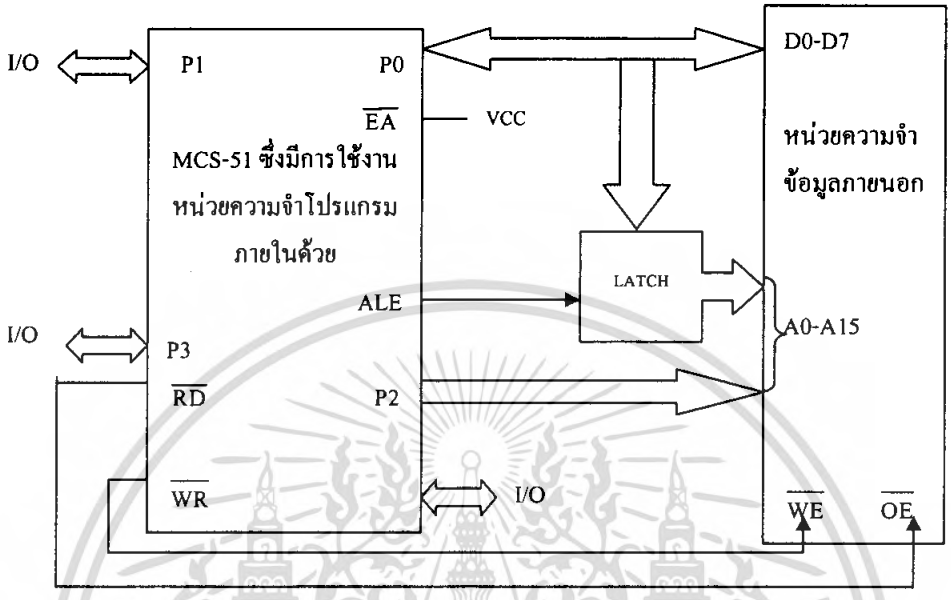


รูปที่ 2.4 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์

2.1.4 หน่วยความจำข้อมูล

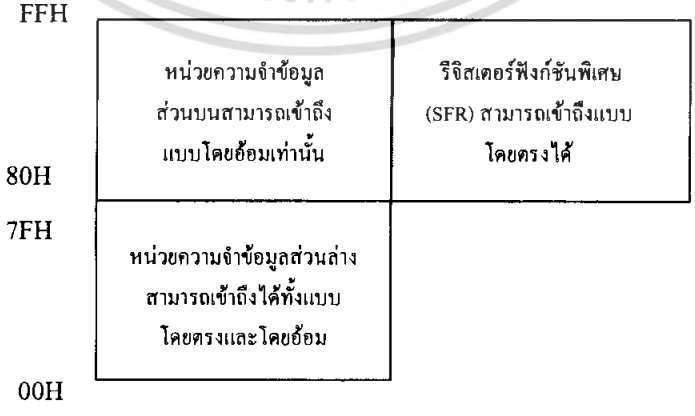
หน่วยความจำข้อมูลมี 2 แบบคือ หน่วยความจำข้อมูลภายนอกและภายใน โดยไมโครคอนโทรลเลอร์ MCS-51 สามารถติดต่อกับหน่วยความจำข้อมูลภายนอกได้สูงสุด 64

กิโลไบต์ โดยการใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก การติดต่อหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แสดงได้ดังรูปที่ 2.5 จะเห็นได้ว่ามีลักษณะคล้ายกับการติดต่อหน่วยความจำโปรแกรมภายนอก แตกต่างกันที่มีสัญญาณที่ใช้สำหรับการอ่านและเขียนหน่วยความจำข้อมูลภายนอก นั่นคือขา $\overline{RD}$ และ $\overline{WR}$



รูปที่ 2.5 การเชื่อมต่อหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51

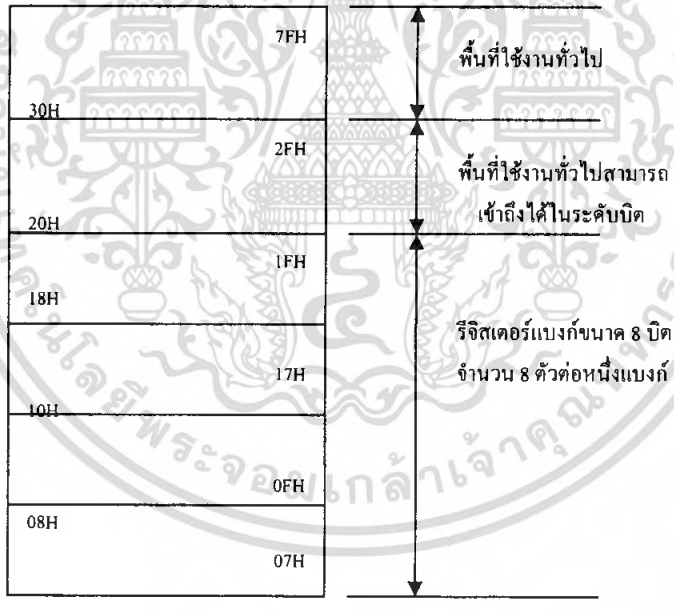
สำหรับไมโครคอนโทรลเลอร์ MCS-51 จะมีหน่วยความจำข้อมูลภายในเป็นแบบแรม โดยแต่ละเบอร์จะมีขนาดแตกต่างกันไป ในเบอร์ AT89C51 มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ในขณะที่เบอร์ AT89C52 มีขนาด 256 ไบต์ สำหรับการจัดสรรหน่วยความจำข้อมูลภายในแบ่งเป็น 3 ส่วนคือ หน่วยความจำข้อมูลส่วนล่าง หน่วยความจำข้อมูลส่วนบน และรีจิสเตอร์ฟังก์ชันพิเศษ(SFR : Special Function Register) แต่ละส่วนมีขนาด 128 ไบต์ ดังแสดงการจัดสรรในรูปที่ 2.6



จะเห็นได้ว่า หน่วยความจำข้อมูลส่วนบนและรีจิสเตอร์ฟังก์ชั่นพิเศษมีตำแหน่งที่ทับซ้อนกัน แต่จะใช้การติดต่อที่แตกต่างกัน และในไมโครคอนโทรลเลอร์ MCS-51บางเบอร์จะไม่มีหน่วยความจำข้อมูลส่วนบน

ขนาดหน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดยแท้จริงแล้วมีเพียง 256 ไบต์ แต่ด้วยการจัดการเข้าถึงที่แตกต่างกัน จึงดูเสมือนว่า ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีหน่วยความจำข้อมูลภายในสูงถึง 384ไบต์ โดยในหน่วยความจำข้อมูลส่วนล่างขนาด 128 ไบต์ มีแอดเดรสอยู่ที่ 00H-7FH ซึ่งสามารถเข้าถึงได้โดยตรงและโดยอ้อม สำหรับหน่วยความจำข้อมูลส่วนบนก็มีขนาด 128 ไบต์เช่นกัน ซึ่งมีแอดเดรสอยู่ที่ 80H-FFH แต่สามารถเข้าถึงแบบโดยอ้อมเท่านั้น ในขณะที่รีจิสเตอร์ SFR มีแอดเดรสอยู่ที่ 80H-FFH เช่นเดียวกับหน่วยความจำข้อมูลส่วนบน แต่จะใช้การเข้าถึงแบบโดยตรง

ในรูปที่ 2.7 แสดงการจัดสรรหน่วยความจำข้อมูลส่วนล่าง หน่วยความจำ 32 ไบต์ต่ำสุดที่แอดเดรส 00H-1FH แบ่งเป็น 4 กลุ่ม เรียกว่า 4 แบงก์ (Bank) แต่ละแบงก์มีรีจิสเตอร์ 8 ตัวคือ R0-R7 การติดต่อกับหน่วยความจำในแบงก์ใดให้กำหนดที่รีจิสเตอร์ PSW(Program Status Word Register)



รูปที่ 2.7 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในส่วนล่างของไมโครคอนโทรลเลอร์

หน่วยความจำข้อมูล 16 ไบต์ถัดมาที่แอดเดรส 20H-2FH เป็นพื้นที่สำหรับใช้งานทั่วไป สามารถเข้าถึงได้ในระดับบิต (Bit Addressable) และหน่วยความจำข้อมูลที่เหลือ 80ไบต์ จะต้อง

แบ่งส่วนหนึ่งสำรองไว้เป็นพื้นที่ของสแต็ก (Stack : ที่พักข้อมูลชั่วคราวในกรณีที่ซีพียูมีการกระโดดไปทำงานโปรแกรมย่อย) การเข้าถึงหน่วยความจำในส่วนนี้ต้องใช้การเข้าถึงในระดับไบต์

2.1.5 หน่วยความจำรีจิสเตอร์

หน่วยความจำส่วนบน (80H-FFH) เป็นส่วนที่เข้าถึงข้อมูลได้โดยตรง และเป็นส่วนที่สำคัญมาก เพราะทำหน้าที่เป็นรีจิสเตอร์ที่ใช้งานเฉพาะ ซึ่งเป็นการทำงานหลักของไมโครคอนโทรลเลอร์ เช่น การคำนวณ การตั้งเวลา/นับเวลา การอินเทอร์รัปต์ การส่งข้อมูลแบบอนุกรม และรีจิสเตอร์อื่นๆ ที่จำเป็น ดังรูปที่ 2.8

ตำแหน่งข้อมูล 1 ไบต์

อ้างโดยตรง (direct byte address)	bit address (MSB)							(LSB)	รีจิสเตอร์ที่ใช้งานเฉพาะ (Special Function Register)
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B Register
E0H	E7	E6	E5	E4	E3	E2	E1	E0	Acc (Accumulator)
	C	AC	FO	RS1	RS0	OV		P	
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW (Program Status Word)
	PCT		PT2	PS	PT1	PX1	PT0	PX0	
B8H	BF	BE	BD	BC	BB	BA	B9	B8	IP (Interrupt Priority Control)
B0H	B7	B6	B5	B4	B3	B2	B1	B0	**Port P3**
	EA		ET2	ES	ET1	EX1	ET0	EX0	
A8H	AF		AD	AC	AB	AA	A9	A8	IE (Interrupt Enable Control)
A0H	A7	A6	A5	A4	A3	A2	A1	A0	**Port P2**
99H	ใช้ได้เป็นไบต์เท่านั้น								SBUF (Serial Data Buffer)
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
98H	9F	9E	9D	9C	9B	9A	99	98	SCON (Serial Control)
90H	97	96	95	94	93	92	91	90	**Port P1**
8DH	ใช้ได้เป็นไบต์เท่านั้น								TH1 (Timer 1 High Byte)
8CH	ใช้ได้เป็นไบต์เท่านั้น								TH0 (Timer 0 Low Byte)
8BH	ใช้ได้เป็นไบต์เท่านั้น								TL1 (Timer 1 High Byte)

8AH	ใช้ได้เป็นไบต์เท่านั้น								TL0 (Timer 0 Low Byte)
89H	ใช้ได้เป็นไบต์เท่านั้น								TMOD (Timer/Counter Mode Control
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
88H	8F	8E	8D	8C	8B	8A	89	88	TCON (Timer/Counter Control)
87H	ใช้ได้เป็นไบต์เท่านั้น								PCON (Power Control)
83H	ใช้ได้เป็นไบต์เท่านั้น								DPH (Data Pointer High)
82H	ใช้ได้เป็นไบต์เท่านั้น								DPL (Data Pointer Low)
81H	ใช้ได้เป็นไบต์เท่านั้น								SP (Stack Pointer)
80H	87	86	85	84	83	82	81	80	**Port P0**

รูปที่ 2.8 พื้นที่ใช้งานของหน่วยความจำภายในส่วนบน

การกำหนดพื้นที่หน่วยความจำภายในที่ใช้เป็นตำแหน่งรีจิสเตอร์ต่างๆ ทำให้การเขียนคำสั่งควบคุมสามารถเข้าถึงค่ารีจิสเตอร์ได้ 2 วิธี คือ เข้าถึงโดยใช้ตำแหน่งหน่วยความจำและใช้ชื่อของรีจิสเตอร์ ในไมโครคอนโทรลเลอร์ MCS-51 ได้ออกแบบค่ารีจิสเตอร์บางตัวให้สามารถเข้าถึงได้ในระดับบิตและไบต์ ดังตารางที่ 2.1

ตารางที่ 2.1 ชื่อและตำแหน่งรีจิสเตอร์ในไมโครคอนโทรลเลอร์ MCS-51

สัญลักษณ์	ชื่อ	ตำแหน่ง (address)
*Acc	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Point	81H
DPTR	Data Pointer 2 ไบต์ DPH,DPL	
DPL	ไบต์ต่ำ (Low Byte)	82H
DPH	ไบต์สูง (High Byte)	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H

ตารางที่ 2.1 ชื่อและตำแหน่งรีจิสเตอร์ในไมโครคอนโทรลเลอร์ MCS-51 (ต่อ)

สัญลักษณ์	ชื่อ	ตำแหน่ง (address)
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	0B8H
*TCON	Timer/Counter Control	0A8H
*T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High byte	8CH
TL0	Timer/Counter 0 Low byte	8AH
TH1	Timer/Counter 1 High byte	8DH
TL1	Timer/Counter 1 Low byte	8BH
#TH2	Timer/Counter 2 High byte	0CDH
#TL2	Timer/Counter 2 High byte	0CCH
#RCAP2H	T/C 2 Capture Reg. High byte	0CBH
#RCAP2L	T/C 2 Caoture Reg. Low byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

หมายเหตุ * เป็นรีจิสเตอร์ที่สามารถเข้าถึงระดับบิต

มีใช้เฉพาะรุ่น 80x52 (AT89xx52)

2.1.6 การทำงานของรีจิสเตอร์

- **รีจิสเตอร์ A (Accumulator = Acc)**

เป็นรีจิสเตอร์ขนาด 8 บิต สามารถเข้าถึงได้ระดับบิต ทำหน้าที่หลักในการคำนวณทางคณิตศาสตร์และลอจิก เก็บผลลัพท์จากการประมวลผล และสามารถเก็บข้อมูลขนาด 8 บิตได้ อยู่ที่ตำแหน่ง E0H

- **รีจิสเตอร์ B (B Register)**

เป็นรีจิสเตอร์ขนาด 8 บิด ใช้เก็บข้อมูลทั่วไป และยังมีหน้าที่พิเศษคือเก็บข้อมูลตัวกระทำการคูณหรือการหาร เช่น รีจิสเตอร์ A เก็บค่าตัวตั้ง และรีจิสเตอร์ B เป็นตัวคูณหรือหาร สามารถเข้าถึงได้ระดับบิต อยู่ที่ตำแหน่ง 0F0H

- **รีจิสเตอร์สถานะ PSW (Program Status Word)**

ทำหน้าที่เก็บสภาวะการทำงานของคำสั่ง มีขนาด 7 บิต อยู่ที่ตำแหน่งหน่วยความจำ 0D0H ซึ่งมีการเก็บสภาวะการทำงานของแต่ละบิตดังนี้

D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	-	P

CY (Carry Flag) ทำหน้าที่เก็บค่าตัวทดเมื่อมีการกระทำทางคณิตศาสตร์และลอจิก ถ้าผลลัพท์จากการประมวลผลที่ได้มีค่าเกิน 8 บิต (0FHH) บิตที่เกินจะเก็บไว้ที่บิต CY ซึ่งเป็นบิตตัวทด

AC (Auxiliary Carry Flag) เป็นตัวทดช่วยเมื่อมีการคำนวณ และเกิดการยืมค่าหรือเกิดตัวทดขึ้นระหว่างบิตที่ 3 กับบิตที่ 4 ของค่ารีจิสเตอร์ บิตนี้จะมีค่าลอจิกเป็น"1" ส่วนมากใช้กับระบบเลขฐานสอง (Binary Code Decimal)

F0 (Flag 0) ทำหน้าที่เป็นตัวเก็บสถานะทั่วไป ถ้ามีการตั้งค่าสถานะนี้ไว้ เมื่อมีการกระทำตามคำสั่งที่มีผลต่อสถานะ (Flag) ก็จะไม่มีผลกระทบต่อค่าที่กำหนดไว้

RS1 และ RS0 (Register Select 1และ0) ทำหน้าที่เลือกใช้ตำแหน่งของรีจิสเตอร์ R0-R7 มีอยู่ 4 ตำแหน่ง ตำแหน่งละ 8 ตัว (R0-R7) สามารถเลือกใช้ได้โดยการตั้งค่าบิตดังตารางที่ 2.2

ตารางที่ 2.2 ตำแหน่งรีจิสเตอร์ R0-R7

RS1	RS0	ตำแหน่ง R0-R7
0	0	Bank 0 ตำแหน่ง 00H-07H
0	1	Bank 1 ตำแหน่ง 08H-0FH
1	0	Bank 2 ตำแหน่ง 10H-17H
1	1	Bank 3 ตำแหน่ง 18H-1FH

OV (Overflow) เป็นค่าแสดงสถานะเมื่อมีการคำนวณหรือการกระทำทางลอจิกผลลัพท์ที่ได้จะมีค่าเกินกว่าที่ตำแหน่งหน่วยความจำจะรับได้ ทำให้สถานะนี้เป็น "1" และยังสามารถใช้เป็นค่าแสดงผลลัพท์ที่เป็นลบ

(– บิตนี้ไม่ได้ใช้งาน ผู้ใช้สามารถนำไปใช้งานได้)

P (Parity Flag) ใช้ตรวจสอบค่าที่เก็บอยู่ในรีจิสเตอร์ A แต่ละบิตมีค่าบิตเป็น 1 รวมกันเป็นจำนวนคู่หรือคี่ ถ้ามีจำนวนเป็นคู่ ค่าสถานะนี้จะเป็น "1" แต่ถ้าเป็นจำนวนคี่จะมีค่าสถานะเป็น "0"

- **รีจิสเตอร์สแตก (SP = Stack Pointer)**

เป็นรีจิสเตอร์สแตกขนาด 8 บิต มีตำแหน่งอยู่ที่ 81H ใช้เก็บตำแหน่งตัวชี้ค่าสแตก ใช้เก็บค่าตำแหน่งของโปรแกรมหลัก เมื่อไมโครคอนโทรลเลอร์กระโดดไปทำงานโปรแกรมย่อยเสร็จ ไมโครคอนโทรลเลอร์จะกลับมาทำงานที่โปรแกรมหลักอีกที ก่อนที่จะกระโดดไป ไมโครคอนโทรลเลอร์จะเก็บค่าตำแหน่งไว้ที่สแตกก่อน ในกรณีของการเขียนโปรแกรมย่อยหลายๆ โปรแกรม ค่าสแตกก็จะมีการเก็บค่าตำแหน่งมาก พื้นที่ที่ต้องใช้งานก็จะเพิ่มขึ้น ค่ารีจิสเตอร์สแตกสามารถกำหนดตำแหน่งใหม่ได้ แต่ถ้าไม่กำหนดไมโครคอนโทรลเลอร์ MCS-51 จะกำหนดตำแหน่งไว้ที่ 07H

- **รีจิสเตอร์ตัวนับขั้นตอนในโปรแกรม (PC = Program Counter)**

เป็นรีจิสเตอร์ขนาด 16 บิต มีหน้าที่บอกตำแหน่งคำสั่งการทำงานของไมโครคอนโทรลเลอร์แต่ละคำสั่งจะมีขนาดความยาวโค้ดคำสั่งต่างกัน ซึ่งมีความสำคัญมากในการตรวจสอบการเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์ประมวลผลและปฏิบัติตามลำดับขั้นตอนแผนงานที่วางไว้ รีจิสเตอร์ตัวนับขั้นตอนในโปรแกรม (PC) จะเป็นตัวเก็บค่าตำแหน่งทุกตำแหน่งที่ไมโครคอนโทรลเลอร์ทำงานอยู่ปัจจุบัน

- **รีจิสเตอร์ตัวชี้ข้อมูล (DPTR = Data Pointer)**

เป็นรีจิสเตอร์ขนาด 8 บิต 2 ชุด ประกอบด้วยรีจิสเตอร์ไบต์ต่ำ (DPL) และไบต์สูง (DPH) อยู่ที่ตำแหน่ง 82H และ 83H สามารถใช้ได้ 8 บิต และใช้รวมเป็นขนาด 16 บิต ทำหน้าที่กำหนดตำแหน่งข้อมูลในหน่วยความจำและอุปกรณ์ภายนอกที่ไมโครคอนโทรลเลอร์ต้องการติดต่อ

- **รีจิสเตอร์พอร์ต (Port Register)**

ไมโครคอนโทรเลอร์ MCS-51 ได้กำหนดให้การติดต่อทุกอย่างกระทำผ่านพอร์ต ซึ่งมีอยู่ 4 พอร์ต มีขนาดพอร์ตละ 8 บิต คือ พอร์ต 0 (P0 = 80H) พอร์ต 1 (P1 = 90H) พอร์ต 2 (P2 = A0H) และ พอร์ต 3 (P3 = B0H) ทุกพอร์ตสามารถใช้เป็นอินพุตและเอาต์พุตที่ควบคุมได้ระดับบิต

- **รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (SBUF = Serial Data Buffer)**

เป็นรีจิสเตอร์ที่ทำหน้าที่รับส่งข้อมูลแบบอนุกรม มีขนาด 8 บิต อยู่ที่ตำแหน่ง 99H ซึ่งปกติใช้การติดต่อข้อมูลอนุกรม มีบัฟเฟอร์อยู่ 2 ชุด คือ บัฟเฟอร์สำหรับส่งข้อมูล (Transmit Buffer Register) ผ่านไปยังขา TxD และบัฟเฟอร์สำหรับรับข้อมูล (Receive Buffer Register) ผ่านมายังขา RxD เมื่อต้องการที่จะส่งข้อมูลอนุกรมสามารถเขียนข้อมูลไปยังรีจิสเตอร์ SBUF และข้อมูลจะถูกส่งไปยังบัฟเฟอร์ เพื่อให้ไมโครคอนโทรลเลอร์ส่งออกไปภายนอก

- **รีจิสเตอร์เกี่ยวกับเวลา (Timer Register)**

เป็นรีจิสเตอร์ขนาด 16 บิต แบ่งเป็นรีจิสเตอร์คู่ คือ ไบต์ต่ำและไบต์สูง ใช้เก็บค่าตัวนับเวลา (Counter) ภายในไมโครคอนโทรลเลอร์ เพื่อใช้เป็นฐานเวลา หรือใช้จับเวลานับจำนวนสัญญาณนาฬิกา (Pulse) ในไมโครคอนโทรลเลอร์ MCS-51 ประกอบด้วย TH0 TL0 TH1 TL1 และ TH2 TL2

- **รีจิสเตอร์แคปเจอร์ (Capture Register)**

เป็นรีจิสเตอร์ขนาด 16 บิต คือ RCAP2L ไบต์ต่ำ และ RCAP2H ไบต์สูง ใช้ร่วมกับ TL2 TH2 ทำหน้าที่ให้ไมโครคอนโทรลเลอร์ตรวจจับเวลาการเปลี่ยนแปลงสถานะลอจิกที่ขา T2EX เพื่อใช้วัด คาบเวลา ความถี่ และการเปลี่ยนแปลงสัญญาณนาฬิกาที่ขา T2EX

- **รีจิสเตอร์ควบคุม (Control Register)**

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานส่วนต่างๆ ของไมโครคอนโทรลเลอร์ ซึ่งประกอบด้วยรีจิสเตอร์ต่อไปนี้

รีจิสเตอร์ PCON (Power Control Register) : ทำหน้าที่ควบคุมการทำงานของไมโครคอนโทรลเลอร์ โดยการกำหนดให้สัญญาณนาฬิกาหยุดทำงาน ทำให้ส่วนต่างๆ ภายในไมโครคอนโทรลเลอร์หยุดทำงานด้วย ซึ่งเป็นการลดพลังงานเมื่อไม่ต้องการให้ไมโครคอนโทรลเลอร์ทำงาน (Sleep Mode)

รีจิสเตอร์ SCON (Serial Control Register) : เป็นรีจิสเตอร์ที่ทำหน้าที่ควบคุมระบบวงจรการสื่อสารแบบอนุกรมภายในไมโครคอนโทรลเลอร์

รีจิสเตอร์ TCON T2CON และ TMOD T2MOD : ใช้ควบคุมการทำงานของวงจรจับเวลา/นับเวลาภายในไมโครคอนโทรลเลอร์ (Timer/Counter)

รีจิสเตอร์ IE และ IP (Interrupt Enable Control และ Interrup Priority Control) : เป็นรีจิสเตอร์ที่ใช้ควบคุมการอินเทอร์รัปต์ของไมโครคอนโทรลเลอร์ มี IE (Interrupt Enable) เป็นรีจิสเตอร์ตอบสนองการทำงานของสัญญาณอินเตอร์รัปต์ และรีจิสเตอร์ IP (Priority Interrupt) ทำหน้าที่จัดลำดับความสำคัญของสัญญาณตอบสนองการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์

ในกรณีเริ่มต้นการทำงานใหม่ของไมโครคอนโทรลเลอร์หรือรีเซตเครื่อง ค่ารีจิสเตอร์ต่างๆ ที่อยู่ภายในไมโครคอนโทรลเลอร์จะถูกเซตค่าใหม่ ดังตารางที่ 2.3

ตารางที่ 2.3 ค่ารีจิสเตอร์ต่างๆ เมื่อเริ่มต้นทำงานของไมโครคอนโทรลเลอร์ MCS-51

สัญลักษณ์	ชื่อ	เลขฐานสอง
*Acc	Accumulator	00000000
*B	B Register	00000000
*PSW	Proggram Status Word	00000000
SP	Stack Point	00000111
DPTR	Data Pointer 2 ไบต์ DPH,DPL	
DPL	ไบต์ต่ำ (Low byte)	00000000
DPH	ไบต์สูง (High byte)	00000000
*P0	Port 0	11111111
*P1	Port 1	11111111
*P2	Port 2	11111111
*P3	Port 3	11111111
*IP	Interrupt Priority Control	8051 xxx00000, 8052 xx000000
*IE	Interrupt Enable Control	8051 0xx00000, 8052 0x000000
TMOD	Timer/Counter Mode Control	00000000
*TCON	Timer/Counter Control	00000000
*+T2CON	Timer/Counter 2 Control	00000000
TH0	Timer/Counter 0 High byte	00000000
TL0	Timer/Counter 0 Low byte	00000000
TH1	Timer/Counter 1 High byte	00000000
TL1	Timer/Counter 1 Low byte	00000000
#TH2	Timer/Counter 2 High byte	00000000
#TL2	Timer/Counter 2 High byte	00000000
#RCAP2H	T/C 2 Capture Reg. High byte	00000000
#RCAP2L	T/C 2 Caoture Reg. Low byte	00000000
*SCON	Serial Control	00000000
SBUF	Serial Data Buffer	ไม่ได้กำหนด
PCON	Power Control	0xxxxxxxx

หมายเหตุ *รีจิสเตอร์ที่สามารถเข้าถึงได้ระดับบิต +มีใช้เฉพาะรุ่น 80x52 (AT89xx52)

2.1.7 รีจิสเตอร์ฟังก์ชันพิเศษ

เป็นรีจิสเตอร์ที่ใช้การควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 มีด้วยกัน 22 ตัวสำหรับเบอร์ AT89C51 และ 28 ตัวในเบอร์ AT89C52 และ อนุกรม AT89Sxx ทั้งนี้เนื่องจากใน AT89C52 และ AT89Sxx มีจำนวนไทเมอร์เคาน์เตอร์มากกว่า AT89C51

รีจิสเตอร์ SFR มีแอดเดรสอยู่ระหว่าง 80H-FFH ในพื้นที่ของหน่วยความจำข้อมูลส่วนบน สามารถเข้าถึงโดยตรง (Direct Addressing) สำหรับรายละเอียดเบื้องต้นของรีจิสเตอร์ SFR มีดังนี้

- **รีจิสเตอร์แสดงสถานะของโปรแกรม (Program Status Word : PSW)**

เป็นรีจิสเตอร์ขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต จึงสามารถกำหนดค่าในแต่ละบิตของรีจิสเตอร์ตัวนี้ได้อย่างอิสระ มีแอดเดรสอยู่ที่ D0H ทำหน้าที่เก็บสถานะของการทำงานของโปรแกรมในขณะนั้นเรียกสถานะต่างๆ ของโปรแกรมว่า แฟลก (Flag) เมื่อซีพียูกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้วเกิดการเปลี่ยนแปลงสถานะขึ้น ผลของการเปลี่ยนแปลงนั้นจะมาปรากฏที่บิตต่างๆ ของรีจิสเตอร์ PSW รายละเอียดของแต่ละบิตในรีจิสเตอร์ PSW สามารถแสดงได้ดังรูปที่ 2.10

จะเห็นได้ว่า นอกจากรีจิสเตอร์ PSW ถูกใช้ในการเก็บสถานะของโปรแกรมแล้ว ที่บิต RS0 และ RS1 ยังใช้ในการเลือกแบงก์ของหน่วยความจำส่วนล่าง ซึ่งเป็นพื้นที่ของรีจิสเตอร์ R0-R7 ด้วยโดยปกติแล้วในการใช้งานรีจิสเตอร์ R0-R7 มักนิยมใช้แบงก์ 0 เป็นลำดับแรก หากไม่เพียงพอจึงเลือกในแบงก์อื่นๆ มาใช้

- **แอกคิวมูเลเตอร์ (Accumulator) หรือ ACC**

เป็นรีจิสเตอร์ขนาด 8 บิต ทำหน้าที่ในการเก็บข้อมูลที่จะส่งให้กับหน่วยทำงานภายในหน่วยประมวลผลกลาง และเก็บผลลัพธ์ที่ได้จากการทำงานเท่านั้น การทำงานของรีจิสเตอร์มีลักษณะเช่นเดียวกับตัวแอกคิวมูเลเตอร์ของโปรเซสเซอร์ทั่วไป การใช้งานในโปรแกรมซึ่งใช้เรียกเป็น รีจิสเตอร์ A

2.1.8 ชุดคำสั่งของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ประกอบด้วยคำสั่งทั้งหมดจำนวนมาก ซึ่งนำมาแสดงไว้ในตารางของชุดคำสั่งต่างๆ โดยสามารถจัดกลุ่มคำสั่งเหล่านี้ตามลักษณะและหน้าที่การทำงานที่คล้ายคลึงกันเพื่อความสะดวกในการใช้งานได้ดังนี้

- **กลุ่มการถ่ายเทข้อมูล**

กลุ่มถ่ายเทข้อมูล คือ กลุ่มคำสั่งในการโอนย้ายข้อมูล ทำหน้าที่ในการโอนย้ายข้อมูลระหว่างรีจิสเตอร์หรือหน่วยความจำภายในแรม ชุดคำสั่งในการถ่ายเทข้อมูลของแรมภายในนั้นแสดงดังตารางที่ 2.1 ซึ่งเวลาที่ใช้ในหนึ่งคำสั่งนั้น จะเท่ากับคสบเวลาขณะความถี่ในการทำงานของหน่วยประมวลผลกลางที่ความถี่ 12 เมกะเฮิรตซ์ และรายละเอียดของแต่ละคำสั่งมีดังนี้

MOV : จะทำงานในลักษณะเป็นการถ่ายเทข้อมูลที่มีขนาดเป็นไบต์ หรือ บิต จากแหล่งกำเนิดเข้าสู่ตัวรับข้อมูลในฟิลด์โอเปอร์แรนด์

PUSH : จะทำงานโดยเพิ่มค่ารีจิสเตอร์ SP ก่อนแล้วจึงทำการถ่ายเทข้อมูลขนาด 1 ไบต์จากแหล่งกำเนิดไปบริเวณสแต็กตามตำแหน่งที่รีจิสเตอร์ SP กำหนด

POP : ถ่ายเทข้อมูลขนาด 1 ไบต์จากบริเวณตำแหน่งที่รีจิสเตอร์ SP กำหนดไปยังรีจิสเตอร์ที่โอเปอร์แรนด์กำหนดและหลังจากนั้นรีจิสเตอร์ส SP จะลดค่าลง

XCH : แลกเปลี่ยนข้อมูลขนาด 1 ไบต์ ระหว่างแหล่งกำเนิดโอเปอร์แรนด์กับรีจิสเตอร์ AXCHD คำสั่งในการแลกเปลี่ยนขนาดนิบเบิลทางอันดับต่ำของแหล่งกำเนิดโอเปอร์แรนด์กับนิบเบิลอันดับต่ำลงของแอกคิวมูเลเตอร์ ตัวอย่างเช่น ทำการเลื่อนข้อมูลไป 2 ไบต์ทางขวามือซึ่งจะมี 2 วิธีคือ ใช้คำสั่ง MOV หรือใช้คำสั่ง XCH รายละเอียดการใช้คำสั่งทั้ง 2 แบบ แสดงดังตารางที่ 2.4

ตารางที่ 2.4 ชุดคำสั่งการถ่ายเทข้อมูลในแรมภายใน

นีโมนิค	การกระทำ	โหมดการแอดเดรส				เวลาการเอ็กซีคิวต์
		Dir	Ind	Reg	Imm	
MOV<SCR>	A=<SCR>	X	X	X	X	1
MOVDEST>,A	<DEST>=<SCR>	X	X	X		1
MOV<DEST>,<SCR>	<DEST>=<SCR>	X	X	X	X	2
MOV DRPT,#Data16	DRPT=6Bit Immediate Constant				X	2
PUSH<SCR>	INC SP:MOV"@SP",<SCR>	X				2
POP<DEST>	MOV<DEST>,"@SP",DEC SP	X				2
XCH A<BYTE>	ACC AND<Byte>Exchange Data	X	X	X		1
XCHD A,@RI	ACC AND @RI Exchange Low Nibbles		X			1

- **กลุ่มคำสั่งทางคณิตศาสตร์**

ช่วงเวลาการทำงานของแต่ละคำสั่งการบวก ลบ คูณ และหารข้อมูลภายในตัวรีจิสเตอร์ต่างๆนั้นจะกำหนดที่ความถี่ของสัญญาณนาฬิกาเมกะเฮิรตซ์ คำสั่งทางคณิตศาสตร์ส่วนใหญ่ใช้เวลา 1 μs ยกเว้นคำสั่ง INC และ DPTR ซึ่งใช้เวลา 2 μs โดยที่การคูณและการหารใช้เวลา 4 μs โดยมีรายละเอียดการใช้คำสั่งมีดังนี้

