ระบบรักษาความปลอดภัยพื้นฐานโดยใช้บัตรหน่วยความจำ

BASIC SECURITY SYSTEM USING EEPROM CARD

โดย

นายพิเชษฐ์ ชัชวารี

นายสมนิพนธ์ รัตนพงศ์

นายสุริยา สาอาจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ระบบรักษาความปลอดภัยพื้นฐานโดยใช้บัตรหน่วยความจำ

BASIC SECURITY SYSTEM USING EEPROM CARD

โดย

นายพิเชษฐ์ ชัชวารี รหัส 41013180

นายสมนิพนธ์ รัตนพงศ์ รหัส 41013192

นายสุริยา สาอาจ รหัส 41013200

อาจารย์ที่ปรึกษา

ดร. สุริภณ สมควรพานิชย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ระบบรักษาความปลอดภัยพื้นฐานโดยใช้บัตรหน่วยความจำ

Basic Security System using EEPROM Card

1. นายพิเชษฐ์ ชัชวารี รหัส 41013180
2. นายสมนิพนธ์ รัตนพงศ์ รหัส 41013192
3. นายสุริยา สาอาจ รหัส 41013200

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(.....)

อาจารย์ที่ปรึกษา

ปริญญานิพนธ์ปีการศึกษา 2543

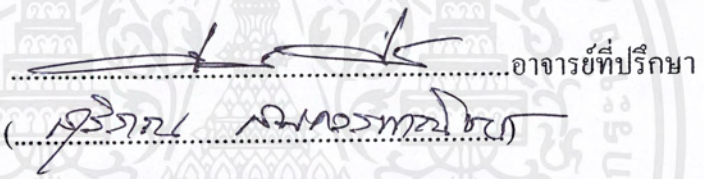
ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบรักษาความปลอดภัยพื้นฐานโดยใช้บัตรหน่วยความจำ

ผู้จัดทำ

1. นายพิเชษฐ์ ชัชวารี รหัส 41013180
2. นายสมนิพนธ์ รัตนพงศ์ รหัส 41013192
3. นายสุริยา สาอาจ รหัส 41013200



.....อาจารย์ที่ปรึกษา

(.....)

ระบบรักษาความปลอดภัยพื้นฐานโดยใช้บัตรหน่วยความจำ

นายพิเชษฐ์ ชัชวารี
นายสมนิพนธ์ รัตนพงศ์
นายสุริยา สาอาจ
ดร. สุริภณ สมควรพานิชย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการนำเสนอระบบรักษาความปลอดภัยพื้นฐาน ซึ่งใช้หน่วยความจำ EEPROM เป็นตัวเก็บข้อมูลรหัสประจำตัว (บัตร) ของบุคคลที่จะใช้อาคาร และใช้ไมโครคอนโทรลเลอร์ทำการประมวลผลโดยจะทำการเปรียบเทียบรหัสในบัตรว่าตรงกับข้อมูลในฐานข้อมูลหรือไม่ ถ้าตรงกันก็จะทำให้ประตูล็อคไฟฟ้าทำงานสามารถเปิดประตูได้ จากนั้นจะทำการส่งข้อมูลให้กับคอมพิวเตอร์ เพื่อทำการเก็บรวบรวมข้อมูลของผู้เข้าออกอาคาร โดยข้อมูลที่เก็บก็คือ รหัส, ชื่อ-นามสกุล, วันและเวลาการเข้า-ออกอาคาร จากการทดสอบ 20 ครั้งโดยไม่พบความผิดพลาดความผิดพลาดในการประมวลผลแต่อย่างไร

Basic Security System using EEPROM Card

Mr. Pichet Chatchawaree
Mr. Somnipon Rattanapong
Mr. Suriya Sa-ard
Dr. Suripon Somkuarnpanit Advisor
2000

Abstract

This Thesis is presented basic security system which uses EEPROM, that is memory unit, to save identity data (Identity Card) of person who use building, and using Microcontroller to compile by comparison that card 's code is similar to database. If they are same, electric door will open automatically, then sending data to computer for collecting data of using building person, which is code, name, surname, day and times of using building. After testing in 20 times, not appear any mistakes in compilation.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

หน้า

บทที่ 2 หลักการและทฤษฎี

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญรูปภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

ความเป็นมาของโครงการ

ในปัจจุบันสถานศึกษาต่าง ๆ นั้นจะประกอบไปด้วยจำนวนนักศึกษาที่มีจำนวนมาก จึงเป็นการยากที่ควบคุมความเป็นระเบียบเรียบร้อย ยกตัวอย่าง การใช้งานในห้องปฏิบัติการของนักศึกษาซึ่งเราไม่สามารถทราบได้เลยว่า มีจำนวนนักศึกษาเท่าใดที่เข้าออกห้องปฏิบัติการนี้ในวัน ๆ หนึ่ง เนื่องจากไม่สามารถตรวจสอบจำนวนนักศึกษาที่เข้ามาใช้งานในห้องปฏิบัติการได้อย่างแน่นอน ดังนั้นเมื่อเกิดการชำรุด สูญหาย ของอุปกรณ์ของเครื่องมือต่าง ๆ เราไม่สามารถหาบุคคลใดมารับผิดชอบได้ ซึ่งโครงการนี้สามารถแก้ไขปัญหาเหล่านี้ได้ในระดับหนึ่ง และยังสามารถป้องกันบุคคลภายนอกที่ไม่มีส่วนเกี่ยวข้องในห้องปฏิบัติการนี้ และโครงการนี้สามารถนำไปประยุกต์ใช้งานได้ในสถานที่ต่าง ๆ อาทิเช่น ในที่พักอาศัยที่อยู่ในรูปแบบของคอนโดมิเนียม หอพัก อพาร์ทเมนท์

วัตถุประสงค์ของโครงการ

1. ศึกษาการอ่านเขียนข้อมูลของ อุปกรณ์ I^2C BUS
2. ศึกษาการเชื่อมต่อของ อุปกรณ์ I^2C BUS กับไมโครคอนโทรลเลอร์
3. ศึกษาการควบคุมการแสดงผลบนคอมพิวเตอร์ด้วยโปรแกรมสำเร็จรูป Vitual Basic
4. ศึกษาการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์
5. เขียนโปรแกรมควบคุมการทำงานของระบบ

ขอบเขตของโครงการ

ระบบจะประกอบไปด้วย 2 ส่วน คือ ส่วนของไมโครคอนโทรลเลอร์ และในส่วนของคอมพิวเตอร์ ซึ่งการทำงานในส่วนของ ไมโครคอนโทรลเลอร์ ซึ่งอยู่ภายในตัวเครื่องนั้นก็คือเครื่องจะแสดง วัน เวลา ที่จอ LCD เมื่อมีการเสียบบัตรไมโครคอนโทรลเลอร์จะทำการประมวลผลข้อมูลว่าตรงกันหรือไม่ ถ้าตรงก็จะทำให้ประตูปลดล็อค และนำรหัส วัน เวลา ของผู้เสียบบัตรเก็บในหน่วยความจำ และเครื่องนี้สามารถตั้งวันเวลา และเปลี่ยนแปลงข้อมูลในบัตรได้ ส่วนการทำงานของคอมพิวเตอร์ มีหน้าที่แสดง ชื่อ-นามสกุล รหัส วัน เวลา ของผู้เสียบบัตร โดยการนำข้อมูลมาจากหน่วยความจำของไมโครคอนโทรลเลอร์

ขีดความสามารถของโครงการ

ขีดความสามารถของระบบสามารถแบ่งออกเป็นข้อๆได้ดังต่อไปนี้

1. แสดงวันและเวลาปัจจุบันที่ LCD และสามารถตั้งวันและเวลาได้
2. สามารถเปลี่ยนแปลงข้อมูลในบัตรได้
3. ควบคุมการ เปิด-ปิด ประตูได้โดยการเสียบบัตร
4. หากต้องการตั้งค่าวันและเวลาหรือเปลี่ยนแปลงข้อมูลในบัตรระบบจะถามรหัสผ่านก่อนถ้ารหัสผ่านไม่ถูกต้องก็ไม่สามารถตั่งค่าวันและเวลาหรือเปลี่ยนแปลงข้อมูลในบัตรได้
5. สามารถบันทึกข้อมูลของการเสียบบัตรในแต่ละครั้งได้ 1,400 ครั้งโดยข้อมูลไม่สูญหาย
6. สามารถเชื่อมต่อกับคอมพิวเตอร์ เพื่อแสดงข้อมูลของการเสียบบัตรในแต่ละครั้งที่หน้าจอมอนิเตอร์ของคอมพิวเตอร์
7. สามารถเพิ่มหรือลดจำนวนของสมาชิกที่ต้องการใช้งานระบบนี้ได้

ประโยชน์ที่ได้รับ

- สามารถตรวจสอบ วัน เวลา การเข้าออกของผู้ใช้อาคารได้
- เพิ่มประสิทธิภาพในการรักษาความปลอดภัย

บทที่ 2
หลักการและทฤษฎี

2.1 ไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลช

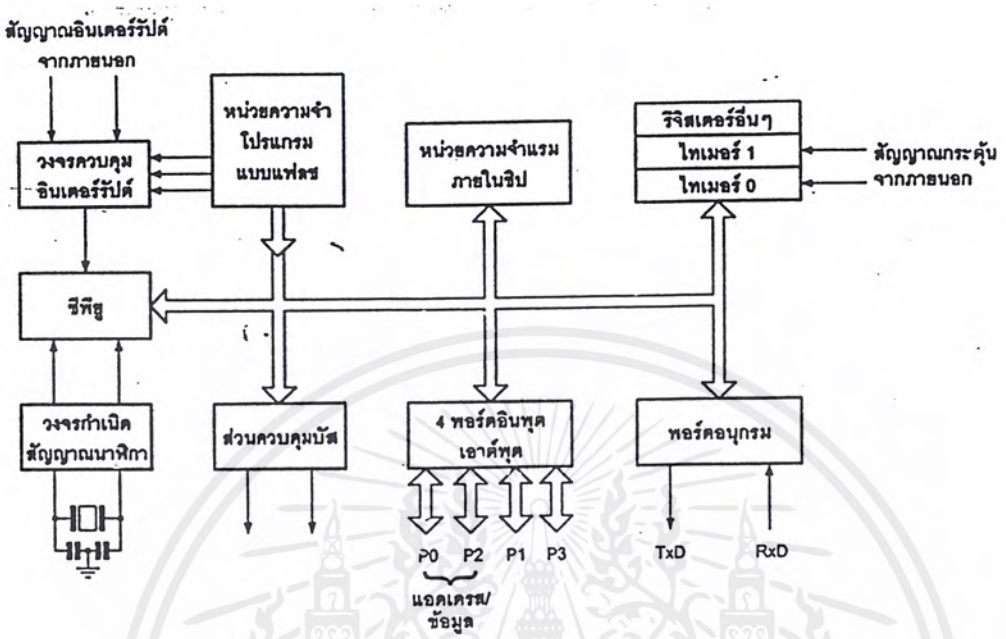
2.1.1 โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลช

คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 อนุกรม AT89xx

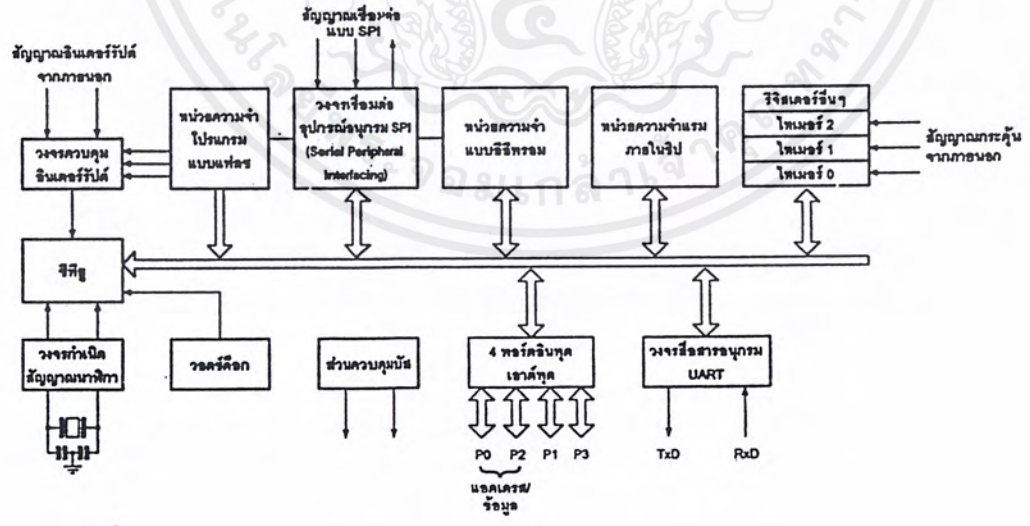
- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางครั้งเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์เพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

ในรูปที่ 2-1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx จะเห็นได้ว่า โครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแตกต่างกันเฉพาะหน่วยความจำโปรแกรมแบบเฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ ในอนุกรม 87xx หน่วยความจำโปรมแกรมภายในจะเป็นแบบอีพรอม และบางเบอร์สามารถโปรแกรมได้เพียงครั้งเดียว

สำหรับในรูปที่ 2-2 เป็นโครงสร้างพื้นฐานของอนุกรม AT89Sxx จะเห็นได้ว่า มีส่วนที่เพิ่มแตกต่างจาก AT89Cxx อยู่หลายส่วน อาทิ วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์ อนุกรมนี้จะใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบ หรือเรียกว่า การโปรแกรมในวงจร ไทเมอร์/ เคาน์เตอร์ขนาด16 บิต ที่เพิ่มเติมเข้ามาอีกหนึ่งตัวเป็นไทเมอร์ 2 และวงจรวอตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียู



รูปที่ 2-1 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89Cxx



รูปที่ 2-2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชในอนุกรม AT89Sxx

2.1.2 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS –51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกันดังแสดงในรูปที่ 2-3 และ 2-4 โดยมีรายละเอียดขั้นตอนดังนี้

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5v

ขา GND เป็นขากราวด์สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต0 (P0.0-P0.7) มี 8 ขาแต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่ง เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตที่นั้นมีสถานะปล่อยลอย (float) จึงมีอินพุร์ตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุร์ตได้ นอกจากนั้น ส่งผลให้ขาพอร์ตที่นั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้น ขาพอร์ตโดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานนั้นให้เป็นไปได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

ขาพอร์ต1 (P1.1–P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 1 ขาใดขาหนึ่ง เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทเมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

ขาพอร์ต2 (P2.0–P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 2 ขาใดขาหนึ่ง เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตที่นั้นมีสถานะปล่อยลอย (float) จึงมีอินพุร์ตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุร์ตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

ขาพอร์ต3 (P3.0–P3.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 3 ขาใดขาหนึ่ง เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตที่นั้นมีสถานะปล่อยลอย (float) จึงมีอินพุร์ตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุร์ตได้ นอกจากนั้นยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นตอนต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่0 หรือขา $\overline{INT0}$