INC : เป็นการเพิ่มค่าในโอเปอร์แรนด์และใส่ค่าใหม่กลับเข้าที่โอเปอร์แรนด์

DEC : เป็นการลบออกจากตัวเลขที่อยู่ในแหล่งกำเนิดโอเปอร์แรนด์และนำผลลัพธ์ที่ได้มาเก็บไว้ทีตัวโอเปอร์แรนด์นั้น

ADD : เป็นการบวกค่าในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์

ADDC : เป็นการบวกค่าต่างๆในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์ และบวกกับบิตทดด้วย

SUBB : เป็นการนำเลขที่แหล่งกำเนิดโอเปอร์แรนด์ลบออกจากค่าใน A และนำค่าบิตตัวทดมาลบออกอีก และผลลัพธ์ที่ได้นำมาใส่ลงในแอกคิวมูเลเตอร์ A

MUL : เป็นการคูณแบบไม่คิดตัวเครื่องหมายของค่าที่อยู่ในแอกคิวมูเลเตอร์กับค่าในรีจิสเตอร์ B แล้วได้ผลลัพธ์ 2 ไบต์ นำมาเก็บไว้ที่ AB โดย A จะรับอันดับต่ำส่วน B จะรับอันดับสูง

DIV : เป็นคำสั่งในการหารแบบไม่คิดเครื่องหมายที่อยู่ในแอกคิวมูเลเตอร์แล้วหารตัวเลขในรีจิสเตอร์ B แล้วนำผลลัพธ์ไปเก็บในแอกคิวมูเลเตอร์ และเศษของการหารตัวเลขจะเก็บไว้ในรีจิสเตอร์ B

DA : สำหรับการบวกกันทางตัวเลข BCD เป็นการปรับค่ารวม ซึ่งเป็นผลมาจากการบวกกันทางไบนารี่ของระบบตัวเลข BCD ขนาด 2 หลักสองจำนวน การปรับค่าตัวเลขผลรวมด้วยการใช้คำสั่ง DA จะได้ผลลัพธ์กลับมาที่แอกคิวมูเลเตอร์ รายละเอียดการใช้คำสั่งแสดงได้ดังตารางที่ 2.5

ตารางที่ 2.5 ชุดคำสั่งทางคณิตศาสตร์

นีโมนิค	การกระทำ	โหมดการแอดเดรส				เวลาการเอ็กซีคิวต์
		Dir	Ind	Reg	Imm	
ADD A,<byte>	A=A+<byte>	X	X	X	X	1
ADDC A,<byte>	A=A+<byte>+C	X	X	X		1
SUBB A,<byte>	A=A-<byte>-C	X	X	X	X	2
INC A	A=A+1	Accumulator Only				1
INC <byte>	<byte>=<byte>+1	X	X	X		1
INC DPTR	DPTR=DPTR+1	Data Pointer Only				2
DEC A	A=A-1	Accumulator Only				1
DEC <byte>	<byte>=<byte>-1	X	X	X		1
MUL AB	B:A=B x A	ACC and B Only				4
DIV	A=Int[A/B]	ACC and B Only				4
	B=Mod[A/B]					
DA A	Decimal Adjust	Accumulator Only				1

- **กลุ่มคำสั่งทางตรรกศาสตร์หรือแบบลอจิก**

ทำหน้าที่เกี่ยวกับการประมวลผลแบบลอจิกต่างๆ เช่น การAND OR หรือ EX-OR ระหว่างข้อมูลในรีจิสเตอร์ A นั่นเอง โดยมีการใช้คำสั่งดังนี้

CPL : เป็นการใช้คำสั่งกลับค่าหรือการคอมพลีเมนต์นั่นเอง ข้อมูลในแอกคิวมูเลเตอร์จะไม่มีผลใดๆต่อค่าของแฟลก หรือการอ้างถึงตำแหน่งแอดเดรสนั้นตามบิตนั้นๆ

RL RLC RR RRC SWAP : ทั้ง 5 คำสั่งนี้เป็นคำสั่งในการทำงานการวนบิตบนตัวของแอกคิวมูเลเตอร์ซึ่ง RL เป็นการวนบิตทางซ้าย RLC เป็นการทำการวนทางซ้ายผ่านบิตทด RRC เป็นการวนขวาผ่านบิตทด และ SWAP เป็นการวนซ้ายสี่ครั้ง

ANL : เป็นการ ADD กันทางตรรกศาสตร์ ระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ซึ่งจะสั่งให้ทำงานในรูปแบบของตรรกศาสตร์ทางข้อมูลขนาดเป็นไบต์หรือบิต รายละเอียดการใช้คำสั่งของชุดคำสั่งทางตรรกศาสตร์แสดงได้ดังตารางที่ 2.6

เป็นความสามารถพิเศษของไมโครคอนโทรลเลอร์ MCS-51 ที่จะดำเนินการประมวลผลในระดับบิต โดยมีชุดคำสั่งที่จัดการโดยตรง ทุกคำสั่งจะเข้าถึงข้อมูลโดยตรงในระดับบิต โดยมีการบิตแอดเดรสได้ตั้งแต่ 00H-7FH ในพื้นที่ 128 บิต หน่วยความจำข้อมูลภายในและบิตแอดเดรส 80H-FFH ในบริเวณกลุ่มรีจิสเตอร์ฟังก์ชันพิเศษ (SFR) โดยรายละเอียดการใช้คำสั่งแสดงได้ดังตารางที่ 2.7

- **กลุ่มคำสั่งในการกระโดดไปยังตำแหน่งต่างๆภายในโปรแกรม**

ใช้ในการเปลี่ยนลำดับของการประมวลผลภายในโปรแกรมให้ไปทำงานยังส่วนต่างๆตามต้องการ แทนที่จะดำเนินการไปเป็นลำดับต่อเนื่อง โดยที่คำสั่ง JMP จะแบ่งเป็น 3 ลักษณะ คือ SJMP LJMP AJMP ซึ่งในแต่ละคำสั่งจะมีข้อแตกต่างของลักษณะและระยะทางในการกระโดดที่ต่างกัน คำสั่ง JMP ซึ่งเป็นแบบโมนีชิกที่สามารถจะใช้ได้ โดยมีรายละเอียดการใช้งานของคำสั่งดังต่อไปนี้

SJMP : เป็นการกระโดดแบบการย้ายอันดับตำแหน่งแอดเดรสตำแหน่งเดิมซึ่งจะสามารถกระโดดได้ -128 ถึง +127 ไบต์

AJMP : การกระโดดแบบนี้จะสามารถกระโดดได้ไกลสุดประมาณ 2 กิโลไบต์ ซึ่งใช้หน่วยความจำเพียง 2 ไบต์เท่านั้นในการกำหนด

LJMP : การกระโดดแบบนี้จะสามารถกระโดดได้ไกลสุดประมาณ 64 กิโลไบต์ ซึ่งใช้หน่วยความจำเพียง 3 ไบต์เท่านั้นในการกำหนด

JMP@A+DPTR : เป็นการควบคุมการกระโดดไปยังโปรแกรมที่ต้องการเฉพาะภายในส่วนต่างๆ โดยสามารถแสดงรายละเอียดการใช้คำสั่งได้ดังตารางที่ 2.8

ตารางที่ 2.6 ชุดคำสั่งของกลุ่มทางตรรกศาสตร์

นีโมนิค	การกระทำ	โหมดการแอดเดรส				เวลาการเอ็กซีคิวต์
		Dir	Ind	Reg	Imm	
ANL A,<byte>	A=A AND <byte>	X	X	X	X	1
ANL <byte>,A	<byte>=<byte>AND A	X				1
ANL <byte>,#Data	<byte>=<byte>AND # Data	X				2
ORL A,<byte>	A=A OR <byte>	X	X	X	X	1
ORL <byte>,A	<byte>=<byte>OR A	X				1
ORL <byte>,#Data	<byte>=<byte>OR # Data	X				2
XRL A,<byte>	A=A XOR <byte>	X	X	X	X	1
XRL <byte>,A	<byte>=<byte> XOR A	X				1
XRL <byte>,#Data	<byte>=<byte> XOR # Data	X				2
CLR A	A=00H	Accumulator Only				1
CPL A	A=NOT A	Accumulator Only				1
RL A	Rotate ACC Left 1 bit	Accumulator Only				1
RLC A	Rotate Left Through Carry	Accumulator Only				1
RR A	Rotate ACC Right 1 bit	Accumulator Only				1
RRC A	Rotate Right Through Carry	Accumulator Only				1
SWAP A	Swap Nibbles in A	Accumulator Only				1

ตารางที่ 2.7 ชุดคำสั่งกระโดดโดยไม่มีเงื่อนไข

นีโมนิค	การกระทำ	โหมดการแอดเดรส				เวลาการเอ็กซีคิวต์
		Dir	Ind	Reg	Imm	
J rel	Jump if A=0	Accumulator Only				2
JNZ rel	Jump if A=0	Accumulator Only				2
DJNZ <byte>,rel	Decrement and Jump if not zero	X		X		2
CJNE A, <byte>,rel	Jump if A=<byte>	X			X	2
CJNE <byte>,#data,rel	Jump if A=#data		X	X		2

ตารางที่ 2.8 ชุดคำสั่งบูลีน

นีโมนิค	การกระทำ	เวลาการเอ็กซีคิวต์
ANL C,bit	C = CAND bit	2
ANL C,/bit	C + C AND (NOT bit)	2
ORL C,bit	C = C OR bit	2
ORL C/bit	C = C OR (NOT bit)	2
MOV C,bit	C = bit	1
MOV bit,C	Bit = C	2
CLR C	C = 0	1
CLR bit	Bit = 01	1
SETB C	C = 1	1
SETB bit	Bit = 1	1
CPL C	C = NOT C	1
CPL bit	Bit = NOT bit	1
JC Rel	Jump if C = 1	2
JNC Rel	Jump if C = 0	2
JB bit,Rel	Jump if C = 1	2
JnNB bit,Rel	Jump if C = 0	2
JBC bit,Rel	Jump if C = 1;CLR bit	2

2.1.9 โครงสร้างการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ MCS-51

จากแผนภาพโครงสร้างระบบอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ MCS-51 สัญญาณที่เข้ามาทำการอินเตอร์รัปต์ MCS-51 นั้นเกิดขึ้นได้ 5 ลักษณะตามตารางที่ข้อมูลในรูปที่ 2.9 ซึ่งจะเห็นได้ว่าสามารถที่จะกำหนดเลือกเพื่อยินยอม (หรือเอนเอเบิล : enable) และห้าม (ดิสเอเบิล : disable) ไม่ให้มีการอินเตอร์รัปต์แต่ละประเภทได้โดยการกำหนดบิตของข้อมูลที่เกี่ยวข้องซึ่งมักจะอยู่ภายในรีจิสเตอร์ TCON และ SCON

ตารางที่ 2.9 ชุดคำสั่งกระโดดโดยมีเงื่อนไขต่างๆ

นีโมนิค	การกระทำ	เวลาการเอ็กซีคิวต์
JMP Addr	Jump to Addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL Addr	Call subroutine at Addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

นอกจากนี้ยังมีตำแหน่งบิตภายในรีจิสเตอร์ IE (Interrupt enable register) ซึ่งทำหน้าที่เสมือนกับเป็นสวิตช์หลักที่ควบคุมสัญญาณอินเตอร์รัปต์ทั้งหมด หากว่ากำหนดไม่ให้มีการอินเตอร์รัปต์แล้ว การกำหนดบิตเพื่อห้ามหรือยินยอมของแต่ละอินเตอร์รัปต์ก็จะไม่มีผลใด ๆ เกิดขึ้น สัญญาณอินเตอร์รัปต์แต่ละประเภทยังสามารถกำหนดระดับความสำคัญ (Priority) ของการอินเตอร์รัปต์ได้สองลักษณะคือ ระดับความสำคัญมากหรือน้อย กล่าวคือขณะที่กำลังประมวลผลอยู่ภายในส่วนของโปรแกรมย่อยบริการอินเตอร์รัปต์ของสัญญาณที่มีระดับความสำคัญต่ำอยู่ ก็อาจจะถูกขัดจังหวะให้ไปประมวลผลของสัญญาณอินเตอร์รัปต์ที่มีระดับความสำคัญสูงกว่า แต่หากว่าเป็นสัญญาณอินเตอร์รัปต์ที่มีระดับความสำคัญระดับเดียวกันแล้ว ก็ต้องรอให้เสร็จสิ้นการประมวลผลที่กำลังดำเนินการอยู่ก่อน

2.1.10 การรีเซต

การรีเซต เป็นการบังคับให้มีการเริ่มต้นใหม่อีกครั้งหนึ่ง ซึ่งมักจะกระทำโดยการกำหนดสภาวะของสัญญาณที่ขารีเซตของไอซี MCS-51 ให้เป็นระดับลอจิกที่เหมาะสมเท่านั้น การรีเซตนี้ถือว่าเป็นการอินเตอร์รัปต์อย่างหนึ่งได้ แต่จะมีลักษณะต่างออกไปจากการอินเตอร์รัปต์ทั่วไป ซึ่งมีศัพท์เฉพาะเรียกว่า Non-Maskable Interrupt คือเป็นการอินเตอร์โดยไม่สนระดับความสำคัญ นอกจากนี้การดำเนินการของโปรแกรมก็แตกต่างออกไปด้วย โดยจะไม่มีการเก็บค่าของคำสั่งที่กำลังจะไปทำในลำดับต่อไปภายในรีจิสเตอร์ PC เมื่อมีการรีเซตเกิดขึ้นโดยโปรแกรมจะถูกสั่งให้กระโดดไปยังแอดเดรส 0000 ทันที ซึ่งตำแหน่งนี้จะเป็นตำแหน่งเริ่มต้นของการทำงานของไมโครคอนโทรลเลอร์ MCS-51 และค่าสภาวะต่างๆ ภายในไมโครคอนโทรลเลอร์จะถูกกำหนดกลับไปเป็นค่าเริ่มต้นใหม่อีกครั้ง ดังตารางที่ 2.10

ตารางที่ 2.10 ค่าเริ่มต้นในการใช้คำสั่งในกลุ่มรีจิสเตอร์

รีจิสเตอร์	ค่าเริ่มต้น (ฐานสิบหก)
PC	0000
DPTR	0000
A	00
B	00
SP	07
PSW	00
P0-P3	FF
IE	0XX0000(ฐานสอง)

2.2 อุปกรณ์รับแสง

อุปกรณ์รับแสง LDR (LSR-light sensitive resistor) มีชื่อที่เรียกกันอีกหลายชื่อ เช่น โฟโต้คอนดักตีฟเซล(Photoconductive cell) หรือ ตัวต้านทานไวแสง ส่วนใหญ่จะทำด้วยสารแคดเมียมซัลไฟด์ (CdS) หรือแคดเมียมซีนิไนด์ (CdSe) ซึ่งทั้งสองตัวนี้เป็นสารประเภทกึ่งตัวนำ เอามาฉาบลงบนแผ่นเซรามิกที่เป็นฐานรองแล้วต่อขาจากสารที่ฉาบไว้ออกมาภายนอก

รูปที่ 2.9 โครงสร้างของ LDR

โครงร่างของ LDR แสดงดังรูปที่ 2.9 ส่วนที่ขดเป็นแนวเล็กๆสีดำ ทำหน้าที่เป็นตัวต้านทานไวแสง และแนวสีดำนั้นจะแบ่งพื้นที่ของตัว LDR ออกเป็น 2 ข้าง คือส่วนที่เป็นตัวนำไฟฟ้าที่ทำหน้าที่สัมผัสกับตัวต้านทานไวแสง หรือเรียกว่า อิเล็กโทรด กับส่วนที่เป็นฐานเซรามิกและอุปกรณ์สำหรับหุ้มตัว LDR ซึ่งมีได้หลายแบบ

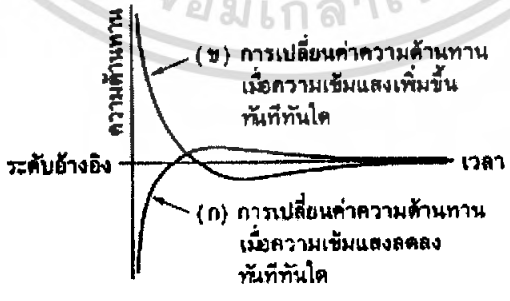
2.2.1 สมบัติทางแสง

การทำงานของ LDR จะมีหลักการทำงานแบบง่ายเพราะ LDR เป็นสารกึ่งตัวนำ เมื่อมีแสงตกกระทบลงไปที่ตัวของ LDR จะถ่ายทอดพลังงานให้กับสารที่ฉาบอยู่ ทำให้เกิดโฮลและอิเล็กตรอนวิ่งกันอย่างอิสระ การที่มีจำนวนโฮลและอิเล็กตรอนอิสระนี้มากจะทำให้มีความต้านทานลดลง ดังนั้นยิ่งความเข้มของแสงที่ตกกระทบมากเท่าไร ความต้านทานก็จะมีค่าน้อยลงไปด้วย

แสงที่ตกกระทบนั้นจะต้องเป็นแสงในช่วงความยาวคลื่นประมาณ 4,000 อังสตรอม ถึง 10,000 อังสตรอมเท่านั้น (1 อังสตรอม เท่ากับ 10^{-10} เมตร) ซึ่งก็เป็นช่วงคลื่นแคบๆเมื่อเทียบกับการทำงานของอุปกรณ์ไวแสงประเภทอื่นๆ แต่อย่างไรก็ตาม แสงในช่วงความยาวคลื่นนี้มีอยู่มากในแสงอาทิตย์ แสงจากหลอดไฟแบบไส้ และแสงจากหลอดฟลูออเรสเซนต์ แต่ความยาวคลื่นที่LDR จะตอบสนองไวที่สุดแล้วมีอยู่หลายความยาวคลื่นตามชนิดของ LDR เช่นชนิดที่ทำจากแคดเมียมซัลไฟด์จะไวต่อแสงที่มีความยาวคลื่นในช่วง 5,000 อังสตรอม ส่วน LDR ที่ทำจาก แคดเมียมซีลิไนด์ก็จะไวต่อความยาวคลื่นในช่วง 7,000 อังสตรอม ซึ่งอยู่ในช่วงของแสงอินฟราเรด

2.2.2 ผลตอบสนองทางไฟฟ้า

อัตราส่วนระหว่างความต้านทานของ LDR ในขณะที่ไม่มีแสงกับขณะที่มีแสง อาจจะเป็นได้ตั้งแต่ 100 เท่า 1,000 เท่า หรือ10,000 เท่าแล้วแต่รุ่น ซึ่งโดยทั่วไปแล้ว ค่าความต้านทานในขณะที่ไม่มีแสงจะอยู่ในช่วง ประมาณ 0.5 โอห์มขึ้นไป ในที่มืดสนิทอาจขึ้นไปได้มากกว่า 2 โอมห์ และในขณะที่มีแสงจะเป็นประมาณ 10 - 20 กิโลวัตต์ลงไป และอาจจะเหลือเพียงไม่ถึงโอห์มก็ได้ โดย LDR สามารถทนแรงดันสูงสุดได้มากกว่า 100 โวล์ท และกำลังสูญเสียอย่างต่ำประมาณ 50 มิลลิวัตต์



รูปที่ 2.10 ผลของการเปลี่ยนความเข้มแสงในทันทีทันใดกับ LDR

นอกเหนือจากคุณสมบัติต่างๆเหล่านี้แล้ว ยังมีคุณสมบัติอีกอย่างหนึ่งที่สำคัญ คือ ปรากฏการณ์ที่เกิดขึ้นจากความเข้มแสดงถึงการเปลี่ยนแปลงอย่างฉับพลัน ซึ่งแสดงดังรูปที่ 2.10 ถ้า LDR ได้รับแสงที่มีความเข้มสูงดังเส้น (ก) ความต้านทานจะมีค่าต่ำ และในทันทีที่ความเข้มของแสงถูกลดลงเหลือเพียงระดับอ้างอิง ความต้านทานจะค่อยๆ เพิ่มขึ้นไปจนถึงค่าความต้านทานที่ควรจะเป็นในระดับอ้างอิง แต่แทนที่จะไปหยุดอยู่ระดับอ้างอิง กลับเพิ่มเลยขึ้นไปอีกแล้วจึงจะลดลงมาอยู่ในระดับอ้างอิง เหมือนกับว่าเบรกไม่ค่อยดี และในทำนองเดียวกันถ้าเก็บไว้ในที่ความเข้มแสงน้อยๆ แล้วเปลี่ยนความเข้มเป็นระดับอ้างอิงทันที ดังในรูป (ข) ความต้านทานจะลดเลยต่ำลงมาจากระดับอ้างอิง แล้วจึงขึ้นไปใหม่ยิ่งความเข้มของแสงเท่ากัน LDR แบบแคดเมียมซีนิไนด์ จะใช้เวลา ในการเข้าสู่สภาวะที่ควรจะเป็นน้อยกว่าแบบแคดเมียมซัลไฟต์ แต่ก็จะวิ่งเลยไปไกลกว่าด้วย และอีกอย่างหนึ่งความเร็วในการเปลี่ยนระดับความต้านทานจากค่าหนึ่งไปอีกค่าหนึ่งช้ามาก ซึ่งจะอยู่ในช่วงของมิลลิวินาทีหรือบางทีก็เป็นวินาที จึงทำให้ LDR ใช้ได้กับงานความถี่ต่ำๆ เท่านั้น

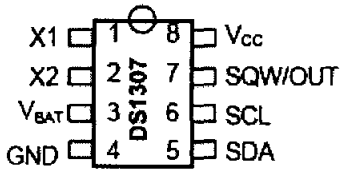
2.2.3 การประยุกต์ใช้อุปกรณ์รับแสงกับไมโครคอนโทรลเลอร์ MCS-51

ในโครงงานนี้มีการใช้อุปกรณ์รับแสง LDR ทั้งหมด 4 วงจร คือชั้นที่หนึ่ง 2 วงจร และชั้นที่สองอีก 2 วงจร โดยวงจรอุปกรณ์รับแสงแต่ละวงจรจะเชื่อมต่อเข้ากับไมโครคอนโทรลเลอร์แบบอนุกรมที่ ขา32 ขา33 ขา34 และขา35 ของไมโครคอนโทรลเลอร์

2.3 ไอซีสร้างฐานเวลา DS1307

ไอซีสร้างฐานเวลาเบอร์ DS1307 ที่สร้างฐานเวลาจริงให้แก่ระบบไมโครคอนโทรลเลอร์ โดย DS1307 จะให้ข้อมูลเกี่ยวกับเวลาทั้งหมด โดยละเอียดถึงหลักวินาที ซึ่งสามารถปรับเวลาให้ตรงตามปฏิทินได้อย่างถูกต้อง และมีคุณสมบัติทางเทคนิคดังนี้

- เป็นไอซีสร้างฐานเวลาให้ข้อมูลตั้งแต่วินาทีจนถึงปี
- มีหน่วยความจำนอนโวลาไทแรม 56 ไบต์อยู่ภายใน สามารถเก็บข้อมูลทั่วไปได้
- มีวงจรตรวจจับระดับของไฟเลี้ยงว่าต่ำหรือหายไปอย่างอัตโนมัติ และสามารถรักษาข้อมูลไว้ได้แม้ไม่มีไฟเลี้ยง



รูปที่ 2.11 การจัดวางขาของ DS1307

โครงสร้างของไอซีเบอร์ DS1307 แสดงได้ดังรูป 2.11 โดยมีรายละเอียดดังนี้

Vcc และ GND ขา 8 และ ขา 4 ต่อไฟเลี้ยง 5V

V_{BAT} ขา 3 ใช้ต่อกับแบตเตอร์รี่ 3V เพื่อรักษาการทำงานของวงจรสร้างฐานเวลาของ DS13071 ให้คงอยู่ต่อไป แม้ว่าจะไม่มีไฟเลี้ยงให้แก่ DS1307 ชนิดของแบตเตอร์รี่ที่เหมาะสมคือแบตเตอร์รี่แบบลิเธียม ซึ่งมีความจุ 40 mAhr หรือมากกว่า จะสามารถรักษาข้อมูลได้นาน 10 ปี ที่อุณหภูมิ 25 องศาเซลเซียส

SDA และ SCL ขา 5 และ ขา 6 เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์บนระบบบัส I^2C

SQW/OUT ขา 7 ที่ขานี้จะมีสัญญาณรูปสี่เหลี่ยมส่งออกมาโดยสามารถเลือกความถี่ได้ 1Hz 4.096Hz 8.192kHz และ 32kHz ในการใช้งานต้องต่อตัวต้านทาน 1k พูลอัปที่ขานี้ด้วย

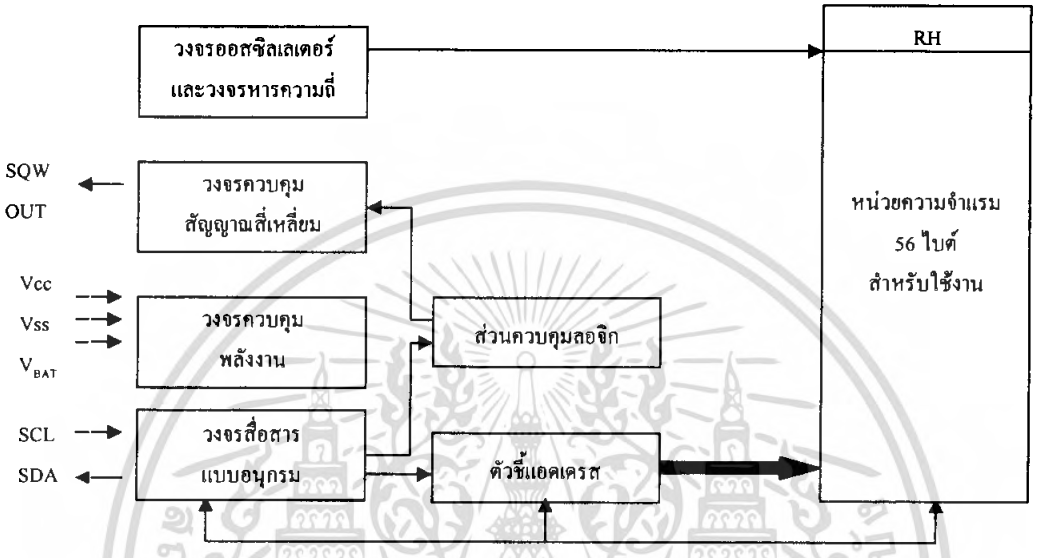
X1 และ X2 ขา 1 และ ขา 2 ใช้ต่อกับคริสตอลความถี่มาตรฐาน 32.768kHz เพื่อใช้เป็นฐานเวลาเพื่อใช้ในการสร้างเวลาจริง ในการใช้งานต้องต่อคริสตอลเข้ากับขาทั้งสองนี้ และที่แต่ละขาต้องต่อตัวเก็บประจุค่าต่ำๆ ประมาณ 15F คร่อมกับขากราวด์ด้วย

2.3.1 การทำงานของ DS1307

ไอซี DS1307 จัดการเชื่อมต่อโดยระบบบัส I^2C โดยจะทำงานเป็นอุปกรณ์สเลฟเสมอ ดังนั้นในการติดต่อเพื่อใช้งานจึงต้องกำหนดรูปแบบตามที่กำหนดไว้ในการติดต่อแบบ I^2C ในรูป 2.12 แสดงส่วนประกอบหลักที่สำคัญและไดอะแกรมการทำงานของไอซีสร้างฐานเวลา DS1307 วงจรออสซิลเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง ในขณะที่ DS1307 ทำงานที่ขา SQW/OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมาตลอดเวลากรณีที่มีการอีนาเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจิสเตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ พร้อมกันนั้นก็จะมีการเก็บค่าของเวลาไว้ในหน่วยความจำนอนโวลาไทแรมซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไปอีก 56 ไบต์

วงจรควบคุมพลังงานไฟฟ้า จะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า 1.25 x V_{BAT} ก็จะควบคุมให้ DS1307 หยุดการทำงานรีเซตค่าตัวนับแอดเดรสภายในทำให้ไม่

สามารถติดต่อกับ DS1307 ได้ดังนั้นในการใช้งาน DS1307 ต้องระวังอย่าให้ไฟเลี้ยงต่ำกว่า 1.25 เท่าของ V_{BAT} หรือประมาณ 3.75 V ในกรณีที่ใช้ V_{BAT} เท่ากับ 3V ซึ่งถ้าหากไฟเลี้ยงมีค่าต่ำกว่า V_{BAT} ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที และจะไม่มีการส่งสัญญาณพัลส์ออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่มีผิดพลาด เมื่อไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็สามารถให้ค่าของเวลาที่เป็นจริงได้



รูปที่ 2.12 วงจรภายใน DS1307

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

2.3.2 การจัดสรรหน่วยความจำใน DS1307

การจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7 ไบต์แรกตั้งแต่แอดเดรส 00H-06H เป็นพื้นที่ของรีจิสเตอร์ค่าเวลาใช้ในเก็บข้อมูลเกี่ยวกับเวลา ไบต์ต่อมาที่แอดเดรส 07H เป็นพื้นที่ของรีจิสเตอร์ควบคุมการทำงานของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ ทำให้ผู้ใช้งานสามารถเรียกข้อมูลเวลาออกมาได้ตามที่ต้องการโดยไม่จำเป็นต้องอ่านออกมาทั้งหมดก็ได้ ค่าของเวลาทั้งหมดจะอยู่ในรูปของเลขฐานสิบ สำหรับการแสดงเวลาในรูปของชั่วโมงสามารถเลือกได้ว่าต้องการแบบ 12 หรือ 24 โดยกำหนดที่บิต 6 ของแอดเดรส 02H และเมื่อเลือกแบบ 12 ชั่วโมงที่บิต 5 ในแอดเดรสเดียวกันจะใช้ในการแสดง

ค่า AM/PM โดยถ้าบิตนี้เป็น 1 หมายถึง ค่าชั่วโมงในขณะนี้เป็นช่วงเวลาหลังเที่ยงวัน ในกรณีที่เป็นแบบ 24 ชั่วโมงบิตนี้จะใช้ในการแสดงค่า 2 ของหลักสิบในหน่วยชั่วโมง

2.3.3 การประยุกต์ใช้ไอซีสร้างฐานเวลา DS1307 กับไมโครคอนโทรลเลอร์ MCS-51

ในโครงงานนี้ใช้การต่อไอซีสร้างฐานเวลา DS1307 เข้ากับไมโครคอนโทรลเลอร์แบบขนาน โดยใช้ขา 5 (ขาสัญญาณ SDA) ของ DS1307 เชื่อมต่อเข้ากับขา 1 (P1.0) ของไมโครคอนโทรลเลอร์ และขา 6 (ขาสัญญาณ SCL) ของ DS1307 เชื่อมต่อเข้ากับขา 2 (P1.1) ของไมโครคอนโทรลเลอร์

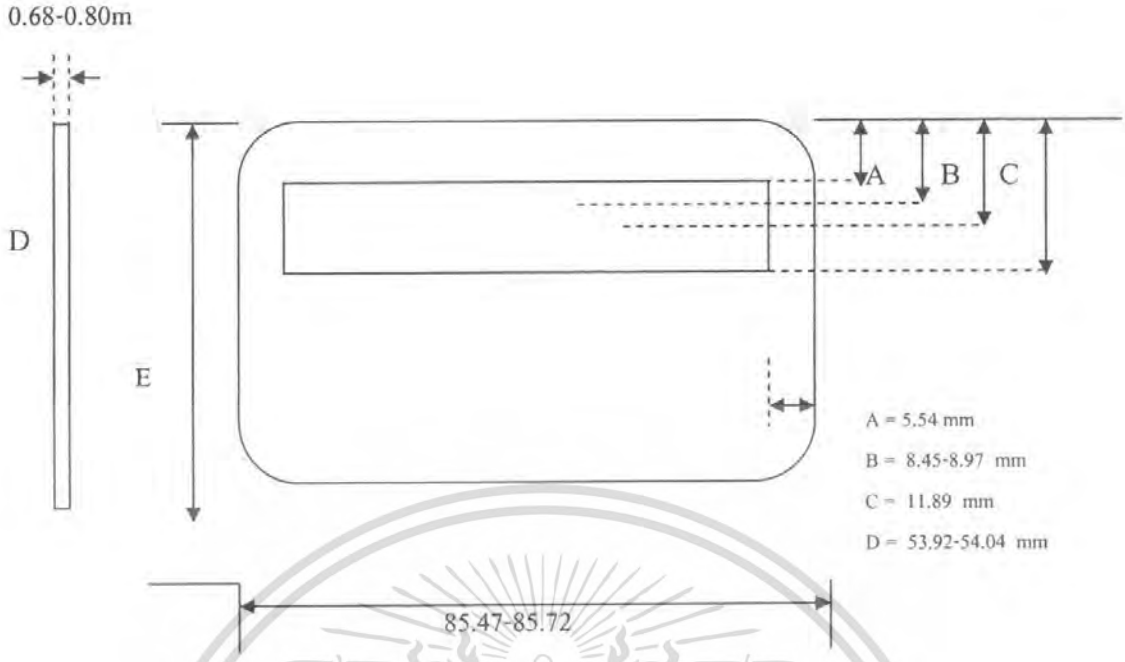
2.4 บัตรแถบแม่เหล็ก

Magnetic Card หรือบัตรแถบแม่เหล็ก ที่เราคุ้นเคยกันดีในรูปแบบของ บัตรเงินสด บัตรATM บัตรเครดิต บัตร VISA และบัตรอื่นๆ อีกมากมาย เป็นต้น เป็นการ์ดแถบแม่เหล็ก ซึ่งบนตัวบัตรเองจะบันทึกข้อมูลและรายละเอียดต่างๆ ของบัตรไว้ในรูปของเส้นแรงแม่เหล็ก ภายในส่วนที่เป็นแถบแม่เหล็กในการ์ดโดยที่แถบแม่เหล็กที่อยู่บนการ์ด ซึ่งเรียกว่า "Track" นั้น ตามปกติแล้วจะถูกแบ่งออกเป็น 3 ส่วนด้วยกัน ซึ่งในแต่ละส่วนจะใช้กับข้อมูลซึ่งมีความหนาแน่นและลักษณะของข้อมูลที่แตกต่างกัน โดยที่ความหนาแน่นของการบันทึกข้อมูลนั้นมีหน่วยเป็น BPI (byte per inch)

โครงสร้างของบัตรแถบแม่เหล็ก แสดงได้ดังรูปที่ 2.13 ซึ่งบนตัวบัตรจะบันทึกข้อมูลและรายละเอียดต่างๆ ของบัตรไว้ในรูปของเส้นแรงแม่เหล็ก โดยแถบแม่เหล็กบนการ์ด ซึ่งเรียกว่า "Track" นั้นจะถูกแบ่งออกเป็น 3 ส่วน ซึ่งแต่ละส่วนจะใช้เก็บข้อมูลซึ่งมีความหนาแน่นและลักษณะของข้อมูลที่มีความแตกต่างกัน ดังนี้คือ

ตารางที่ 2.11 ความหนาแน่นของการบันทึกข้อมูลมีหน่วยเป็น BPI (BYTE PER INCH)

แถบแม่เหล็ก	ความหนาแน่น	การเข้ารหัส	จำนวนอักษร	ลักษณะของข้อมูลที่เก็บ
TRACK 1	210 BPI	ALPHA	79	ชื่อเจ้าของบัตรและหมายเลขบัตร
TRACK 2	75 BPI	BCD	40	หมายเลขบัตรและวันหมดอายุ
TRACK 3	210 BPI	BCD	107	หมายเลขบัตรและรหัสพิเศษ



รูปที่ 2.13 ขนาดของบัตรแถบแม่เหล็ก

2.5 เครื่องอ่านบัตรแถบแม่เหล็ก (Magnetic Card Reader)

รูปที่ 2.14 เครื่องอ่านบัตรแถบแม่เหล็ก รุ่น MCR-B02TTL

เครื่องอ่านแถบแม่เหล็กรุ่น MCR-B02TTL แสดงดังรูปที่ 2.14 นี้จะอ่านได้เฉพาะข้อมูลที่บันทึกด้วยความละเอียด 75 BPI คือเฉพาะในส่วนของแถบแม่เหล็กที่ 2 (Track2) เท่านั้น ซึ่งใช้สัญญาณที่ใช้เชื่อมต่อเพื่ออ่านข้อมูลจากเครื่องอ่านบัตรแถบแม่เหล็กทั้งหมด 5 สัญญาณ โดยเป็นสัญญาณเอาท์พุตที่ส่งออกมาจากเครื่องอ่านแถบแม่เหล็กทั้งหมด รวมถึงขาไฟเลี้ยงวงจรที่ต้องต่อให้เครื่องอ่านดังนี้ คือ

ไฟเลี้ยงวงจร (+VCC : สายสีแดง) เป็นแหล่งจ่ายไฟเลี้ยงให้กับเครื่องอ่านบัตรแถบแม่เหล็ก ซึ่งต้องมีค่าเท่ากับ +SVDC

กราวนด์ (GND : สายสีดำ) เป็นระดับอ้างอิงของสัญญาณต่างๆ ระหว่างเครื่องอ่านแถบแม่เหล็ก

สัญญาณข้อมูล (Data : สายสีดำ) เป็นสัญญาณข้อมูลเอาท์พุตที่ส่งออกมาจากเครื่องอ่านแถบแม่เหล็ก ซึ่งเป็นสัญญาณของข้อมูลที่อ่านได้จากบัตรแถบแม่เหล็ก โดยในการอ่านสัญญาณข้อมูลนี้ต้องอ่านแบบอนุกรมทีละบิต และต้องพิจารณาให้สอดคล้องกับสัญญาณนาฬิกาของบัตรด้วย ซึ่งสัญญาณข้อมูลที่อ่านได้นี้จะกลับสภาวะกับสัญญาณจริง คือมีสภาวะเป็นตรงข้ามกันอยู่ ดังนั้นเมื่ออ่านสัญญาณจากเครื่องอ่านแถบแม่เหล็กได้แล้วจะต้องกลับสภาวะของสัญญาณ (Complement) เสียก่อนจึงจะได้ข้อมูลที่ถูกต้อง

สัญญาณนาฬิกา (Clock : สายสีเขียว) เป็นสัญญาณนาฬิกาที่ส่งออกมาจากเครื่องอ่านแถบแม่เหล็ก ซึ่งจะใช้เป็นสัญญาณอ้างอิงในการอ่านสัญญาณข้อมูลจากบัตร โดยการอ่านสัญญาณข้อมูลแต่ละบิตนั้นต้องอ่านในขณะที่สัญญาณนาฬิกานี้เป็นขอบขาลง (Falling edge) สัญญาณข้อมูลที่อ่านได้นั้นจะเริ่มต้นจากบิตที่มีนัยสำคัญต่ำสุด (LSB) ก่อนเป็นอันดับแรก

สัญญาณสถานะ (Present : สายสีขาว) เป็นสัญญาณแสดงสถานะของเครื่องอ่านแถบแม่เหล็ก เมื่อสัญญาณมีลอจิกศูนย์เมื่อสัญญาณนี้เกิด (“0”) จะบ่งบอกให้ทราบว่าเครื่องอ่านแถบแม่เหล็กจะทำการเริ่มต้นส่งข้อมูลออกมาหรือมีการนำบัตรแถบแม่เหล็กไปรูดผ่านหัวอ่าน สัญญาณของเครื่อง และเมื่อสัญญาณนี้หมดลง คือกลับเป็นลอจิกหนึ่ง แสดงว่าข้อมูลจากการอ่านในครั้งนี้ถูกส่งออกไปหมดสิ้นเรียบร้อยแล้ว ซึ่งเราอาจใช้ประโยชน์จากสัญญาณนี้ ส่งเป็นสัญญาณ Interrupt เพื่อให้ CPU เพื่อทำการอ่านข้อมูลจากเครื่องอ่านแถบแม่เหล็ก

2.5.1 การอ่านข้อมูลจากบัตรแถบแม่เหล็ก

การใช้เครื่องอ่านแถบแม่เหล็กเพื่อให้อ่านข้อมูลจากบัตรแถบแม่เหล็กนั้น จะใช้วิธีการนำบัตรส่วนที่เป็นแถบแม่เหล็กไปรูดผ่านหัวอ่านของเครื่องอ่านไปตามทิศทางและตำแหน่งที่กำหนดไว้เท่านั้น โดยให้สังเกตด้านตำแหน่งของหัวอ่านกับส่วนที่เป็นแถบแม่เหล็กของบัตรต้องอยู่ในแนวเดียวกัน และเนื่องจากเครื่องอ่านแถบแม่เหล็กสามารถอ่านข้อมูลจากบัตรแถบแม่เหล็กได้ เฉพาะข้อมูลของ Track2 เท่านั้น ดังนั้นจึงขอกล่าวถึงเฉพาะวิธีการอ่านข้อมูลจาก Track2 เพียงอย่างเดียว โดยรูปแบบของข้อมูลใน Track2 ซึ่งใช้บันทึกสัญญาณด้วยความหนาแน่นของสนามแม่เหล็ก 75 BPI ซึ่งจะบรรจุจำนวนรหัสของข้อมูลต่างๆใน Track2 นี้ ได้สูงสุดไม่เกิน 41 ตัวอักษร โดยนับรวมรหัสควบคุมและรหัสตรวจสอบด้วย ซึ่งมีรายละเอียด ดังนี้คือ

SS	PAN	FS	Addition Data	ES	LRC

SS = รหัสข้อมูลแสดงการเริ่มต้นของข้อมูลในบัตร (Start Sentinel) มีรหัสข้อมูลเป็น OBH (;)

PAN = ข้อมูลแสดงหมายเลขของบัตร (มีจำนวนข้อมูลในส่วนสูงสุดไม่เกิน 19 หลัก)

FS = รหัสข้อมูลแสดงการแบ่งแยกข้อมูล (Field Separator) มีรหัสข้อมูลเป็น 0DH (=)

Additional Data = เป็นข้อมูลเพิ่มเติมบัตรอื่นๆของบัตร เช่น เดือน/ปี ที่ออกบัตรและหมดอายุ

ES = รหัสข้อมูลแสดงการสิ้นสุดของข้อมูลในบัตร (End Sentinel) มีรหัสข้อมูลเป็น 0FH(?)

LRC = ข้อมูลตรวจสอบความผิดพลาด (Longitudinal Redundancy Check)

0000000000	SS	data,data,data,…,data	ES	LRC	0000000000

ลักษณะการจัดเรียงของข้อมูลที่เก็บไว้ในแถบแม่เหล็กในส่วนของ Track2 จะจัดเรียงลำดับความสำคัญจากซ้ายไปขวาแบบอนุกรมทีละบิต ซึ่งลักษณะการจัดเก็บข้อมูลของ Track2 จะเก็บข้อมูลด้วยการเข้ารหัสแบบ Modulo5 ซึ่งในแต่ละชุดข้อมูลจะประกอบด้วยข้อมูล 5 บิต โดย 4 บิตแรกเป็นรหัสข้อมูลแบบ BCD ส่วนบิตที่ 5 เป็นพาริตี้บิตแบบคี่ (Odd) ที่ใช้ตรวจสอบความถูกต้องของข้อมูลในแต่ละไบต์ (5bit) ที่อ่านได้ ซึ่งหากค่าพาริตี้ผิดพลาด แสดงว่าการอ่านข้อมูลนั้นล้มเหลว โดยเริ่มทำการอ่านข้อมูลเมื่อสัญญาณ Present มีค่าเป็นศูนย์ก่อน และทำการอ่านข้อมูลในทุกๆ ขอบขาลงของสัญญาณนาฬิกาเสมอ ซึ่งข้อมูลในส่วนก่อนเริ่มต้นและหลังจากสิ้นสุดของการอ่านนี้จะมีค่าเป็นศูนย์ (สัญญาณ data = logic“ l”) นำหน้า และปิดท้ายข้อมูลจริงอยู่ ซึ่งเรียกว่า “Clocking Bit” ซึ่งข้อมูลในส่วนนี้เราไม่ต้องสนใจแต่ให้ตรวจสอบและรอจนกว่าจะเริ่มเป็นข้อมูล Start บิต (0BH 11010 ในที่นี้เรียงลำดับความสำคัญจากซ้ายไปขวา ซึ่งบิตเริ่มต้นของรหัส Start หรือ 0BH ต้องเริ่มด้วย 1 เป็นบิตแรกเสมอ) ดังนั้นในการอ่านเราต้องรอจนกว่าจะพบสัญญาณข้อมูลมีค่าเป็น “1” (สัญญาณ data = logic “0” เพราะกลับสภาวะกันอยู่) จึงเริ่มเก็บข้อมูลชุดละ 5 บิตไปเรื่อย ๆ จนถึงรหัสจบ (0FH) ซึ่งเมื่อพบรหัสจบแล้วจะมีข้อมูลตามมาอีก 1 ไบต์ ซึ่งเป็นข้อมูลสำหรับตรวจสอบความผิดพลาดของการอ่านข้อมูลทั้งหมดใน Track 2 เรียกว่า “LRC” ซึ่งค่าของ “LRC” สามารถหาได้จากการนำเอาข้อมูลในแต่ละไบต์ (ไม่คิดพาริตี้บิต) ตั้งแต่เริ่มต้นจนสิ้นสุดมาทำการ XOR กัน โดยครั้งแรกให้นำไบต์เริ่มต้น (0BH) ทำการ XOR กับศูนย์ แล้วนำผลลัพท์ที่ได้ไป XOR เรื่อย ๆ ตามลำดับ หากผลลัพท์สุดท้ายที่ได้ไม่เท่ากับค่าของ “LRC” ที่อ่านมาได้แสดงว่าการอ่านข้อมูลทั้งหมดล้มเหลว สำหรับข้อมูลของรหัส BCD ที่ใช้สำหรับเครื่องอ่านแถบแม่เหล็กนี้จะเป็นข้อมูลชุดละ 5 บิต โดยเป็นข้อมูลจริง 4 บิต และเป็นรหัสตรวจสอบพาริตี้อีก 1 บิต ซึ่งมีจำนวนทั้งหมด 16 อักขระ ดังนี้ คือ

ตารางที่ 2.12 แสดงข้อมูลของรหัส BCD สำหรับเครื่องอ่านแถบแม่เหล็ก

Parity	D3	D2	D1	D0	Character	Function
1	0	0	0	0	0(0H)	Data
0	0	0	0	1	1(1H)	Data
0	0	0	1	0	2(2H)	Data
1	0	0	1	1	3(3H)	Data
0	0	1	0	0	4(4H)	Data
1	0	1	0	1	5(5H)	Data
1	0	1	1	0	6(6H)	Data
0	0	1	1	1	7(7H)	Data
0	1	0	0	0	8(8H)	Data
1	1	0	0	1	9(9H)	Data
1	1	0	1	0	:(AH)	Control
0	1	0	1	1	;(BH)	Start Sentinel
1	1	1	0	0	<(CH)	Control
0	1	1	0	1	=(DH)	Field Seprator
0	1	1	1	0	>(EH)	Control
1	1	1	1	1	?(FH)	End Sentinel

2.5.2 การประยุกต์ใช้เครื่องอ่านบัตรแถบแม่เหล็กกับไมโครคอนโทรลเลอร์ MCS-51

โครงงานนี้เชื่อมต่อเครื่องอ่านบัตรแถบแม่เหล็กกับไมโครคอนโทรลเลอร์ MCS-51 แบบอนุกรม โดยเชื่อมต่อขา Data ขา Clock และขา Presentกของเครื่องอ่านบัตรแถบแม่เหล็กกเข้ากับขา 14 (P3.4/T0) ขา 15 (P3.5/T1) และขา 17 (P3.7/RD) ของไมโครคอนโทรลเลอร์ ตามลำดับ เนื่องจากสัญญาณข้อมูลของเครื่องอ่านบัตรแถบแม่เหล็ก ต้องสัมพันธ์สอดคล้องกับสัญญาณนาฬิกา จึงต้องใช้ขา Timer ของไมโครคอนโทรลเลอร์เพื่อให้สอดคล้องกัน

2.6 รายละเอียดเกี่ยวกับโมดูลLCD

รูปที่ 2.15 โมดูลLCD

ในโมดูลLCD แสดงดังรูปที่ 2.15 มีส่วนประกอบหลักๆ 3 ส่วน ดังนี้

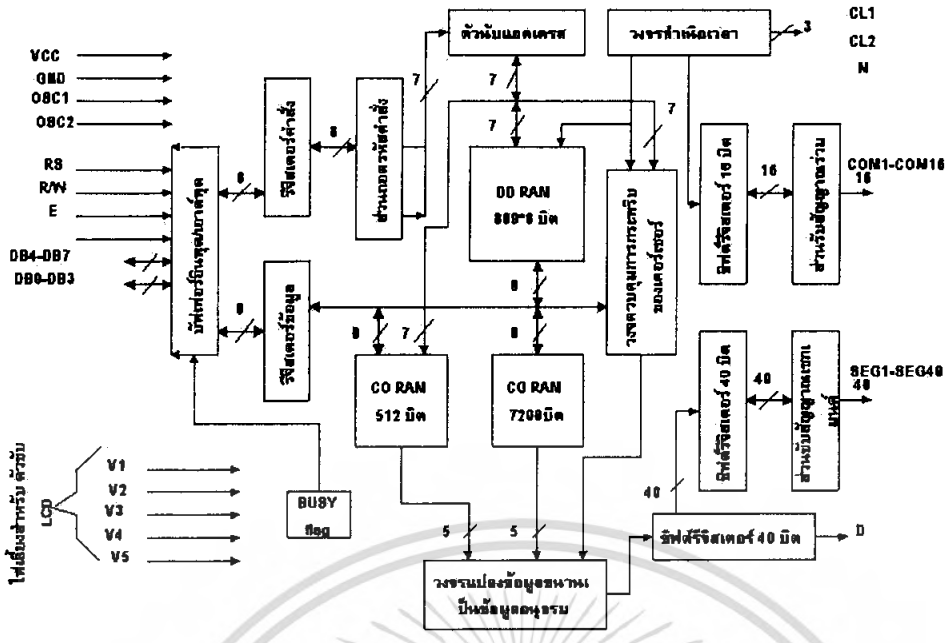
ตัวแสดงผล (Display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนโมดูลLCD

ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูลLCD เช่น ลบจอภาพ แสดงตัวอักษร หรือ เลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิพควบคุมโดยเฉพาะชิพที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟฟิก

ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนดชิพที่ใช้ทำหน้าที่ในการขับนี้ ได้แก่ HD44100H และ MSM5259 เป็นต้น

2.6.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD

การใช้งานโมดูลLCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ต่อไปจะขอยกข้อมูลโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย ดังรูปที่ 2.16 เป็นบล็อกไดอะแกรมภายในของชิพควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระประกอบด้วย



รูปที่ 2.16 ไดอะแกรมการทำงานของโมดูลLCD แบบอักขระ

บัฟเฟอร์อินพุตเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับและส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจีสเตอร์คำสั่ง (Instructtion Register : IR) เป็นรีจีสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจีสเตอร์ข้อมูล (Data Register : DR) เป็นรีจีสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผลหรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

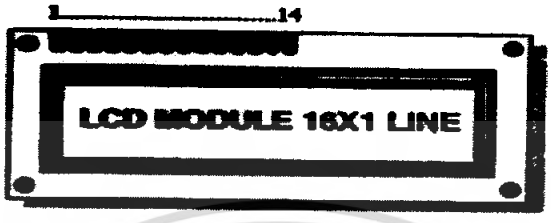
แรมเก็บข้อมูลแสดงผล (Display Data RAM : DDRAM) เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจีสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up-label) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษร เพื่อที่จะนำไปแสดงที่ตัวแสดงผล

รอมเก็บตัวอักษร (Character Generator ROM : CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ โดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ การเขียนและ

การอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือเขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟลกBUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุม เพื่อให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลกBUSY นี้เสียก่อน



ขา 1 : GND
ขา 2 : +V
ขา 3 : Brightness ปรับความสว่าง
ขา 4 : RS
ขา 5 : R/W
ขา 6 : E
ขา 7-14 : D0-D7

รูปที่ 2.17 รูปร่างและการจัดขาโมดูล LCD แบบอักขระ

2.6.2 โมดูลLCD ขนาด 16 ตัวอักษร 1 บรรทัด

สำหรับโมดูลLCD ที่ยกมาเป็นขนาด 16 ตัวอักษร 1 บรรทัด ซึ่งเป็นโมดูลLCD ที่มีโครงสร้างเป็นมาตรฐาน มีผู้ผลิตหลายรายและมีการระบุเบอร์แตกต่างกันออกไปตามผู้ผลิต เช่น LM020Lของฮิตาชิ DMC-16117A ของออปเทร็กซ์ (Optrex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันคือเบอร์ HD44750 ของฮิตาชิ

โมดูลLCD ขนาด 16x1 มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังรูปที่ 2.17 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

V_{SS} (ขา 1) ต่อกราวด์

V_{DD} (ขา 2) ต่อไฟเลี้ยง +5 โวลต์

V_O (ขา 3) เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจีสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจีสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

$\overline{R/W}$ (ขา 5) เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา 6) เป็นขาอีนาเบิล LCD ให้ทำงาน

D0-D7 (ขา 7-14) เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอก ขนาด 8 บิต

2.6.3 คำสั่งควบคุม LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ $\overline{R/W}$ เป็น "0" เขียนคำสั่งลงไป คำสั่งควบคุมโมดูลLCD ของชิพควบคุม HD 44780 ที่สำคัญมี 10 คำสั่ง ดังนี้

- **คำสั่งเคลียร์ตัวแสดงผล (Clear Display)**

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

- **คำสั่ง return home**

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลง นั่นคือข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

- **คำสั่งเลือกโหมดการป้อนข้อมูล (Entry Mode Set)**

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

บิต S เป็นบิตที่ใช้ในการกำหนดลักษณะการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิต S เป็น "1" เมื่อเกิดข้อมูลใหม่บนจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย แต่ถ้าหากบิตนี้เป็น "0" เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวา

บิต I/D เป็นบิตที่ใช้ในการกำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว ทำให้แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้นแต่ถ้าเป็น "0" แอดเดรสจะลดลง

ดังนั้นข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H-07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดให้เมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรสของ DDRAM เพิ่มขึ้น

- **คำสั่งควบคุมการแสดงผล**

มีรายละเอียดของข้อมูลคำสั่งดังนี้

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 ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนตัวแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปิดเคอร์เซอร์ หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกระพริบ ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H-0FH (8รูปแบบคำสั่ง) ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์ และ 0FH เป็นการสั่งให้เปิดจอภาพแสดงเคอร์เซอร์ และสั่งให้เคอร์เซอร์กระพริบ

- **คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลอักษร**

มีรายละเอียดของรูปแบบคำสั่งดังนี้

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

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผลขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

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

- **คำสั่งกำหนดฟังก์ชันการทำงาน**

มีรูปแบบข้อมูลคำสั่ง ดังนี้

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

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ในการติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น "0" จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าเป็น "1" จะเป็นแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น "0" จะแสดงผล 1 บรรทัด ถ้าเป็น "1" จะแสดง 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัด ก็กำหนดบิต N นี้ให้เป็น "1"

บิต F ใช้เลือกความละเอียดของอักษรให้การแสดงผล ถ้าบิตนี้เป็น "0" จะเป็นการแสดงผลแบบ 5*7 จุด และถ้าเป็น "1" จะแสดงผลแบบเป็น 5x10 จุด

ข้อมูลคำสั่งที่ใช้บ่อยคือ 38H เป็นการกำหนดให้โมดูล LCD แบบ 8 บิตแสดงผล 2 บรรทัด และเลือกความละเอียดเป็น 5x7 จุด

จุดที่น่าสังเกตคือ โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง 1 บรรทัด แต่จะต้องกำหนด N ให้เป็น "1" เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่ช่อง คือ 00H และ 40

- **คำสั่งเลือกแอดเดรสของ CGRAM**

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

- **คำสั่งเลือกแอดเดรสของ DDRAM**

ในการเลือกแอดเดรสของ 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

- **คำสั่งอ่านแฟลก BUSY และแอดเดรส**

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
BF	A	A	A	A	A	A	A

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

ทำงานภายใน หรือกำลังประมวณผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง และเมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา $\overline{R/W}$ เป็น "1" ด้วย แต่สัญญาณที่ RS ยังต้องเป็น "0" อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0 – 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

2.6.4 การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูลLCD แสดงผลตามที่ผู้ใช้งานต้องการ ต้องส่งคำสั่ง (Instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูลLCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (Data) ที่ต้องการแสดงผลเนื่องจากบัสข้อมูลของโมดูลLCD มี 8 เส้น คือ D0-D7 และใช้เป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจะต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ถ้าหากที่ขา RS ได้ลอจิก "0" หมายความว่า ข้อมูลที่ป้อนให้แก่โมดูล LCD ขณะนั้นเป็นคำสั่ง ในทางตรงข้ามหากขา RS ได้รับ ลอจิก "1" ข้อมูลที่ป้อนให้ขณะนั้นเป็นข้อมูลที่ใช้ในการแสดงผล

เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อนโดยใช้คำสั่งเลือกแอดเดรส จากนั้นกำหนดให้ขา RS เป็น "1" เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากกฏต่อไปนี้เป็นข้อมูลปกติไม่ใช่คำสั่ง ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา $\overline{R/W}$ เป็น "1" ข้อมูลขนาด 8 บิต (หรือ 4 บิต) ก็จะปรากฏบนบัสข้อมูล โดยข้อมูลที่อ่านออกมาได้จะเป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ ในกรณีที่ต้องการเขียนข้อมูล เมื่อกำหนดแอดเดรสและป้อนลอจิก "1" ให้ขา R แล้วต้องกำหนดให้ขา $\overline{R/W}$ เป็น "0" ข้อมูลที่อยู่บนบัสข้อมูลจะถูกเขียนลงในรีจีสเตอร์ DR จากนั้นจึงถ่ายทอดลงใน DDRAM ต่อไป

2.6.5 จังหวะการทำงานของ LCD โมดูล

การติดต่อกับโมดูลLCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายในLCD โมดูล แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อน จากนั้นจึงจะรับข้อมูลหรือดำเนินการต่อไป ดังนั้นในการใช้งานโมดูล LCD ต้องมีการเขียนโปรแกรมเพื่อหน่วงเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้โมดูลLCD ต้องรอประมาณ 10 มิลลิวินาที เพื่อให้ โมดูล LCD ทำการเตรียมความพร้อมหรือการกำหนดค่าเริ่มต้น (Initial) หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขา RS ของโมดูลLCD แล้วต้องหน่วงเวลาอีกประมาณ 2 มิลลิวินาที เพื่อให้คอนโทรลเลอร์ในโมดูลLCD แปลความหมายของลอจิกที่ขา RS ว่าข้อมูลต่อไปที่จะได้รับเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E ของโมดูลLCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาที

ทั้งหมดที่กล่าวมาคือขั้นตอนและจังหวะในการทำงาน 1 รอบของโมดูลLCD จะเห็นได้ว่ามีโปรแกรมย่อยอยู่ 3 โปรแกรมคือ โปรแกรมการกำหนดค่าเริ่มต้นของLCD โปรแกรมหน่วงเวลา และโปรแกรมย่อยการส่งพัลส์เพื่อสั่งให้โมดูลLCDทำงาน

2.6.6 การประยุกต์ใช้โมดูล LCD กับไมโครคอนโทรลเลอร์ MCS-51

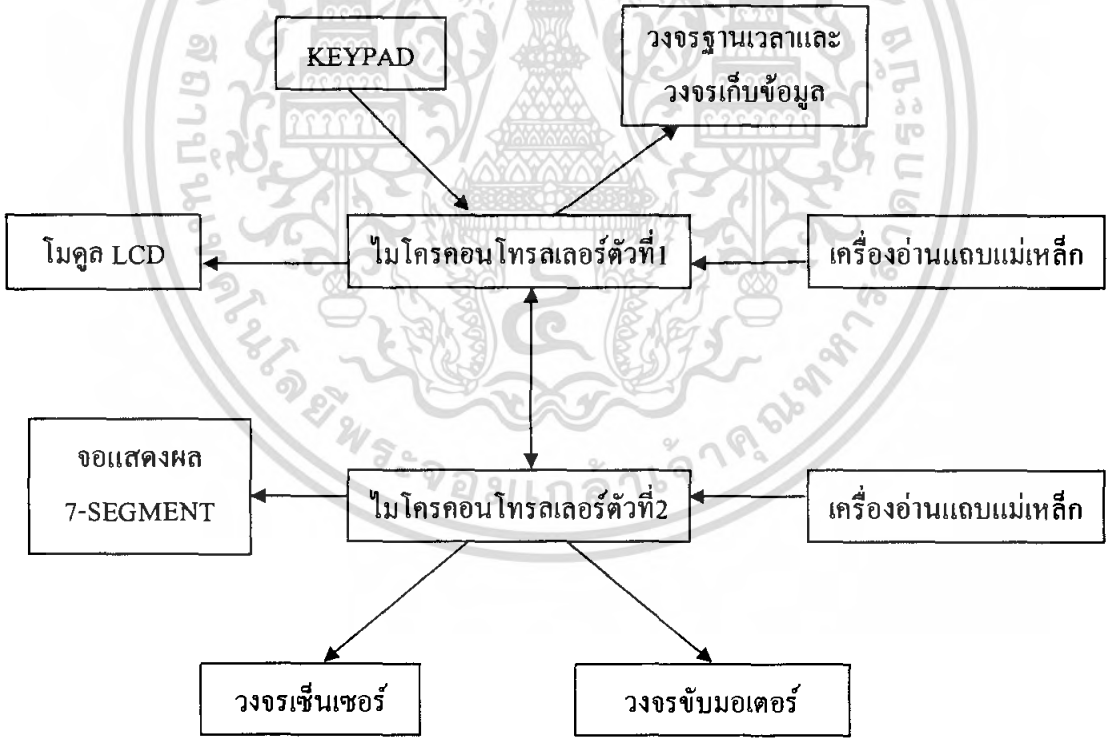
จากรายละเอียดการทำงานของแต่ละขาในโมดูลLCD ที่กล่าวไปแล้วในข้างต้นการออกแบบวงจรเชื่อมต่อระหว่างโมดูลLCD กับไมโครคอนโทรลเลอร์ จะใช้พอร์ต 2 ของไมโครคอนโทรลเลอร์ โดยเริ่มจากพอร์ต P2.0 ถึง P2.5 เชื่อมต่อกับขาของโมดูลLCD

บทที่ 3
การคำนวณและการสร้าง

โครงงานนี้ประกอบไปด้วย 2 ส่วนใหญ่ๆ คือ ส่วนของฮาร์ดแวร์และส่วนของซอฟต์แวร์ โดยฮาร์ดแวร์จะประกอบไปด้วย อุปกรณ์ทางอิเล็กทรอนิกส์ต่างๆ ที่นำมาใช้ในการประกอบวงจรของโครงงานเช่น หัวอ่านบัตรแถบแม่เหล็ก ไมโครคอนโทรลเลอร์ และส่วนของซอฟต์แวร์จะประกอบไปด้วยโปรแกรมภาษาซี เป็นต้น

3.1 แผนผังวงจรของโครงงาน

โครงงานนี้ใช้ไมโครคอนโทรลเลอร์ 2 ตัวในการควบคุมระบบ โดยไมโครคอนโทรลเลอร์ทั้ง 2 ตัวจะเชื่อมต่อกันแบบอนุกรม โดยไมโครคอนโทรลเลอร์ตัวแรกจะควบคุมโมดูลLCD KEYPAD วงจรฐานเวลา วงจรเก็บข้อมูลและ เครื่องอ่านแถบแม่เหล็ก ส่วนไมโครคอนโทรลเลอร์ตัวที่สองควบคุมจอแสดงผล7-SEGMENT วงจรเซ็นเซอร์ วงจรขับมอเตอร์ และเครื่องอ่านแถบแม่เหล็ก ดังรูปที่ 3.1



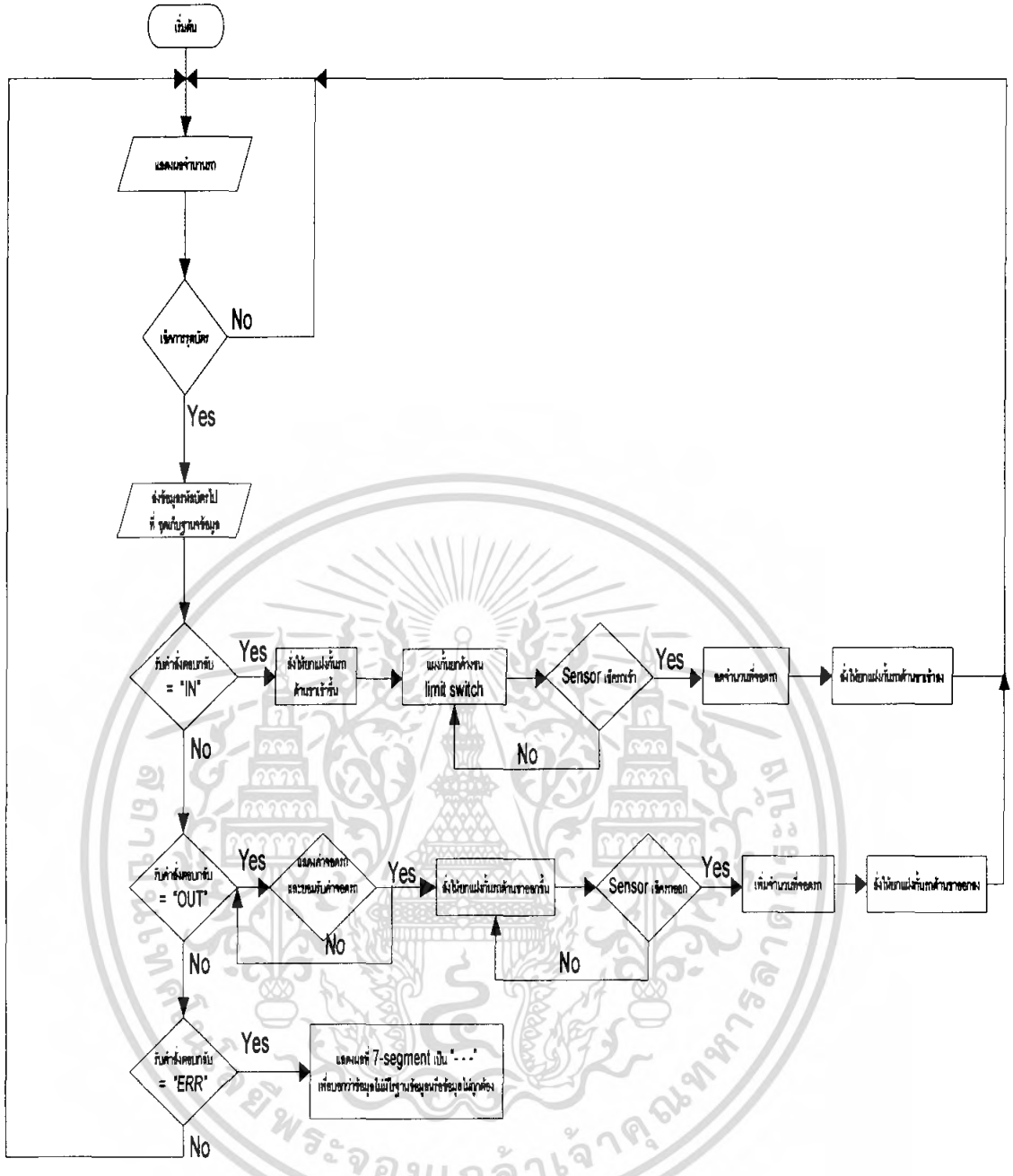
รูปที่ 3.1 แผนผังของระบบลานจอดรถอัตโนมัติ

3.2 ขั้นตอนการดำเนินการสร้าง

- วิเคราะห์โครงงานและหาข้อมูล
- ออกแบบโครงสร้างด้านฮาร์ดแวร์
- ทำการเขียนโปรแกรมควบคุมฮาร์ดแวร์ (ซอฟต์แวร์)
- ทดสอบฮาร์ดแวร์และโปรแกรมควบคุมฮาร์ดแวร์
- ออกแบบลายวงจรฮาร์ดแวร์
- ประกอบอุปกรณ์ลงลายวงจรและประกอบอุปกรณ์ลงแบบจำลองที่จอดรถ
- ทดสอบระบบทั้งหมดเพื่อดูข้อผิดพลาดและทำการแก้ไข
- แล้วทำการทดสอบอีกครั้ง