P3.3 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่ 1 หรือขา $\overline{INT1}$

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่องที่ 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่องที่ 1 หรือขา T1

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

P3.7 ใช้เป็นขาสัญญาณ $\overline{RD}$ ในกรณีที่เชื่อมต่อกับหน่วยความจำภายนอก

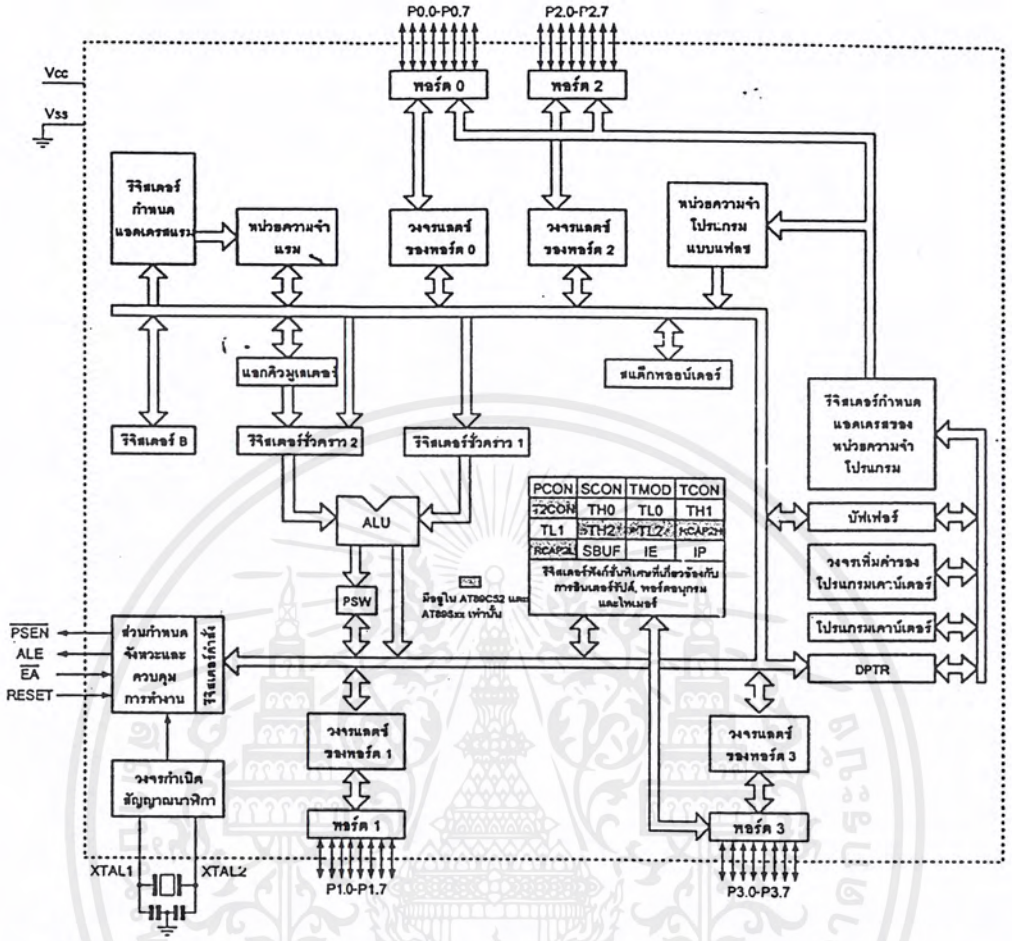
ขารีเซต ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการป้อนสัญญาณเพื่อรีเซต สถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างปกติ

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

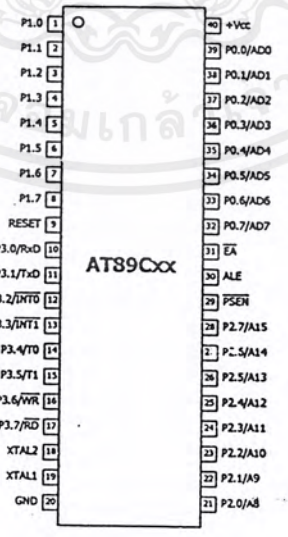
ขา $\overline{PSEN}$ (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์ จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง ในแต่ละแมชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆ ออกมา

ขา $\overline{EA}$/Vpp (External Access enable / Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอก หรือ ภายในตัวไมโครคอนโทรลเลอร์ถ้าหากขานี้เป็น “0” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมจากภายนอก แต่ถ้าหากขานี้เป็น “1” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมจากภายในไมโครคอนโทรลเลอร์ นอกจากนี้ ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12V

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริศตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์



รูปที่ 2-3 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟรช



รูปที่ 2-4 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51

2.1.3 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำภายในหลักๆ อยู่ 2 ส่วน คือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งก็มีขนาดและการจัดสรรแตกต่างกันไปในแต่ละเบอร์

หน่วยความจำโปรแกรม (Program memory)

ในรูปที่ 2-5 แสดงการจัดหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช จะเห็นได้ว่า ทั้งสองเบอร์สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียวหรือรวมกับภายนอกหรือเลือกใช้หน่วยความจำภายนอกอย่างเดียวก็ได้ดังในรูปที่ 2-5(ก) โดยภายใน AT89C51 จะมีหน่วยความจำโปรแกรมภายใน 4 กิโลไบต์ ในขณะที่ AT89C52 จะมีขนาด 8 กิโลไบต์

ในกรณีที่ใช้หน่วยความจำภายในและภายนอกรวมกัน หากใช้ AT89C51ห ก็จะสามารถติดต่อกับหน่วยความจำภายนอกได้ 60 กิโลไบต์ และถ้าใช้เบอร์ AT89C52 จะสามารถติดต่อกับหน่วยความจำโปรแกรมภายนอกได้ 56 กิโลไบต์

หน่วยความจำโปแกรมใช้เก็บข้อมูลของโปแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์หรือที่เรียกว่า โปรแกรมมอนิเตอร์ (moniter program) หากใช้หน่วยความจำภายนอกมักจะบรรจุอยู่ในหน่วยความจำชนิดอีพรอม (EPROM:Erasable Programmable Read-Only Memory) ซึ่งจะสามารถกระทำการอ่านได้เพียงอย่างเดียว

หน่วยความจำโปรแกรมมีแอดเดรสเริ่มต้นที่ 0000H เมื่อซีพียูได้รับการรีเซตให้เริ่มต้นการทำงาน จะต้องเริ่มต้นที่แอดเดรส 0000H นี้เสมอ อย่างไรก็ตาม ในพื้นที่ของหน่วยความจำโปรแกรมไม่ว่าจะใช้งานจากภายในหรือภายนอกก็ตาม ต้องมีการสงวนพื้นที่บางตำแหน่งเอาไว้สำหรับบริการอินเตอร์รัปต์ 6 ประเภท ประเภทละ 8 ไบต์ประกอบด้วย

พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากภายนอก กำหนดไว้ที่แอดเดรส 003H

พื้นที่สำหรับบริการอินเตอร์รัปต์ จากไทเมอร์ 0 กำหนดไว้ที่แอดเดรส 00BH

พื้นที่สำหรับบริการอินเตอร์รัปต์ 1 จากภายนอก กำหนดไว้ที่แอดเดรส 0013H

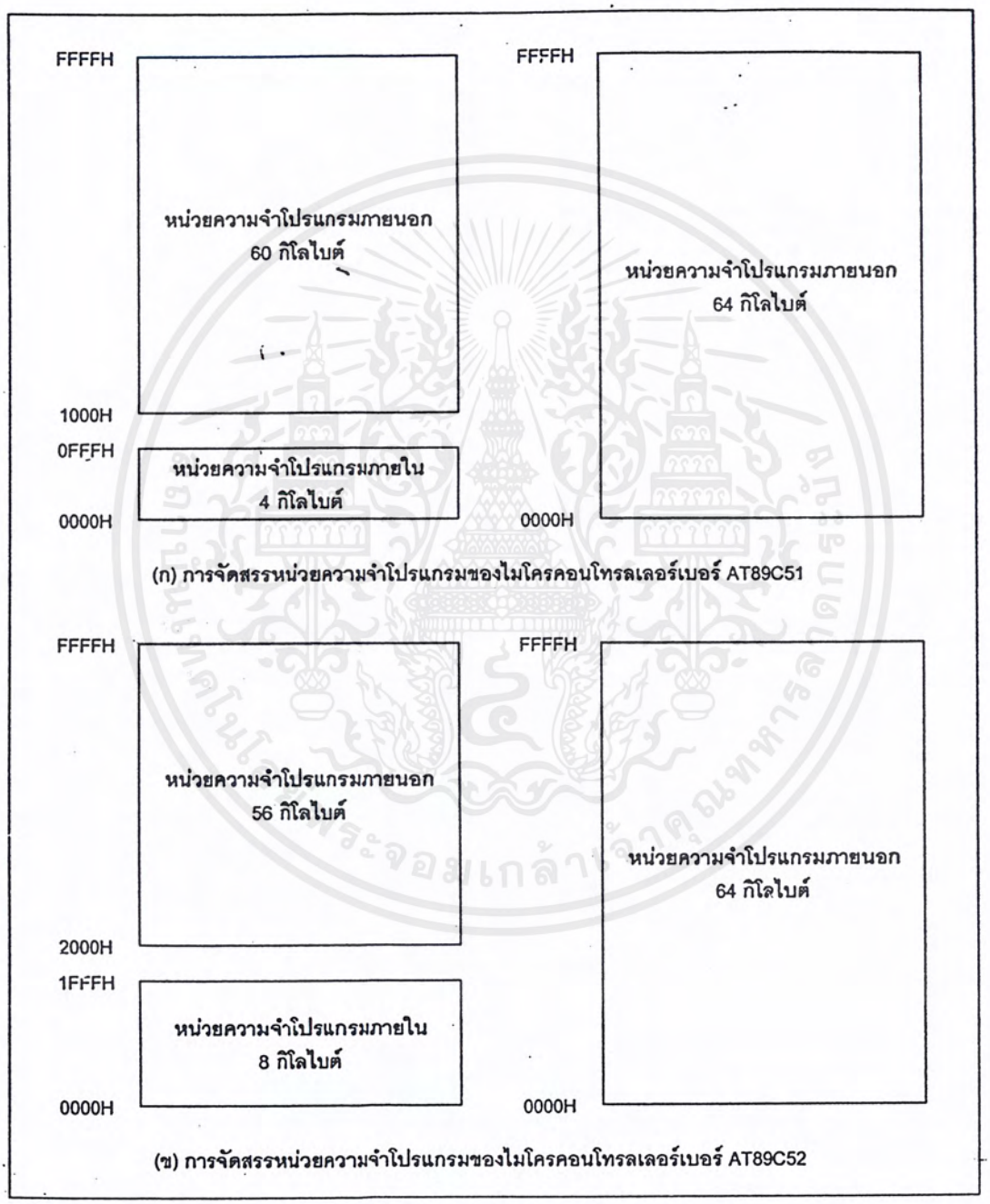
พื้นที่สำหรับบริการอินเตอร์รัปต์ จากไทเมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH

พื้นที่สำหรับบริการอินเตอร์รัปต์ ของการสื่อสารอนุกรม กำหนดไว้ที่แอดเดรส 0023H

พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH

กรณีที่ใช้ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่มีหน่วยความจำโปแกรมภายใน แต่ต้องการติดต่อกับหน่วยความจำโปรแกรมภายนอกด้วย สามารถทำได้โดยต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายในของไม

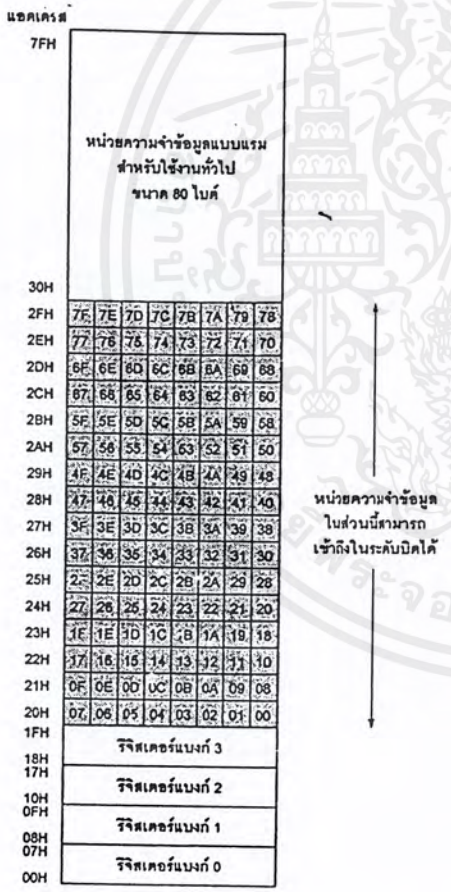
โครคอนโทรลเลอร์ ยกตัวอย่าง ไมโครคอนโทรลเลอร์ AT89C51 มีหน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์ มีแอดเดรสอยู่ระหว่าง 0000H-0FFFH เมื่อหน่วยความจำโปรแกรมภายนอกต้องกำหนดให้แอดเดรสอยู่ในช่วง 1000H-FFFFH



รูปที่ 2-5 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

หน่วยความจำข้อมูล (Data memory)

มีด้วยกัน 2 แบบคือ หน่วยความจำข้อมูลภายนอกและภายในโดยไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89 สามารถติดต่อกับหน่วยความจำข้อมูลภายนอกได้สูงสุด 64 กิโลไบต์ โดยการใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในเป็นแบบแรม (RAM: Random Access Memmory) โดยแต่ละเบอร์จะมีขนาดแตกต่างกันไป ในเบอร์ AT89C51 มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ในขณะที่เบอร์ AT89C52 มีขนาด 256 ไบต์ สำหรับการจัดสรรหน่วยความจำข้อมูลภายในแบ่งเป็น 2 ส่วนคือ หน่วยความจำข้อมูล และรีจิสเตอร์ฟังก์ชั่นพิเศษ (SER:Special Function Register) แต่ละส่วนมีขนาด 128 ไบต์



รูปที่ 2-6โครงสร้างของหน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

แอดเดรส	บิต								
FFH									
F0H	B7	B6	B5	B4	B3	B2	B1	B0	รีจิสเตอร์ B
E0H	A7	A6	A5	A4	A3	A2	A1	A0	รีจิสเตอร์ ACC
D0H	D7	D6	D5	D4	D3	D2	-	D0	รีจิสเตอร์ PSW
B8H	-	-	-	D4	D3	D2	D1	D0	รีจิสเตอร์ IP
B0H	3.7	3.6	3.5	3.4	3.3	3.2	3.1	3.0	รีจิสเตอร์ P3
A8H	D7	-	-	D4	D3	D2	D1	D0	รีจิสเตอร์ IE
A0H	2.7	2.6	2.5	2.4	2.3	2.2	2.1	2.0	รีจิสเตอร์ P2
99H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ SBUF
98H	S7	S6	S5	S4	S3	S2	S1	S0	รีจิสเตอร์ SCON
90H	1.7	1.6	1.5	1.4	1.3	1.2	1.1	1.0	รีจิสเตอร์ P1
8DH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TH1
8CH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TH0
8BH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TL1
8AH	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TL0
89H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ TMOD
88H	T7	T6	T5	T4	T3	T2	T1	T0	รีจิสเตอร์ TCON
87H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ PCON
83H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ DPH
82H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ DPL
81H	ไม่สามารถเข้าถึงระดับบิตได้								รีจิสเตอร์ SP
80H	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	รีจิสเตอร์ P0