3.3 ขั้นตอนการทำงานทั้งหมดของระบบ

ลานจอดรถสามารถรองรับรถเข้าจอดได้จำนวน 2 ชั้น ชั้นละ 200 คันโดยเริ่มต้นการทำงานจาก วงจรเซ็นเซอร์จะทำหน้าที่ตรวจเช็คตลอดเวลาว่ามีรถเข้ามาจอดในแต่ละชั้นหรือไม่ ซึ่งวงจรเซนเซอร์นี้จะถูกติดตั้งไว้กับพื้นตรงบริเวณทางเข้าของแต่ละชั้น เมื่อมีรถขับมาจอดบริเวณหน้าทางเข้าซึ่งมีจอแสดงผล 7-SEGMENT แสดงจำนวนที่ว่างที่สามารถนำรถเข้าไปจอดได้ เมื่อเจ้าของรถทำการรูดบัตร โมดูล LCD บริเวณทางเข้าจะแสดงจำนวนเงินคงเหลือในบัตร และข้อมูลในบัตรจะถูกส่งไปเปรียบเทียบยังชุดเก็บฐานข้อมูล เมื่อข้อมูลในบัตรถูกต้องไมโครคอนโทรลเลอร์ตัวที่ 2 จะส่งคำสั่งตอบกลับมายังไมโครคอนโทรลเลอร์ตัวที่ 1 ผ่านทางสายเชื่อมต่อ เมื่อคำสั่งที่ส่งมาเป็น " IN " แผงกั้นด้านทางเข้าจะถูกยกขึ้น เมื่อเจ้าของรถขับรถเข้าไปยังชั้นที่จอดจะผ่านวงจรเซ็นเซอร์เช็ครถเข้า วงจรเซ็นเซอร์จะส่งค่ากลับมายังไมโครคอนโทรลเลอร์ตัวที่ 1 ว่ามีรถเข้ามาจอดแล้วไมโครคอนโทรลเลอร์ตัวที่ 1 จะทำการลดจำนวนรถลงและแสดงผลออกทาง จอแสดงผล 7-SEGMENT แผงกั้นด้านทางเข้าจะยกลง เมื่อมีการรูดบัตรออกโมดูล LCD จะแสดง ค่าใช้บริการและจำนวนเงินคงเหลือในบัตร เมื่อเจ้าของบัตรกดปุ่ม Enter แผงกั้นทางออกจะยกขึ้นเพื่อให้รถออก วงจรเซ็นเซอร์เช็ครถออกจะส่งค่าไปยังไมโครคอนโทรลเลอร์ตัวที่ 1 ว่ามีรถออกจากที่จอดแล้วไมโครคอนโทรลเลอร์ตัวที่ 1 จะสั่งให้แผงกั้นทางออกปิด และทำการเพิ่มจำนวนรถขึ้นและแสดงผลออกทาง จอแสดงผล 7-SEGMENT เมื่อมีการนำบัตรที่ไม่ได้มีการบันทึกข้อมูลไว้มาใช้งาน เมื่อทำการรูดบัตร จอแสดงผล 7-SEGMENT จะแสดงค่าเป็น "----" เพื่อบอกว่าข้อมูลในบัตรไม่ถูกต้อง ซึ่งขั้นตอนการทำงานของไมโครคอนโทรลเลอร์ในการเช็ครถเข้าและออกแสดงดังรูปที่ 3.2

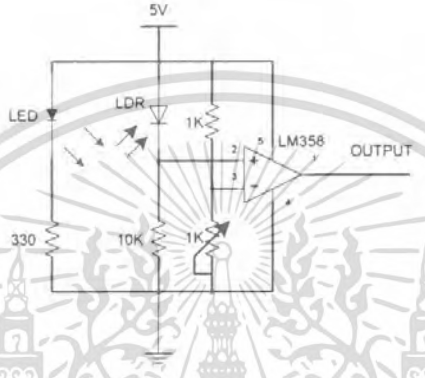


รูปที่ 3.2 แผงผังการทำงานของระบบลานจอดรถอัตโนมัติ

3.4 ภาควงจร

3.4.1 เซ็นเซอร์

โครงงานนี้ใช้ตัวต้านทานไวแสงหรือ LDR เป็นเซ็นเซอร์ ตัว LDR นี้เมื่อมีแสงตกกระทบลงไปจะถ่ายทอดพลังงานให้กับสารที่ฉาบอยู่ทำให้เกิดโฮลกับอิเล็กตรอน ซึ่งการที่มีโฮลกับอิเล็กตรอนอิสระนี้มากก็เท่ากับความต้านทานลดลง เมื่อนำมาใช้กับวงจรเปรียบเทียบแรงดันที่ใช้ไอซีเบอร์ LM358 จะได้วงจรและชุดเซ็นเซอร์ที่ประกอบแล้วดังรูปที่ 3.3 และรูปที่ 3.4

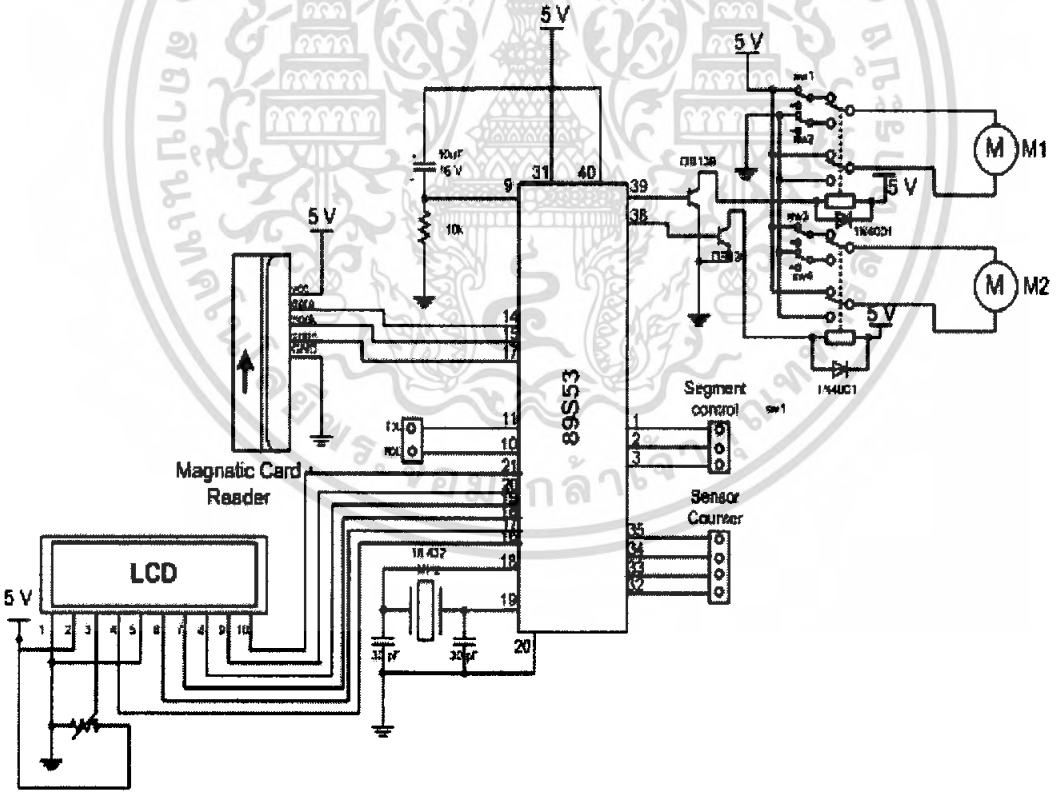


รูปที่ 3.3 วงจรเซ็นเซอร์

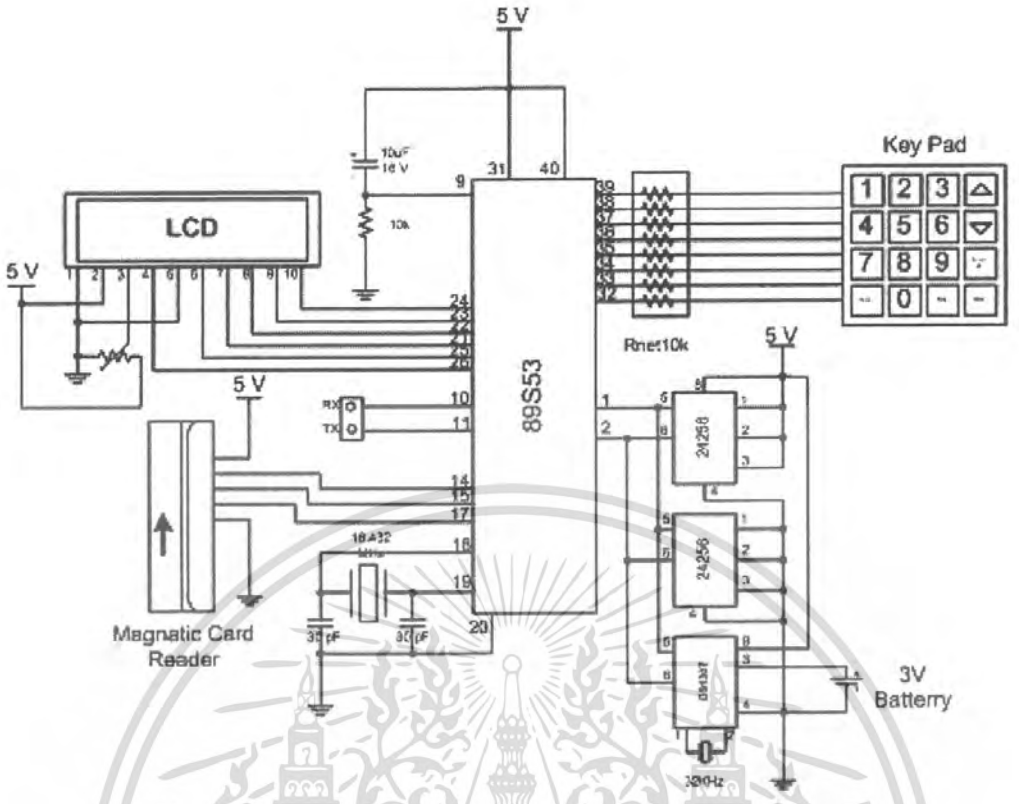
รูปที่ 3.4 วงจรเซ็นเซอร์เมื่อประกอบแล้ว

3.4.2 ควบคุมทางเข้าและควบคุมทางออก

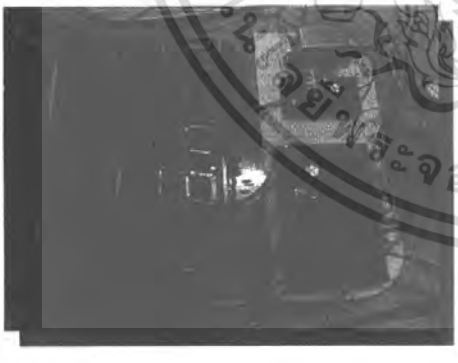
วงจรควบคุมทางเข้าและวงจรควบคุมทางออกประกอบไปด้วย ไมโครคอนโทรลเลอร์ 2 ตัวโดยไมโครคอนโทรลเลอร์ตัวที่ 2 จะติดต่อกับ เครื่องอ่านบัตรแถบแม่เหล็ก วงจรเซ็นเซอร์ จอแสดงผล 7-SEGMENT และวงจรควบคุมมอเตอร์ ไมโครคอนโทรลเลอร์ตัวที่ 1 จะติดต่อกับ โมดูล LCD เครื่องอ่านบัตรแถบแม่เหล็ก KEYPAD วงจรฐานเวลา และชุดเก็บฐานข้อมูล โดยที่ ไมโครคอนโทรลเลอร์ทั้ง 2 ตัวนี้จะเชื่อมต่อกันแบบอนุกรมโดยใช้ขา RxD เป็นขารับข้อมูลและ ขา TxD เป็นขาส่งข้อมูล เมื่อมีการรูดบัตรเข้า ไมโครคอนโทรลเลอร์ตัวที่ 1 จะส่งข้อมูลของ บัตรผ่านทางขา TxD ไปยังขา RxD ของไมโครคอนโทรลเลอร์ตัวที่ 2 เมื่อไมโครคอนโทรลเลอร์ ตัวที่ 2 ได้รับข้อมูลของบัตรแล้ว จะทำการส่งข้อมูลไปเปรียบเทียบกับฐานข้อมูลว่าข้อมูลที่ได้ รับมานั้นตรงกับฐานข้อมูลที่มีอยู่หรือไม่ เมื่อทำการเปรียบเทียบเรียบร้อยแล้ว ไมโครคอนโทรลเลอร์ตัวที่ 2 จะส่งเป็นคำสั่งโปรโตคอลกลับมายังไมโครคอนโทรลเลอร์ตัวที่ 1 เพื่อให้ไมโครคอนโทรลเลอร์ตัวที่ 1 ทำงานต่อไป โดยในการต่อวงจรควบคุมทางเข้าแสดงได้ดัง รูปที่ 3.5 และวงจรทางเข้าเมื่อประกอบแล้วแสดงดังรูปที่ 3.7 วงจรควบคุมทางออกแสดงได้ดัง รูปที่ 3.6 และวงจรทางออกเมื่อประกอบแล้วแสดงดังรูปที่ 3.8



รูปที่ 3.5 วงจรควบคุมทางเข้า



รูปที่ 3.6 วงจรควบคุมทางออก



รูปที่ 3.7 วงจรควบคุมทางเข้าเมื่อประกอบแล้ว

รูปที่ 3.8 วงจรควบคุมทางออกเมื่อประกอบแล้ว

3.4.3 ชุดเก็บฐานข้อมูลและฐานเวลา

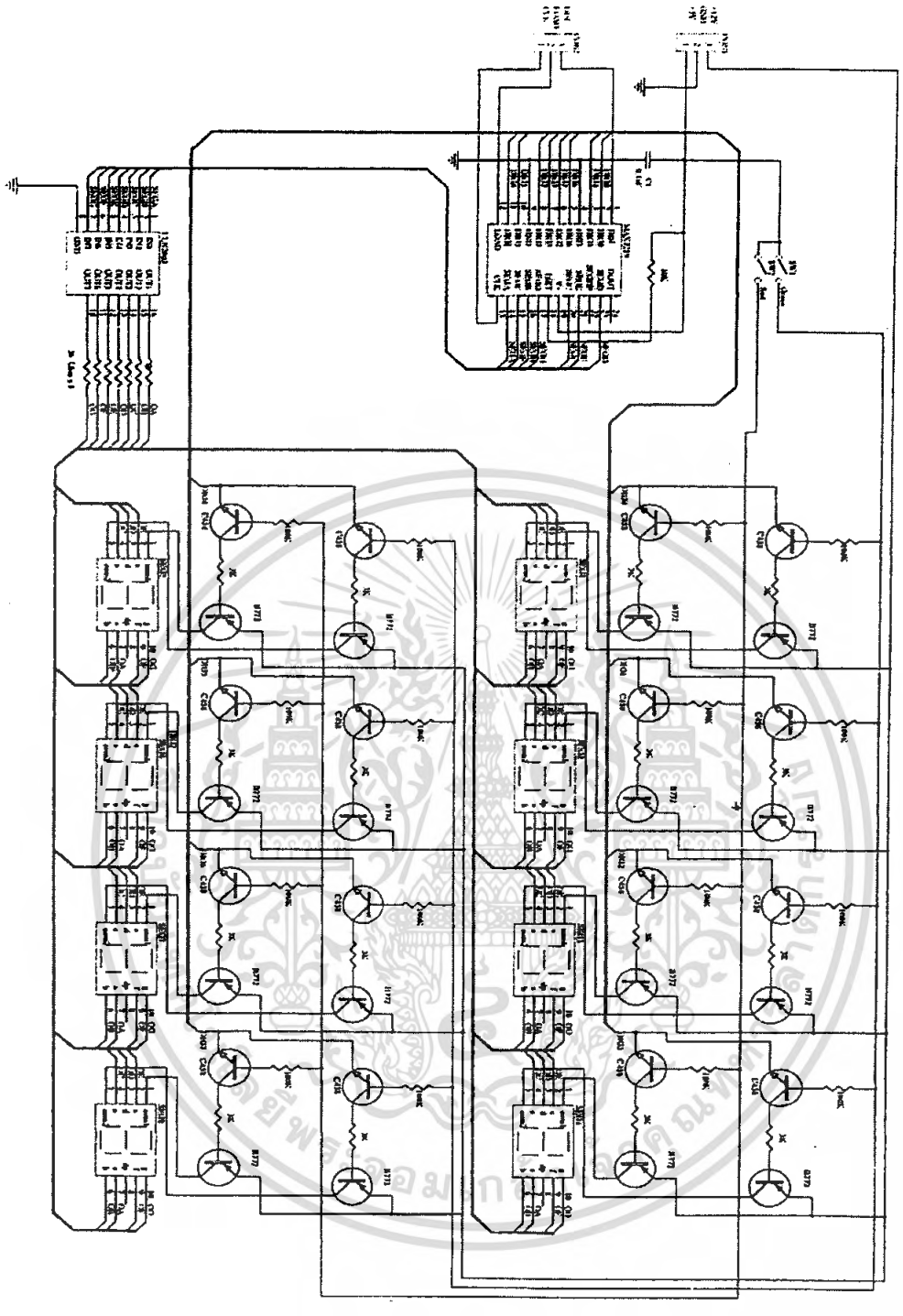
วงจรชุดเก็บฐานข้อมูลประกอบด้วย หน่วยความจำข้อมูลภายนอก (EEPROM) เบอร์ AT24LC256 ขนาดหน่วยความจำ 32 กิโลไบต์ จำนวน 2 ตัว เนื่องจากในการเก็บข้อมูลของบัตรหนึ่งใบใช้หน่วยความจำเท่ากับ 70 ไบต์ ตามตารางตำแหน่งหน่วยความจำ ในโครงงานนี้จะต้องมีการเก็บข้อมูลของบัตรทั้งสิ้น 200 ใบ จึงใช้หน่วยความจำข้อมูลภายนอก (EEPROM) จำนวน 2 ตัว มีหน่วยความจำ 64 กิโลไบต์ เชื่อมต่อกันแบบขนาน และเชื่อมต่อเข้ากับไมโครคอนโทรลเลอร์ตัวที่ 2 โดยระบบบัส I^2C

วงจรฐานเวลา DS1307 เชื่อมต่อกับไมโครคอนโทรลเลอร์ตัวที่ 2 โดยระบบบัส I^2C แสดงดังรูปที่ 3.9

รูปที่ 3.9 วงจรชุดเก็บฐานข้อมูลและฐานเวลา

3.4.4 วงจรควบคุมการแสดงผลทางจอแสดงผล 7 – SEGMENT

วงจรควบคุมการแสดงผลทางจอแสดงผล 7– SEGMENT ใช้ไอซีเบอร์MAX7219 ซึ่งสามารถที่จะรับข้อมูลจำนวนรถจากไมโครคอนโทรลเลอร์ของวงจรทางเข้ามาแสดงผลออกทางจอแสดงผล 7-SEGMENT ได้สูงสุด 8 หลัก แบบสแกนดิสเพลย์ โดยโครงงานนี้จะใช้บอกจำนวนรถที่จอดได้ในแต่ละชั้นจำนวน 2 ชั้น ชั้นละ 4 หลักซึ่ง ไอซีเบอร์MAX7219 สามารถที่จะแสดงผลออกทางจอแสดงผล 7-SEGMENT ขนาดเล็กได้โดยตรง ซึ่งการต่อวงจรควบคุมการแสดงผลทางจอแสดงผล 7-SEGMENT จะต่อดังรูปที่ 3.10



รูปที่ 3.10 การต่อวงจรควบคุมการแสดงผลทาง 7-SEGMENT

3.5 การคำนวณ

3.5.1 เวลาจอดรถ

- เก็บค่าเวลาที่จอดรถขณะรถเข้า
- เก็บค่าเวลาที่จอดรถขณะรถออก
- เปลี่ยนแปลงชั่วโมงที่จอดรถเป็นหน่วยนาที
- นำหน่วยนาทีที่จอดรถมาลบกัน

เปลี่ยนแปลงเวลาเข้าและออกเป็นหน่วยนาที = (เวลาเป็นชั่วโมง x 60) + เวลาเป็นนาที

เวลาที่จอดรถเป็นนาที = เวลาเป็นนาทีขณะรถออก - เวลาเป็นนาทีขณะรถเข้า

เวลาที่จอดรถเป็นชั่วโมง = เวลาที่จอดรถเป็นนาที DIV 60

เศษเวลาที่จอดรถเป็นนาที = เวลาที่จอดรถเป็นนาที MOD 60

กรณีที่ 1 ถ้าเศษมากกว่า 0 เวลาที่จอดรถทั้งหมด = เวลาที่จอดรถเป็นชั่วโมง + 1

กรณีที่ 2 ถ้าเศษเท่ากับ 0 เวลาที่จอดรถทั้งหมด = เวลาที่จอดรถเป็นชั่วโมง

ตัวอย่างเช่น นำรถเข้าจอดเวลา 13.30 น. นำรถออกเวลา 18.50 น.

เวลารถเข้าเป็นนาที = (13x60) + 30 = 810 นาที

เวลารถออกเป็นนาที = (18x60) + 50 = 1130 นาที

เวลาที่จอดรถเป็นนาที = 1130 - 810 = 320 นาที

เวลาที่จอดรถเป็นชั่วโมง = 320 DIV 60 = 5 ชั่วโมง

เศษเวลาที่จอดรถเป็นนาที = 320 MOD 60 = 20 นาที

เวลาที่จอดรถทั้งหมด = 5 + 1 = 6 ชั่วโมง

3.5.2 ค่าจอดรถ

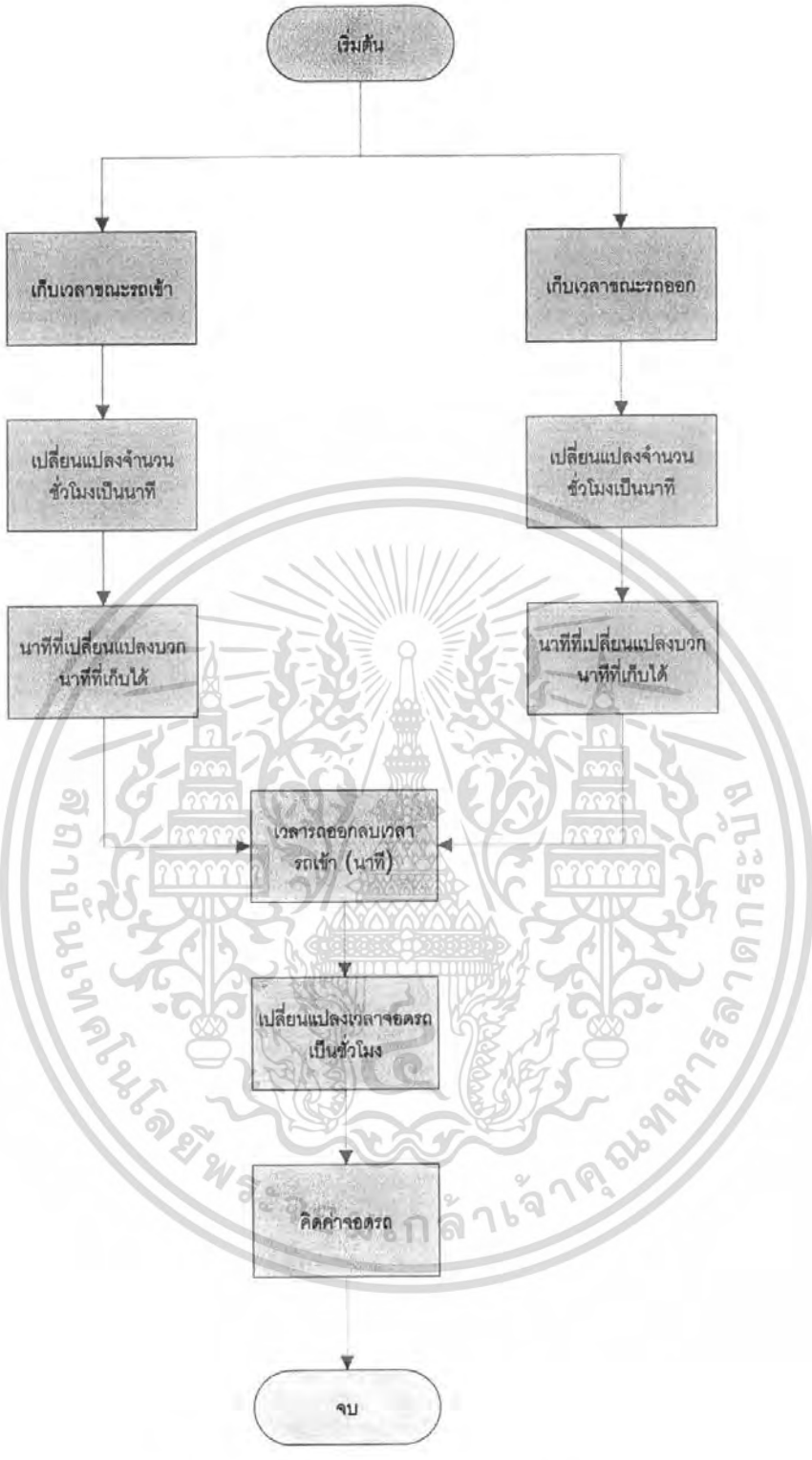
ค่าจอดรถ = เวลาที่จอดรถทั้งหมด x อัตราค่าจอดรถ (บาทต่อชั่วโมง)

เช่น จอดรถทั้งหมด 6 ชั่วโมง ค่าจอดรถชั่วโมงละ 30 บาท

ค่าจอดรถ = 6 x 30 = 180 บาท

3.6 การพัฒนาโปรแกรมควบคุมระบบลานจอดรถอัตโนมัติ

โครงงานนี้ใช้ภาษาซีพื้นฐานเป็นโปรแกรมในการควบคุมไมโครคอนโทรลเลอร์ โดยมีโปรแกรมเดียวเป็นโปรแกรมที่ใช้ในการคอมไพล์ โดย Main ใช้ Library ดังนี้ commu_lib func1 lcd_lib main_game reg52_new และSlave ใช้ Library ดังนี้ commu_lib func1 i2c_lib lab0101 lcd_lib i2c



รูปที่ 3.11 แผนผังการคำนวณอัตราค่าจอดรถ

ตารางที่ 3.1 ตำแหน่งหน่วยความจำ

ตำแหน่ง	หน้าที่	หมายเหตุ
1-25	เก็บข้อมูลรหัสบัตร	
26 - 40	เก็บข้อมูลชื่อ	
41 - 44	เก็บข้อมูลทะเบียนรถ	
45 - 50	เก็บข้อมูลจำนวนเงิน	
51	เก็บข้อมูลชั่วโมง	
52	เก็บข้อมูลนาที	
53	เก็บข้อมูลวัน	
54	เก็บข้อมูลเดือน	
55	เก็บข้อมูลปี	
56	สถานะการจอดรถ	100 = มีการจอด 0 = ไม่มีการจอด
69	มีข้อมูลในตำแหน่งนี้หรือไม่	100 = มีข้อมูล 0 = ไม่มีข้อมูล

ตารางที่ 3.2 ตำแหน่งหน่วยความจำพิเศษ

ตำแหน่ง	หน้าที่	หมายเหตุ
0XFFF0	เก็บจำนวนข้อมูลทั้งหมด	
0XFFFA	เก็บค่าอัตราการจอดรถ	

โปรโตคอลที่ใช้ในการสื่อสาร

ERR = คำสั่งที่บอร์ดเก็บข้อมูลส่งไปให้บอร์ดเช็คข้อมูลขาเข้า เพื่อบอกว่า ไม่มีข้อมูลที่ค้นหา

IN = คำสั่งที่บอร์ดเก็บข้อมูลส่งไปให้บอร์ดเช็คข้อมูลขาเข้า เพื่อบอกว่า มีข้อมูลที่ค้นหา

OUT = คำสั่งที่บอร์ดเก็บข้อมูลส่งไปให้บอร์ดเช็คข้อมูลขาเข้า เพื่อบอกว่า มีรถจะออกโดยได้ทำการคิดเงินและจ่ายเงินแล้ว

CARxxx = คำสั่งที่บอร์ดเก็บข้อมูลส่งไปให้บอร์ดเช็คข้อมูลขาเข้า เพื่อแสดงผลจำนวนรถที่ต้องการ โดย xxx คือจำนวนที่ต้องการแสดงผล

;xxxxxxxxxxxxxxxxxxxxxxxx = คำสั่งที่บอร์ดเก็บข้อมูลใช้รับข้อมูลจากบอร์ดเช็คข้อมูลขาเข้า เพื่อ บอกรหัสของบัตรที่ใช้ในการรูดเข้า โดย xxx ทั้ง 24 ตัวคือ ข้อมูล ; คือ ตัวเช็ครหัสบนบัตร

บทที่ 4

ผลการทดลอง

ในบทนี้จะกล่าวถึงการทำงานและผลการทดลองของระบบลานจอดรถอัตโนมัติซึ่ง ประกอบไปด้วย วิธีใช้ปุ่มฟังก์ชันต่างๆ ของเครื่อง และผลการทดสอบแบบจำลองลานจอดรถ แบบจำลองลานจอดรถเมื่อประกอบอุปกรณ์อิเล็กทรอนิกส์ลงบนแบบจำลองเรียบร้อยแล้ว จะแสดงดังรูปที่ 4.1

รูปที่ 4.1 แบบจำลองลานจอดรถ

4.1 วิธีใช้ปุ่มฟังก์ชันต่างๆ ของเครื่อง

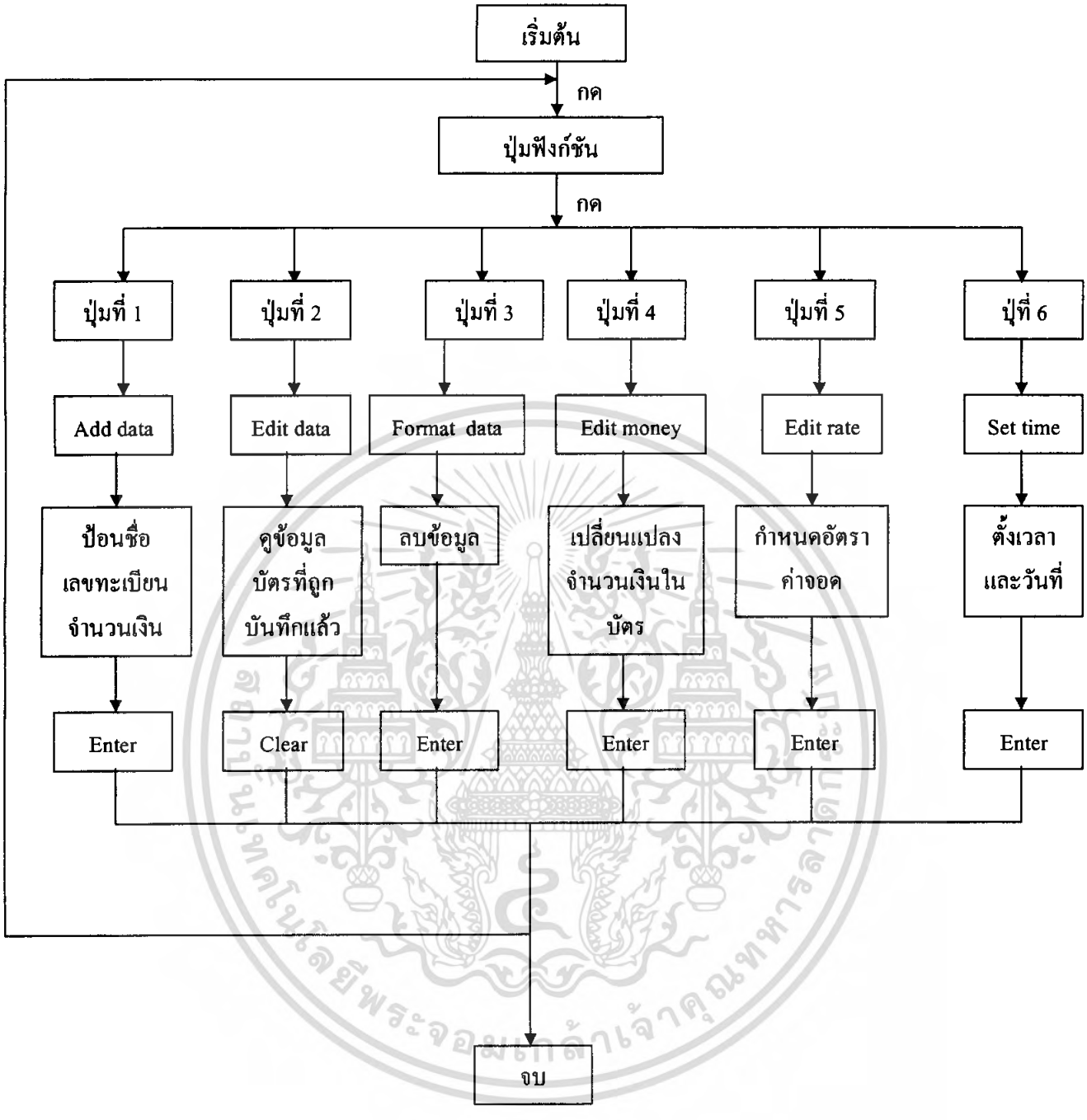
การป้อนข้อมูลต่างๆ เข้าไปในฐานข้อมูลจะใช้วิธีการป้อนข้อมูลผ่าน KEYPAD ดังรูปที่ 4.2

1) กดปุ่ม FUNCTION และปุ่ม 1 เพื่อใช้ Add Data คือการเพิ่มข้อมูลของบัตร ซึ่งประกอบด้วย
   - ชื่อเจ้าของรถ สามารถเก็บได้สูงสุดไม่เกิน 15 ตัวอักษร
   - หมายเลขทะเบียนรถ สามารถเก็บได้ สูงสุดไม่เกิน 4 หลัก
   - จำนวนเงิน สามารถเก็บได้สูงสุดไม่เกิน 6 หลัก
2) กดปุ่ม FUNCTION และปุ่ม 2 เพื่อใช้ Edit Data คือ เรียกดูข้อมูลบัตรที่มีการบันทึกไว้แล้ว
3) กดปุ่ม FUNCTION และปุ่ม 3 เพื่อใช้ Format Data คือ ใช้ลบข้อมูลของบัตร
4) กดปุ่ม FUNCTION และปุ่ม 4 เพื่อใช้ Edit Money คือ การแก้ไขจำนวนเงินในบัตร
5) กดปุ่ม FUNCTION และปุ่ม 5 เพื่อใช้ Edit Rate คือ การกำหนดอัตราค่าบริการ จอดรถต่อชั่วโมงซึ่งสามารถแก้ไขอัตราค่าบริการจอดรถเป็นเท่าไรก็ได้ โดยในโครงงานนี้กำหนดให้เป็นชั่วโมงละ 30 บาท
6) กดปุ่ม FUNCTION และปุ่ม 6 เพื่อใช้ Set Time คือ การตั้งเวลาและวันที่

โครงงานนี้กำหนดให้เป็นชั่วโมงละ 30 บาท

6) กดปุ่ม FUNCTION และปุ่ม 6 เพื่อใช้ Set Time คือ การตั้งเวลาและวันที่

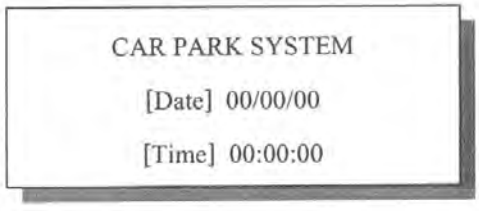
รูปที่ 4.2 KEYPAD ที่ใช้ป้อนข้อมูล



รูปที่ 4.3 แผนผังการทำงานของ KEYPAD

4.2 ผลการทดสอบแบบจำลองการจอดรถ

- เริ่มต้นเมื่อยังไม่มีรถเข้ามาจอดข้อความที่โมดูล LCD จะแสดงเวลาและวัน เดือน ปีในขณะนั้น แสดงดังรูปที่ 4.4

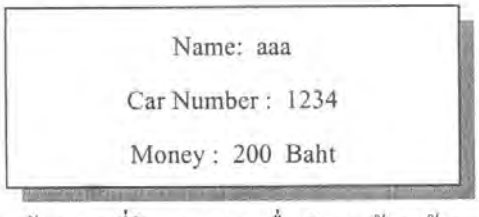


รูปที่ 4.4 ข้อความที่โมดูล LCD เมื่อยังไม่มีรถเข้ามาจอด

จอแสดงผล 7-SEGMENT ในขณะที่ยังไม่มีรถเข้าจอดจะแสดงจำนวนที่ว่างที่สามารถนำรถเข้าจอดได้ โดยในโครงงานนี้กำหนดให้ในแต่ละชั้นสามารถจอดรถได้จำนวน 200 คัน แสดงดังรูปที่ 4.5

รูปที่ 4.5 จอแสดงผล 7-SEGMENT แสดงค่าเมื่อยังไม่มีรถเข้าจอด

- เริ่มโดยทำการป้อนข้อมูลของบัตรโดยในการป้อนข้อมูล คือ ชื่อ เลขทะเบียนรถ และจำนวนเงินในบัตร แสดงดังรูปที่ 4.6



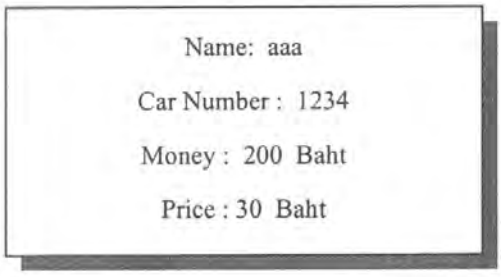
รูปที่ 4.6 แสดงข้อความที่โมดูล LCD เมื่อทำการป้อนข้อมูลของบัตรลงในเครื่อง

- เมื่อทำการรูดบัตรผ่านถ้าข้อมูลของบัตรถูกต้อง โมดูล LCD บริเวณทางเข้าจะแสดงจำนวนเงินคงเหลือในบัตร และประตูกั้นทางเข้าจะเปิดขึ้น เมื่อขับรถผ่านเซ็นเซอร์เช็ครถเข้าชั้นใด จอแสดงผล 7-SEGMENT จะแสดงจำนวนที่ว่างที่ลดลง ประตูกั้นทางเข้าจะปิด จากรูปที่ 4.7 จอแสดงผล 7-SEGMENT แสดงค่าที่ว่างลดลงเมื่อมีรถเข้ามาจอดชั้นที่หนึ่งจำนวน 1 คันโดยในขณะที่ชั้นที่สองยังไม่มีรถเข้าจอดจำนวนที่ว่างจึงแสดงเป็น 200 คันเท่าเดิม

รูปที่ 4.7 จอแสดงผล 7-SEGMENT แสดงค่าเมื่อมีรถเข้าจอดชั้นที่หนึ่งจำนวน 1 คัน

รูปที่ 4.8 จอแสดงผล 7-SEGMENT แสดงค่าเมื่อมีรถเข้าจอดชั้นที่หนึ่งจำนวน 2 คันและมีรถเข้าจอดชั้นที่สองจำนวน 1 คัน

- เมื่อทำการรูดบัตรออกข้อความที่โมดูล LCD ประกอบไปด้วยชื่อเจ้าของรถ หมายเลขทะเบียนรถ จำนวนเงินในบัตรและค่าบริการจอดรถ แสดงดังรูปที่ 4.9



รูปที่ 4.9 แสดงข้อความที่โมดูล LCD เมื่อทำการรูดบัตร

- เมื่อทำการรูดบัตรออกประตูกั้นทางออกจะเปิด เมื่อขับผ่านเซ็นเซอร์เช็ครถออก จอแสดงผล 7-SEGMENT จะแสดงจำนวนที่ว่างเพิ่มขึ้น ประตูกั้นทางออกจะปิด จากรูปที่ 4.10 จอแสดงผล 7-SEGMENT แสดงจำนวนที่ว่างชั้นที่หนึ่งที่เพิ่มขึ้นจาก 198 เป็น 199 เมื่อมีรถออกจากชั้นที่หนึ่งจำนวน 1 คัน และเมื่อรถทุกคันออกจากชั้นที่หนึ่งและชั้นที่สองแล้ว จอแสดงผล 7-SEGMENT จะแสดงดังรูปที่ 4.11

รูปที่ 4.10 จอแสดงผล 7-SEGMENT แสดงค่าเมื่อมีรถออกจากชั้นที่หนึ่งจำนวน 1 คัน

รูปที่ 4.11 จอแสดงผล 7-SEGMENT แสดงค่าเมื่อรถทุกคันออกจากชั้นที่หนึ่งและชั้นที่สอง

- เมื่อนำบัตรที่ไม่ได้บันทึกข้อมูลเอาไว้มาใช้จอแสดงผล 7-SEGMENT จะแสดง "---" ดังรูปที่ 4.12

รูปที่ 4.12 จอแสดงผล 7-SEGMENT แสดงค่าเมื่อเป็นบัตรที่ไม่มีการบันทึกข้อมูลไว้

บทที่ 5
บทวิจารณ์และสรุป

5.1 สรุปผลการทดลอง

สำหรับโครงงานนี้ก็ได้ใช้ไมโครคอนโทรลเลอร์เข้ามาควบคุมระบบ เพื่อใช้ในการเก็บเงินค่าจอดรถแบบอัตโนมัติ โดยผู้ที่นำรถเข้ามาจอดจะรูดบัตรผ่านเครื่องอ่านบัตรแถบแม่เหล็กซึ่งมีข้อมูลชื่อเจ้าของรถและหมายเลขทะเบียนรถปรากฏอยู่ ประตูทางเข้าจึงเปิดให้นำรถเข้าไปจอด วงจรนับจำนวนรถก็จะทำงาน เมื่อต้องการจะนำรถออกจากที่จอดรถจะต้องทำการรูดบัตรผ่าน จากนั้นเครื่องจะทำการคิดคำนวณเวลาและค่าจอดรถ และหักเงินออกจากมูลค่าในบัตรที่มีอยู่ กดปุ่มยอมรับข้อตกลงประตูทางออกจึงจะเปิดให้นำรถออกไปได้ โครงงานนี้ยังสามารถทำการบอกจำนวนรถที่สามารถจอดได้ในแต่ละชั้น ทำให้ผู้ที่นำรถเข้ามาจอดไม่ต้องเสียเวลาในการเข้าไปหาที่จอดรถ ในกรณีที่ที่จอดรถในชั้นนั้นเต็มจะได้นำรถเข้าไปจอดในชั้นอื่นๆที่มีที่ว่างได้เลย เพื่อประหยัดเวลาในการหาที่จอดรถ

จากผลการทดลองสามารถสรุปผลการทดลองได้ว่า โครงงานแบบจำลองที่จอดรถแบบเก็บเงินอัตโนมัติโดยมีบัตรผ่านสามารถทำงานได้ตรงกับขอบเขตของโครงงานดังนี้

1) สามารถคิดเงินค่าจอดรถได้อัตโนมัติ
2) สามารถหักเงินค่าจอดรถได้จากมูลค่าในบัตร
3) มีบัตรผ่านที่มีข้อมูลเจ้าของรถและเลขทะเบียนรถ
4) สามารถแสดงจำนวนรถที่จอดได้ในแต่ละชั้น
5) บัตรผ่านเข้าและออกต้องเป็นบัตรเดียวกันที่มีข้อมูลตรงกันจึงจะนำรถออกไปได้
6) สามารถเปิดปิดประตูอัตโนมัติ

5.2 ปัญหาและแนวทางพัฒนา

ในโครงงานนี้ได้ใช้อุปกรณ์ LDR เป็นเซ็นเซอร์เพื่อรับแสงจึงต้องระมัดระวังให้มีแสงพอเพียงต่อการทำงานของอุปกรณ์ LDR ปัญหาเกิดขึ้นคือ เรื่องความแม่นยำในการตรวจเช็ค เนื่องจากแสงแต่ละที่ไม่เท่ากัน จึงเพิ่มแสงเข้าไปในจุดอับต่างๆ ประกอบกับการปรับค่าความต้านทานเพื่อให้การอ่านค่าได้ถูกต้องยิ่งขึ้น แนวทางในการพัฒนา คือ พัฒนาระบบลานจอดรถอัตโนมัติให้สามารถบอกตำแหน่งที่ว่างในแต่ละชั้นที่สามารถนำรถเข้าจอดได้ เพื่อประหยัดเวลาในการวนหาที่จอดรถ เป็นต้น

เอกสารอ้างอิง

1 เจน สงสมพันธุ์, นิคม อนันต์ทิพย์ . คู่มือไอซี 1 (ฉบับภาษาไทย). กรุงเทพมหานคร : เม็ดทราย-พริ้นติ้ง, 2533

2 รศ. ยื่น ภู่วรรณ. ทฤษฎีและการใช้งานอิเล็กทรอนิกส์. เล่ม 2. กรุงเทพมหานคร : หจก.นำอักษรการพิมพ์, 2531

3 ผศ. อุทัย สุขสิงห์. ไมโครโพรเซสเซอร์และไมโครคอนโทรลเลอร์. กรุงเทพมหานคร : สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2547

4 อุดม จีนประดับ. ไมโครคอนโทรลเลอร์ MCS-51. สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ : โรงพิมพ์สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ, 2546

5 Ayala, K.J. **"The 8051 Microcontroller Architecture. Programming and Applications"** West: Publishing Company, 1991.

6 Intel Corporation. **"Microcontroller HandBook"** Intel Corporation : Sata Clara, 1992.

ภาคผนวก ก
โปรแกรม

MAIN

```
#include <reg52.h>     // Header file for generic 80C51 and 80C31.
#include <stdio.h>     // prototype declarations for I/O functions
/*** Constants ***************************************************/
#define TRUE 1

/* set pin for magnatic card reader */
#define pdat    P3_4
#definepclk     P3_5
#definepcin     P3_7

static unsigned char idata str_buf[22],         rcard[25],char_card[25];
unsigned char  S_Sec,S_Min,S_Hour,S_Day,S_Date,S_Mon,S_Year;
unsigned char  Sec,Min,Hour,Day,Date,Mon,Year;
// ประกาศ Libraly
#include <func1.c>
#include <lcd_lib.c>
#include <i2c.c>

//===============================================================
//                Function RS232
//===============================================================

#define serial_buffer_size 50
static unsigned char idata serial_buffer[serial_buffer_size];
static unsigned char idata index_in=0,index_out=0,send_status=0,data_count=0,tt=0;
unsigned char DISBUF[8],tot;

void init_serial(void) {
SCON = 0x52;          // Setup serial port control register Mode 1:
PCON &= 0x10;   // Clear SMOD bit in power ctrl reg This bit
TMOD |= 0x22;   // Set M1 for 8-bit autoreload timer
TH1 = 0xF6;     // Set autoreload value for timer1 9600 baud
TR1 = 1;        // Start timer 1
TI = 1;         // Set TI to indicate ready to xmit
}
//===============================================================
// ฟังก์ชั่น แสดงข้อมูล ID  ชื่อ เลขทะเบียนรถ  จำนวนเงิน
void f_show_data(int temp)
{
int i;
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf," ID = [%d] ",temp));
lcd_gotoxy(1,2);
```

```
out_lcd(sprintf(str_buf,"%c",RD_AT24C256(((temp-1)*70)+26+i)));

lcd_gotoxy(1,3);
out_lcd(sprintf(str_buf," Car Number: "));
for(i=0;i<4;i++)
out_lcd(sprintf(str_buf,"%bu",RD_AT24C256(((temp-1)*70)+41+i)));
lcd_gotoxy(17,4);
out_lcd(sprintf(str_buf,"Bath"));
lcd_gotoxy(1,4);
out_lcd(sprintf(str_buf," Monny : "));
for(i=0;i<6;i++)
if (RD_AT24C256(((temp-1)*70)+45+i) != '*')
out_lcd(sprintf(str_buf,"%bu",RD_AT24C256(((temp-1)*70)+45+i)));
}
void f_show_data1(int temp)
{
int i;
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf," Name: "));
for(i=0;i<15;i++)
out_lcd(sprintf(str_buf,"%c",RD_AT24C256(((temp-1)*70)+26+i)));

lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf," Car Number: "));
for(i=0;i<4;i++)
out_lcd(sprintf(str_buf,"%bu",RD_AT24C256(((temp-1)*70)+41+i)));
lcd_gotoxy(16,3);
out_lcd(sprintf(str_buf,"Bath"));
lcd_gotoxy(1,3);
out_lcd(sprintf(str_buf," Monny : "));
for(i=0;i<6;i++)
if (RD_AT24C256(((temp-1)*70)+45+i) != '*')
out_lcd(sprintf(str_buf,"%bu",RD_AT24C256(((temp-1)*70)+45+i)));
}
#include<commu_lib.c>

// ฟังก์ชั่น อ่านค่าจากคีย์บอร์ด
//=====================================================================
//                  Function Key
//=====================================================================
unsigned char read_key(void){
unsigned char temp1,temp2;
int key;
P0 = 0x0f;
temp1 = P0 & 0xff;
```

```
P0 = 0xf0;
temp2 = P0 & 0xff;
temp1 = (temp1 | temp2)& 0xff;

if (temp1 == 238)  // 1
key = 1;
if (temp1 == 237)  // 2
key = 2;
if (temp1 == 235)  // 3
key = 3;
if (temp1 == 222)  // 4
key = 4;
if (temp1 == 221)  // 5
key = 5;
if (temp1 == 219)  // 6
key = 6;
if (temp1 == 190)  // 7
key = 7;
if (temp1 == 189)  // 8
key = 8;
if (temp1 == 187)  // 9
key = 9;
if (temp1 == 125)  // 0
key = 0;
if (temp1 == 126)  // Clear
key = 11;
if (temp1 == 123)  // HELP
key = 12;
if (temp1 == 119)  // Enter
key = 13;
if (temp1 == 183)  // Function
key = 14;
if (temp1 == 231)  // UP
key = 15;
if (temp1 == 215)  // Down
key = 16;
if (temp1 == 255)  // Not Put
key = 100;
return (key); // ส่งค่า key ที่กดเข้ากลับ
}
// ฟังก์ชั่น อ่านค่าจาก บัตรแม่เหล็ก
//=============================================================================
//                Read magnetic Card
//=============================================================================
void cardclk(void)
{
```

```
while( pclk == 0 ){}
while( pclk == 1 ){}
}
void getcard(void)
{
unsigned char i,j,din;
bit fstart;
fstart = 0;
pdat = 1;
pclk = 1;
pcin = 1;
while( pcin == 1 )
{}

for( j=0 ; j < 25; j++)
{
din = 0;
for( i=0 ; i < 5 ; i++)
{

cardclk();

if( fstart == 0 )
{
while(pdat == 1){ cardclk(); }
fstart = 1;
din = 0x40;
}
else
{
if(pdat == 0) { din = din | 0x80; }
else { din = din & 0x7F; }
din = din >> 1;
}
}
din = din >> 2;
din = din | 0x30;
rcard[j] = din;
}
}

void getcard1(void)
{
unsigned char i,j,din;
bit fstart;
fstart = 0;
```

```
pdat = 1;
pclk = 1;
pcin = 1;

if ( pcin != 1 )
{

for( j=0 ; j < 25; j++)
{
din = 0;
for( i=0 ; i < 5 ; i++)
{
cardclk();
if( fstart == 0 )
{
while(pdat == 1){ cardclk(); }
fstart = 1;
din = 0x40;
}
else
{
if(pdat == 0) { din = din | 0x80; }
else { din = din & 0x7F; }
din = din >> 1;
}
}
din = din >> 2;
din = din | 0x30;
rcard[j] = din;
}

tot = 20;

}
}
// ฟังก์ชั่น การใช้งานเมนูที่ 1
//=============================================================================
//  Menu Funcition 1
//=============================================================================
void func_menu1 (void)
{
char ch= '_';
int tmp;
unsigned char dd,chk_key = 0 , font[8] ,i , x = 8 , j , k ,chk_menu = 0 ,number_data = 0;
unsigned char  card[25],chk_card=0;
number_data =  RD_AT24C256(0xfff0);
```

```
do
{
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf," [Add Data] ID = %bu   ",number_data+1));
lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf," Read CARD data 1."));
getcard();
for (j=0;j<25;j++)
card[j]=rcard[j];
chk_card = 0;
for (j=0;j<25;j++)
{
if (card[j] == rcard[j])
{
chk_card = chk_card + 1;
}
}
}
while(chk_card!=25);

for(i=0;i<24;i++)
WR_AT24C256((number_data*70)+i+1,rcard[i]);

clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf," [Add Data] ID = %bu   ",number_data+1));
lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf," NAME: "));
x = 8;
do
{
// การกดแสดงตัวอักษรเพื่อตั้งชื่อ
dd = read_key();
if (dd == 100)
chk_key = 1;
if (dd== I && chk_key == 1)
{
chk_key = 0;
ch = '1' ;
}
if (dd== 2 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=0)
font[i] = 0       ;
```

```
chk_key = 0;
font[0] ++;
if (font[0]==1)
ch = 'a';
if (font[0]==2)
ch = 'b';
if (font[0]==3)
ch = 'c';
if (font[0]==4)
ch = '2';
if (font[0]>3)
font[0]=0;
}
if (dd== 3 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=1)
font[i] = 0       ;
chk_key = 0;
font[1] ++;
if (font[1]==1)
ch = 'd';
if (font[1]==2)
ch = 'e';
if (font[1]==3)
ch = 'f';
if (font[1]==4)
ch = '3';
if (font[1]>3)
font[1]=0;
}
if (dd== 4 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=2)
font[i] = 0       ;
chk_key = 0;
font[2] ++;
if (font[2]==1)
ch = 'g';
if (font[2]==2)
ch = 'h';
if (font[2]==3)
ch = 'i';
if (font[2]==4)
ch = '4';
```

```
if (font[2]>3)
font[2]=0;
}
if (dd== 5 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=3)
font[i] = 0       ;
chk_key = 0;
font[3] ++;
if (font[3]==1)
ch = 'j';
if (font[3]==2)
ch = 'k';
if (font[3]==3)
ch = 'l';
if (font[3]==4)
ch = '5';
if (font[3]>3)
font[3]=0;
}
if (dd== 6 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=4)
font[i] = 0       ;
chk_key = 0;
font[4] ++;
if (font[4]==1)
ch = 'm';
if (font[4]==2)
ch = 'n';
if (font[4]==3)
ch = 'o';
if (font[4]==4)
ch = '6';
if (font[4]>3)
font[4]=0;
}
if (dd== 7 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=5)
font[i] = 0       ;
chk_key = 0;
font[5] ++;
```

```
if (font[5]==1)
ch = 'p';
if (font[5]==2)
ch = 'q';
if (font[5]==3)
ch = 'r';
if (font[5]==4)
ch = 's';
if (font[5]==5)
ch = '7';
if (font[5]>4)
font[5]=0;
}
if (dd== 8 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=6)
font[i] = 0       ;
chk_key = 0;
font[6] ++;
if (font[6]==1)
ch = 't';
if (font[6]==2)
ch = 'u';
if (font[6]==3)
ch = 'v';
if (font[6]==4)
ch = '8';
if (font[6]>3)
font[6]=0;
}
if (dd== 9 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=7)
font[i] = 0       ;
chk_key = 0;
font[7] ++;
if (font[7]==1)
ch = 'w';
if (font[7]==2)
ch = 'x';
if (font[7]==3)
ch = 'y';
if (font[7]==4)
ch = 'z';
```

```
if (font[7]==5)
ch = '9';
if (font[7]>4)
font[7]=0;
}
if (dd== 11 && chk_key == 1)
{
x--;
ch = '_';
chk_key = 0;
}
if (dd== 15 && chk_key == 1)
{

chk_key = 0;
x++;
ch = '_';
}
if (dd== 0 && chk_key == 1)
{
for (i=0;i<8;i++)
if (i!=7)
font[i] = 0       ;
chk_key = 0;
ch = '0';
}
if (dd== 13 && chk_key == 1)
{
chk_key = 0;
break;
}

lcd_gotoxy(x,2);
out_lcd(sprintf(str_buf,"%c",ch));
card[x-8]  = ch;
}while(1);

for (i=0;i<15;i++)
{
if (i >(x-8))
{
ch = ' ';
WR_AT24C256((number_data*70)+26+i,ch);
}
else
WR_AT24C256((number_data*70)+26+i,card[i]);
```

```
}
lcd_gotoxy(1,3);
out_lcd(sprintf(str_buf," CAR NUMBER: "));
chk_key = 0;
x = 0;
do
{
dd = read_key();
if (dd == 100)
chk_key = 1;

if (dd <=9 && dd >=0 && chk_key == 1 && x < 4 )
{
chk_key = 0;
out_lcd(sprintf(str_buf,"%bu",dd));
WR_AT24C256((number_data*70)+41+x,dd);
x++;
}
if (x == 4 && dd == 13)
{
break;

}
}
while(1);
lcd_gotoxy(17,4);
out_lcd(sprintf(str_buf,"Bath"));
lcd_gotoxy(1,4);
out_lcd(sprintf(str_buf," Monny : "));
chk_key = 0;
x = 0;
do
{
dd = read_key();
if (dd == 100)
chk_key = 1;

if (dd <=9 && dd >=0 && chk_key == 1 && x < 6 )
{
chk_key = 0;
out_lcd(sprintf(str_buf,"%bu",dd));
WR_AT24C256((number_data*70)+45+x,dd);
x++;
}
if (x > 1 && dd == 13)
```

```
break;

}
while(1);
for(i=0;i<6;i++)
if (i >= x)
{
ch = '*';
WR_AT24C256((number_data*70)+45+i,ch);
}
WR_AT24C256((number_data*70)+69,100);
dmsec(2000);
number_data = number_data + 1;
WR_AT24C256(0xfff0,number_data);
}
// ฟังก์ชั่นเมนู 6
//=============================================================================
//                  Function Menu 6
//=============================================================================
void func_menu6(void)
{
int temp,set_time[6],set_day[6],dd,chk_key = 0;
temp = 0;
clr_lcd();
RD_DS1307();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"[Set Time] %.2bx:%.2bx:%.2bx ",Hour,Min,Sec));
lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf," Time "));
lcd_gotoxy(7,2);
do
{
dd = read_key();
if (dd >=0 && dd <= 9 && chk_key == 1 && temp <=5)
{
chk_key = 0 ;
out_lcd(sprintf(str_buf,"%d",dd));
set_time[temp] = dd;
temp = temp +1;
if (temp==2)
{
out_lcd(sprintf(str_buf,":"));
}
if (temp==4)
out_lcd(sprintf(str_buf,":"));
}
```

```
if (dd == 100)
chk_key = 1;
if (dd==13 && chk_key == 1 && temp >=5)
break;
}while(1);
temp = 0;
clr_lcd();
RD_DS1307();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf," [Set DAY]  %.2bx/%.2bx/%.2bx ",Date,Mon,Year));
lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf," DAY "));
lcd_gotoxy(7,2);
do
{
dd = read_key();
if (dd >=0 && dd <= 9 && chk_key == 1 && temp <=5)
{
chk_key = 0 ;
out_lcd(sprintf(str_buf,"%d",dd));
set_day[temp] = dd;
temp = temp +1;
if (temp==2)
{
out_lcd(sprintf(str_buf,"/"));
}
if (temp==4)
out_lcd(sprintf(str_buf,"/"));
}
if (dd == 100)
chk_key = 1;
if (dd==13 && chk_key == 1 && temp >=6)
break;

}while(1);
S_Sec =  (set_time[4]*16) + set_time[5];
S_Min =  (set_time[2]*16) + set_time[3];
S_Hour =  (set_time[0]*16) + set_time[1];
S_Year =  (set_day[4]*16) + set_day[5];
S_Mon =  (set_day[2]*16) + set_day[3];
S_Date =  (set_day[0]*16) + set_day[1];

WR_DS1307();
lcd_gotoxy(1,4);
out_lcd(sprintf(str_buf,"%.2bx/%.2bx/%.2bx ",S_Date,S_Mon,S_Year));
```

```
out_lcd(sprintf(str_buf,"  %.2bx/%.2bx/%.2bx ",S_Hour,S_Min,S_Sec));
}


unsigned char xtod(char temp)
{unsigned char a,b,c;
a =  temp /16;
b =  temp % 16;
a = (a *10) + b;
return (a);
}
// ฟังก์ชั่นคำนวณ
unsigned int calulate(unsigned char dd,unsigned char mm,unsigned char yy,unsigned char
hh,unsigned char mi,unsigned char j)
{
unsigned char mm_day[12],Hour_m,Min_m,Date_m,Mon_m,Year_m,i;
int temp1,temp2,temp3,temp4;
//RD_AT24C256(((j*70)+56),100);
Hour_m = RD_AT24C256(((j*70)+51));
Min_m= RD_AT24C256(((j*70)+52));
Date_m =RD_AT24C256(((j*70)+53));
Mon_m = RD_AT24C256(((j*70)+54));
Year_m = RD_AT24C256(((j*70)+55));
mm_day[0] = 31;
mm_day[1] = 28;
mm_day[2] = 31;
mm_day[3] = 30;
mm_day[4] = 31;
mm_day[5] = 30;
mm_day[6] = 31;
mm_day[7] = 31;
mm_day[8] = 30;
mm_day[9] = 31;
mm_day[10] = 30;
mm_day[11] = 31;

temp1  = xtod(yy)- xtod(Year_m);
temp2  = xtod(dd)- xtod(Mon_m);

if (temp2 == 0)
{
temp3 = xtod(hh) - xtod(Hour_m);
temp4 = xtod(mi) - xtod(Min_m);
if (temp4 > 0)
temp3 = temp3+1;
}
```

```
return (temp3);
}

//=============================================================================



//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//                                  Main Program
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void main(void) {
char ch= '_';
int temp=0,temp1,temp2;
unsigned char dd,chk_key = 0 , font[8] , i , x = 8 , j , k ,chk_menu = 0, key_menu = 0
,data_chk=0,number_data=0;
unsigned char hk_card=0,chk_counter=0;
char buffer_char1=0;
dmsec(500);
// ประกาศตั้ง ค่า LCD
lcd_init();

dmsec(100);
// ประกาศตั้งค่า Serail
init_serial();


dmsec(100);
EA = 1;
ES=1;

for(i=0;i<8;i++)
font[i]= 0;

k = RD_AT24C256(0xfff0);
for(i=0;i<k;i++)
{
if (RD_AT24C256((i*70)+56)==100)
chk_counter++;
}
//=============================================
//                set display data
//=============================================
do
{
RD_DS1307();
lcd_gotoxy(1,1);
```

```
out_lcd(sprintf(str_buf,"  CAR PARK SYSTEM "));
lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf," [Date]   %.2bx/%.2bx/%.2bx ",Date,Mon,Year));

do
{
// แสดงผลข้อมูลวันที่เวลา และรอการกด คีย์แพด และการ รูดบัตร
dd = read_key();
RD_DS1307();
lcd_gotoxy(1,3);
out_lcd(sprintf(str_buf," [Time]   %.2bx:%.2bx:%.2bx ",Hour,Min,Sec)); // แสดงผลออกจอ lCD

//dmsec(20);
chk_serial();

tot = 0;
getcard1();// เช็คการรูดบัตร

if (tot==20)
{
j=0;
data_chk=0;
number_data =  RD_AT24C256(0xfff0);
do
{
if (RD_AT24C256(((j*70)+56) != 20))

{
data_chk=0;
for(i=0;i<25;i++)
                        {
                        char_card[i]=RD_AT24C256((j*70)+i+1);
                        if (char_card[i]==rcard[i])
                                        data_chk++;
                        }
if (data_chk >= 24)
                {
                f_show_data1(j+1);
                i = 0;
                temp2=0;
                do
                                {
                                if (RD_AT24C256(((j)*70)+45+i) != '*')
                                                temp2 =(temp2 *10) +
(RD_AT24C256(((j)*70)+45+i));
```

```
                else
                                        break;
                    i++;
                }while(1);



        lcd_gotoxy(15,4);
        out_lcd(sprintf(str_buf," Bath"));
        lcd_gotoxy(2,4);
        out_lcd(sprintf(str_buf," Price  "));


        for(i=0;i<4;i++)
                    if (RD_AT24C256(0xfffa+i) != '*')

    out_lcd(sprintf(str_buf,"%bu",RD_AT24C256(0xfffa+i)));
        i = 0;
        temp1=0;
        do
                    {
                    if (RD_AT24C256(0xfffa+i)!= '*')
                                    temp1 =(temp1 *10) +
(RD_AT24C256(0xfffa+i));

                    else
                                    break;
                i++;
              }while(1);
        temp1 = temp2 - temp1;
                        do
                        {
                                if (temp1 < 10)
                                    {
                                    WR_AT24C256((j*70)+45,temp1);
                                    WR_AT24C256((j*70)+46,'*');
                                    WR_AT24C256((j*70)+47,'*');
                                    WR_AT24C256((j*70)+48,'*');
                                    WR_AT24C256((j*70)+49,'*');
                                    WR_AT24C256((j*70)+50,'*');
                                    break;

                                    }
                                if (temp1 <100)
                                    {
                                    WR_AT24C256((j*70)+45,temp1/10);
```

```
                                        WR_AT24C256((j*70)+46,temp1%10);
                                        WR_AT24C256((j*70)+47,'*');
                                        WR_AT24C256((j*70)+48,'*');
                                        WR_AT24C256((j*70)+49,'*');
                                        WR_AT24C256((j*70)+50,'*');

                                        break;
                                        }
                                if (temp1 < 1000)
                                        {
                                        WR_AT24C256((j*70)+45,temp1/100);

                                WR_AT24C256((j*70)+46,(temp1/10)%10);
                                        WR_AT24C256((j*70)+47,(temp1%10));
                                        WR_AT24C256((j*70)+48,'*');
                                        WR_AT24C256((j*70)+49,'*');
                                        WR_AT24C256((j*70)+50,'*');
                                        break;
                                        }
                                if (temp1 < 10000)
                                        {
                                        WR_AT24C256((j*70)+45,temp1/1000);

WR_AT24C256((j*70)+46,(temp1/100)%100);

WR_AT24C256((j*70)+47,(temp1/10)%10);
                                        WR_AT24C256((j*70)+48,(temp1%10));
                                        WR_AT24C256((j*70)+49,'*');
                                        WR_AT24C256((j*70)+50,'*');
                                        break;

                                        }
                                        }while(1);

        do
                {
                dd = read_key();
                if (dd == 13)
                                break;
                }
        while(1);

        calulate(Date,Mon,Year,Hour,Min,j);

        data_chk=0;
        WR_AT24C256(((j*70)+56),20);
```

```
                out_serial(sprintf(str_buf,"OUT"));
                break;
                }
 }
if (j > number_data)
                {
                lcd_gotoxy(1,4);
                out_lcd(sprintf(str_buf,"      No data "));
                break;
                }
  j++;
}
while(1);
dmsec(2000);
j=0;
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"  CAR PARK SYSTEM "));
lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf," [Date]   %.2bx/%.2bx/%.2bx ",Date,Mon,Year));
}

}
while(dd != 14);
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"    FUNCTION SETUP  "));
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1. Add Data    "));
chk_menu =1;
do
{
dd = read_key();

if (dd == 1)
{
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1. Add Data    "));
chk_menu =1;
}
if (dd == 2)
{
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"2. Edit  Data    "));
chk_menu =2;
}
```

```
if (dd == 3)
{
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"3. Format DATA   "));
chk_menu =3;
}
if (dd == 4)
{
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"4. Edit Monny   "));
chk_menu =4;
}
if (dd == 5)
{
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"5. Edit Rate  "));
chk_menu =5;
}
if (dd == 6)
{
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"6. Set Time   "));
chk_menu =6;
}
if (dd == 11)
{
break;
}
if (dd== 13 )
{
//1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
// เข้าเมนู 1 เพื่อ ทำการ เพิ่มข้อมูล
if(chk_menu ==1)
{
func_menu1();
chk_menu =1;
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"    FUNCTION SETUP  "));
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1. Add Data   "));
chk_menu =1;
dmsec(2000);
}
//2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
// เข้าเมนู 2 เพื่อแสดงผลข้อมูลที่ได้บันทึกไว้ และลบข้อมูล
```

```
if (chk_menu == 2)
{
k = RD_AT24C256(0xfff0);
temp = 0;
chk_key = 0;
do
                {
                dd = read_key();
                if (dd == 100)
                                chk_key = 1;

                if (dd== 13 && chk_key == 1)
                                {
                                chk_key = 0;
                                break;
                                }
                if (dd== 11 && chk_key == 1)
                                {
                                clr_lcd();
                                lcd_gotoxy(1,1);
                                out_lcd(sprintf(str_buf,"     Do you want "));
                                lcd_gotoxy(1,2);
                                out_lcd(sprintf(str_buf,"     to Delete "));
                                lcd_gotoxy(1,3);
                                out_lcd(sprintf(str_buf," Y. press Key Enter"));
                                lcd_gotoxy(1,4);
                                out_lcd(sprintf(str_buf," N. press Key Help"));
                                do
                                                {
                                                dd = read_key();
                                                if (dd == 13)
                                                                {
                                                                WR_AT24C256(((temp-
1)*70)+69,20);
                                                                break;
                                                                }
                                                if (dd == 12)
                                                                {
                                                                lcd_gotoxy(1,2);

                                                                break;
                                                                }
                                                }
                                while(1);
                                f_show_data(temp);
                dmsec(1000);
```

```
                    chk_key = 0;

                    }
          if (dd == 15 && chk_key == 1 && temp < k )
                    {
                    clr_lcd();
                    temp = temp + 1;
                     do
                              {
                              if (RD_AT24C256(((temp-1)*70)+69) == 20)
                                        {
                                        temp = temp + 1;
                                        }
                              else
                                        break;

                              }
                     while(1);
                              chk_key = 0;
                              lcd_gotoxy(1,1);
                              out_lcd(sprintf(str_buf," ID = [%d] ",temp));
                              f_show_data(temp);
                    }
          if (dd== 16 && chk_key == 1 && temp >1 )
                    {
                    clr_lcd();
                    temp = temp - 1;

                     do
                              {
                              if (RD_AT24C256(((temp-1)*70)+69) == 20 &&
temp > 0 )
                                        temp = temp - 1;
                              else
                                        break;

                              }
               while(1);

                    chk_key = 0;
                    lcd_gotoxy(1,1);
                    out_lcd(sprintf(str_buf," ID = [%d] ",temp));
                    f_show_data(temp);
               }
          }
while(1);
```

```
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"    FUNCTION SETUP  "));
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1. Add Data   "));
chk_menu =1;
dmsec(2000);
}
//3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
// เข้าเมนู 3 เพื่อformatข้อมูล
if (chk_menu == 3)
{
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"     Do you want "));
lcd_gotoxy(1,2);
out_lcd(sprintf(str_buf,"     to Format"));
lcd_gotoxy(1,3);
out_lcd(sprintf(str_buf," Y. press Key Enter"));
lcd_gotoxy(1,4);
out_lcd(sprintf(str_buf," N. press Key Help"));
chk_key = 0;
do
                {
          dd = read_key();
                if (dd == 100)
                                chk_key = 1;

                if (dd == 13 && chk_key == 1)
                                {
                                chk_key = 0;
                                WR_AT24C256(0xfff0,0);
                                break;

                                }
          if (dd == 12 && chk_key == 1)
                                {
                                chk_key = 0;
                                break;
                                }
                }
while(1);
dmsec(1000);
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"    FUNCTION SETUP  "));
```

```
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1. Add Data   "));
chk_menu =1;
}
//444444444444444444444444444444444444444444444444444444444444444444444444444444
//        เข้าเมนู 4 เพื่อตั้งค่าจำนวนเงิน
if (chk_menu == 4)
{
k = RD_AT24C256(0xfff0);
temp = 0;
chk_key = 0;
do
                {
                dd = read_key();
                if (dd == 100)
                                chk_key = 1;
                if (dd== 13 && chk_key == 1)
                                {
                                chk_key = 0;
                                break;
                                }

                if (dd== 11 && chk_key == 1)
                                {
                                clr_lcd();
                                lcd_gotoxy(1,1);
                                out_lcd(sprintf(str_buf,"     EDIT Monny"));
                                lcd_gotoxy(17,3);
                                out_lcd(sprintf(str_buf,"Bath"));
                                lcd_gotoxy(1,3);
                                out_lcd(sprintf(str_buf," Monny : "));
                                chk_key = 0;
                                x = 0;
                                do
                                {
                                                dd = read_key();
                                                if (dd == 100)
                                                                chk_key = 1;


                                                if (dd <=9 && dd >=0 && chk_key == 1
&& x < 6 )

                                                                {
                                                                chk_key = 0;

        out_lcd(sprintf(str_buf,"%bu",dd));
```

```
                                                WR_AT24C256(((temp-
1)*70)+45+x,dd);
                                                x++;
                                                }
                                        if (x > 1 && dd == 13)
                                                break;


                        }
                        while(1);
                        for(i=0;i<6;i++)
                                if (i >= x)
                                                {
                                ch = '*';
                                                WR_AT24C256(((temp-
1)*70)+45+i,ch);
                                                }
                        f_show_data(temp);
                        dmsec(2000);
                        chk_key = 0;
                        }
        if (dd == 15 && chk_key == 1 && temp < k )
                        {
                        clr_lcd();
                        temp = temp + 1;
                         do
                                {
                                if (RD_AT24C256(((temp-1)*70)+69) == 20)
                                                temp = temp + 1;
                                else
                                                break;

                                }
                         while(1);
                                chk_key = 0;
                                lcd_gotoxy(1,1);
                                out_lcd(sprintf(str_buf," ID = [%d] ",temp));
                                f_show_data(temp);
                        }
        if (dd== 16 && chk_key == 1 && temp >1 )
                        {
                        clr_lcd();
                        temp = temp - 1;
                         do
                                {
```

```
                                        if (RD_AT24C256(((temp-1)*70)+69) == 20 &&
temp > 0 )
                                                temp = temp - 1;
                                        else
                                                break;

                                        }
                        while(1);
                                chk_key = 0;
                                lcd_gotoxy(1,1);
                                out_lcd(sprintf(str_buf," ID = [%d] ",temp));
                                f_show_data(temp);
                                }
                }
while(1);
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"    FUNCTION SETUP  "));
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1. Add Data    "));
chk_menu =1;
dmsec(2000);
}
// เข้าเมนู 5 ตั้งค่า rate ที่จอครถ
//5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
if(chk_menu ==5)
{
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"Rate Car Park"));
lcd_gotoxy(16,2);
out_lcd(sprintf(str_buf,"Bath"));
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1 Hour = "));
for(i=0;i<4;i++)
                if (RD_AT24C256(0xfffa+i) != '*')
                                out_lcd(sprintf(str_buf,"%bu",RD_AT24C256(0xfffa+i)));
lcd_gotoxy(16,3);
out_lcd(sprintf(str_buf,"Bath"));
lcd_gotoxy(2,3);
out_lcd(sprintf(str_buf,"1 Hour = "));
chk_key = 0;
x=0;
do
                {
                dd = read_key();
```

```
                if (dd == 100)
                                chk_key = 1;

                if (dd <=9 && dd >=0 && chk_key == 1 && x < 4 )
                        {
                                chk_key = 0;
                                out_lcd(sprintf(str_buf,"%bu",dd));
                                WR_AT24C256((0xfffa)+x,dd);
                                x++;
                                }
                 if (x > 1 && dd == 13)
                                break;
                 if (dd == 11)
                                break;
}
while(1);
for(i=0;i<4;i++)
                if (i >= x)
                                {
                ch = '*';
                                WR_AT24C256((0xfffa)+i,ch);
                                }
clr_lcd();
lcd_gotoxy(1,1);
out_lcd(sprintf(str_buf,"    FUNCTION SETUP  "));
lcd_gotoxy(2,2);
out_lcd(sprintf(str_buf,"1. Add Data   "));
dmsec(2000);
chk_menu =1;
}
// เข้าเมนู 6 ตั้งเวลา
//6666666666666666666666666666666666666666666666666666666666666666666666666666666
if(chk_menu ==6)
{
func_menu6();
chk_menu =1;
}
}
}
while(1);
}
while(1);
}
```

```
//=========================================
// set ค่า interrupt serial โดยเก็บค่ามาใส่ตัวแปล
void serial_service() interrupt 4
{ if(TI)
   { TI=0;
         send_status=0;
   }
 if(RI)
   { RI=0;
            serial_buffer[index_in]=SBUF;  // เก็บค่าจาก serial มาใส่ไว้ใน ตัวแปล array
                 index_in++;
                   if(index_in>=serial_buffer_size)
                            index_in=0;
   }
}
//=========================================================
// อ่านค่าจาก serail
unsigned char read_serial()
{ unsigned char dat;
    dat=serial_buffer[index_out];
          index_out++;
              if(index_out>=serial_buffer_size)
                           index_out=0;
 return dat;
}
//=========================================================
// ส่งค่าออก serial
void out_serial(unsigned char dat)
{ unsigned char i;
 for(i=1;i<=dat;i++)
   { while(send_status==1);
            SBUF=str_buf[i-1];
                     send_status=1;

         }
}
//=========================================================

// เช็คค่าใน serial

void chk_serial()
{ static unsigned char da[10],number_data=0;
 unsigned char a_char,k=0,data_chk=0;
 unsigned char i,j;

 if(index_in==index_out)
```

```
   enabled=0;//output_low(enabled);  //lcd.enable = 0;
   delay_ms(20);
     lcd_send_nibble(3);delay_ms(3);
          lcd_send_nibble(3);delay_ms(3);
     lcd_send_nibble(3);delay_ms(3);
     lcd_send_nibble(2);delay_ms(3);

         lcd_send_byte(0,0x28);delay_ms(3);
   lcd_send_byte(0,0x0c);delay_ms(3);
   lcd_send_byte(0,0x01);delay_ms(3);
   lcd_send_byte(0,0x06);delay_ms(3);
}
//=====================================================
//=====================================================
ฟังก์ชั่นตั้งค่าแสดงผลตำแหน่งของ LCD
void lcd_gotoxy( unsigned char x, unsigned char y)
  {unsigned char address;
   switch(y) {
     case 1 : address=0x80;break;
     case 2 : address=0xc0;break;
     case 3 : address=0x94;break;
     case 4 : address=0xd4;break;
   }
   address=address+(x-1);
   lcd_send_byte(0,address);
   delay_us(50);
  }
//=====================================================
//=====================================================
void lcd_putc( unsigned char c)
{ lcd_send_byte(1,c);

}
// ฟังก์ชั่น แสดงผลออกจอ LCD
//=====================================================
void out_lcd(unsigned char n_char)
{ unsigned char i;
  for(i=1;i<=n_char;i++)
    {lcd_putc(str_buf[i-1]);
          }
}
//=====================================================
// ฟังก์ชั่น ลบข้อมูลการแสดงผลของ จอ LCD
void clr_lcd()
{lcd_send_byte(0,0x01);
// delay_ms(1);
```

```
}
// ฟังก์ชั่น ตั้งค่า cursor มาที่ตำแหน่งเริ่มต้น
void cursor_home()
{lcd_send_byte(0,0x02);
}
// ฟังก์ชั่น เลื่อนตำแหน่งการแสดงผลไปทางขวามือ
void lcd_shif_right()
{lcd_send_byte(0,0x05);
}
// ฟังก์ชั่น เลื่อนตำแหน่งการแสดงผลไปทางซ้ายมือ
void lcd_shif_left()
{lcd_send_byte(0,0x07);
}
// ฟังก์ชั่น สั่งงานให้ cursor กระพริบ
void cursor_blink()
{lcd_send_byte(0,0x0f);
}
// ฟังก์ชั่น สั่งงานให้ cursor หยุดกระพริบ
void cursor_no_blink()
{lcd_send_byte(0,0x0e);
}
//=======================
```