หมายเหตุ : ชื่อของแต่ละบิตที่กำหนดในรูปเป็นการกำหนดให้เห็นว่ามีการเรียงลำดับนัยสำคัญของรีจิสเตอร์แต่ละตัว โดยเรียงจากบิตสูงมายังบิตต่ำ สำหรับชื่อที่แท้จริงของแต่ละบิต ให้ตรวจสอบกับรายละเอียดของรีจิสเตอร์ตัวนั้นๆ ต่อไป

รูปที่ 2-7 การจัดสรรพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR)

รีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register:SFR)

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชทั้งหมดมีด้วยกัน 22 ตัว สำหรับในไมโครคอนโทรลเลอร์MCS-51 แบบเฟลชเบอร์ AT89C51 และ28 ตัวในไมโครคอนโทรลเลอร์MCS-51 แบบเฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx ทั้งนี้เนื่องจากใน AT89C52 และ AT89Sxxมีจำนวนไทเมอร์เคาน์เตอร์มากกว่า AT89C51

รีจิสเตอร์ SFR มีแอดเดรสอยู่ระหว่าง 80H-FFH ในพื้นที่ของหน่วยความจำข้อมูลสามารถเข้าถึงได้โดยตรง(direct addressing) ในรูปที่ 2-7 แสดงการจัดสรรพื้นที่ของรีจิสเตอร์ SFR แต่ละตัว สำหรับรายละเอียดเบื้องต้นของรีจิสเตอร์ SFR มีดังนี้

รีจิสเตอร์แสดงสถานะของโปรแกรม (Program Status Word:PSW)

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

CY : แฟลกทด (Carry flag) เป็น "1" เมื่อมีการกระทำคำสั่งทางทางคณิตศาสตร์และลอจิกแล้วค่าของแอกคิวมูลเลเตอร์เกิน 225 (ฐานสิบ) หรือ FFH

AC : แฟลกทดเสริม (Auxiiiary Carry flag) เป็น "1" เมื่อมีการกระทำคำสั่งทางทางคณิตศาสตร์แล้วทำให้เกิดการทดข้ามจากบิต 3 มายังบิต 4 มักใช้ในการ แปลงค่าเป็นเลขฐานสิบ (BCO operat ion)

FO : แฟลกใช้งานทั่วไป เมื่อผู้เขียนโปรแกรมกำหนดค่าที่บิตนี้แล้ว ไม่ว่ากระทำคำสั่งใด ๆ ที่บิตนี้จะไมมีการเปลี่ยนแปลง

RS1 : บิตเลือกรีจิสเตอร์แบงก์ (Register Select1) ใช้งานร่วมกับบิต RS0 เพื่อเลือกแบงก์ของรีจิสเตอร์ R0-R7

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

OV : บิตเกิน (Ovenlow) เป็น "1" เมื่อมีการกระทำคำสั่งทางทางคณิตศาสตร์และลอจิกแล้วทำให้เกิดการทดข้ามจากบิต 6 มายังบิต 7 ของแอกคิวมูลเลเตอร์ หรือแอกคิวมูลเลเตอร์มีค่าเกิน 127 (ฐานสิบ) นอกจากนั้นยังใช้เป็นการแสดงค่าลบอีกด้วย

- : บิตนี้ผู้ใช้งานสามารถกำหนดการใช้งานได้อย่างอิสระ
- P : บิตพาริตี้ (Parity) ใช้ในการตรวจสอบจำนวนค่า "1" ภายในแอกคิวมูลเลเตอร์ ถ้าหากในแอกคิวมูลเลเตอร์ มีจำนวนบิตที่เป็น "1" รวมกันเป็นเลขคู่ และเป็น "0" ถ้าเป็นคี่

แอกคิวมูเลเตอร์ (Accumulator:ACC)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของโมโครคอนโทรลเลอร์ โดยเฉพาะอย่างยิ่งในการคำนวณทางคณิตศาสตร์และลอจิก ก่อนที่จะส่งข้อมูลหรือผลลัพธ์ที่ได้นั้นให้แก่ซีพียูเพื่อทำการประมวลผผลต่อไป อาจเรียกรีจิสเตอร์แอกคิวมูลเตอร์อย่างสั้นๆว่า รีจิสเตอร์ A หรือ ACC

รีจิสเตอร์ A นี้สามารถเข้าถึงระดับบิตได้ นั้นหมายความว่าสามารถกระทำคำสั่งหรือกำหนดค่าในแต่ละบิตของรีจิสเตอร์ตัวนี้ได้อย่างอิสระ

รีจิสเตอร์ B

มีขนาด 8 บิต มีแอดเดรสอยู่ที่F0H มีหน้าที่พิเศษ คือ หากมีความต้องการคูณหรือหารทางคณิตศาสตร์ จะต้องนำข้อมูลที่ต้องการหารหรือคูณนั้นมาเก็บไว้ในรีจิสเตอร์ B นี้ แล้วจึงทำกระทำคำสั่งการคูณหรือหารกับค่าในรีจิสเตอร์ A ต่อไป

ในกรณีที่ไม่ได้มีความต้องการคูณหรือหารข้อมูล สามารถใช้รีจิสเตอร์ B นี้ในการเก็บข้อมูลทั่วไปได้เหมือนกับรีจิสเตอร์ปกติ และสามารถเข้าถึงในระดับบิตได้เช่นเดียวกับรีจิสเตอร์A

โปรแกรมเคาน์เตอร์ (Program Counter : PC)

มีขนาด 16 บิตมีหน้าที่แจ้งแอดเดรสของหน่วยความจำโปรแกรมในตำแหน่งถัดไปที่ซีพียูจะต้องไปทำงาน รีจิสเตอร์ PC เป็นรีจิสเตอร์ตัวเดียวที่ไม่ได้จัดสรรรวมกับรีจิสเตอร์ SFR ตัวอื่นๆ การเปลี่ยนแปลงค่าของรีจิสเตอร์ PC จะขึ้นอยู่กับผลกระทำคำสั่งแต่ละคำสั่งภายในหน่วยความจำโปรแกรมที่ผู้เขียนโปรแกรมกำหนด

รีจิสเตอร์ PC มีความสำคัญมาก โดยเฉพาะอย่างยิ่งในการตรวจสอบการทำงานของโปรแกรมว่าดำเนินไปตามลำดับขั้นตอนตามที่กำหนดไว้หรือไม่

สแต็กพอยน์ตอร์ (Stack Point : SP)

หรือรีจิสเตอร์ตัวชี้สแต็ก มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 81H ใช้ในการเก็บค่าตำแหน่งตัวชี้สแต็ก ซึ่งสามารถเปลี่ยนแปลงได้ เมื่อซีพียูมีการกระโดดไปทำงานที่โปรแกรมย่อย หรือกระโดดโปแกรมย่อยกลับมายังโปรแกรมหลัก เมื่อมีการรีเซตเกิดขึ้น (รีเซต:การกระทำที่ส่งผลให้ซีพียูต้องเริ่มต้นการทำงานใหม่ตั้งแต่ต้น) ค่าของรีจิสเตอร์ SP จะเท่ากับ 07H นั่นหมายความว่า ตัวชี้สแต็กมีค่า 07H แอดเดรสแรกของพื้นที่ที่สำรองไว้ทำหน้าที่เป็นสแต็กจะเท่ากับ 08H

รีจิสเตอร์ชี้ข้อมูลหรือดาต้าพอยน์เตอร์ (Data Pointer : DPTR)

มีขนาด 16 บิต โดยแบ่งเป็นรีจิสเตอร์ (DPH) และรีจิสเตอร์ (DPL) แต่ละตัวมีขนาด 8 บิต มีแอดเดรสอยู่ที่ 82H สำหรับ DPL และ 83H สำหรับ DPH รีจิสเตอร์ DPTR นี้ใช้ในการเก็บค่าแอดเดรสของหน่วยความจำหรืออุปกรณ์ภายนอกที่ไมโครคอนโทรลเลอร์ต้องการติดต่อด้วย

รีจิสเตอร์พอยน์ (Port register)

เป็นรีจิสเตอร์ขนาด 8 บิต ที่ใช้เก็บข้อมูลของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 มีด้วยกันทั้งสิ้น 4 ตัว รีจิสเตอร์พอร์ต 0 หรือ P0 มีแอดเดรสอยู่ที่ 80H, รีจิสเตอร์พอร์ต1 หรือ P1 มีแอดเดรสอยู่ที่ 90H, รีจิสเตอร์พอร์ต 2 หรือ P2 มีแอดเดรสอยู่ที่ A0H และรีจิสเตอร์พอร์ต 3 หรือ P3 มีแอดเดรสอยู่ที่ B0H รีจิสเตอร์ ทุกตัวสามารถเข้าถึงในระดับบิตได้ เมื่อต้องการอ่านข้อมูลหรือเขียนข้อมูลออกไปยังพอร์ตต่างๆ ของไมโครคอนโทรลเลอร์จะต้องผ่านกระทำรีจิสเตอร์นี้ทุกครั้ง

รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

รีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ทำการส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วนคือรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล(receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจาก ไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขาP3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51 นั้นเอง

รีจิสเตอร์ไทเมอร์ (Timer register)

เป็นรีจิสเตอร์ขนาด 16 บิต แต่จะจัดแบ่งเป็นไบต์สูงและไบต์ต่ำเช่นเดียวกับรีจิสเตอร์ DPTR รีจิสเตอร์ไทเมอร์ใช้ในการเก็บค่าของตัวนับหรือเคาน์เตอร์ (counter) ภายในไมโครคอนโทรลเลอร์เพื่อใช้ในการสร้างฐานเวลา,จับเวลาหรือนับจำนวนพัลส์สัญญาณนาฬิกาภายในบางทีเรียกรีจิสเตอร์ตัวนี้ว่ารีจิสเตอร์ไทเมอร์/เคาน์เตอร์

ในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชเบอร์AT89C51 จะมีรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ 2 ตัว แบ่งเป็น T0 หรือ Timer 0 และT1หรือ Timer 1 ในรีจิสเตอร์ยังแบ่งเป็นมีรีจิสเตอร์ไทเมอร์ไบต์ต่ำ(TL) และรีจิสเตอร์ไทเมอร์ไบต์สูง(TH) เหมือนกัน โดยมีรีจิสเตอร์ TL0 มีแอดเดรสอยู่ที่ 8AH รีจิสเตอร์ TH0 มีแอดเดรสอยู่ที่ 8CH ในขณะที่ TL1 และ TH1 มีแอดเดรสอยู่ที่ 8BH และ 8DH ตามลำดับ

สำหรับในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชเบอร์ AT89C52 และในอนุกรม AT89Sxx จะมีรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ถึง 3 ตัว โดยมีรีจิสเตอร์ TL2 และ TH2 ซึ่งมีแอดเดรสอยู่ที่ CCH และ CDH ตามลำดับเพิ่มเติมเข้ามา

รีจิสเตอร์แคปเจอร์(Capture register)

เป็นรีจิสเตอร์ขนาด 16 บิต มีเฉพาะในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชเบอร์AT89C52 และในอนุกรม AT89Sxx เท่านั้น เนื่องจากต้องใช้ร่วมกับไทเมอร์/เคาน์เตอร์ 2 (Time 2 ซึ่งมีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชเบอร์ AT89C52 และในอนุกรมAT89Sxx โดยรีจิสเตอร์แคปเจอร์นี้มีชื่อเรียกอย่างย่อว่า รีจิสเตอร์ RCAP2 ซึ่งแบ่งเป็นไบต์ต่ำคือ RCAP2L มีแอดเดรสอยู่ที่ CAH และที่ไบต์สูงคือ RCAP2H มีแอดเดรสอยู่ที่ CBH

รีจิสเตอร์แคปเจอร์จะถูกใช้งานเมื่อกำหนดให้ไทเมอร์ 2 ทำงานในโหมดแคปเจอร์ ซึ่งเป็นโหมดที่กำหนดไมโครคอนโทรลเลอร์ทำการตรวจจับการเปลี่ยนแปลงสถานะทางลอจิกที่ขา T2EX ทั้งนี้เพื่อใช้ประโยชน์ในการวัดคาบเวลา ความถี่ ตลอดจนการเปลี่ยนแปลงของสัญญาณพัลส์ที่ที่ขา T2EX นี้

รีจิสเตอร์ควบคุม(Control register)

รีจิสเตอร์ SFR ที่ใช้ในการควบคุมการทำงานในส่วนต่างๆ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชยังมีอีกหลายตัวประกอบด้วย

รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของวงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ TCON และ T2CON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดย T2CON ใช้สำหรับไทเมอร์/เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และในอนุกรม AT89Sxx

รีจิสเตอร์ TMOD และ **T2MOD** เป็นรีจิสเตอร์ที่ใช้กำหนดโหมดหรือลักษณะในการทำงานของไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดย T2MOD ใช้สำหรับไทเมอร์/เคาน์เตอร์2ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และในอนุกรม AT89Sxx

รีจิสเตอร์ IE และ IP เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการตอบสนองการอินเตอร์รัปต์ (Interrupt:การขัดจังหวะการทำงานปกติของซีพียู) โดย IE เป็นรีจิสเตอร์สำหรับอีนนาเบิลหรือใช้ในการกำหนดลักษณะของการตอบสนองการอินเตอร์รัปต์ ในขณะที่ IP เป็นรีจิสเตอร์สำหรับกำหนดลำดับความสำคัญก่อนหลัง ของการตอบสนองการอินเตอร์รัปต์ว่า ก่อนหรือหลัง

2.1.4 ไทเมอร์/เคาน์เตอร์ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ไทเมอร์/เคาน์เตอร์ (Time/Counter) เป็นอีกส่วนประกอบที่สำคัญของไมโครคอนโทรลเลอร์ เนื่องจากในการทำงานของไมโครคอนโทรลเลอร์จะต้องมีการเก็บและตรวจสอบค่าของเวลาและจำนวนของสัญญาณนาฬิกาอยู่ตลอดเวลาทั้งนี้เพื่อประโยชน์ในการสร้างฐานเวลา สร้างสัญญาณพัลส์ เปรียบเทียบค่าเวลา หรือเปรียบเทียบค่าของการนับ รวมไปถึงการกำหนดอัตราเร็วในการสื่อสารข้อมูลของพอร์ตอนุกรมด้วย

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 มีวงจรไทเมอร์/เคาน์เตอร์ขนาด 16 บิต 2 ตัว โดยค่าของวงจรไทเมอร์/เคาน์เตอร์นี้จะเก็บไว้ใน รีจิสเตอร์ขนาด 16 บิตที่ชื่อไทเมอร์ 0 (Timer 0) และ ไทเมอร์ 1 (Timer 1) เรียกสั้นๆว่า T0 และ T1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx จะมี ไทเมอร์/เคาน์เตอร์ถึง 3 ตัว คือ มีไทเมอร์ 2 (Timer2 : T2) เพิ่มเติมเข้ามา โดยรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ทั้งสามตัวสามารถกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือไทเมอร์และตัวนับหรือ/เคาน์เตอร์ได้อย่างอิสระต่อกัน การนับค่าของ ไทเมอร์/เคาน์เตอร์ในไมโครคอนโทรลเลอร์ MCS-51 จะเป็นการนับเพิ่มขึ้นเพียงทางเดียว

การทำงานเป็นไทเมอร์

เมื่อกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือไทเมอร์ ค่าของรีจิสเตอร์จะเพิ่มขึ้นในทุกๆ แมชีนไซเกิล ดังนั้นเมื่อทำงานเป็นไทเมอร์ รีจิสเตอร์จะทำการนับค่าของแมชีนไซเกิลนั่นเอง และเนื่องจากแมชีนไซเกิลประกอบด้วยคาบเวลาของวงจรกำเนิดสัญญาณนาฬิกา 12 คาบ ดังนั้นอัตราในการนับของรีจิสเตอร์จึงเท่ากับ1/12 ของความถี่สัญญาณนาฬิกา

การทำงานเป็นเคาน์เตอร์

เมื่อทำงานเป็นตัวนับหรือเคาน์เตอร์ค่าของรีจิสเตอร์จะเพิ่มขึ้นก็ต่อเมื่อมีการเปลี่ยนแปลงของระดับลอจิก “1” เป็น “0” เกิดขึ้นที่ขาพุตทางฮาร์ดแวร์ของวงจรไทเมอร์/เคาน์เตอร์ ซึ่งก็คือขา T0(P3.4) และขาT1(P3.5) สำหรับในไมโครคอนโทรลเลอร์ MCS-51แบบแฟลชเบอร์ AT89C51 รวมทั้งขา T2(P1.0) ไมโครคอนโทรลเลอร์ MCS-51แบบแฟลชเบอร์AT89C52 และอนุกรม AT89Sxx โดยจะมีการสุ่มรับสัญญาณจากขาอินพุตในทุกๆ คาบเวลาที่ 2 ของสเตตที่ 5 (S5P2) ในแต่ละแมชีนไซเกิล

เมื่อสัญญาณอินพุตเปลี่ยนแปลงจาก “1” เป็น “0” เป็นเวลาหนึ่งไซเกิลต่อมาค่าของการนับจะเพิ่มขึ้นหนึ่งค่า และจะไปปรากฏในรีจิสเตอร์ภายในคาบเวลาที่ 1 ของสเตตที่ 3(S3P1) ของแมชีนไซเกิลต่อไปหลังจากที่ตรวจจับพบการเปลี่ยนแปลงที่ขาไทเมอร์อินพุตแล้ว เมื่อเป็นเช่นนี้ในกระบวนการตรวจจับการเปลี่ยนแปลงของสัญญาณอินพุตที่ขาไทเมอร์ 2 แมชีนไซเกิลอัตราการนับของเคาน์เตอร์จึงเท่ากับ1/2.4 ของความถี่สัญญาณนาฬิกา ดังนั้น ความถี่สูงสุดของสัญญาณอินพุตที่

ไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถตรวจจับได้จึงเท่ากับความถี่ของสัญญาณนาฬิกาหารด้วย 24 ยกตัวอย่าง ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช เบอร์ AT89C51 สาสมารถใช้สัญญาณนาฬิกาได้สูงสุด 24MHz ดังนั้นความถี่สูงสุดของสัญญาณอินพุตที่ไทเมอร์/เคาน์เตอร์สามารถตรวจจับได้คือ 1MHz

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทเมอร์/เคาน์เตอร์0 แและ1

ในการทำงานของไทเมอร์/เคาน์เตอร์0 แและ1 ไมโครคอนโทรลเลอร์ MCS-51แบบแฟลช มีรีจิสเตอร์ที่เกี่ยวข้องเป็นพื้นฐานอยู่ 6 ตัว ดังมีรายละเอียดต่อไปนี้

รีจิสเตอร์ไทเมอร์

มีด้วยกัน 4 ตัวคือ TL0 มีแอดเดรสอยู่ที่ 8AH,TH0 มีแอดเดรสอยู่ที่ 8CH,TL1 มีแอดเดรสอยู่ที่ 8BH และ TH1 มีแอดเดรสอยู่ที่ 8DH รีจิสเตอร์ทั้ง 4ตัวจะอยู่ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR รีจิสเตอร์แต่ละตัวมีขนาด 8 บิต แต่ในการใช้งานโดยทั่วไปมักใช้รวมกันโดยจัดเป็นคู่คือ TL0 และ TH0 รวมกันเป็นรีจิสเตอร์ Timer 0 ขนาด 16 บิต และ TL1 กับ TH1 รวมกันเป็นรีจิสเตอร์ Timer 1 ขนาด 16 บิตโดย TL0และ TL1 จะเก็บข้อมูล 8 บิตล่าง ส่วน TH0 และ TH1เก็บข้อมูล 8บิตบนรีจิสเตอร์ไทเมอร์ทั้งคู่เมื่อนำมาใช้งานร่วมกันจะสามารถเก็บค่าของการนับได้สูงสุด 65,536 ค่าหรือ FFFFH เมื่อนับค่าถึงค่านี้แล้วก็จะวนไปเริ่มนับ 0000H ใหม่ และเมื่อเกิดการนับรอบใหม่ จะมีการเซต TF0 หรือ TF1 ในรีจิสเตอร์ TCON ที่ใช้ควบคุมการทำงานของไทเมอร์ เพื่อแจ้งล่วงหน้าให้ทาบว่าเกิดการนับเกินค่าสูงสุดแล้ว การเซตบิต TF0 หรือ TF1 ขึ้นอยู่กับว่าเลือกใช้งานรีจิสเตอร์ไทเมอร์ตัวใด

รีจิสเตอร์ควบคุมการทำงานของไทเมอร์ / เคาน์เตอร์หรือ

TCON (Timer/Counter Control Register)

เป็นรีจิสเตอร์ ขนาด 8 บิต มีแอดเดรสอยู่ที่ 88H ในพื้นที่ของรีจิสเตอร์ SER สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 (Timer 1 overflow flag) : เซตด้วยกระบวนการทางฮาร์ดแวร์เมื่อค่าของรีจิสเตอร์ Timer 1 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นเดียวกัน โดยบิตนี้จะเคลียร์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

TR1 (Timer 1 run control bit) : ใช้ในการเปิดปิดการทำงานของไทเมอร์ 1 (อีนาเบิลหรือดิสเอเบิล) ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทเมอร์ 1 ทำงานต้องเซตบิตนนี้ให้เป็น "1"

TF0 (Timer 0 overflow flag) : เซตด้วยกระบวนการทางฮาร์ดแวร์เมื่อค่าของรีจิส Timer 0 เกิดการนับเกิน หรือ โอเวอร์โฟลว การเคลียร์บิตนี้จะทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นกัน โดยบิตนี้จะเคลียร์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

TR0 (Timer 0 run control bit) : ใช้ในการเปิดปิดการทำงานไทเมอร์ 0 (อีนาเบิลหรือดิสเอเบิล) ทำการเซววววตและจะเคลียร์ด้วยกระบวนการทางซอฟแวร์ ถ้าต้องการให้ไทเมอร์ 0 ทำงานต้องเซตบิตนี้ให้เป็น "1"

IE1 (External Interrupt 1 edge flag) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์สามารถเซตได้ด้วยกระบวนการทางฮาร์ดร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 1 (INT1) ได้ และจะทำการเคลียร์เมื่อมีบริการอินเตอร์รัปต์เกิดขึ้น IT1 (Interrupt 1 type control bit) : บิตนี้จะใช้ในกระบวนการอินเตออร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปตี (INT1) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกขอบขาลงของสัญญาณ (falling ede)

"I" เลือกระดับลอจิกต่ำ (low level triggered)

IE0(External Interrupt 0 edge flag) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรววจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 0 (INT0) ได้และจะทำการเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น

IT0 (Interrupt 0 type control bit) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์ 0 (INTO) การเซตและเคลียทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกขอบขาลงของสัญญาณ (falling edge)

"1" เลือกระดับลอจิกต่ำ (low level triggered)

รีจิสเตอร์เลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 89H ในพื้นที่ของรีจิสเตอร์ SFR ไม่สามารถเข้าถึงได้ในระดับบิต แบ่งการทำงานออกเป็น 2 ส่วน คือ 4 บิตล่าง ใช้ในการเลือกโหมดการทำงานของไทเมอร์ 0 และ 4 บิตบนใช้ในการเลือกโหมดการทำงานของไทเมอร์ 1 ดังนั้นในการอธิบายการทำงานจะขออธิบายเพียงส่วนเดียวดังนี้

TMOD (Timer/Counter Mode Control Register)

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
ไทเมอร์ 1				ไทเมอร์ 0			

GATE : ใช้เลือกลักษณะการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์

"0" ไทเมอร์/เคาน์เตอร์ จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น "1" เรียกการควบคุมแบบนี้ว่า การควบคุมทางซอฟต์แวร์

"1" ไทเมอร์/เคาน์เตอร์ จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น "1" และสถานะลอจิกที่ขาอินพุตอินเตอร์รัปต์ INTI เป็น "1" เรียกการควบคุมแบบนี้ว่า การควบคุมทางฮาร์ดแวร์

C/T(Timer or Counter selector) : ใช้เลือกลักษณะการทำงานของไทเมอร์/เคาน์เตอร์

"0" เลือกให้ทำงานเป็นไทเมอร์ โดยใช้สัญญาณอินพุตจากสัญญาณนาฬิการภายในไมโครคอนโทรลเลอร์

"1" เลือกให้ทำงานเป็นเคาน์เตอร์ โดยรับสัญญาณอินพุตจากภายนอกที่เข้ามาทางขา T0 หรือ T1

M1,M0 (Mode selector bit) : ใช้เลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์

"00" เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์ 13 บิต

"01" เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์ 16 บิต

"10" เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์ขนาด 8 บิตแบบตั้งค่าอัตโนมัติ

"11" สำหรับไทเมอร์ 0เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์แยกส่วน โดยแยกออกเป็นไทเมอร์/เคาน์เตอร์ขนาด 8 บิต 2 ตัว รีจิสเตอร์ TL0 จะได้รับการควบคุมการเปิดปิดจากบิต TR0 ในรีจิสเตอร์ TCONและรีจิสเตอร์ TH0 ซึ่งเป็นไทเมอร์/เคาน์เตอร์ 8 บิตอีกตัวหนึ่ง จะได้รับการควบคุมจากบิต TR1ในรีจิสเตอร์ TCON

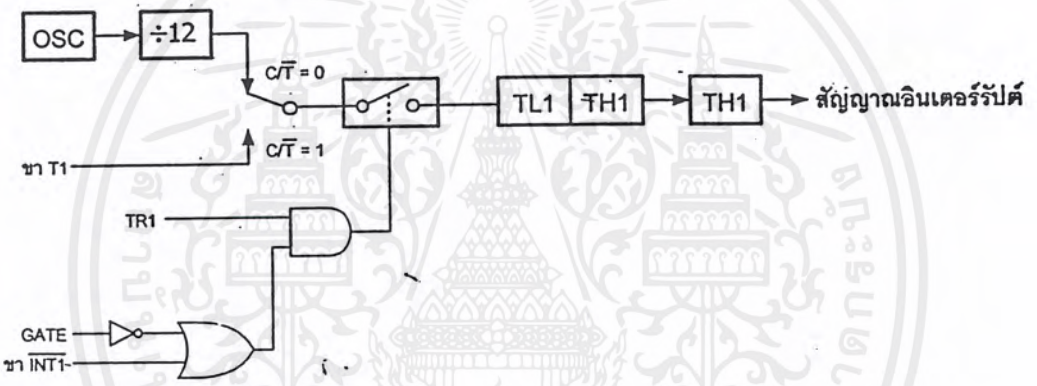
สำหรับไทเมอร์ 1 เป็นการสั่งให้ไทเมอร์/เคาน์เตอร์ 1 หยุดทำงาน (ดิสเอเบิล)

โหมดการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1

ไทเมอร์ 0 และไทเมอร์ 1 สามารถเลือกโหมดการทำงานได้ 4 โหมด คือ โหมด 0: ไทเมอร์/เคาน์เตอร์ 13 บิต,โหมด 1: ไทเมอร์/เคาน์เตอร์ 16 บิต, โหมด 2: ตั้งค่าอัตโนมัติขนาด 8 บิตและโหมด 3: ไทเมอร์/เคาน์เตอร์แยกส่วน หรืออาจเรียกว่าโหมดไทเมอร์/เคาน์เตอร์ 8 บิตก็ได้ ในขณะที่

ไทเมอร์ 2 มีโหมดการทำงาน 3 โหมด คือ โหมดแคปเจอร์หรือตรวจสัญญาณ(capture),โหมดตั้งค่าอัตโนมัติ(auto-reload) และโหมดกำเนิดอัตราเร็วในการสื่อสารข้อมูลอนุกรมหรืออัตราบอด(baud rate generator)

ต่อไปนี้จะเป็นการอธิบายถึงการทำงานในแต่ละโหมดของไทเมอร์ 0 และไทเมอร์ 1 ไปพร้อมกัน ส่วนไทเมอร์ 2 จะอธิบายแยกต่างหากในหัวข้อโหมดกาดทำงานของไทเมอร์ 2 การเลือกโหมดการทำงานของไทเมอร์ /เคาน์เตอร์ 0 และ1 สามารถกระทำได้ที่รีจิสเตอร์ TCON และ TMOD ร่วมกันโดยที่รีจิสเตอร์ TCONใช้ในการอีนาเบิลหรือดิสเอเบิลไทเมอร์/เตอร์ ส่วน TMOD ใช้ในการเลือกโหมดและลักษณะการทำงาน



รูปที่ 2-8 ไดอะแกรมการทำงานในโหมด 0 ของไทเมอร์/เคาน์เตอร์ 1

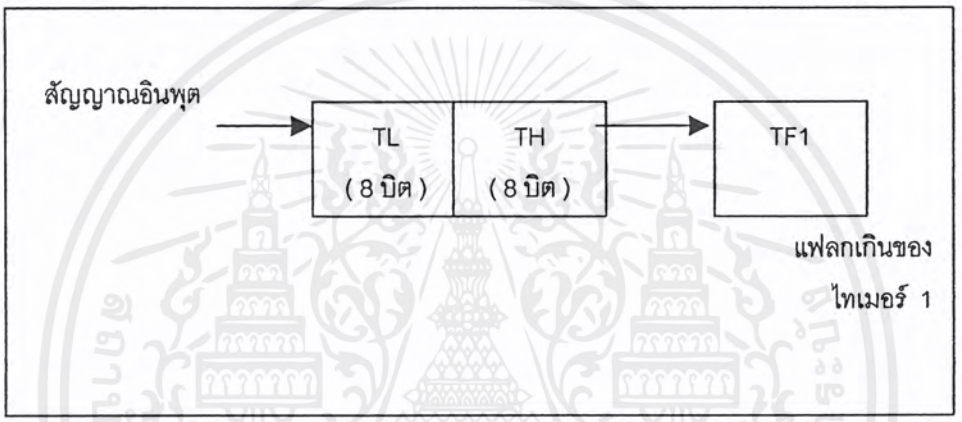
การทำงานในโหมด 0: ไทเมอร์ / เคาน์เตอร์ 13 บิต