SLAVE

```
// ประกาศเรียกใช้ Libraly
#include <reg52.h>
#include <stdio.h>
#include <stdlib.h>


// ประกาศการใช้ตัวแปล
#define serial_buffer_size 50   // ตัวแปล ใช้เก็บค่า serial
static unsigned char idata serial_buffer[serial_buffer_size],chk_door1 = 0 ,chk_door2 = 0;
static unsigned char idata index_in=0,index_out=0,send_status=0, str_buf[22];

/*** Constants *****************************************************/
#define TRUE 1

// ประกาศเรียกใช้ Libraly ที่สร้างขึ้น
#include <func1.c>
#include <lcd_lib.c>


/* set pin for magnatic card reader */
#define pdat    P1_3
#definepclk     P1_4
#definepcin     P1_5

/* set pin for MAX7219 */
sbit MXCLK = P1^0;
sbit MXLDB = P1^1;
sbit MXDAT = P1^2;


static unsigned char idata str_buf[22],        rcard[25],chk_data=200,chk_data1=200;


// ฟังก์ชันการใช้งาน max 7219
//=============================================================
============//              Function MAX7219
//=============================================================
============
// ตั้งค่าตัวเลขแสดงผลสำหรับ 7-segment
unsigned char code SEGTAB[16] = {0x7E,0x30,0x6D,0x79,0x33,0x5B,0x5F,0x70,
0x7F,0x7B,0x77,0x1F,0x4E,0x3D,0x4F,0x47};
unsigned char DISBUF[8];
```

```
void mxbyte (unsigned char add,unsigned char dat) { // max7219 send one byte
        unsigned char i;
        for (i=1;i<=8;i++) { // 8 bit address
                MXDAT = add & 0x80;
                add = add << 1;
                MXCLK = 1;
                MXCLK = 0;
                }
        for (i=1;i<=8;i++) { // 8 bit data
                MXDAT = dat & 0x80;
                dat = dat << 1;
                MXCLK = 1;
                MXCLK = 0;
                }
        MXLDB = 1; // load clock
        MXLDB = 0;
}

void mxset (void) { // max7219 setup
        MXCLK = 0;
        MXLDB = 0;
        mxbyte (0x0f,0); // display - normal
        mxbyte (0x0c,1); // shutdown - normal
        mxbyte (0x09,0); // decode - no decode
        mxbyte (0x0a,5); // intensity
        mxbyte (0x0b,7); // scan limit - 8 digit
}


void mxload (void) { // max7219 load to display
        mxset ();
        mxbyte (1,DISBUF[0]);
        mxbyte (2,DISBUF[1]);
        mxbyte (3,DISBUF[2]);
        mxbyte (4,DISBUF[3]);
        mxbyte (5,DISBUF[4]);
        mxbyte (6,DISBUF[5]);
        mxbyte (7,DISBUF[6]);
        mxbyte (8,DISBUF[7]);
}
// ฟังก์ชั่น หน่วงเวลา
void dmsec (unsigned int count)
unsigned int i; // Keil v5.2 (Speed x 1)
        while (count) {
```

```
        i = 115; while (i>0) i--;
        count--;
        }
}
// ฟังก์ชั่น แสดงค่าออก 7-segment digit สามแถวบน
void show_segment(int temp)
{
int k;
k = SEGTAB[temp/100];
DISBUF[1] = k;
mxload ();
k = SEGTAB[(temp/10)%10];
DISBUF[2] = k;
mxload ();
k = SEGTAB[temp % 10];
DISBUF[3] = k;
mxload ();
}

// ฟังก์ชั่น แสดงค่าออก 7-segment digit สามแถวล่าง
void show_segment1(int temp)
{
int k;
k = SEGTAB[temp/100];
DISBUF[5] = k;
mxload ();
k = SEGTAB[(temp/10)%10];
DISBUF[6] = k;
mxload ();
k = SEGTAB[temp % 10];
DISBUF[7] = k;
mxload ();
}

// ประกาศเรียกใช้ Libraly ที่สร้างขึ้น
#include<commu_lib.c>

// ฟังก์ชั่นตั้งค่า port
void set_serial_port()
{ PCON=PCON|0x80; // setup scon in bit 7 (smod)
  SCON=0x52; // mode 1
  TMOD=0x20; // timer mode 2
  TH1=246;  // set br9600 for 18.432
```

```
TR1=1;
}

// ฟังก์ชัน อ่านค่า Read magnetic Card
//=====================================================
============/                Read magnetic Card
//=====================================================
============
// ฟังก์ชันตั้งค่า Clock
void cardclk(void)
        {
        while( pclk == 0 ){}
        while( pclk == 1 ){}
        }
// ฟังก์ชันอ่านบัตร
void getcard(void)
{
                unsigned char i,j,din;
                bit fstart;
                fstart = 0;
                pdat = 1;
                pclk = 1;


                pcin = 1;
                while( pcin == 1 )
                                {
                                chk_serial();
                                dmsec(5);

                                }
                for( j=0 ; j < 25; j++)
                                {
                                din = 0;
                                for( i=0 ; i < 5 ; i++)
                                                {

                                                cardclk();

                                                if( fstart == 0 )
                                                                {
                                                                while(pdat == 1){ cardclk(); }
                                                                fstart = 1;
                                                                din = 0x40;
                                                                }
```

```
                    else
                                        {
                                        if(pdat == 0) { din = din | 0x80; }
                                        else { din = din & 0x7F; }
                                        din = din >> 1;
                                        }
                                }
                    din = din >> 2;
                    din = din | 0x30;
                    rcard[j] = din;
                    }

          for( j=0 ; j < 25; j++)
                    {
                    // ส่งค่าที่อ่านได้จาก Magnetic Card ออก serial
                    out_serial(sprintf(str_buf,"%c",rcard[j]));

                    }
}

// เริ่ม Main Program
void main()
{

unsigned int k;
P0_0 =0;
P0_1 =0;

  delay_ms(1000);
//  ตั้งค่า config ก่อนเริ่มการใช้งาน LCD
  lcd_init();
//  ตั้งค่า config ก่อนเริ่มการใช้งาน Serial
  set_serial_port();
//แสดงผล ออกจอ LCD
  lcd_gotoxy(1,1);
  out_lcd(sprintf(str_buf," CarPark System"));

//เปิดการใช้ Interrupt
EA = 1;                 /* Start Timer 1 Running */
ES=1;
// แสดงผลค่า 200 ออก 7-segment สาม  digit บน
show_segment(200);
// แสดงผลค่า 200 ออก 7-segment สาม digit ล่าง
show_segment1(200);
```

```
do
{
        // อ่านค่าจากบัตร
        getcard();
        หน่วงเวลา
        delay_ms(1000);

while(1);  // วนลูป

}// end main
```

```
 if(index_in==index_out)
     goto out_1;
 a_char=read_serial();
    for(i=0;i<=8;i++)
            { j=i+1;
                  da[i]=da[j];
                }
        da[9]=a_char;
if (a_char == ';' )
        tt = 1;
if (tt==1)
        {
        // เก็บค่า จากบัตรลงตัวแปลที่ controller อีกตัวส่งเข้ามา
        char_card[data_count]= a_char;
        data_count++;
        // เช็คว่าข้อมูลบัตร ครบหรือยัง
        if (data_count >= 25)
                        {
                        number_data =  RD_AT24C256(0xfff0); // อ่านค่าจำนวนบัตรที่ได้ลงทะเบียนไว้จาก
eeeprom                         j=0;
                        do
                                {
                                data_chk=0;
                                //  วนลูปรับค่าจาก บัตรทั้งหมด 25 ตัวอักษร
                                for(i=0;i<25;i++)
                                                {
                                                // อ่านค่า รหัสบัตร จาก eeeprom

        rcard[i]=RD_AT24C256((j*70)+i+1);
                                                if (char_card[i]==rcard[i]) //
เปรียบเทียบข้อมูลที่ส่งมาจาก Serial กับข้อมูลใน eeeprom
                                                        data_chk++;
                                                }
                                // เช็คข้อมูล ว่าตรงกับ ฐานข้อมูล
                                if (data_chk >= 24)
                                        {
                                        out_serial(sprintf(str_buf,"IN")); // ส่ง
protocol "IN"ไป controller อีกตัวนึง
                                         dmsec(100);
                                        for(i=0;i<6;i++)


                                                                if
(RD_AT24C256((j*70)+45+i) != '*')
```

```
                                                                    {
      // ส่งค่าจำนวนเงินที่เหลือ ของบัตรนั้น ไปที่คอนโทรลเลอร์อีกตัวนึง

      out_serial(sprintf(str_buf,"%bu",RD_AT24C256((j*70)+45+i)));
                                                                    dmsec(400);
                                                                    }

                                                  data_chk=0;

                                                  // เก็บค่าวันที่ เวลาลง หน่วยความจำ eeprom
                                                  WR_AT24C256(((j*70)+51),Hour);
                                                  WR_AT24C256(((j*70)+52),Min);
                                                  WR_AT24C256(((j*70)+53),Date);
                                                  WR_AT24C256(((j*70)+54),Mon);
                                                  WR_AT24C256(((j*70)+55),Year);
                                                  WR_AT24C256(((j*70)+56),100);


                                                  break;
                                                  }
                                            // เช็คว่าข้อมูลไม่มีในฐานข้อมูล
                                            if (j > number_data)
                                                  {
                                                  out_serial(sprintf(str_buf,"ERR"));
                                                  // ส่ง Protocol แจ้ง ERR ออกไปที่ คอนโทรลเลอร์อีกตัว
                                                  break;
                                                  }
                                       j++;
                              }while(1);
                              data_count = 0;
                              tt = 0;
                              }
      }
      out_1:
      a_char=0;
}
//=========================================
```

```
// ฟังก์ชั่นตั้งค่า Config  Serial Interrupt
//=========================================
void serial_service() interrupt 4
{ if(TI)
   { TI=0;
         send_status=0;
   }
 if(RI)
   { RI=0;
            serial_buffer[index_in]=SBUF;    // รับค่าจาก serial มาเก็บไว้ในตัวแปล serial_buffer
                 index_in++;
                   if(index_in>=serial_buffer_size)
                            index_in=0;
   }
}
//=========================================================
// ฟังก์ชั่นอ่านค่า Serial Interrupt
unsigned char read_serial()
{ unsigned char dat;
     dat=serial_buffer[index_out];
          index_out++;
              if(index_out>=serial_buffer_size)
                           index_out=0;
 return dat;
}
//=========================================================
void out_serial(unsigned char dat)
{ unsigned char i;
 for(i=1;i<=dat;i++)
   { while(send_status==1);
              SBUF=str_buf[i-1];
                        send_status=1;

         }
}
//=========================================================
// ฟังก์ชั่นเช็คค่าจาก serial
void chk_serial()
{ static unsigned char da[10];
 unsigned char a_char,k;
 unsigned char i,j;

 if(index_in==index_out)
    { goto out_1;
         }
 a_char=read_serial();
```

```
 if(index_in==index_out)
    { goto out_1;
          }
 a_char=read_serial();
    for(i=0;i<=8;i++)
          { j=i+1;
                da[i]=da[j];
               }
       da[9]=a_char;

out_serial(sprintf(str_buf," %c ",da[9]));

// เช็คค่า protocol จาก serial เพื่อบอกว่าไม่มีข้อมูลที่ใช้ในการค้นหา
 if(da[7]=='E' && da[8]=='R' && da[9]=='R')
     {
//  แสดงผล "- - - " ออก 7-segment
               DISBUF[1] = 1;
               mxload ();
               DISBUF[2] = 1;
               mxload ();
               DISBUF[3] = 1;
               mxload ();
               dmsec(2000);          //   หน่วงเวลา 2 วินาที
          show_segment(chk_data);  // แสดงค่า จำนวนที่จอดรถที่เหลือ อยู่

          }
// เช็คค่า protocol จาก serial เพื่อแสดงว่ามีข้อมูลในฐานข้อมูล
 if(da[4]=='I' && da[5]=='N')
     {
               chk_door1 = 1;
               out_serial(sprintf(str_buf,"Open1"));  // ส่งข้อมูลกลับเพื่อแสดงว่า ประตูกั้นรถถูกยกขึ้น
               P0_0 =1;    // Motor 1 On Open the door
               lcd_gotoxy(1,2);
    out_lcd(sprintf(str_buf,"monny = %c%c%c%c",da[6],da[7],da[8],da[9])); // แสดงค่า
จำนวนเงินคงเหลือ ไปที่จอ LCD
               dmsec(4000);   //หน่วงเวลา 4 วินาที
               lcd_gotoxy(1,2); // แสดงผลตำแหน่ง LCD ที่ 1,2
    out_lcd(sprintf(str_buf,"              "));   // ลบการแสดงผลที่จอ LCD
               }
// เช็คค่า protocol จาก serial เพื่อบอกว่ามีรถออก
 if(da[7]=='O' && da[8]=='U' && da[9]=='T')
    {
         chk_door2 = 1;
         out_serial(sprintf(str_buf,"Open2")); ส่งข้อมูลกลับเพื่อแสดงว่า ประตูกั้นรถถูกยกขึ้น
```

```
            P0_1 =1;    // Motor 2 On Open the door

            }
          out_1:
          a_char=0;
// เช็ค sensor เพื่อลดจำนวนที่จอดรถ ชั้นที่ 1
if (chk_door1 == 1 && P0_4 == 0 )
          {
          chk_data1--;  // รถจำนวนที่จอดรถ
          show_segment1(chk_data1); // แสดงค่าออก 7-segment

          out_serial(sprintf(str_buf,"Close1"));
          dmsec(1000);  // หน่วงเวลา
          P0_0 = 0;
          chk_door1 = 0;
          }
// chk sensor close the door 2
if (chk_door2 == 1 && P0_5 == 0 )
          {
    chk_data1++;  // เพิ่มจำนวนที่จอดรถ
    show_segment1(chk_data1);  // แสดงค่าออก 7-segment
          out_serial(sprintf(str_buf,"Close2"));
          dmsec(1000); // หน่วงเวลา
          P0_1 =0;
          chk_door2 = 0;
          }

// เช็ค sensor เพื่อลดจำนวนที่จอดรถ ชั้นที่ 2
// chk sensor close the door 1
if (chk_door1 == 1 && P0_6 == 0 )
          {
          chk_data--;  // ลดจำนวนที่จอดรถ
          show_segment(chk_data);   // แสดงผลออก 7-segment
          out_serial(sprintf(str_buf,"Close1"));
          dmsec(1000); // หน่วงเวลา
          P0_0 = 0;
          chk_door1 = 0;
          }
// chk sensor close the door 2
if (chk_door2 == 1 && P0_7 == 0 )
          {
    chk_data++;     // เพิ่มจำนวนที่จอดรถ
    show_segment(chk_data);  // แสดงผลออก 7-segment
          out_serial(sprintf(str_buf,"Close2"));
```

```
        dmsec(1000);  // หน่วงเวลา
        P0_1 =0;
        chk_door2 = 0;

        }

  }
//=========================================
```

```
//=============================================================
// ฟังก์ชั่นตรวจสอบข้อมูล
unsigned char bit_test(unsigned char buffer1,unsigned char buffer2)
{   return ((buffer1>>buffer2)&0x01);
}

// ฟังก์ชั่นหน่วงเวลา msec
//=============================================================
void delay_ms(unsigned int ms)
{ unsigned int i;
  unsigned int j;
  for(j=0;j<=ms;j++)
     {for(i=0;i<=120;i++);

          }
}
// ฟังก์ชั่นหน่วงเวลา microsec
//=================================
void delay_us(unsigned int us)
{ unsigned int j;
  for(j=0;j<=us;j++)
          { //_nop_();
                //_nop_();
               }


}
//=============================================================
==========================================
```

```
// ประกาศการเชื่อมต่อ ขาสัญญาณ ของ LCD
//=============================================================================
====================
#define rs P2_0
#define enabled P2_1
#define DB4 P2_2
#define DB5 P2_3
#define DB6 P2_4
#define DB7 P2_5
#define lcd_line_two 0x40
//=============================================================================
===
//unsigned char bit_test(unsigned char buffer1,unsigned char buffer2)
//{   return ((buffer1>>buffer2)&0x01);
//}
//=============================================================================
===
void make_out_data(unsigned char buffer_data)
{ DB4=bit_test(buffer_data,0);
  DB5=bit_test(buffer_data,1);
  DB6=bit_test(buffer_data,2);
  DB7=bit_test(buffer_data,3);
}
//=====================================
//=====================================
void lcd_send_nibble(unsigned char buffer_nibble)
{ make_out_data(buffer_nibble);
  delay_us(2);
  enabled=1;//output_high(enabled);
  delay_us(2);
  enabled=0;//output_low(enabled);
  delay_us(2);
}
//=====================================
//=====================================
void lcd_send_byte( unsigned char address, unsigned char n )
  {   rs=0;//output_low(rs); //rs= = 0;
     delay_us(100);
          rs=address;//lcd.rs = address;
     delay_us(2);
     enabled=0;//output_low(enabled);//lcd.enable = 0;
     lcd_send_nibble(n >> 4);
          lcd_send_nibble(n & 0xf);
}
//=============================================
//=============================================
```

```
// ฟังก์ชั่นตั้งค่า เริ่มต้นการใช้งาน
void lcd_init()
{  rs=0;//output_low(rs);     //lcd.rs = 0;
   enabled=0;//output_low(enabled);  //lcd.enable = 0;
   delay_ms(20);
     lcd_send_nibble(3);delay_ms(3);
          lcd_send_nibble(3);delay_ms(3);
     lcd_send_nibble(3);delay_ms(3);
     lcd_send_nibble(2);delay_ms(3);

         lcd_send_byte(0,0x28);delay_ms(3);
   lcd_send_byte(0,0x0c);delay_ms(3);
   lcd_send_byte(0,0x01);delay_ms(3);
   lcd_send_byte(0,0x06);delay_ms(3);
}
//=====================================================
//=====================================================
ฟังก์ชั่นตั้งค่าแสดงผลตำแหน่งของ LCD
void lcd_gotoxy( unsigned char x, unsigned char y)
 {unsigned char address;
  switch(y) {
    case 1 : address=0x80;break;
    case 2 : address=0xc0;break;
    case 3 : address=0x94;break;
    case 4 : address=0xd4;break;
  }
  address=address+(x-1);
  lcd_send_byte(0,address);
  delay_us(50);
 }
//=====================================================
//=====================================================
void lcd_putc( unsigned char c)
{ lcd_send_byte(1,c);

}
// ฟังก์ชั่น แสดงผลออกจอ LCD
//=====================================================
void out_lcd(unsigned char n_char)
{ unsigned char i;
 for(i=1;i<=n_char;i++)
   {lcd_putc(str_buf[i-1]);
        }
}
//=====================================================
// ฟังก์ชั่น ลบข้อมูลการแสดงผลของ จอ LCD
```

```
void clr_lcd()
{lcd_send_byte(0,0x01);
// delay_ms(1);
}
// ฟังก์ชั่น ตั้งค่า cursor มาที่ตำแหน่งเริ่มต้น
void cursor_home()
{lcd_send_byte(0,0x02);
}
// ฟังก์ชั่น เลื่อนตำแหน่งการแสดงผลไปทางขวามือ
void lcd_shif_right()
{lcd_send_byte(0,0x05);
}
// ฟังก์ชั่น เลื่อนตำแหน่งการแสดงผลไปทางซ้ายมือ
void lcd_shif_left()
{lcd_send_byte(0,0x07);
}
// ฟังก์ชั่น สั่งงานให้ cursor กระพริบ
void cursor_blink()
{lcd_send_byte(0,0x0f);
}
// ฟังก์ชั่น สั่งงานให้ cursor หยุดกระพริบ
void cursor_no_blink()
{lcd_send_byte(0,0x0e);
}
//=====================================================
```

```
// ตั้งค่าตำแหน่งขา สัญญาณการเชื่อมต่อ I2C
#define SCL P1_0
#define SDA P1_1

// ฟังก์ชั่นหน่วงเวลา สำหรับ I2C
void    I2c_Wait(void)
{ unsigned char i;
          for (i=0;i<30;i++) { }
}
// ฟังก์ชั่น clock สำหรับ I2C
void I2c_Clk(void)
{       SCL   =       1;
        I2c_Wait();
        SCL   =       0;
        I2c_Wait();
}

void I2c_Ack(void)
{       SDA   = 0;
        I2c_Clk();
}

void I2c_Nack(void)
{       SDA   = 1;
        I2c_Clk();
}


// ฟังก์ชั่น start I2C
void I2c_Start(void)
{       SCL   =       0;
        SDA   =       1;
        SCL   =       1;
        I2c_Wait();
        SDA   =       0;
        I2c_Wait();
}
// ฟังก์ชั่น stop I2C
void I2c_Stop(void)
{       SCL   =       0;
        SDA   =       0;
        SCL   =       1;
        I2c_Wait();
        SDA   =       1;
        I2c_Wait();
}
```

```
// ฟังก์ชั่น เขียนค่า ลงบัส I2C
void  I2c_Write(unsigned char Data )
{ unsigned char  i;
 bit     out;
          SCL  =        0;
    for (i=0;i<8;i++)
        {
          out   =       Data & 0x80;
          Data =        Data << 1;
          SDA =         out;
          I2c_Clk();
        }
     SDA     =          1;
     I2c_Clk();
}
// ฟังก์ชั่น  อ่านค่าจากบัส I2C
unsigned  char  I2c_Read(void)
{ unsigned char         Data,i;
 bit     out;
        SCL     =       0;
        SDA     =       1;
        I2c_Wait();
        for (i=0;i<8;i++)
      {
                SCL     =       1;
                I2c_Wait();
                out     =       SDA;
                Data    =       Data << 1;
                Data    =       Data | out;
                SCL     =       0;
                I2c_Wait();
          }
      return(Data);
}
// ฟังก์ชั่น หน่วงเวลา
void dmsec (unsigned int count) {
unsigned int i; // Keil v5.2 (Speed x 1)
        while (count) {
        i = 115; while (i>0) i--;
        count--;
        }
}

//=========================================================================
//        Function  RTC
//=========================================================================
```

```
// ฟังก์ชั่น เขียนข้อมูลเวลาลง RTC DS1307
void  WR_DS1307(void)
{
        I2c_Start();
        I2c_Write(0xD0);
        I2c_Write(0x00);
        I2c_Write(S_Sec);   //  บันทึกค่าวินาทีลง DS1307
        I2c_Write(S_Min);  // บันทึกค่านาทีลง DS1307
        I2c_Write(S_Hour); // บันทึกค่าชั่วโมงลง DS1307
        I2c_Write(S_Day); // บันทึกค่าวันลง DS1307
        I2c_Write(S_Date);  // บันทึกค่าวันที่ลง DS1307
        I2c_Write(S_Mon); // บันทึกเดือนลง DS1307
        I2c_Write(S_Year); // บันทึกปีลง DS1307
        I2c_Stop();
        dmsec(10);
        }
// ฟังก์ชั่น อ่านค่าเวลาจาก DS1307
void  RD_DS1307(void)
{       I2c_Start();
        I2c_Write(0xD0);
        I2c_Write(0x00);
        I2c_Start();
        I2c_Write(0xD1);
        Sec     =       I2c_Read();  I2c_Ack();
        Min     =  I2c_Read();  I2c_Ack();
        Hour    =  I2c_Read();  I2c_Ack();
        Day     =  I2c_Read();  I2c_Ack();
        Date    =  I2c_Read();  I2c_Ack();
        Mon     =  I2c_Read();  I2c_Ack();
        Year    =  I2c_Read();  I2c_Nack();
        I2c_Stop();
}

//=================================================================
//ฟังก์ชั่นบันทึกข้อมูลลง EEEProm
void WR_AT24C256(unsigned int address_eeprom,char data_write)
{unsigned char buffer1,buffer2;

 //TR0=0;
 buffer2=address_eeprom>>8;
 buffer1=address_eeprom;
 I2c_Start();
 I2c_Write(0xa0);
 I2c_Write(buffer2);
 I2c_Write(buffer1);
```

```
  I2c_Write(data_write);
  I2c_Stop();
  dmsec(20);
  //delay_ms(20);
  //TR0=1;
}
// อ่านข้อมูลจาก EEEProm
char RD_AT24C256(unsigned int address_eeprom)
{unsigned char buffer1,buffer2;
char dat1;
  //TR0=0;
  buffer2=address_eeprom>>8;
  buffer1=address_eeprom;
  I2c_Start();
  I2c_Write(0xa0);
  I2c_Write(buffer2);
  I2c_Write(buffer1);
  I2c_Stop();
  I2c_Start();
  I2c_Write(0xa1);
  dat1=I2c_Read();
        I2c_Stop();
  //TR0=1;
  //delay_ms(1);
        dmsec(10);
  return (dat1);
}
```

```
// ประกาศการเชื่อมต่อ ขาสัญญาณ ของ LCD
//=====================================================================
#define rs P2_5
#define enabled P2_4
#define DB4 P2_0
#define DB5 P2_1
#define DB6 P2_2
#define DB7 P2_3
#define lcd_line_two 0x40
//=====================================================================
//unsigned char bit_test(unsigned char buffer1,unsigned char buffer2)
//{   return ((buffer1>>buffer2)&0x01);
//}
//=====================================================================
void make_out_data(unsigned char buffer_data)
{ DB4=bit_test(buffer_data,0);
  DB5=bit_test(buffer_data,1);
  DB6=bit_test(buffer_data,2);
  DB7=bit_test(buffer_data,3);
}
//=====================================
//=====================================
void lcd_send_nibble(unsigned char buffer_nibble)
{ make_out_data(buffer_nibble);
  delay_us(2);
  enabled=1;//output_high(enabled);
  delay_us(2);
  enabled=0;//output_low(enabled);
  delay_us(2);
}
//=====================================
//=====================================
void lcd_send_byte( unsigned char address, unsigned char n )
  {   rs=0;//output_low(rs); //rs= = 0;
      delay_us(100);
            rs=address;//lcd.rs = address;
      delay_us(2);
      enabled=0;//output_low(enabled);//lcd.enable = 0;
      lcd_send_nibble(n >> 4);
            lcd_send_nibble(n & 0xf);
}
//=============================================
//=============================================
// ฟังก์ชั่นตั้งค่า เริ่มต้นการใช้งาน
void lcd_init()
{   rs=0;//output_low(rs);      //lcd.rs = 0;
```

ภาคผนวก ข

Data Sheet

'9-4452; Rev 4; 7/03

MAXIM

Serially Interfaced, 8-Digit LED Display Drivers

General Description

he MAX7219/MAX7221 are compact, serial input/out-ut common-cathode display drivers that interface icroprocessors (µPs) to 7-segment numeric LED dis-lays of up to 8 digits, bar-graph displays, or 64 indi-idual LEDs. Included on-chip are a BCD code-B ecoder, multiplex scan circuitry, segment and digit rivers, and an 8x8 static RAM that stores each digit.)nly one external resistor is required to set the seg-ent current for all LEDs. The MAX7221 is compatible ith SPI™, QSPI™, and MICROWIRE™, and has slew-ate-limited segment drivers to reduce EMI.