มีไดอะแกรมการทำงานแสดงในรูปที่ 2-8 ในที่นี้จะใช้ไทเมอร์/เคาน์เตอร์ 1 ในการอธิบายการทำงานในโหมดนี้จะเป็นการกำหนดให้ใช้งานรีจิสเตอร์TL1 เพียง 5 บิต และTH1 ครบ 8 บิต โดยTL0จะทำหน้าที่คล้ายกับเป็นปรีสเกลเลอร์หาร 32 สัญญาณอินพุตสำหรับการนับจะเลือกจากสัญญาณนาฬิกาภายในหรือภายนอกผ่านทางขา T1 ขึ้นอยู่กับการควบคุมของบิต C/TGATE ในรีจิสเตอร์ TMOD, บิต TR1 ในรีจิสเตอร์ TCON และสถานะของลอจิกที่ขา INT1 เมื่อ TL1 นับครบ 32 คือ จาก 0-31 ก็จะส่งสัญญาณไปยัง TH1 เพื่อทำการเพิ่มค่า ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 13 บิต เมื่อทำการนับครบรอบก็จะทำการเซตบิต TF1 ในรีจิสเตอร์ TCON ส่วนการทำงานในโหมดนี้ของไทเมอร์/เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการ เพียงแต่เปลี่ยนรีจิสเตอร์และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์/เคาน์เตอร์ 0

การทำงานในโหมด 1 ไทเมอร์/เคาน์เตอร์ 16 บิต

มีไดอะแกรมการทำงานแสดงในรูปที่ 2-9 ในที่นี้จะใช้ไทเมอร์/เคาน์เตอร์ 1 ในการอธิบายการทำงานในโหมดนี้จะคล้ายกับโหมด 0 แต่จะใช้งานรีจิสเตอร์ TL1 ครบ 8 บิต , TH1 ครบ 8 บิต ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 16 บิต 0000H-FFFFH เมื่อทำการนับครบรอบค่าของการนับเปลี่ยนจาก FFFFH เป็น 0000H ก็จะทำการเซตบิต TF1 ในรีจิสเตอร์ TCON

ส่วนในการทำงานของโหมดนี้ของไทเมอร์ / เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการเพียงแต่เปลี่ยนรีจิสเตอร์ และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์ / เคาน์เตอร์ 0



รูปที่ 2-9 ไดอะแกรมการทำงานในโหมด 1 ของไทเมอร์ / เคาน์เตอร์ 1

ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 16 บิต 0000H-FFFFH เมื่อทำการนับครบรอบค่าของการนับเปลี่ยนจาก FFFFH เป็น 0000H ก็จะทำการเซตบิต TF1 ในรีจิสเตอร์ TCON

ส่วนในการทำงานของโหมดนี้ของไทเมอร์ / เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการเพียงแต่เปลี่ยนรีจิสเตอร์ และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์ / เคาน์เตอร์ 0

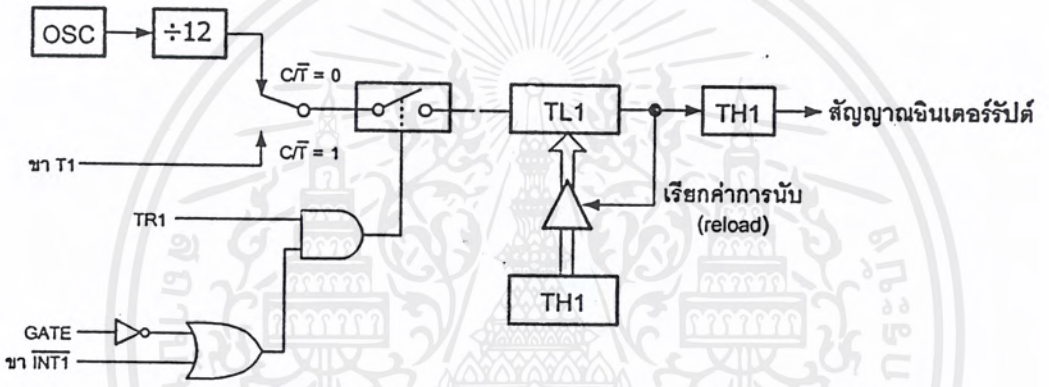
การทำงานในโหมด 2 : ไทเมอร์ / เคาน์เตอร์ 8 บิต แบบตั้งค่าอัตโนมัติ

มีไดอะแกรมการทำงานแสดงในรูปที่ 2-10 ในที่นี้จะใช้ไทเมอร์/เคาน์เตอร์ 1 ในการอธิบายการทำงานในโหมดนี้จะแยกรีจิสเตอร์ไทเมอร์ออกเป็น 2 ตัว ตัวละ 8 บิต โดยรีจิสเตอร์ TL1 ทำหน้าที่เป็นตัวนับค่า ส่วน TH1 ใช้ในการเก็บค่าเริ่มต้นของการนับ เมื่อเริ่มต้นการทำงานค่าของรีจิสเตอร์ TH1 จะถูกส่งไปยังรีจิสเตอร์ TL1 และ TH1 จะเหมือนกันเมื่อ TL1 ใหม่โดยอัตโนมัติ หรือเรียกกระบวนการนี้ว่า รีโหลด (reloard) แม้ว่าจะมีการส่งค่าเริ่มต้นไปยัง TL1 แล้วก็ตาม ค่าของข้อมูลใน รีจิสเตอร์ TH ก็ยังคงเดิม ไม่มีการเปลี่ยนแปลงจนกว่าจะมีการกำหนดค่าใหม่ด้วยกระบวนการทาง

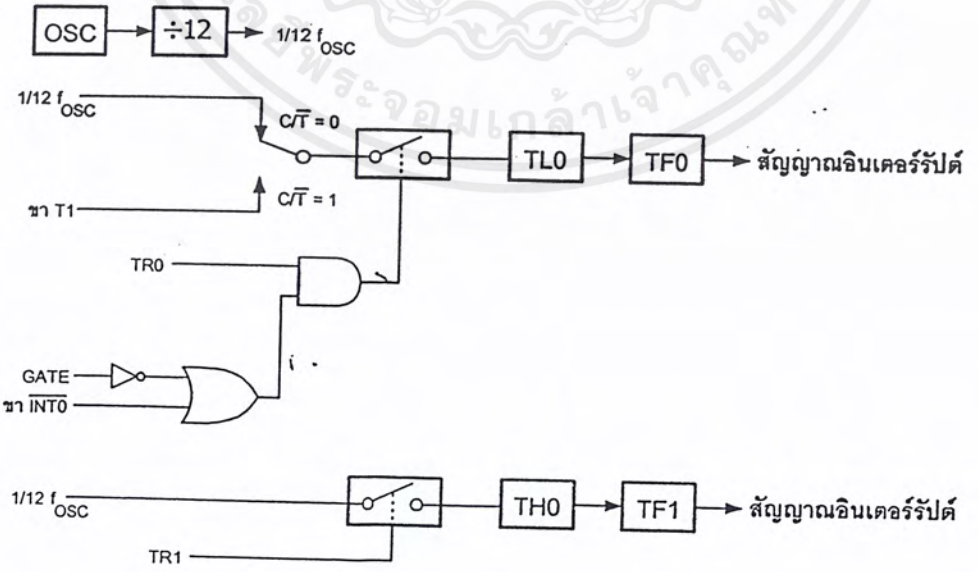
ส่วนในการทำงานของโหมดนี้ของไทเมอร์ / เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการเพียงแต่เปลี่ยนรีจิสเตอร์ และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์ / เคาน์เตอร์ 0

การทำงานในโหมด 3 : ไทเมอร์ / เคาน์เตอร์แยกส่วนหรือไทเมอร์ / เคาน์เตอร์ 8 บิต

ในโหมดนี้เป็นโหมดเดียวที่การทำงานของไทเมอร์ 0 และไทเมอร์ 1 ไม่เหมือนกัน ขออธิบายในส่วนของไทเมอร์ 1 ก่อน เมื่อเข้าสู่โหมดนี้ จะเป็นการสั่งให้ไทเมอร์ / เคาน์เตอร์หยุดนับ ค่าของการนับก่อนหน้านี้จะถูกเก็บไว้ในรีจิสเตอร์ไทเมอร์ 1 มีลักษณะการทำงานเหมือนกับการดิสเอเบิลไทเมอร์/เคาน์เตอร์ 1 ด้วยการเคลียร์บิต TR1 ในรีจิสเตอร์ TCON



รูปที่ 2-10 ไดอะแกรมการทำงานในโหมด 2 ของไทเมอร์ / เคาน์เตอร์ 1



รูปที่ 2-11 ไดอะแกรมการทำงานในโหมด 3 ของไทเมอร์ / เคาน์เตอร์ 1

ส่วนการทำงานของไทเมอร์ 0 ในโหมดนี้มีไดอะแกรมการทำงานแสดงในรูปที่ 2-11 การทำงานในโหมดนี้จะแยกรีจิสเตอร์ไทเมอร์ 0 ออกเป็น 2 ตัว ตัวละ 8 บิต คือ รีจิสเตอร์ TL0 และ TH1 โดยแยกการทำงานออกจากกัน รีจิสเตอร์ TL0 สามารถเลือกการทำงานได้เหมือนกับไทเมอร์/คาน์เตอร์ ตามปกติ ส่วนรีจิสเตอร์ TH0 สามารถทำงานในโหมดไทเมอร์ได้เพียงอย่างเดียว กล่าวคือ สามารถรับสัญญาณอินพุตจากสัญญาณนาฬิกาภายในเพียงทางเดียวเท่านั้น แต่การแจ้งการนับเกินยังคงเหมือนเดิมหากแต่ TL0 แจ้งผ่านบิต TF0 ในขณะที่ TF1 จะแจ้งผ่านทางบิต TF1

ไทเมอร์/เคาน์เตอร์ 2 ในไมโครคอนโทรลเลอร์ MCS-51

ในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชเบอร์ AT89C52 และในอนุกรม AT89Sxx ยังมีไทเมอร์/เคาน์เตอรขนาด 16 บิต ให้ใช้งานเพิ่มเติมอีก 1 ตัวคือ ไทเมอร์/เคาน์เตอร์ 2 รีจิสเตอร์ฟังก์ชั่นพิเศษที่เกียวข้องกับไทเมอร์/เคาน์เตอร์ 0 และ 1 ก็คือ รีจิสเตอร์ CTON สำหรับโหมดการทำงานของไทเมอร์/เคาน์เตอร์ 2 มีด้วยกัน 3 โหมดคือ โหมดแคปเจอร์หรือตรวจจับสัญญาณ (capture), โหมดตั้งค่าอัตโนมัติ (autoreload) และโหมดกำเนิดอัตราเร็วในการสื่อสารข้อมูลอนุกรมหรืออัตราบอด (baud rate generator)

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทเมอร์/เคาน์เตอร์

ในการทำงานของไทเมอร์/เคาน์เตอร์ 2 ในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลช มีรีจิสเตอร์ ที่เกี่ยวข้องเป็นพื้นฐานอยู่ 5 ตัว ดังมีรายละเอียดดังต่อไปนี้

รีจิสเตอร์ไทเมอร์

มีอยู่ด้วยกัน 2 ตัว คือ TL2 มีแอดเดรสอยู่ที่ CCH และ TH2 มีแอดเดรสอยู่ CDH รีจิสเตอร์ทั้ง 2 ตัวจะอยู่ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR รีจิสเตอร์แต่ละตัวมีขนาด 8 บิต แต่ในการใช้งานโดยทั่วไปมักใช้ร่วมกันเป็นรีจิสเตอร์ไทเมอร์ 2 ขนาด 16 บิต โดยใน TI2 จะเก็บข้อมูล 8 บิตล่างส่วน TH2 เก็บข้อมูล 8 บิตบนรีจิสเตอร์ไทเมอร์ทั้งคู่เมื่อนำมาใช้งานร่วมกนจะสามารถเก็บค่าของการนับได้สูงสุด 65,536 ค่าหรือ FFFFH เมื่อนับถึงค่านี้แล้วก็จะวนไปเริ่มนับ 0000H ใหม่ และเมื่อเกิดการนับใหม่จะมีการเซตบิต TF2 ในรีจิสเตอร์ T2CON ที่ใช้ควบคุมการทำงานของไทเมอร์ เพื่อแจ้งให้ทราบว่าเกิดการนับค่าสูงสุด

รีจิสเตอร์ควบคุมการทำงานของไทเมอร์ / เคาน์เตอร์ 2 หรือ T2CON

(Timer/ Counter 2 Control Register)

เป็นรีจิสเตอร์ ขนาด 8 บิต มีแอดเดรสอยู่ที่ C8H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
TF1	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

TF2 (Timer 2 overflow flag) : จะเซตเมื่อค่าของรีจิสเตอร์ Timer 2 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางซอฟต์แวร์เท่านั้น และจะไม่เกิดการเซตถ้าบิต RCLK หรือบิต TCLK เป็น "1"

EXF2 (Timer 2 external flag) : จะเกิดการเซตเมื่อเกิดการแคปแจร์หรือ รีโหลดขึ้น ซึ่งเกิดจากการเปลี่ยนแปลงระดับลอจิกจาก "1" เป็น "0" ที่ขาอินพุต T2EX(ขา P1.1) และบิต EXEN2 เป็น "1" ในกรณีที่อินเตอร์รัปต์ในไทเมอร์ 2 ได้รับการอีนาเบิล และบิต EXF2 นี้เกิดการเซต จะมีผลทำให้ซีพียูไปอ่านค่าเวกเตอร์ของการบริการอินเตอร์รัปต์ บิตนี้สามารถเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์เท่านั้น

RCLK (Receive clock flag) : ถ้าบิตนี้เกิดการเซต ทำให้วงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ใช้พัลส์โอเวอร์โฟลวของไทเมอร์ 2 เป็นสัญญาณนาฬิกา ในการรับข้อมูลในโหมด 1 และ 3 ถ้าหากบิตนี้เป็น"0" จะใช้พัลส์จากไทเมอร์ 1

TCLK (Transmit clock flag) : ถ้าบิตนี้เกิดการเซต ทำให้วงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ใช้พัลส์โอเวอร์โฟลวของไทเมอร์ 2 เป็นสัญญาณนาฬิกาในการรับข้อมูลในโหมด 1 และ 3 ถ้าหากบิตนี้เป็น "0" จะใช้พัลส์จากไทเมอร์ 1

EXEN2 (Timer 2 external enable flag) : เมื่อบิตนี้เซตเป็น "1" จะเป็นการอีนาเบิลการแคปแจร์หรือรีโหลดที่ขา T2EX ให้เกิดขึ้นได้ภายใต้เงื่อนไขว่า ไทเมอร์ 2 ต้องไม่ถูกนำไปใช้ในการสื่อสารพอร์ตอนุกรม หรือบิต RCLK หรือ TCLK เป็น "1" ถ้าบิตนี้เป็น "0" ไทเมอร์ 2 จะไม่สนใจเหตุการณ์ที่ขา T2EX

TR2 : ใช้ในการควบคุมเริ่มต้นและหยุดการทำงานของไทเมอร์ 2 เป็นการควบคุมทางซอฟแวร์

"0" หยุดการทำงานของไทเมอร์ 2 (STOP)

"1" เริ่มต้นการทำงานของไทเมอร์ 2 START)

C/T (Timer or Counter selector) : ใช้ลักษณะการทำงานของไทเมอร์/เคาน์เตอร์ 2

"0" เลือกให้ทำงานเป็นไทเมอร์ โดยใช้สัญญาณอินพุตจากสัญญาณนาฬิการภายในไมโครคอนโทรลเลอร์

"1" เลือกให้ทำงานเป็นเคาน์เตอร์โดยรับสัญญาณอินพุตจากภายนอกที่เข้ามาทางขา T2(P1.0)

CP/RL2 (Capture/Reload flag) :

"0" มีการรีโหลดค่าของการนับแบบอัตโนมัติเกิดขึ้นเมื่อไทเมอร์ 2 เกิดโอเวอร์โฟลวหรือเกิดการเปลี่ยนระดับลอจิกแบบลบ (เปลี่ยนจาก'1" เป็น "0") ที่ขา T2EX ในกรณีที่บิต EXEN2 เป็น "1"แต่ถ้าหากบิต RCLK เป็น "1" จะยอมให้การรีดโหลดแบบอัตโนมัติเกิดขึ้นเมื่อไทเมอร์ 2 เกิดโอเวอร์โฟลวเท่านั้น

"1" แสดงว่ามีการแคปแจร์เกิดขึ้น เมื่อมีการเปลี่ยนแปลงระดับลอจิกแบบบล (เปลี่ยนจาก "1" เป็น "0") ที่ขา T2EX ในกรณีที่บิต บิต EXEN2 เป็น "1"

รีจิสเตอร์เลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์ 2 หรือ

T2MOD(Timer/Counter 2 Mode Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ C9H ในพื้นที่ของรีจิสเตอร์ SFR ไม่สามารถเข้าถึงได้ในระดับบิต มีบิตสำหรับกำหนดการทำงานเพียง 2 บิต คือ

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	-	-	-	-	-	T2OE	DCEN

T2OE (Timer 2 Output Enable bit) : ใช้อีนาเบิลเอาต์ไทเมอร์ 2

"0" ดิสเอเบิล

"1" อีนาเบิล

Dcen (timer 2 Counter Enable bit) : ใช้กำหนดให้ไทเมอร์ 2 ทำงานเป็นตัวนับหรือเคาน์เตอร์แบบขึ้นและลง

"0" ดิสเอบิล

"1" อีนาเบิลให้ทำงานเป็นตัวนับหรือเคาน์เตอร์แบบขึ้นลง

โหมดการทำงานของไทเมอร์ / เคาน์เตอร์ 2

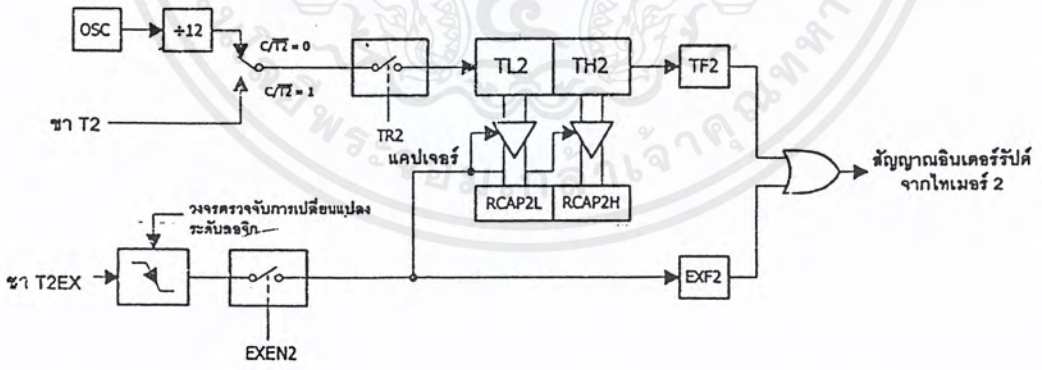
ไทเมอร์ / เคาน์เตอร์ 2 สามารถเลือกโหมดการทำงานได้ 3โหมด โดยการกำหนดที่บิต RCLK+TCLK,CP/RL2 และ TR2 ในรีจิสเตอร์ T2CON ดังแสดงรายละเอียดในรูป2-12 โหมดการทำงานของไทเมอร์/เคาน์เตอร์ 2 ได้แก่ โหมดแคปแจร์หรือตรวจจับสัญญาณ(capture), โหมดตั้งค่าการนับอัตโนมัติ(auto-reload) และโหมดกำเนิดอัตราเร็วในการสื่อสารข้อมูลอนุกรมหรืออัตราบอด (baud rate generator) ต่อไปจะเป็นการอธิบายการทำงานแต่ละโหมดของไทเมอร์/เคาน์เตอร์ 2

RCLK + TCLK	CP/RL2	TR2	โหมด
0	0	1	16 บิตแบบตั้งค่าการนับอัตโนมัติหรือรีโหลด
0	1	1	แคปเจอร์ตรวจจับสัญญาณ
1	x	1	กำเนิดอัตราบอด
X	x	0	ดิสเอเบิล

รูปที่ 2-12 การเลือกโหมดทำงานของไทเมอร์/เคาน์เตอร์ 2

การทำงานในโหมดแคปแจร์หรือตรวจจับสัญญาณ (capture)

การทำงานในโหมดนี้จะเกิดขึ้นได้อย่างสมบูรณ์เมื่อกำหนดให้บิต EXEN2เป็น"1" หลังจากที่เลือกให้ไทเมอร์ 2 ทำงานในโหมดแคปเจอร์แล้ว เมื่อเลือกให้ทำงานในโหมดนี้ขาอินพุสตที่ทำการตรวจจับคือขาอินพุตT2EX ซึ่งตรงกับขาP1.1 ที่ขาT2EX จะมีวงจรตรวจจับการเปลี่ยนแปลงของระดับลอจิกจาก "1"เป็น "0" รีจิสเตอร์TL2 และTH2 จะทำการนับค่าเพิ่มขึ้นเรื่อย ๆ จนกว่าที่ขา T2EX จะตรวจจับการเปลี่ยนแปลงได้ เมื่อตรวจจับ ค่าของรีจิสเตอร์ TL2 และTH2 จะถูกส่งต่อไปยังรีจิสเตอร์ RCAP2Lและ RCAP2H พร้อมกันนั้นยังทำการเซตบิตEXF2 ให้เป็น "1" เพื่อสร้างสัญญาณอินเตอร์รัปต์ให้เกิดขึ้น ในรูปที่ 2-13 แสดงการทำงานของไทเมอร์ 2 โหมดแคปเจอร์นี้



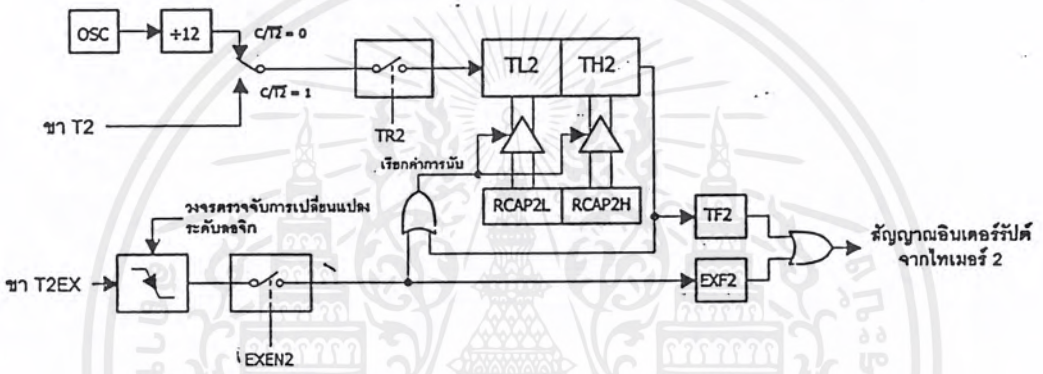
รูปที่ 2-13 ไดอะแกรมการทำงานในโหมดแคปเจอร์ของไทเมอร์/เคาน์เตอร์ 2

การทำงานในโหมดตั้งค่าการนับอัตโนมัติ

การทำงานในโหมดนี้ การกำหนดสถานะที่ EXEN2 จะทำให้เกิดการทำงานใน 2 ลักษณะ คือ ถ้าบิต EXEN2 เป็น "0" เมื่อไทเมอร์ 2 ทำการนับค่าเกินรอบ แล้ว จะเกิดการเซตบิต TF2 และจะ

ทำการเรียกค่าการนับใหม่จากรีจิสเตอร์ RCAP2L และ RCAP2H โดยอัตโนมัติ การกำหนดค่าของรีจิสเตอร์ RCAP2L และ RCAP2H สามารถกระทำได้ด้วยกระบวนการทางซอฟต์แวร์

ในกรณีที่บิต EXEN2 เป็น"1" หลังจากที่เลือกให้ไทเมอร์ 2 จะทำการนับเหมือนเดิม แต่เงื่อนไขของการตั้งค่าการนับใหม่โดยอัตโนมัติจากรีจิสเตอร์ RCAP2L และ RCAP2H จะใช้การเปลี่ยนแปลงของระดับลอจิกจาก"1" เป็น"0" ที่ขาT2EX เป็นตัวกำหนด และทำการเซตบิต EXF2 เพื่อสร้างสัญญาณอินเตอร์รัปต์แทนบิต TF2 ในรูปที่ 2-14 แสดงไดอะแกรมการทำงานของไทเมอร์ 2 ในโหมดตั้งค่ารับอัตโนมัติ



รูปที่ 2-14 ไดอะแกรมการทำงานในโหมดตั้งค่าการนับอัตโนมัติของไทเมอร์/เคาน์เตอร์ 2

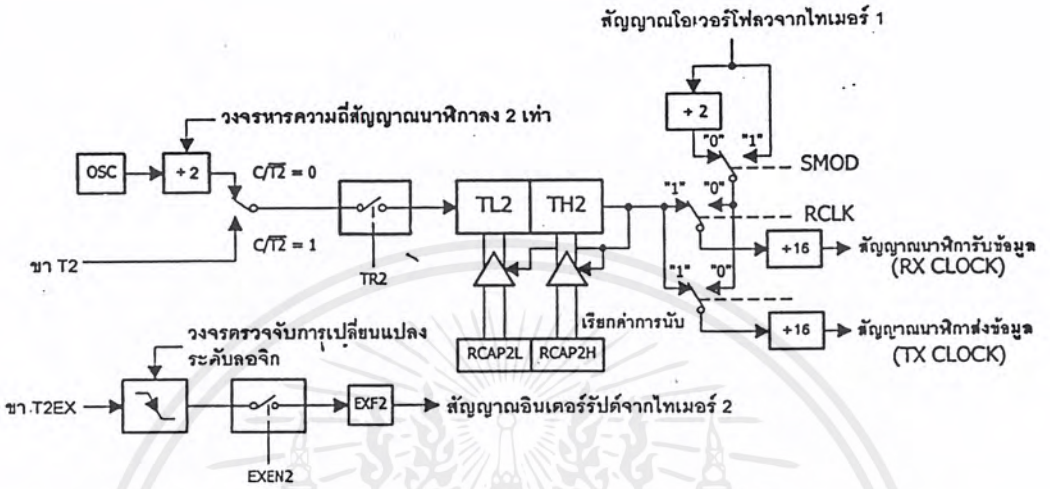
การทำงานในโหมดกำเนิดอัตราเร็วในการสื่อสารข้อมูลอนุกรมกหรืออัตราบอด

การทำงานในโหมดนี้จะเกิดขึ้นเมื่อทำการเซตบิต RCLK และ/หรือ TCLK ในรีจิสเตอร์ T2CON ให้เป็น "1" เมื่อเข้าสู่การทำงานในโหมดนี้ บิต TF2 จะไม่เกิดการเซต และไม่มีการสร้างสัญญาณอินเตอร์รัปต์ แต่การอินเตอร์รัปต์ในไทเมอร์ 2 ก็ไม่ได้ถูกดิสเอเบิล

เมื่ออยู่ในโหมดนี้ บิต EXEN2 ถูกเซต การเปลี่ยนแปลงระดับลอจิกจาก "1" เป็น "0" ที่ขา T2EX จะส่งผลให้บิตEXF2 เกิดการเซต แต่จะไม่มีการเรียกค่าจากรีจิสเตอร์ RCAP2L และ RCAP2H มาแต่อย่างใด เมื่อเป็นเช่นนี้ขา T2EX จึงสามารถใช้เป็นขาอินพุตสำหรับการอินเตอร์รัปต์ได้เป็นกรณีพิเศษ

เมื่อไทเมอร์ 2 ทำงาน (บิตTR2 เป็น"1") จะถูกกำหนดให้ทำงานเป็นไทเมอร์ จึงไม่สามารถอ่านหรือเขียนค่ากับรีจิสเตอร์ TL2 และ TH2 ได้ ค่าของรีจิสเตอร์ไทเมอร์จะเพิ่มขึ้นทุก ๆ สเตต ค่าการนับของรีจิสเตอร์จึงถูกนำมาใช้สร้างสัญญาณนาฬิกาของการรับและส่งข้อมูลในการสื่อสารข้อมูลผ่านพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

ในรูปที่ 2-15 แสดงการทำงานของไทเมอร์ 2 ในการกำเนิดอัตราบอด



รูปที่ 2-15 ไดอะแกรมการทำงานในโหมดกำเนิดอัตราบอดในการสื่อสารข้อมูลผ่านพอร์ตอนุกรมของไทเมอร์/เคาน์เตอร์ 2

อย่างไรก็ตามรายละเอียดเพิ่มเติมของการใช้ไทเมอร์ 2 ในการกำเนิดอัตราบอดจะอธิบายไว้ในบทที่ว่าด้วยการสื่อสารผ่านพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

วอตช์ด็อกไทเมอร์ (Watchgog Timer : WDT)