, convenient 4-wire serial interface connects to all ommon µPs. Individual digits may be addressed and pdated without rewriting the entire display. The IAX7219/MAX7221 also allow the user to select code-decoding or no-decode for each digit.

he devices include a 150µA low-power shutdown ode, analog and digital brightness control, a scan-mit register that allows the user to display from 1 to 8 igits, and a test mode that forces all LEDs on.

or applications requiring 3V operation or segment linking, refer to the MAX6951 data sheet.

Applications

Bar-Graph Displays
Industrial Controllers
Panel Meters
LED Matrix Displays

Features

- ♦ **10MHz Serial Interface**
- ♦ **Individual LED Segment Control**
- ♦ **Decode/No-Decode Digit Selection**
- ♦ **150µA Low-Power Shutdown (Data Retained)**
- ♦ **Digital and Analog Brightness Control**
- ♦ **Display Blanked on Power-Up**
- ♦ **Drive Common-Cathode LED Display**
- ♦ **Slew-Rate Limited Segment Drivers for Lower EMI (MAX7221)**
- ♦ **SPI, QSPI, MICROWIRE Serial Interface (MAX7221)**
- ♦ **24-Pin DIP and SO Packages**

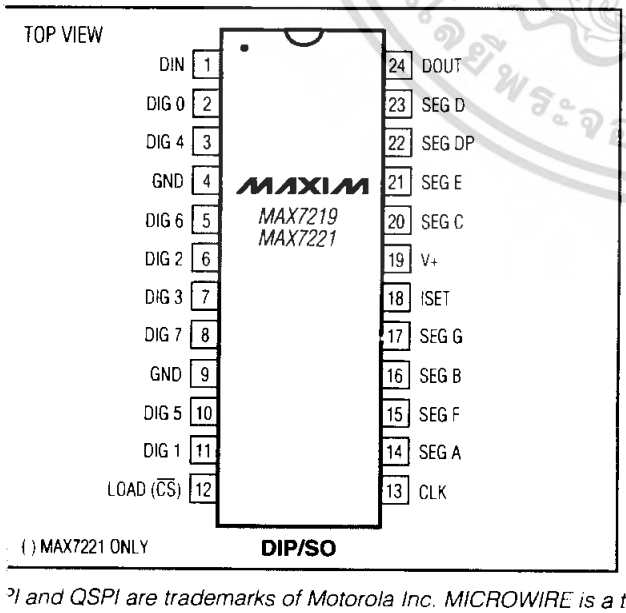
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX7219CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX7219CWG	0°C to +70°C	24 Wide SO
MAX7219C/D	0°C to +70°C	Dice*
MAX7219ENG	-40°C to +85°C	24 Narrow Plastic DIP
MAX7219EWG	-40°C to +85°C	24 Wide SO
MAX7219ERG	-40°C to +85°C	24 Narrow CERDIP

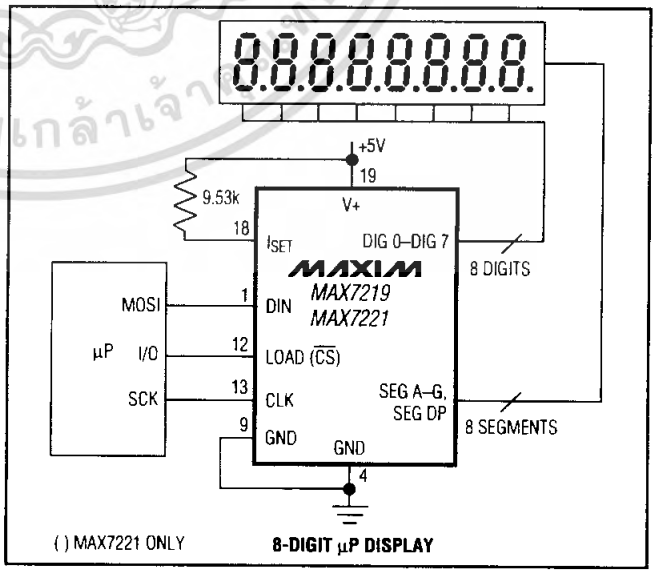
Ordering Information continued at end of data sheet.

**Dice are specified at T_A = +25°C.*

Pin Configuration



Typical Application Circuit



PI and QSPI are trademarks of Motorola Inc. MICROWIRE is a trademark of National Semiconductor Corp.

ABSOLUTE MAXIMUM RATINGS

Voltage (with respect to GND)
V+ .....-0.3V to 6V
DIN, CLK, LOAD, $\overline{CS}$ .....-0.3V to 6V
All Other Pins.....-0.3V to (V+ + 0.3V)
Current
DIG0–DIG7 Sink Current.....500mA
SEGA–G, DP Source Current.....100mA
Continuous Power Dissipation (T_A = +85°C)
Narrow Plastic DIP (derate 13.3mW/°C above +70°C).....1066mW
Wide SO (derate 11.8mW/°C above +70°C).....941mW
Narrow CERDIP (derate 12.5mW/°C above +70°C) ...1000mW

Operating Temperature Ranges (T_{MIN} to T_{MAX})
MAX7219C_G/MAX7221C_G .....0°C to +70°C
MAX7219E_G/MAX7221E_G .....-40°C to +85°C
Storage Temperature Range .....-65°C to +160°C
Lead Temperature (soldering, 10s) .....+300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(V+ = 5V ±10%, R_{SET} = 9.53kΩ ±1%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Operating Supply Voltage	V+		4.0		5.5	V
Shutdown Supply Current	I+	All digital inputs at V+ or GND, T_A = +25°C			150	µA
Operating Supply Current	I+	R_{SET} = open circuit			8	mA
		All segments and decimal point on, $I_{SEG\_}$ = -40mA		330		
Display Scan Rate	f_{OSC}	8 digits scanned	500	800	1300	Hz
Digit Drive Sink Current	I_{DIGIT}	V+ = 5V, V_{OUT} = 0.65V	320			mA
Segment Drive Source Current	I_{SEG}	T_A = +25°C, V+ = 5V, V_{OUT} = (V+ - 1V)	-30	-40	-45	mA
Segment Current Slew Rate (MAX7221 only)	$\Delta I_{SEG}/\Delta t$	T_A = +25°C, V+ = 5V, V_{OUT} = (V+ - 1V)	10	20	50	mA/µs
Segment Drive Current Matching	ΔI_{SEG}			3.0		%
Digit Drive Leakage (MAX7221 only)	I_{DIGIT}	Digit off, V_{DIGIT} = V+			-10	µA
Segment Drive Leakage (MAX7221 only)	I_{SEG}	Segment off, V_{SEG} = 0V			1	µA
Digit Drive Source Current (MAX7219 only)	I_{DIGIT}	Digit off, V_{DIGIT} = (V+ - 0.3V)	-2			mA
Segment Drive Sink Current (MAX7219 only)	I_{SEG}	Segment off, V_{SEG} = 0.3V	5			mA

ELECTRICAL CHARACTERISTICS (continued)

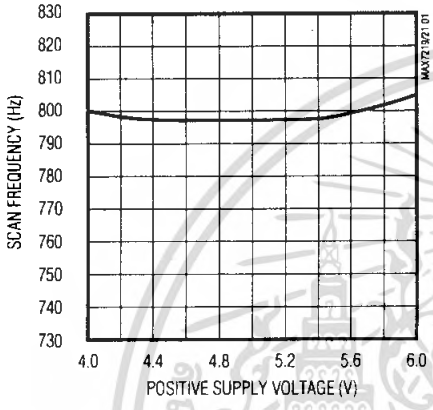
(V+ = 5V ±10%, R_{SET} =9.53kΩ ±1%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
LOGIC INPUTS						
Input Current DIN, CLK, LOAD, $\overline{CS}$	I_{IH}, I_{IL}	V_{IN} = 0V or V+	-1		1	µA
Logic High Input Voltage	V_{IH}		3.5			V
Logic Low Input Voltage	V_{IL}				0.8	V
Output High Voltage	V_{OH}	DOUT, I_{SOURCE} = -1mA	V+ - 1			V
Output Low Voltage	V_{OL}	DOUT, I_{SINK} = 1.6mA			0.4	V
Hysteresis Voltage	ΔV_I	DIN, CLK, LOAD, $\overline{CS}$		1		V
TIMING CHARACTERISTICS						
CLK Clock Period	t_{CP}		100			ns
CLK Pulse Width High	t_{CH}		50			ns
CLK Pulse Width Low	t_{CL}		50			ns
$\overline{CS}$ Fall to SCLK Rise Setup Time (MAX7221 only)	t_{CSS}		25			ns
CLK Rise to $\overline{CS}$ or LOAD Rise Hold Time	t_{CSH}		0			ns
DIN Setup Time	t_{DS}		25			ns
DIN Hold Time	t_{DH}		0			ns
Output Data Propagation Delay	t_{DO}	C_{LOAD} = 50pF			25	ns
Load-Rising Edge to Next Clock Rising Edge (MAX7219 only)	t_{LDCK}		50			ns
Minimum $\overline{CS}$ or LOAD Pulse High	t_{CSW}		50			ns
Data-to-Segment Delay	t_{DSPD}				2.25	ms

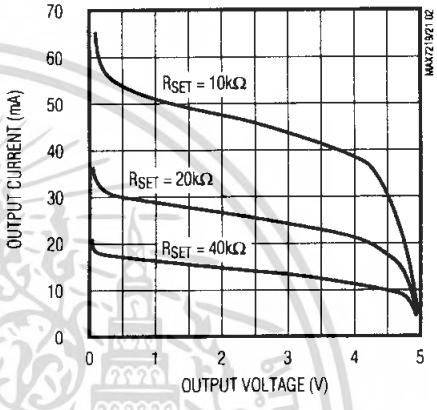
Typical Operating Characteristics

V+ = +5V, T_A = +25°C, unless otherwise noted.)

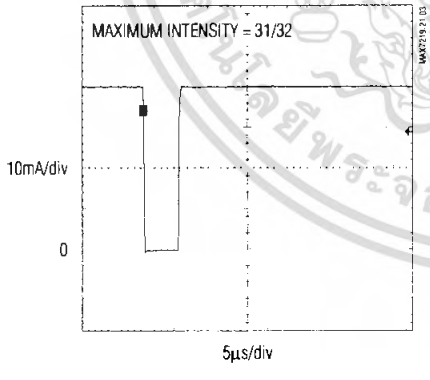
SCAN FREQUENCY vs. POSITIVE SUPPLY VOLTAGE



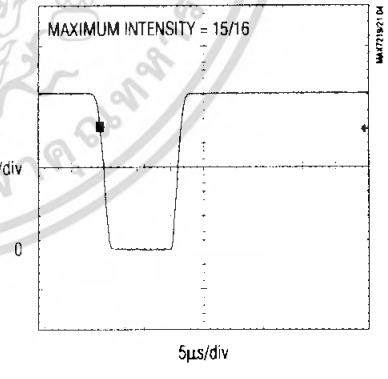
SEGMENT DRIVER OUTPUT CURRENT vs. OUTPUT VOLTAGE



MAX7219 SEGMENT OUTPUT CURRENT



MAX7221 SEGMENT OUTPUT CURRENT

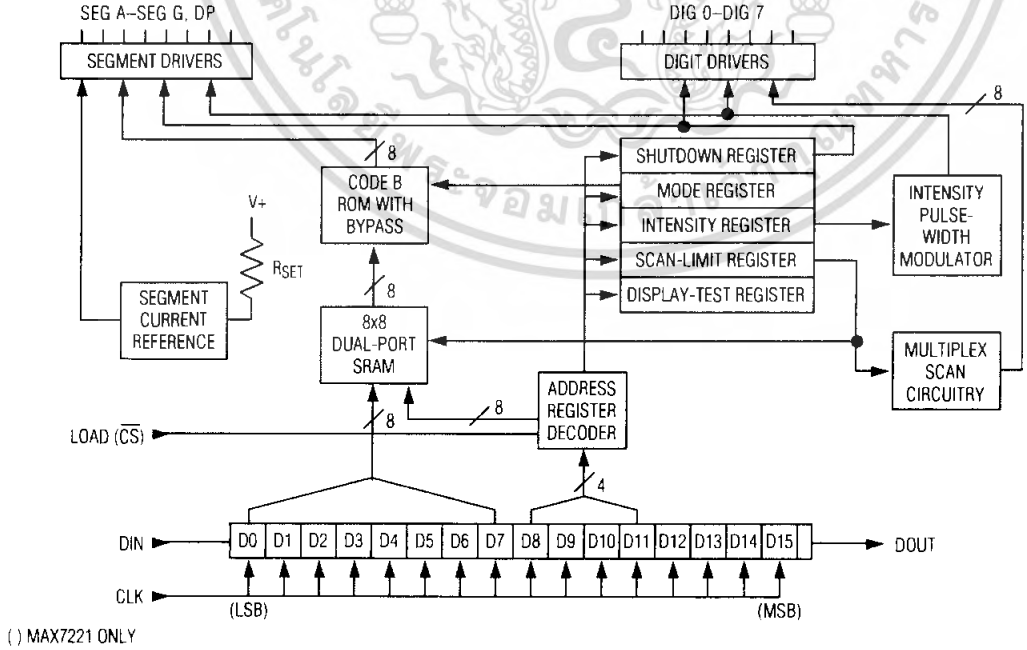


Serially Interfaced, 8-Digit LED Display Drivers

Pin Description

PIN	NAME	FUNCTION
1	DIN	Serial-Data Input. Data is loaded into the internal 16-bit shift register on CLK's rising edge.
2, 3, 5–8, 10, 11	DIG 0–DIG 7	Eight-Digit Drive Lines that sink current from the display common cathode. The MAX7219 pulls the digit outputs to V+ when turned off. The MAX7221's digit drivers are high-impedance when turned off.
4, 9	GND	Ground (both GND pins must be connected)
12	LOAD (MAX7219)	Load-Data Input. The last 16 bits of serial data are latched on LOAD's rising edge.
	$\overline{CS}$ (MAX7221)	Chip-Select Input. Serial data is loaded into the shift register while $\overline{CS}$ is low. The last 16 bits of serial data are latched on $\overline{CS}$'s rising edge.
13	CLK	Serial-Clock Input. 10MHz maximum rate. On CLK's rising edge, data is shifted into the internal shift register. On CLK's falling edge, data is clocked out of DOUT. On the MAX7221, the CLK input is active only while $\overline{CS}$ is low.
14–17, 20–23	SEG A–SEG G, DP	Seven Segment Drives and Decimal Point Drive that source current to the display. On the MAX7219, when a segment driver is turned off it is pulled to GND. The MAX7221 segment drivers are high-impedance when turned off.
18	ISET	Connect to V_{DD} through a resistor (R_{SET}) to set the peak segment current (Refer to *Selecting R_{SET} Resistor* section).
19	V+	Positive Supply Voltage. Connect to +5V.
24	DOUT	Serial-Data Output. The data into DIN is valid at DOUT 16.5 clock cycles later. This pin is used to daisy-chain several MAX7219/MAX7221's and is never high-impedance.

Functional Diagram



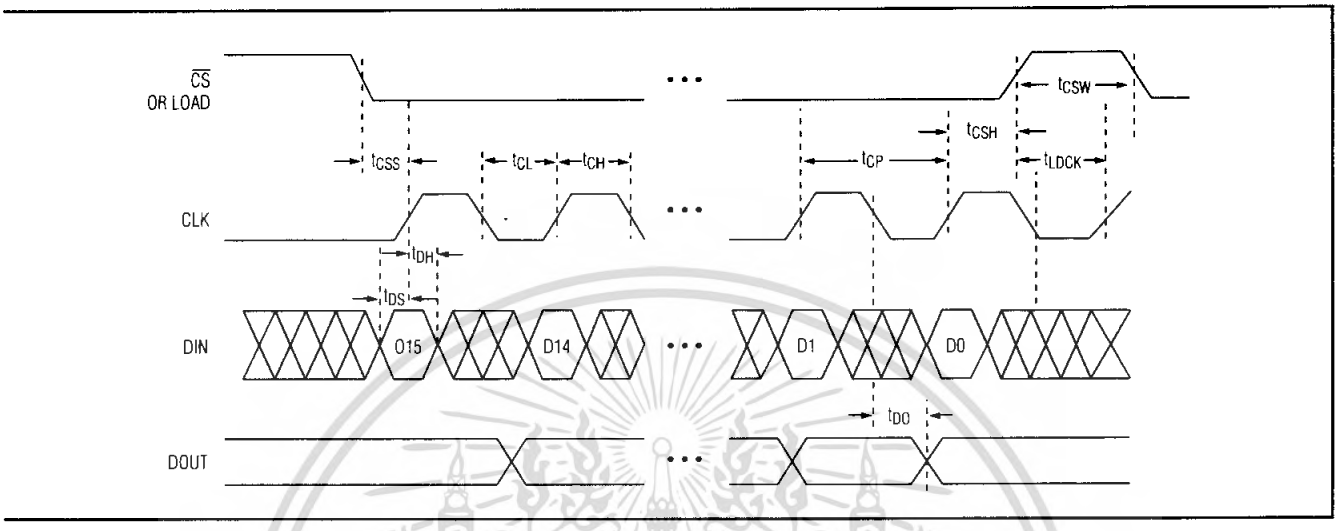


Figure 1. Timing Diagram

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB			DATA				LSB

Detailed Description

MAX7219/MAX7221 Differences

The MAX7219 and MAX7221 are identical except for two parameters: the MAX7221 segment drivers are slew-rate limited to reduce electromagnetic interference (EMI), and its serial interface is fully SPI compatible.

Serial-Addressing Modes

For the MAX7219, serial data at DIN, sent in 16-bit packets, is shifted into the internal 16-bit shift register with each rising edge of CLK regardless of the state of LOAD. For the MAX7221, $\overline{CS}$ must be low to clock data in or out. The data is then latched into either the digit or control registers on the rising edge of LOAD/$\overline{CS}$. LOAD/$\overline{CS}$ must go high concurrently with or after the 16th rising clock edge, but before the next rising clock edge or data will be lost. Data at DIN is propagated through the shift register and appears at DOUT 16.5 clock cycles later. Data is clocked out on the falling edge of CLK. Data bits are labeled D0–D15 (Table 1). D8–D11 contain the register address. D0–D7 contain the data, and D12–D15 are "don't care" bits. The first received is D15, the most significant bit (MSB).

Digit and Control Registers

Table 2 lists the 14 addressable digit and control registers. The digit registers are realized with an on-chip, 8x8 dual-port SRAM. They are addressed directly so that individual digits can be updated and retain data as long as V+ typically exceeds 2V. The control registers consist of decode mode, display intensity, scan limit (number of scanned digits), shutdown, and display test (all LEDs on).

Shutdown Mode

When the MAX7219 is in shutdown mode, the scan oscillator is halted, all segment current sources are pulled to ground, and all digit drivers are pulled to V+, thereby blanking the display. The MAX7221 is identical, except the drivers are high-impedance. Data in the digit and control registers remains unaltered. Shutdown can be used to save power or as an alarm to flash the display by successively entering and leaving shutdown mode. For minimum supply current in shutdown mode, logic inputs should be at ground or V+ (CMOS-logic levels).

Typically, it takes less than 250µs for the MAX7219/MAX7221 to leave shutdown mode. The display driver can be programmed while in shutdown mode, and shutdown mode can be overridden by the display-test function.

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

Initial Power-Up

On initial power-up, all control registers are reset, the display is blanked, and the MAX7219/MAX7221 enter shutdown mode. Program the display driver prior to display use. Otherwise, it will initially be set to scan one digit, it will not decode data in the data registers, and the intensity register will be set to its minimum value.

Decode-Mode Register

The decode-mode register sets BCD code B (0-9, E, H, L, P, and -) or no-decode operation for each digit. Each bit in the register corresponds to one digit. A logic high selects code B decoding while logic low bypasses the decoder. Examples of the decode mode control-register format are shown in Table 4.

When the code B decode mode is used, the decoder looks only at the lower nibble of the data in the digit registers (D3–D0), disregarding bits D4–D6. D7, which sets the decimal point (SEG DP), is independent of the decoder and is positive logic (D7 = 1 turns the decimal point on). Table 5 lists the code B font.

When no-decode is selected, data bits D7–D0 correspond to the segment lines of the MAX7219/MAX7221. Table 6 shows the one-to-one pairing of each data bit to the appropriate segment line.

Table 3. Shutdown Register Format (Address (Hex) = 0xXC)

MODE	ADDRESS CODE (HEX)	REGISTER DATA							
		D7	D6	D5	D4	D3	D2	D1	D0
Shutdown Mode	0xXC	X	X	X	X	X	X	X	0
Normal Operation	0xXC	X	X	X	X	X	X	X	1

Table 4. Decode-Mode Register Examples (Address (Hex) = 0xX9)

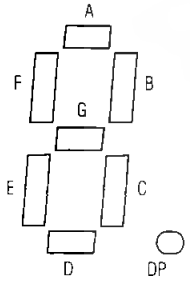
DECODE MODE	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
No decode for digits 7–0	0	0	0	0	0	0	0	0	0x00
Code B decode for digit 0 No decode for digits 7–1	0	0	0	0	0	0	0	1	0x01
Code B decode for digits 3–0 No decode for digits 7–4	0	0	0	0	1	1	1	1	0x0F
Code B decode for digits 7–0	1	1	1	1	1	1	1	1	0xFF

Table 5. Code B Font

7-SEGMENT CHARACTER	REGISTER DATA						ON SEGMENTS = 1							
	D7*	D6–D4	D3	D2	D1	D0	DP*	A	B	C	D	E	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
5		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
—		X	1	0	1	0		0	0	0	0	0	0	1
E		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
L		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
blank		X	1	1	1	1		0	0	0	0	0	0	0

*The decimal point is set by bit D7 = 1

Table 6. No-Decode Mode Data Bits and Corresponding Segment Lines



STANDARD 7-SEGMENT LED

	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Corresponding Segment Line	DP	A	B	C	D	E	F	G

Intensity Control and Interdigit Blanking

The MAX7219/MAX7221 allow display brightness to be controlled with an external resistor (R_{SET}) connected between V+ and ISET. The peak current sourced from the segment drivers is nominally 100 times the current entering ISET. This resistor can either be fixed or variable to allow brightness adjustment from the front panel. Its minimum value should be 9.53kΩ, which typically sets the segment current at 40mA. Display brightness can also be controlled digitally by using the intensity register.

Digital control of display brightness is provided by an internal pulse-width modulator, which is controlled by the lower nibble of the intensity register. The modulator scales the average segment current in 16 steps from a maximum of 31/32 down to 1/32 of the peak current set by R_{SET} (15/16 to 1/16 on MAX7221). Table 7 lists the intensity register format. The minimum interdigit blanking time is set to 1/32 of a cycle.

Table 7. Intensity Register Format (Address (Hex) = 0xXA)

DUTY CYCLE		D7	D6	D5	D4	D3	D2	D1	D0	HEX CODE
MAX7219	MAX7221									
1/32 (min on)	1/16 (min on)	X	X	X	X	0	0	0	0	0xX0
3/32	2/16	X	X	X	X	0	0	0	1	0xX1
5/32	3/16	X	X	X	X	0	0	1	0	0xX2
7/32	4/16	X	X	X	X	0	0	1	1	0xX3
9/32	5/16	X	X	X	X	0	1	0	0	0xX4
11/32	6/16	X	X	X	X	0	1	0	1	0xX5
13/32	7/16	X	X	X	X	0	1	1	0	0xX6
15/32	8/16	X	X	X	X	0	1	1	1	0xX7
17/32	9/16	X	X	X	X	1	0	0	0	0xX8
19/32	10/16	X	X	X	X	1	0	0	1	0xX9
21/32	11/16	X	X	X	X	1	0	1	0	0xXA
23/32	12/16	X	X	X	X	1	0	1	1	0xXB
25/32	13/16	X	X	X	X	1	1	0	0	0xXC
27/32	14/16	X	X	X	X	1	1	0	1	0xXD
29/32	15/16	X	X	X	X	1	1	1	0	0xXE
31/32	15/16 (max on)	X	X	X	X	1	1	1	1	0xXF

Table 8. Scan-Limit Register Format (Address (Hex) = 0xXB)

SCAN LIMIT	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
Display digit 0 only*	X	X	X	X	X	0	0	0	0xX0
Display digits 0 & 1*	X	X	X	X	X	0	0	1	0xX1
Display digits 0 1 2*	X	X	X	X	X	0	1	0	0xX2
Display digits 0 1 2 3	X	X	X	X	X	0	1	1	0xX3
Display digits 0 1 2 3 4	X	X	X	X	X	1	0	0	0xX4
Display digits 0 1 2 3 4 5	X	X	X	X	X	1	0	1	0xX5
Display digits 0 1 2 3 4 5 6	X	X	X	X	X	1	1	0	0xX6
Display digits 0 1 2 3 4 5 6 7	X	X	X	X	X	1	1	1	0xX7

*See *Scan-Limit Register* section for application.

Scan-Limit Register

The scan-limit register sets how many digits are displayed, from 1 to 8. They are displayed in a multiplexed manner with a typical display scan rate of 800Hz with 8 digits displayed. If fewer digits are displayed, the scan rate is $8f_{OSC}/N$, where N is the number of digits scanned. Since the number of scanned digits affects the display brightness, the scan-limit register should not be used to blank portions of the display (such as leading zero suppression). Table 8 lists the scan-limit register format.

f the scan-limit register is set for three digits or less, ndividual digit drivers will dissipate excessive amounts of power. Consequently, the value of the RSET resistor nust be adjusted according to the number of digits displayed, to limit individual digit driver power dissipation. Table 9 lists the number of digits displayed and the corresponding maximum recommended segment current when the digit drivers are used.

Display-Test Register

The display-test register operates in two modes: normal and display test. Display-test mode turns all LEDs on by overriding, but not altering, all controls and digit registers (including the shutdown register). In display-test mode, 8 digits are scanned and the duty cycle is 31/32 (15/16 for MAX7221). Table 10 lists the display-test register format.

Table 9. Maximum Segment Current for 1-, 2-, or 3-Digit Displays

NUMBER OF DIGITS DISPLAYED	MAXIMUM SEGMENT CURRENT (mA)
1	10
2	20
3	30

Table 10. Display-Test Register Format (Address (Hex) = 0xXF)

MODE	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Normal Operation	X	X	X	X	X	X	X	0
Display Test Mode	X	X	X	X	X	X	X	1

Note: The MAX7219/MAX7221 remain in display-test mode (all LEDs on) until the display-test register is reconfigured for normal operation.

No-Op Register

The no-op register is used when cascading MAX7219s or MAX7221s. Connect all devices' LOAD/$\overline{CS}$ inputs together and connect DOUT to DIN on adjacent devices. DOUT is a CMOS logic-level output that easily drives DIN of successively cascaded parts. (Refer to the *Serial Addressing Modes* section for detailed information on serial input/output timing.) For example, if four MAX7219s are cascaded, then to write to the fourth chip, sent the desired 16-bit word, followed by three no-op codes (hex 0xXX0X, see Table 2). When LOAD/$\overline{CS}$ goes high, data is latched in all devices. The first three chips receive no-op commands, and the fourth receives the intended data.

Applications Information

Supply Bypassing and Wiring

To minimize power-supply ripple due to the peak digit driver currents, connect a 10µF electrolytic and a 0.1µF ceramic capacitor between V+ and GND as close to the device as possible. The MAX7219/MAX7221 should be placed in close proximity to the LED display, and connections should be kept as short as possible to minimize the effects of wiring inductance and electromagnetic interference. Also, both GND pins must be connected to ground.

Selecting RSET Resistor and Using External Drivers

The current per segment is approximately 100 times the current in ISET. To select RSET, see Table 11. The MAX7219/MAX7221's maximum recommended segment current is 40mA. For segment current levels above these levels, external digit drivers will be needed. In this application, the MAX7219/MAX7221 serve only as controllers for other high-current drivers or transistors. Therefore, to conserve power, use RSET = 47kΩ when using external current sources as segment drivers.

The example in Figure 2 uses the MAX7219/MAX7221's segment drivers, a MAX394 single-pole double-throw analog switch, and external transistors to drive 2.3" AND2307SLC common-cathode displays. The 5.6V zener diode has been added in series with the decimal point LED because the decimal point LED forward voltage is typically 4.2V. For all other segments the LED forward voltage is typically 8V. Since external transistors are used to sink current (DIG 0 and DIG 1 are used as logic switches), peak segment currents of 45mA are allowed even though only two digits are displayed. In applications where the MAX7219/MAX7221's digit drivers are used to sink current and fewer than four digits are displayed, Table 9 specifies the maximum allowable segment current. RSET must be selected accordingly (Table 11).

Refer to the Power Dissipation section of the Absolute Maximum Ratings to calculate acceptable limits for ambient temperature, segment current, and the LED forward-voltage drop.

DALLAS SEMICONDUCTOR MAXIM

DS1307
64 x 8, Serial, I²C Real-Time Clock

ENERAL DESCRIPTION

e DS1307 serial real-time clock (RTC) is a v-power, full binary-coded decimal (BCD) ck/calendar plus 56 bytes of NV SRAM. dress and data are transferred serially through I²C, bidirectional bus. The clock/calendar vides seconds, minutes, hours, day, date, nth, and year information. The end of the nth date is automatically adjusted for months h fewer than 31 days, including corrections for p year. The clock operates in either the 24- ır or 12-hour format with AM/PM indicator. e DS1307 has a built-in power-sense circuit t detects power failures and automatically tches to the backup supply. Timekeeping ration continues while the part operates from backup supply.

FEATURES

- Real-Time Clock (RTC) 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, Battery-Backed, Nonvolatile (NV) RAM for Data Storage
- I²C Serial Interface
- Programmable Square-Wave Output Signal
- Automatic Power-Fail Detect and Switch Circuitry
- Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
- Optional Industrial Temperature Range: -40°C to +85°C
- Available in 8-Pin Plastic DIP or SO
- Underwriters Laboratory (UL) Recognized

Typical Operating Circuit and Pin Configurations appear at end of data sheet.

RDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
1307	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
1307N	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
1307Z	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
1307ZN	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
1307Z/T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
1307ZN/T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N
1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

notes a lead-free/RoHS-compliant device.
'+" anywhere on the top mark indicates a lead-free device.

Some revisions of this device may incorporate deviations from published specifications known as errata. Multiple revisions of any device be simultaneously available through various sales channels. For information about device errata, click here: www.maxim-ic.com/errata.