นอกจากไทเมอร์/เคาน์เตอร์ทั้งสามตัวที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 แล้ว สำหรับในอนุกรม AT89Sxx ไม่ว่าจะเป็นเบอร์ AT89S8252,AT89LS8252 และเบอร์ AT89S53 ยังมีวงจรไทเมอร์อีกแบบหนึ่งเพิ่มเติม เพื่อให้สามารถนำไปใช้สร้างสัญญาณรีเซตในกรณีที่ซีพียูภายในไมโครคอนโทรลเลอร์เกิดภาวะหยุดทำงานกระทันหันหรือแฮงก์ (hang) นั่นคือ วอตช์ด็อกไทเมอร์ (Watchdog Timer : WDT) โดยวอตช์ด็อกไทเมอร์แบบโปรแกรมได้ โดยการกำหนดค่าของบิต ในรีจิสเตอร์ WMCONสำหรับเบอร์ AT89S8252และรีจิสเตอร์ WCONสำหรับเบอร์AT89S53

วอตช์ด็อกไทเมอร์จะได้รับการดิสเอเบิลโดยอัตโนมัติเมื่อเริ่มต้นจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์ หรือเมื่อเกิดเพาเวอร์ออนรีเซต และเมื่อไมโครคอนโทรลเลอร์ เข้าสู่การทำงานในโหมดประหยัดพลังงานแบบลดพลังงานหรือเพาเวอร์ดาวส์ สำหรับการอีนาเบิลสามารถกระทำได้โดยกระบวนการทางซอฟต์แวร์ โดยการเซตบิต WDTEN ในรีจิสเตอร์ WMCON สำหรับเบอร์ AT89S8252 และรีจิสเตอร์ WCON สำหรับเบอร์ AT89S53

การทำงานของวอตช์ด็อกไทเมอร์ จะเริ่มต้นขึ้นเมื่อมีการกำหนดคาบเวลาในการทำงานของวอตช์ด็อกไทเมอร์ และมีการอีนาเบิลให้วอตช์ด็อกไทเมอร์เริ่มทำงาน วงจรนับภายในวอตช์ดีอกไทเมอร์จะนับค่าเวลาไปตามปกติ หากซีพียูทำงานอยู่เป็นปกติจะมีการส่งสัญญาณมารีเซตหรือเคลียร์ค่าในวอตช์ด็อกไทเมอร์อยู่เป็นระยะเพื่อไม่ให้วอตช์ด็อกไทเมอร์ทำงานจนถึงค่าเวลาที่กำหนดหรือเกิดการไทม์เอาต์ (time out) ขึ้น แต่ถ้าหากซีพียูเกิดภาวะหยุดทำงานขึ้นกระทันหันไม่ว่าจะด้วยสาเหตุใด ยกเว้นการรีเซตและการเข้าสู่โหมดประหยัดพลังงานแบบลดพลังงานหรือเพาว์เวอร์ดาวน์ ก็จะไม่มีสัญญาณมารีเซตหรือเคลียร์ค่าในวอตช์ด็อกไทเมอร์ ทำให้ในที่สุดวอตช์ด็อกไทเมอร์ก็จะทำงานถึงค่าเวลาที่กำหนดเกิดการไทม์เอาต์วอตช์ด็อกไทเมอร์จะทำการสร้างสัญญาณรีเซตส่งไปยังซีพียู เพื่อทำการรีเซตซีพียู ส่งผลให้ซีพียูสามารถกลับมาเริ่มต้นทำงานใหม่ได้อีกครั้ง โดยที่ไม่จำเป็นต้องทำการรีเซตซีพียูจากภายนอกหรือจ่ายไฟเลี้ยงใหม่ให้แก่ไมโครคอนโทรลเลอร์

ในไมโครคอนโทรลเลอร์MCS-51 เบอร์พื้นฐานเช่น 8031, 8032, 8051 หรือแบบเฟลชเบอร์ AT89C51, AT89C52, AT89C55 จะไม่มีวอตช์ด็อกไทเมอร์อยู่ภายใน

รีจิสเตอร์ควบคุมการทำงานของวอตช์ด็อกไทเมอร์ (Watchdog and Memory Control register : WMCON และ Watchdog Control register : WCON)

ในการทำงานของวอตช์ด็อกไทเมอร์ในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชอนุกรม AT89Sxx มีรีจิสเตอร์ ที่ใช้ในการควบคุมการทำงาน 1 ตัวคือ WMCON สำหรับเบอร์ AT89S8252 และ รีจิสเตอร์ WCON สำหรับเบอร์ AT89S53 โดยรีจิสเตอร์ในไมโครคอนโทรลเลอร์แต่ละเบอร์จะมีหน้าที่ในการทำงานเหมือนกันทุกประการ ส่วนแตกต่างมีเพียงจุดเดียวคือ ในเบอร์ AT89S8252 มีหน่วยความจำข้อมูลแบบอีอีพรอมอยู่ จึงต้องมีการควบคุมการทำงานเพิ่มเติมเข้ามา จึงใช้บิต 3 และ 4 ในรีจิสเตอร์ WMCON มาจัดการหน่วยความจำข้อมูลส่วนนี้สำหรับบิตอื่น ๆ ที่เกี่ยวข้องกับการทำงานของวอตช์ด็อกไทเมอร์ในรีจิสเตอร์ WMCON และ WCON จะเหมือนกันทุกประการ

ดังนั้นในการอธิบายรายละเอียดของรีจิสเตอร์ ควบคุมการทำงานของวอตช์ด็อกไทเมอร์จะอธิบายในคราวเดียวกัน โดยจะเพิ่มรายละเอียดของบิตที่ใช้ในการควบคุมหน่วยความจำข้อมูลแบบอีอีพรอมเข้าไปด้วย ดังมีรายละเอียดดังต่อไปนี้

รีจิสเตอร์ควบคุมการทำงานของวอตช์ด็อกไทเมอร์ทั้ง WMCON และ WCON เป็นรีจิสเตอร์ ขนาด 8 บิต มีแอดเดรสอยู่ที่ 96H ในพื้นที่ของรีจิสเตอร์สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
PS2	PS1	PS0	EEMWE	EEMEN	OPS	WDTRST	WDTEN

PS2	PS1	PS0	-	-	OPS	WDTRST	WDTEN

PS2-PS0 (Prescaler bits) : ใช้เลือกค่าเวลาให้แก่วอตช์ด็อกไทเมอร์ ดังมีการกำหนดค่าเวลาดังนี้

บิตปรีสเกลเลอร์			ค่าเวลาของ
PS2	PS1	PS0	วอตช์ด็อกไทเมอร์
0	0	0	16ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1,024 ms
1	1	1	2,048 ms

EEMWE (EEPROM Data Memory Write Enable bit) : บิตนี้เป็นบิต 4 ของรีจิสเตอร์ WMCON ใช้ในการอีนาเบิลการเขียนข้อมูลลงในหน่วยความจำข้อมูลอีอีพรอมภายในไมโครคอนโทรลเลอร์เบอร์ AT89C8252 เมื่อต้องการเขียนข้อมูลต้องการทำเซตบิตนี้ก่อน หลังจากการเขียนข้อมูลเสร็จสิ้นลงต้องการทำการเคลียร์บิตนี้เสมอ สำหรับในรีจิสเตอร์ WCON ในไมโครคอนโทรลเลอร์ AT89C53 บิตนี้จะไม่มีการใช้งานต้องกำหนดให้เป็น "0"

EEMEN (Internal EEPROM Access Enable bit) : บิตนี้เป็นบิต 3 ของรีจิสเตอร์ WMCON ใช้ในการอีนาเบิลการเขียนข้อมูลลงในหน่วยความจำข้อมูลอีอีพรอมภายในไมโครคอนโทรลเลอร์เบอร์ AT89C8252 แทนที่จะใช้หน่วยความจำข้อมูลภายนอกไมโครคอนโทรลเลอร์ เมื่อต้องการเข้าถึงในหน่วยความจำส่วนนี้ โดยใช้คำสั่ง MOVX และรีจิสเตอร์ DPTR ต้องการทำเซตบิตนี้ก่อน ถ้าหากบิตนี้เป็น "0" จะเป็นการกำหนดให้ไมโครคอนโทรลเลอร์ MOVX และรีจิสเตอร์ DPTR ติดต่อกับหน่วยความจำข้อมูลภายนอกไมโครคอนโทรลเลอร์ สำหรับในรีจิสเตอร์ WCON ในไมโครคอนโทรลเลอร์ AT89C53 บิตนี้จะไม่มีการใช้งานต้องกำหนดให้เป็น "0"

DPS (Data Pointer Register Seleect) : เนื่องจากไมโครคอนโทรลเลอร์ อนุกรม AT89Sxx จะมีรี..สำหรับชี้ข้อมูลหรือ DPTR2 ตัวคือ DPTR0 และ DPTR1 บิตนี้จึงมีหน้าที่ใช้ในการเลือกรีจิสเตอร์ DPTR ถ้าเป็น "0" จะเป็นการเลือกรีจิสเตอร์ DPTR0 (ซึ่งก็คือ DP0L และ DP0H) ในกรณีที่บิตนี้เป็น "1" จะเป็นการเลือกรีจิสเตอร์ DPTR1 (ซึ่งก็คือ DP1L และ DP1H)

WDTRST/RDY/BSY (Watchdog Timer Reset and EEPROM Ready / BusyFlag) :

บิตนี้มีความเกี่ยวข้องกับการทำงาน 2 ส่วนคือ การรีเซตของวอตช์ด็อกไทเมอร์ และใช้เป็นบิตแสดงสถานะของหน่วยความจำข้อมูลอีอีพรอมภายในไมโครคอนโทรลเลอร์ AT89C8252

ในกรณีของวอตช์ด็อกไทเมอร์ บิตนี้จะเป็นโดยผู้ใช้งานด้วยกระบวนการทางซอฟต์แวร์ เป็นการสั่งให้วอตช์ด็อกไทเมอร์สร้างสัญญาณรีเซตส่งออกไปในกรณีที่วอตช์ด็อกไทเมอร์เกิดไทมีเอาต์ หลังจากที่ส่วสัญญาณรีเซตออกไปแล้ว บิตนี้จะเคลียร์ตัวเองเป็น "0" โดยอัตโนมัติในไซเกิลของการทำงานถัดไป ดังนั้นบิตนี้จึงสามารถเขียนได้เพียงอย่างเดียว

ในหน่วยความจำข้อมูลอีอีพรอม จะใช้บิตนี้ในการป้องกันการเขียนข้อมูลลงในหน่วยความจำ ถ้าบิตนี้เป็น "1" หน่วยความจำข้อมูลอีอีพรอมจะสามารถอ่านได้เพียงอย่างเดียว เมื่อมีความต้องการเขียนข้อมูล หลังจากอีนาเบิลการเขียนที่บิต EEMWE แล้ว ให้ทำการเคลียร์บิตนี้เป็นเพียง "0" หลังจากที่การเขียนข้อมูลเสร็จสิ้นลง บิตนี้จะเซต "1" โดยอัตโนมัติ เป็นการแจ้งให้ทราบว่าการเขียนข้อมูลเสร็จสิ้นลงแล้ว

สำหรับในไมโครคอนโทรลเลอร์ AT89C53 ใช้ในการสร้างสัญญาณรีเซตของวอตช์ด็อกไทเมอร์เท่านั้น

WDTEN (Wathdog Timer Enable bit) : ใช้ในอีนาเบิลการทำงานของวอตช์ด็อกไทเมอร์

"0" ดิสเอเบิลวอตช์ด็อกไทเมอร์

"1" อีนาเบิลวอตช์ด็อกไทเมอร์

จากรายละเอียดของไทเมอร์/เคาน์เตอร์ทั้งหมดของไม่ไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชที่ได้กล่าวมานั้น จะเห็นได้ว่า ในการสร้างระบบควบคุมที่ดีไทเมอร์/เคาน์เตอร์ เป็นสิ่งที่สำคัญและจำเป็นอย่างยิ่งที่จะต้องมีไมโครคอนโทรลเลอร์ ไม่ว่าจะเป็นการสร้างอัตราบอดสำหรับการสื่อสารข้อมูลผ่านพอร์ตอนุกรมเพื่อเชื่อมต่อกับคอมพิวเตอร์ หรือเพื่อสร้างฐานเวลาและยังสามารถใช้ไทเมอร์/เคาน์เตอร์ในการวัดคาบเวลา และความถี่ของสัญญาณได้อีกด้วย นอกเหนือจากนั้นในไมโครคอนโทรลเลอร์อนุกรม 89Sxx ที่มีวอตช์ด็อกไทเมอร์ในตัวนั้นยังช่วยให้ระบบควบคุมมีเสถียรภาพเพิ่มมากขึ้น ตลอดจนสร้างฐานเวลาก็สามารถทำได้อย่างเที่ยงตรงแม่นยำ

เพิ่มมากขึ้น การมีวอตช์ด็อกไทเมอร์ในตัวจัดได้ว่าเป็นคุณสมบัติขั้นพื้นฐานของไมโครคอนโทรลเลอร์สมัยใหม่ไปแล้ว จึงส่งผลให้ไมโครคอนโทรลเลอร์ MCS-51 มีขีดความสามารถที่ทัดเทียมกับไมโครคอนโทรลเลอร์ในยุคใหม่ และยังคงเป็นที่นิยมใช้งานกันต่อไป

2.1.5 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

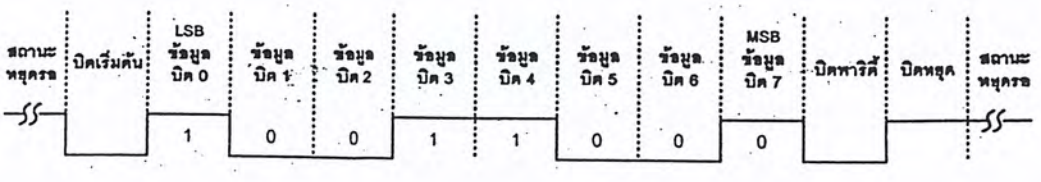
ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟูลดูเพล็กซ์ หมายถึง วงจรสื่อสารที่สามารถทำการรับส่งข้อมูลในลักษณะ 2 ทิศทางได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบแฟลช์เป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐาน RS-422 หรือ RS-485 ได้แล้ว โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือ บอดเรต (baud rate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรม มีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด (stop bit) มีขนาด 1 บิต



รูปที่ 2-16 รูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส

รูปที่ 2-16 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล ขา DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น (start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิต จากนั้นตามด้วยบิตพาริตี้ (parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือ บิตปิดท้ายหรือบิตหยุด (stop bit) โดยจะเป็นการทำให้ขา DATA มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต, 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่า ได้แก่ 110, 150, 300, 600, 1,200, 2,400, 4,800, 9,600 และ19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่า ข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์จะมีความยาวเท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9,600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd),แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์ รวมบิตพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99H หรือ 10011001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของบิตพาริตี้จะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asynchronous Receiver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก 1 ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้เป็น NONE นั้นทั้งภาครับและภาคส่ง

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน MCS-51

ในการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 มีรีจิสเตอร์ที่ต้องเกี่ยวข้องอยู่ 2 ตัว ดังมีรายละเอียดต่อไปนี้

รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต มีการแบ่งเป็น 2 ส่วน คือรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ R3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial port Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0-SM1(Serial port mode bit 0-1) : ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 ดังมีรายละเอียดต่อไปนี้