3SOLUTE MAXIMUM RATINGS

ltage Range on Any Pin Relative to Ground	....-0.5V to +7.0V
erating Temperature Range (Noncondensing)	
Commercial	..0°C to +70°C
Industrial	-40°C to +85°C
rage Temperature Range	-55°C to +125°C
ldering Temperature (DIP, leads)..	.....+260°C for 10 seconds
ldering Temperature (surface mount) ..	See JPC/JEDEC Standard J-STD-020

sses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, ' functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

ECOMMENDED DC OPERATING CONDITIONS

= 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
upply Voltage	V_{CC}		4.5	5.0	5.5	V
ogic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
ogic 0 Input	V_{IL}		-0.3		+0.8	V
BAT Battery Voltage	V_{BAT}		2.0	3	3.5	V

C ELECTRICAL CHARACTERISTICS

C = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
put Leakage (SCL)	I_{LI}		-1		1	µA
O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
ogic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
ctive Supply Current CL = 100kHz)	I_{CCA}				1.5	mA
andby Current	I_{CCS}	(Note 3)			200	µA
BAT Leakage Current	I_{BATLKG}			5	50	nA
ower-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V

C ELECTRICAL CHARACTERISTICS

C = 0V, V_{BAT} = 3.0V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
BAT Current (OSC ON); QW/OUT OFF	I_{BAT1}			300	500	nA
BAT Current (OSC ON); QW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
BAT Data-Retention Current Oscillator Off)	I_{BATDR}			10	100	nA

ARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

C ELECTRICAL CHARACTERISTICS

(CC = **4.5V to 5.5V**; T_A = 0°C to +70°C, T_A = -40°C to +85°C.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
L Clock Frequency	f_{SCL}		0		100	kHz
s Free Time Between a STOP and ART Condition	t_{BUF}		4.7			µs
ld Time (Repeated) START idition	$t_{HD:STA}$	(Note 4)	4.0			µs
W Period of SCL Clock	t_{LOW}		4.7			µs
GH Period of SCL Clock	t_{HIGH}		4.0			µs
up Time for a Repeated START ndition	$t_{SU:STA}$		4.7			µs
ta Hold Time	$t_{HD:DAT}$		0			µs
ta Setup Time	$t_{SU:DAT}$	(Notes 5, 6)	250			ns
e Time of Both SDA and SCL nals	t_R				1000	ns
l Time of Both SDA and SCL nals	t_F				300	ns
up Time for STOP Condition	$t_{SU:STO}$		4.7			µs

APACITANCE

(= +25°C)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
n Capacitance (SDA, SCL)	$C_{I/O}$				10	pF
apacitance Load for Each Bus ne	C_B	(Note 7)			400	pF

te 1: All voltages are referenced to ground.

te 2: Limits at -40°C are guaranteed by design and are not production tested.

te 3: I_{CCS} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.

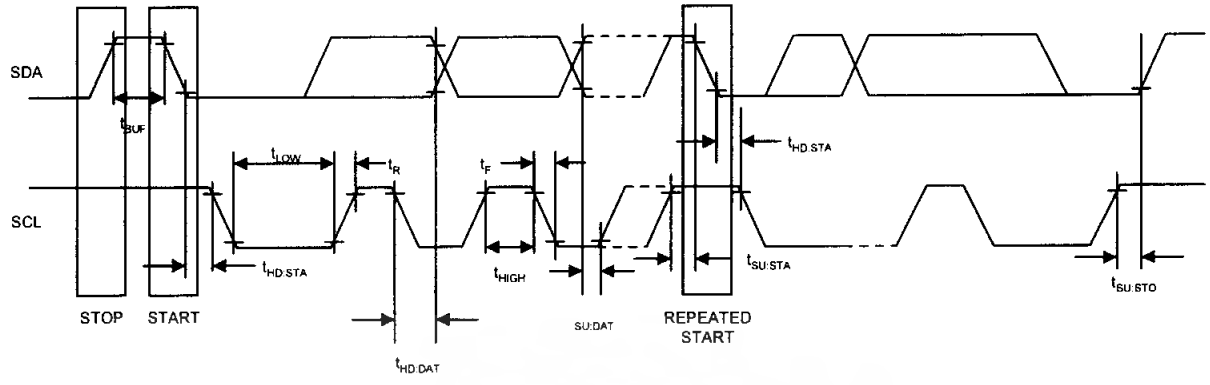
te 4: After this period, the first clock pulse is generated.

te 5: A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the $V_{IH(MIN)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

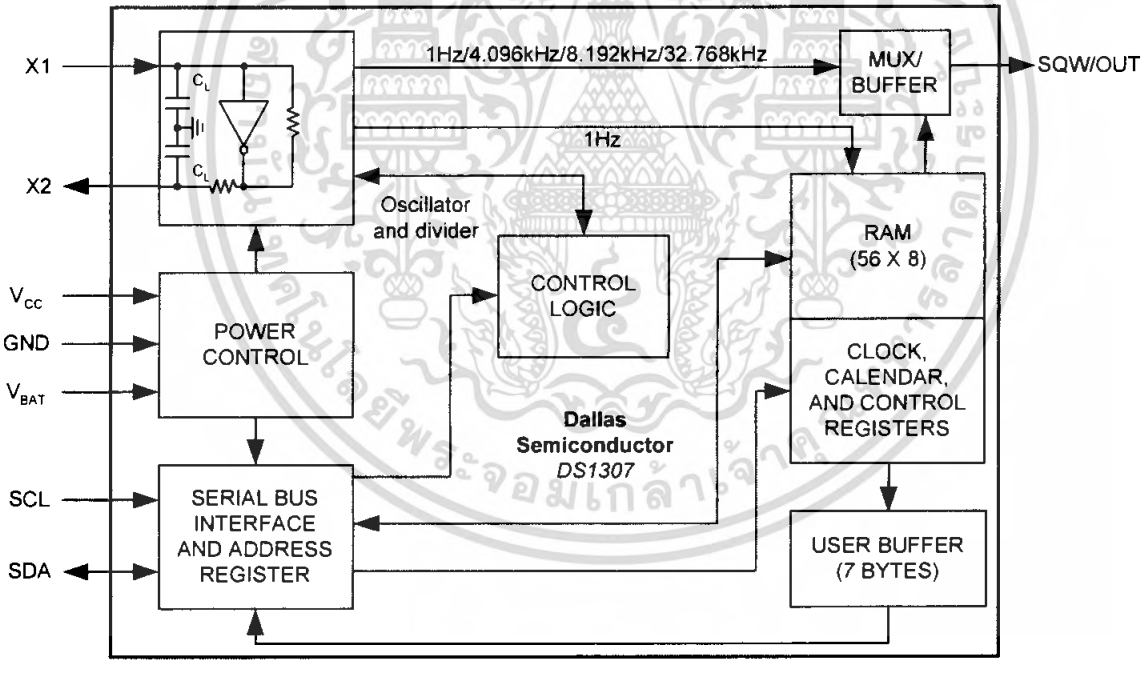
te 6: The maximum $t_{HD:DAT}$ only has to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.

te 7: C_B—total capacitance of one bus line in pF.

MING DIAGRAM

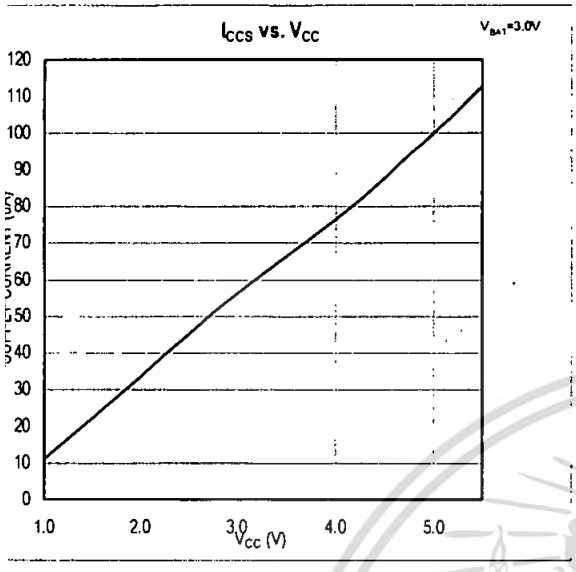


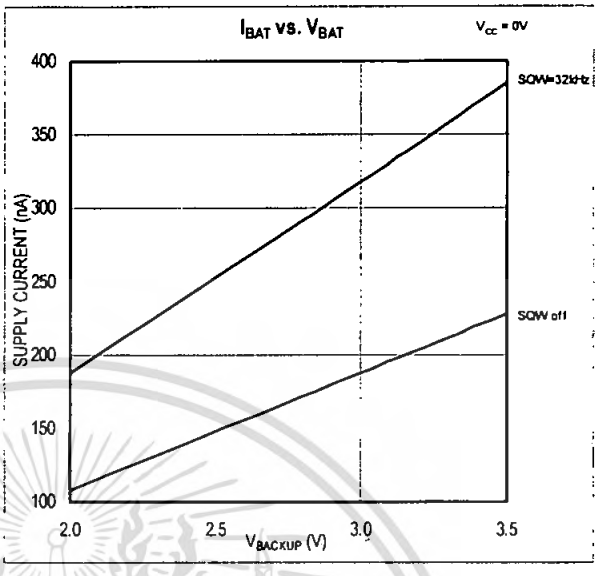
gure 1. Block Diagram

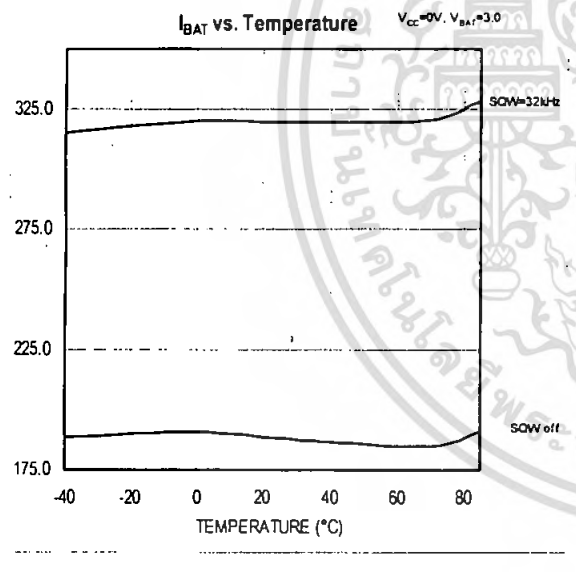


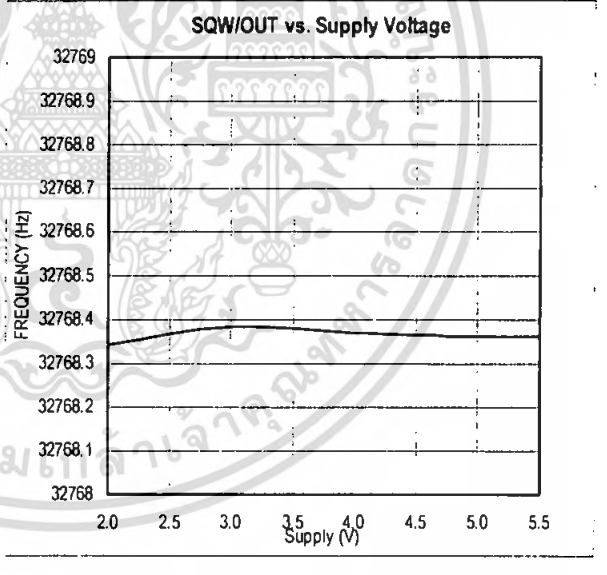
'PICAL OPERATING CHARACTERISTICS

$_C$ = 5.0V, T_A = +25°C, unless otherwise noted.)









N DESCRIPTION

PIN	NAME	FUNCTION
1	X1	Connections for Standard 32.768kHz Quartz Crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (C_L) of 12.5pF. X1 is the input to the oscillator and can optionally be connected to an external 32.768kHz oscillator. The output of the internal oscillator, X2, is floated if an external oscillator is connected to X1. **Note:** For more information on crystal selection and crystal layout considerations, refer to *Application Note 58: Crystal Considerations with Dallas Real-Time Clocks.*
2	X2	
3	V_{BAT}	Backup Supply Input for Any Standard 3V Lithium Cell or Other Energy Source. Battery voltage must be held between the minimum and maximum limits for proper operation. Diodes in series between the battery and the V_{BAT} pin may prevent proper operation. If a backup supply is not required, V_{BAT} must be grounded. The nominal power-fail trip point (V_{PF}) voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAhr or greater will back up the DS1307 for more than 10 years in the absence of power at +25°C. UL recognized to ensure against reverse charging current when used with a lithium battery. Go to: www.maxim-ic.com/qa/info/ul/.
4	GND	Ground
5	SDA	Serial Data Input/Output. SDA is the data input/output for the I²C serial interface. The SDA pin is open drain and requires an external pullup resistor.
6	SCL	Serial Clock Input. SCL is the clock input for the I²C interface and is used to synchronize data movement on the serial interface.
7	SWQ/OUT	Square Wave/Output Driver. When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square-wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pullup resistor. SQW/OUT operates with either V_{CC} or V_{BAT} applied.
8	V_{CC}	Primary Power Supply. When voltage is applied within normal limits, the device is fully accessible and data can be written and read. When a backup supply is connected to the device and V_{CC} is below V_{TP}, read and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage.

ETAILED DESCRIPTION

e DS1307 is a low-power clock/calendar with 56 bytes of battery-backed SRAM. The clock/calendar vides seconds, minutes, hours, day, date, month, and year information. The date at the end of the nth is automatically adjusted for months with fewer than 31 days, including corrections for leap year. e DS1307 operates as a slave device on the I²C bus. Access is obtained by implementing a START idition and providing a device identification code followed by a register address. Subsequent registers be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT}, the vice terminates an access in progress and resets the device address counter. Inputs to the device will not recognized at this time to prevent erroneous data from being written to the device from an out-of-erance system. When V_{CC} falls below V_{BAT}, the device switches into a low-current battery-backup de. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} +0.2V and ognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main ments of the serial RTC.

SCILLATOR CIRCUIT

e DS1307 uses an external 32.768kHz crystal. The oscillator circuit does not require any external istors or capacitors to operate. Table 1 specifies several crystal parameters for the external crystal. ;ure 1. shows a functional schematic of the oscillator circuit. If using a crystal with the specified ıracteristics, the startup time is usually less than one second.

.OCK ACCURACY

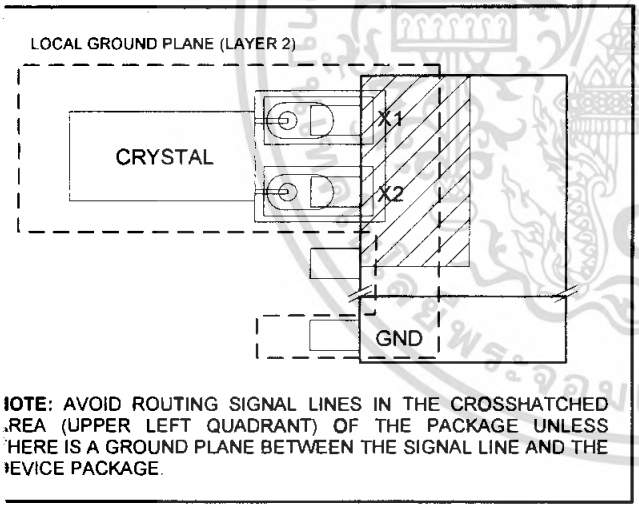
e accuracy of the clock is dependent upon the accuracy of the crystal and the accuracy of the match ween the capacitive load of the oscillator circuit and the capacitive load for which the crystal was nmed. Additional error will be added by crystal frequency drift caused by temperature shifts. External :uit noise coupled into the oscillator circuit may result in the clock running fast. Refer to *Application te 58: Crystal Considerations with Dallas Real-Time Clocks* for detailed information.

ble 1. Crystal Specifications*

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
ominal Frequency	f_O		32.768		kHz
eries Resistance	ESR			45	kΩ
oad Capacitance	C_L		12.5		pF

crystal, traces, and crystal input pins should be isolated from RF generating signals. Refer to lication Note 58: Crystal Considerations for Dallas Real-Time Clocks *for additional specifications.*

ıure 2. Recommended Layout for Crystal



IOTE: AVOID ROUTING SIGNAL LINES IN THE CROSSHATCHED .REA (UPPER LEFT QUADRANT) OF THE PACKAGE UNLESS 'HERE IS A GROUND PLANE BETWEEN THE SIGNAL LINE AND THE EVICE PACKAGE.

C AND RAM ADDRESS MAP

ıle 2 shows the address map for the DS1307 RTC and RAM registers. The RTC registers are located in ress locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a ltibyte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to ıation 00h, the beginning of the clock space.

_OCK AND CALENDAR

e time and calendar information is obtained by reading the appropriate register bytes. Table 2 shows ; RTC registers. The time and calendar are set or initialized by writing the appropriate register bytes. e contents of the time and calendar registers are in the BCD format. The day-of-week register ;rements at midnight. Values that correspond to the day of week are user-defined but must be quential (i.e., if 1 equals Sunday, then 2 equals Monday, and so on.) Illogical time and date entries ult in undefined operation. Bit 7 of Register 0 is the clock halt (CH) bit. When this bit is set to 1, the :illator is disabled. When cleared to 0, the oscillator is enabled.

te that the initial power-on state of all registers is not defined. Therefore, it is important to able the oscillator (CH bit = 0) during initial configuration.

e DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the -hour or 24-hour mode-select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 :he AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20 to hours). The hours value must be re-entered whenever the 12/24-hour mode bit is changed.

ıen reading or writing the time and date registers, secondary (user) buffers are used to prevent errors en the internal registers update. When reading the time and date registers, the user buffers are ıchronized to the internal registers on any I²C START. The time information is read from these ondary registers while the clock continues to run. This eliminates the need to re-read the registers in e the internal registers update during a read. The divider chain is reset whenever the seconds register is itten. Write transfers occur on the I²C acknowledge from the DS1307. Once the divider chain is reset, ıvoid rollover issues, the remaining time and date registers must be written within one second.

ble 2. Timekeeper Registers

DDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00 59
01H	0	10 Minutes			Minutes				Minutes	00 59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1 12 +AM/PM 00 23
		24	PM/ AM							
03H	0	0	0	0	0	DAY			Day	01 07
04H	0	0	10 Date		Date				Date	01 31
05H	0	0	0	10 Month	Month				Month	01 12
06H	10 Year				Year				Year	00 99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
8H-3FH									RAM 56 x 8	00H–FFH

Always reads back as 0.

ONTROL REGISTER

e DS1307 control register is used to control the operation of the SQW/OUT pin.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

7: Output Control (OUT). This bit controls the output level of the SQW/OUT pin when the square-ve output is disabled. If SQWE = 0, the logic level on the SQW/OUT pin is 1 if OUT = 1 and is 0 if IT = 0.

4: Square-Wave Enable (SQWE). This bit, when set to logic 1, enables the oscillator output. The quency of the square-wave output depends upon the value of the RS0 and RS1 bits. With the square-ve output set to 1Hz, the clock registers update on the falling edge of the square wave.

s 1, 0: Rate Select (RS1, RS0). These bits control the frequency of the square-wave output when the uare-wave output has been enabled. The following table lists the square-wave frequencies that can be ected with the RS bits.

RS1	RS0	SQW/OUT OUTPUT	SQWE	OUT
0	0	1Hz	1	X
0	1	4.096kHz	1	X
1	0	8.192kHz	1	X
1	1	32.768kHz	1	X
X	X	0	0	0
X	X	1	0	1

: DATA BUS

e DS1307 supports the I^2C protocol. A device that sends data onto the bus is defined as a transmitter l a device receiving data as a receiver. The device that controls the message is called a master. The ices that are controlled by the master are referred to as slaves. The bus must be controlled by a master ice that generates the serial clock (SCL), controls the bus access, and generates the START and STOP ditions. The DS1307 operates as a slave on the I^2C bus.

ures 3, 4, and 5 detail how data is transferred on the I^2C bus.

Data transfer may be initiated only when the bus is not busy.
During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is high will be interpreted as control signals.

cordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain HIGH.

Start data transfer: A change in the state of the data line, from HIGH to LOW, while the clock is HIGH, defines a START condition.

Stop data transfer: A change in the state of the data line, from LOW to HIGH, while the clock line is HIGH, defines the STOP condition.

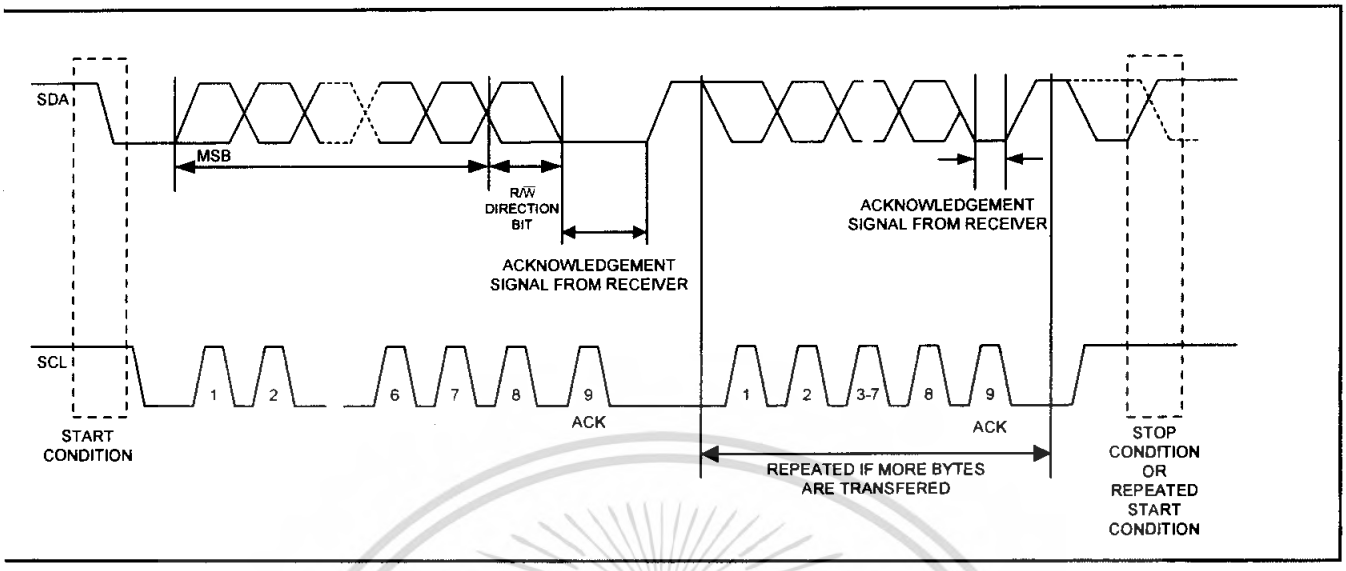
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit. Within the I^2C bus specifications a standard mode (100kHz clock rate) and a fast mode (400kHz clock rate) are defined. The DS1307 operates in the standard mode (100kHz) only.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line HIGH to enable the master to generate the STOP condition.

ıure 3. Data Transfer on I²C Serial Bus



ɔending upon the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

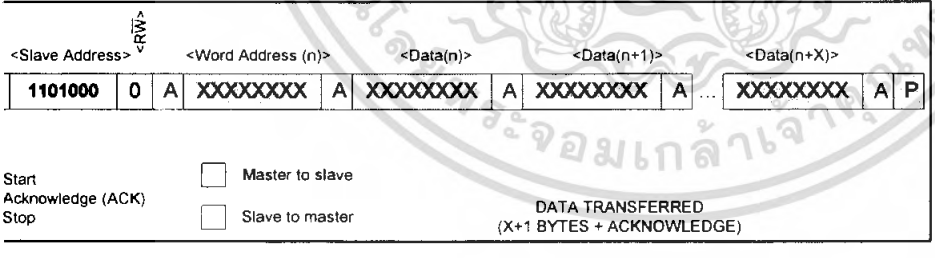
The master device generates all the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

e DS1307 may operate in the following two modes:

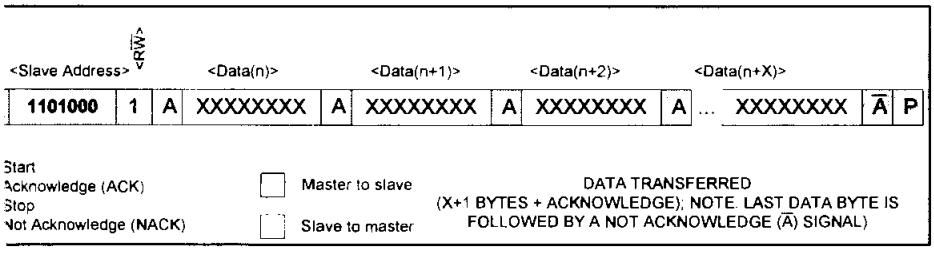
1. **Slave Receiver Mode (Write Mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Hardware performs address recognition after reception of the slave address and direction bit (see Figure 4). The slave address byte is the first byte received after the master generates the START condition. The slave address byte contains the 7-bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$), which for a write is 0. After receiving and decoding the slave address byte, the DS1307 outputs an acknowledge on SDA. After the DS1307 acknowledges the slave address + write bit, the master transmits a word address to the DS1307. This sets the register pointer on the DS1307, with the DS1307 acknowledging the transfer. The master can then transmit zero or more bytes of data with the DS1307 acknowledging each byte received. The register pointer automatically increments after each data byte are written. The master will generate a STOP condition to terminate the data write.

2. **Slave Transmitter Mode (Read Mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. The DS1307 transmits serial data on SDA while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (see Figure 5). The slave address byte is the first byte received after the START condition is generated by the master. The slave address byte contains the 7-bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$), which is 1 for a read. After receiving and decoding the slave address the DS1307 outputs an acknowledge on SDA. The DS1307 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The register pointer automatically increments after each byte are read. The DS1307 must receive a Not Acknowledge to end a read.

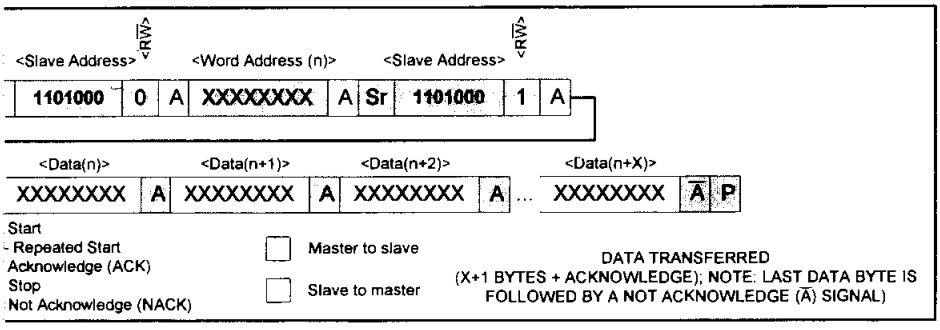
ɟure 4. Data Write—Slave Receiver Mode



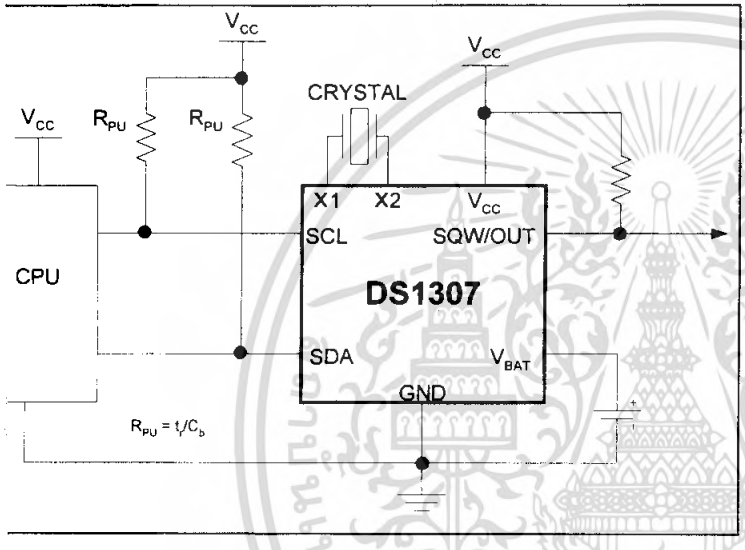
ɟure 5. Data Read—Slave Transmitter Mode



gure 6. Data Read (Write Pointer, Then Read)—Slave Receive and Transmit

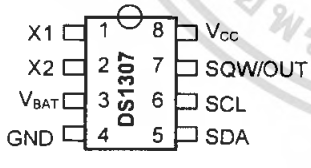


'PICAL OPERATING CIRCUIT

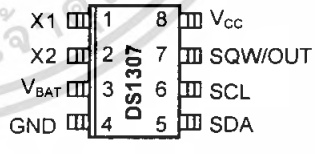


N CONFIGURATIONS

TOP VIEW



PDIP (300 mils)

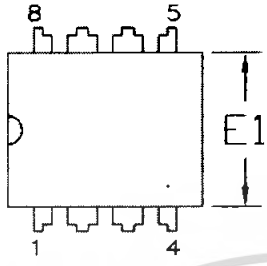


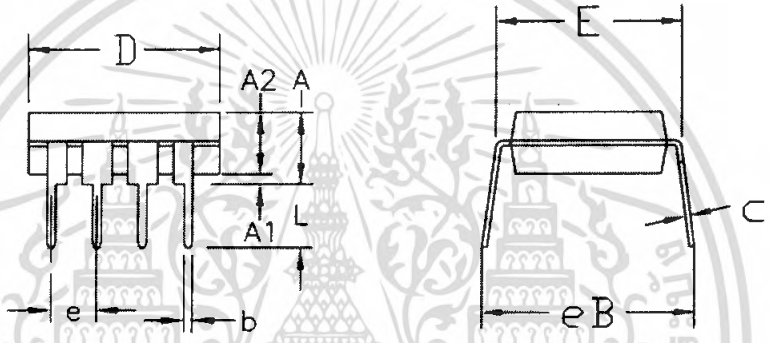
SO (150 mils)

ACKAGE INFORMATION

ıe package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package ;line information, go to **www.maxim-ic.com/DallasPackInfo**.)

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
A	NEW DRAWING	12/01	





	8 PIN	
	MIN	MAX
A	–	0.170
A1	0.015	–
A2	0.115	0.195
b	0.015	0.022
c	0.008	0.012
D	0.360	0.380
E	0.300	0.325
E1	0.240	0.260
e	0.090	0.110
L	0.125	0.135
eB	–	0..430

ALL DIMENSIONS ARE IN INCHES

SIGNATURE		DATE
DOC. CONTROL:		
ENGR. MGR:		
MFC. ENGR:		
CHECKED BY:	TWM	12/01
DRAWN BY:	JFD	12/01

DALLAS SEMICONDUCTOR MAXIM

TITLE: MARKETING OUTLINE, 8 LEAD PLASTIC DUAL-IN-LINE PACKAGE (0.300")

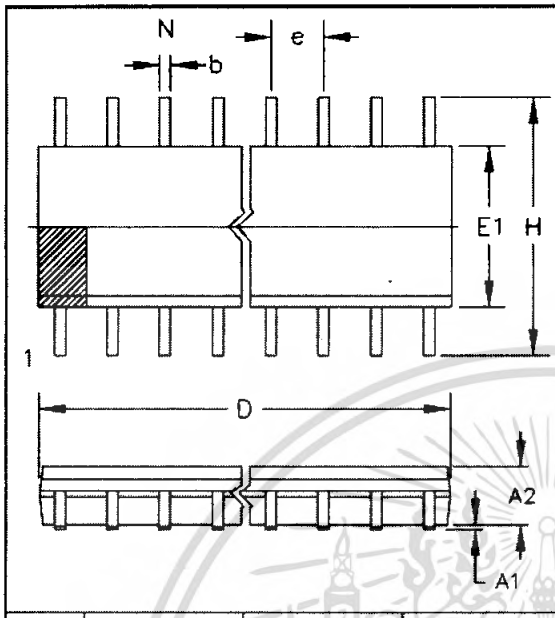
SIZE	FSCM NO	PART NO.	REV
A		56-G5005-000	A

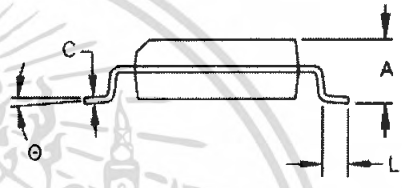
DO NOT SCALE DWG. | SCALE N/A | SHEET 1 OF 1

ACKAGE INFORMATION (continued)

ıe package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package tline information, go to **www.maxim-ic.com/DallasPackInfo**.)

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
A	NEW DRAWING	2/95	J.W.
B	UPDATE DIMENSIONS		





PKG		8 PIN		14 PIN		16 PIN	
LTR		MIN	MAX	MIN	MAX	MIN	MAX
A	IN.	0.053	0.069	0.053	0.069	0.053	0.069
	MM	1.35	1.75	1.35	1.75	1.35	1.75
A1	IN.	0.004	0.010	0.004	0.010	0.004	0.010
	MM	0.10	0.25	0.10	0.25	0.10	0.25
A2	IN.	0.048	0.062	0.048	0.062	0.048	0.062
	MM	1.22	1.57	1.22	1.57	1.22	1.57
b	IN.	0.012	0.020	0.012	0.020	0.012	0.020
	MM	0.030	0.51	0.30	0.51	0.30	0.51
C	IN.	0.007	0.011	0.007	0.011	0.007	0.011
	MM	0.18	0.28	0.18	0.28	0.18	0.28
D	IN.	0.188	0.196	0.337	0.344	0.386	0.393
	MM	4.78	4.98	8.56	8.74	9.80	9.98
e	IN.	.050 BSC		.050 BSC		.050 BSC	
	MM	1.27 BSC		1.27 BSC		1.27 BSC	
E1	IN.	0.150	0.158	0.150	0.158	0.150	0.158
	MM	3.81	4.01	3.81	4.01	3.81	4.01
H	IN.	0.230	0.244	0.230	0.244	0.230	0.244
	MM	5.84	6.20	5.84	6.20	5.84	6.20
L	IN.	0.016	0.050	0.016	0.050	0.016	0.050
	MM	0.41	1.27	0.41	1.27	0.41	1.27
Θ		0°	8°	0°	8°	0°	8°

THE CHAMFER ON THE BODY IS OPTIONAL. IF IT IS NOT PRESENT, A TERMINAL 1 IDENTIFIER MUST BE POSITIONED SO THAT 1/2 OR MORE OF IT'S AREA IS CONTAINED IN THE HATCHED ZONE.

SIGNATURE		DATE
DOC. CONTROL:		
ENGR. MGR:		
MFG. ENGR:		
CHECKED BY:		
DRAWN BY:	M.W.C.	2/95
DO NOT SCALE DWG.		

DALLAS SEMICONDUCTOR MAXIM

TITLE: PACKAGE OUTLINE .150" SOIC 8,14&16 LD.

SIZE	FSCM NO	PART NO	REV
A		56-G2008-001	B

SCALE N/A | SHEET 1 OF 1