SM0	SM1	MODE	ความหมาย	BAUD RATE
0	0	0	Shift Register	เปลี่ยนแปลงไม่ได้ (ความถี่ osc/12)
0	1	1	8-bit UART	เปลี่ยนแปลงได้โดยกำหนดจาก Timer
1	0	2	9-bit UART	เปลี่ยนแปลงไม่ได้ (ความถี่ osc/12 หรือ/64)
1	1	3	9-bit UART	เปลี่ยนแปลงได้โดยกำหนดจาก Timer

SM2 : ใช้ในการอีนาเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ (multiprocessor) ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 ในโหมด 2 และ 3 ถ้าบิตนี้เป็น "1" บิต RI จะไม่แอกตีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น "0" (ข้อมูลบิตที่ 9 เก็บไว้ที่บิต RB8) ในการ

ทำงานโหมด 1 ถ้าบิตนี้เซต บิต RI จะไม่แอกตีฟถ้ายังไม่ได้รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN(Enable serial reception) : ใช้ในการอีนาเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ถ้าต้องการให้มีการรับข้อมูลต้องเซตบิตนี้ให้เป็น "1"

TB8 : ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RE8 : ใช้สำหรับรับข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น "0" ข้อมูลบิตที่ RB8 คือข้อมูลของบิตหยุด (STOP bit) สำหรับในการทำงานโหมด 0 บิตนี้จะไม่ใช้งาน บิต RB8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

TI (Transmit Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์เมื่อทำการส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่นบิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

RI (Receive Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการรับข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อสามารถรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิต SM2 มีการเซต บิตนี้จะเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเลือกโหมดการทำงานได้ถึง 4 โหมด คือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์
2. โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้
3. โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่
4. โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 กระทำได้โดยการกำหนดข้อมูลให้แก่บิต SM0 และ SM1 ในรีจิสเตอร์ SCON

การทำงานในโหมด 0 ของวงจรพอร์ตอนุกรม

มีไดอะแกรมการทำงานและไดอะแกรมเวลาแสดงในรูปที่ 2-17 ข้อมูลอนุกรมจะผ่านเข้าและออกทางขา RxD ส่วนขา TxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล (shift clock) ในโหมดนี้มีจำนวนข้อมูล 8 บิต โดยทำการรับและส่งข้อมูลในบิต LSB ก่อน อัตราในการรับส่งข้อมูลหรืออัตราบอดถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่สัญญาณนาฬิกา

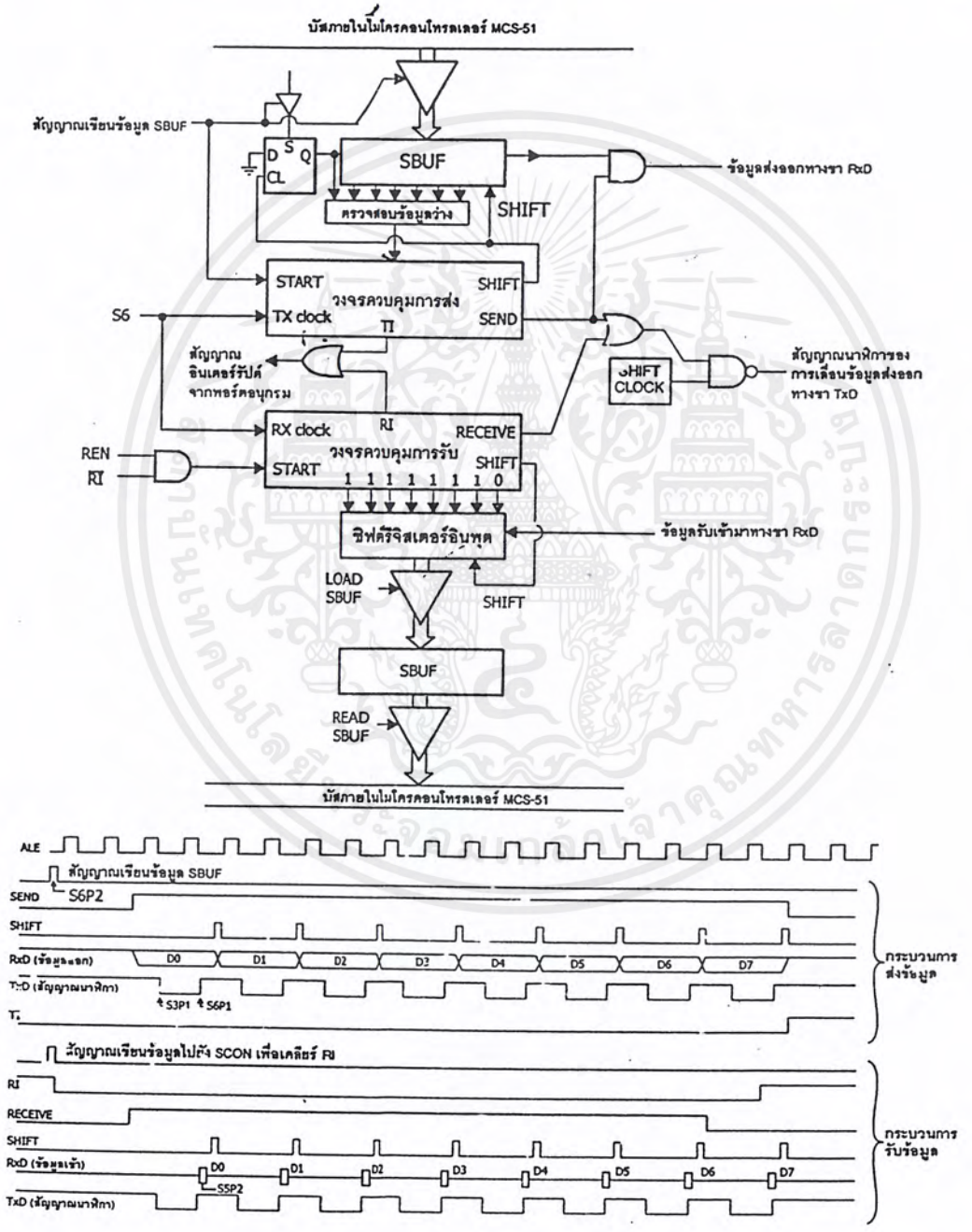
เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูล SBUF แอกตีฟเป็น "1" ที่สเตต 6 เฟส 2 (S6P2) ของแมชีนไซเกิล ส่งมายังวงจรควบคุมการส่ง(TX control) ทำให้วงจรการควบคุมการส่งเริ่มต้นการส่งข้อมูล สัญญาณ SEND จะแอกตีฟเป็น "1" ตลอดกระบวนการส่งข้อมูล

ข้อมูลจากรีจิสเตอร์ SBUF จะถูกเลื่อนออกที่ขา P3.0 หรือขา RxD ครั้งละบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทางขา P3.1 หรือ TxD โดยสัญญาณนาฬิกาของการเลื่อนข้อมูลจะมีขอบขาลงของสัญญาณที่สเตต 3 เฟส และมีขอบขาขึ้นของสัญญาณที่สเตต 6 เฟส 1 ของแต่ละแมชีนไซเกิลในกระบวนการส่งข้อมูล จนกระทั่งเมื่อส่งข้อมูลครบ 8 บิตแล้ว บิต TI ในรีจิสเตอร์ SCON จะเกิดการเซตเป็นการแจ้งให้ทราบว่าส่งข้อมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นโนระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ SEND จะกลายเป็น "0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

ในกระบวนการรับข้อมูล เริ่มต้นด้วยการเซต REN ให้เป็น "1" และเคลียร์บิต RI ในรีจิสเตอร์ SCON ก่อน ที่สเตต 6 เฟส 2 ของแมชีนไซเกิลถัดไป วงจรควบคุมการรับ (RX control) จะทำการเขียนข้อมูล 11111110 ไปยังชิฟต์รีจิสเตอร์สำหรับรับข้อมูลและทำการแอกตีฟสัญญาณ RECEIVE ให้เป็น "1" ในสัญญาณนาฬิกาลูกถัดไป

เมื่อสัญญาณ RECEIVE แอกตีฟ ก็จะเกิดการส่งสัญญาณนาฬิกาของการเลื่อนข้อมูล (Shift clock) ขึ้นผ่านทางขา P3.1 หรือ TxD เพื่อทำการกำหนดจังหวะการรับข้อมูลครั้งละบิต โดยสัญญาณนาฬิกานี้จะเกิดขึ้นในช่วงสเตต 3 เฟส 1 ถึงสเตต 6 เฟส 1 ของแต่ละแมชีนไซเกิล การรับข้อมูลเข้ามาทางขา P3.0 หรือ RxD จะเกิดขึ้นที่สเตต 5 เฟส 2 ในแมชีนไซเกิลเดียวกับสัญญาณนาฬิกาของการเลื่อนข้อมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต บิต RI จะได้รับการเซตเพื่อแจ้งการเสร็จสิ้นกระบวนการรับข้อมูล หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการอีนาเบิลไว้ จะเกิดการอินเตอร์รัปต์ขึ้นใน เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ RECEIVE จะกลายเป็น "0"

การทำงานในโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 จะใช้ประโยชน์ในการเชื่อมต่อกับไอซีรีจิสเตอร์ภายนอก เพื่อทำการขยายจำนวนพอร์ตอินพุตหรือเอาต์พุต แต่ไม่เป็นที่นิยมใช้งานมากนัก เนื่องจากในไมโครคอนโทรลเลอร์ MCS-51 เองมีพอร์ตอยู่ค่อนข้างมาก และสามารถติดต่อกับพอร์ตเหล่านั้นได้ง่ายและเร็วกว่ามาก



รูปที่ 2-17 ไดอะแกรมการทำงานโหมด 0 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

การทำงานในโหมด 1 ของวงจรพอร์ตอนุกรม

มีไดอะแกรมการทำงานและไดอะแกรมเวลาแสดงในรูปที่ 2-18 ในโหมดนี้จะใช้ในการรับส่งข้อมูลรวม 10 บิต โดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลรวมทั้ง 10 บิตประกอบด้วย บิตเริ่มต้น (มีค่าเป็น "0") 1 บิต บิตข้อมูล 8 บิต โดยรับหรือส่งข้อมูลในบิต LSB ก่อน และบิตหยุดหรือบิตปีดท้าย (มีค่าเป็น "1") ในการรับข้อมูล บิตหยุดจะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON อัตราบอดในโหมดนี้ได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 ใน AT89C51 ส่วนในไมโครคอนโทรลเลอร์เบอร์ AT89C52 และในอนุกรม AT89Sxx สามารถเลือกใช้อัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 หรือไทเมอร์ 2 ในการกำหนดเป็นยัตราบอดได้

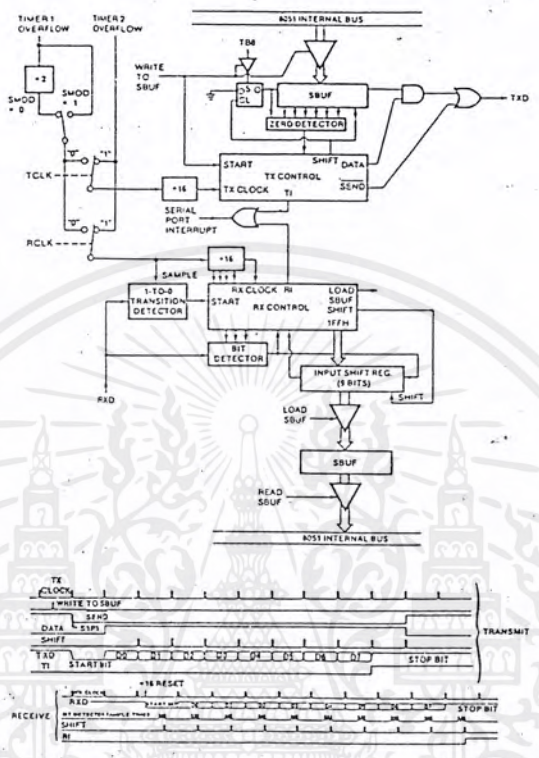
กระบวนการส่งข้อมูลเริ่มต้นด้วยการแอกตีฟสัญญาณเขียนข้อมูลมายังรีจิสเตอร์ SBUF ส่งมายังวงจรควบคุมการส่ง (TX control) จากนั้นวงจรควบคุมการส่งจะทำการแอกตีฟสัญญาณ SEND ที่สเตต 1 เฟส 1 ของแมชีนไซเกิลต่อมา โดยสัญญาณ SEND จะเป็น "0" ตลอดกระบวนการส่งข้อมูลเมื่อสัญญาณ SEND แอกตีฟ จะทำการส่งบิตเริ่มต้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 แมชีนไซเกิล จากนั้นตามด้วยการส่งบิตข้อมูล 8 บิต เรียงลำดับจากบิต LSB โดยข้อมูลที่ทำการส่งนี้จะถูกเรียกออกมาจากรีจิสเตอร์บัฟเฟอร์สำหรับการส่งข้อมูล ในทุก ๆ บิตข้อมูลที่ทำการส่งออกไป จะเกิดสัญญาณพัลส์ SHIFT ขึ้น เพื่อให้เรียกข้อมูลในแต่ละบิตจากรีจิสเตอร์ บัฟเฟอร์ การกำหนดจังหวะการส่งข้อมูลจะใช้สัญญาณนาฬิกาการส่ง (TX clock) เป็นตัวกำหนด โดยสัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณ TCLK จากไทเมอร์ 1 ด้วย 16 หลังจากการส่งบิตข้อมูลก็จะทำการส่งบิตหยุดหรือบิตปีดท้ายขนาด 1 บิต ดังนั้นการส่งข้อมูลและใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการส่งข้อมูลครบเรียบร้อยแล้ว จะทำการเซตบิต TI ในรีจิสเตอร์ SCON หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ

หลังจากที่ทำการบริการอินเตอร์รัปต์หรือส่งข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต TI ก่อนเป็นอันดับแรก เพื่อให้สามารถเริ่มต้นกระบวนการรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ด้านการรับข้อมูล จะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก "1" เป็น "0" ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของอัตราบอด เมื่อตรวจจับพบ ไทเมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดอัตราบอดจะรีเซต และทำการเขียนข้อมูล 1FFH ไปยังชิฟต์รีจิสเตอร์อินพุต

ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามาเป็น "0" หรือ "1" จะใช้ผลการสุ่มข้างมาก โดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 สเตต การสุ่มข้อมูลจะทำการสุ่มสเตตที่ 7,8 และ 9 หาก 2 ใน 3 ของการสุ่มพบว่า

ข้อมูลเป็นลอจิกใด จะตีความข้อมูลในบิตนั้นเป็นตามเสียงข้างมาก ยกตัวอย่างสุ่มพบลอจิก “1” 2 ใน 3 ครั้ง จะตีความว่าบิตของข้อมูลที่รับได้เป็น “1”



รูปที่ 2-18 ไดอะแกรมการทำงานในโหมด 1 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

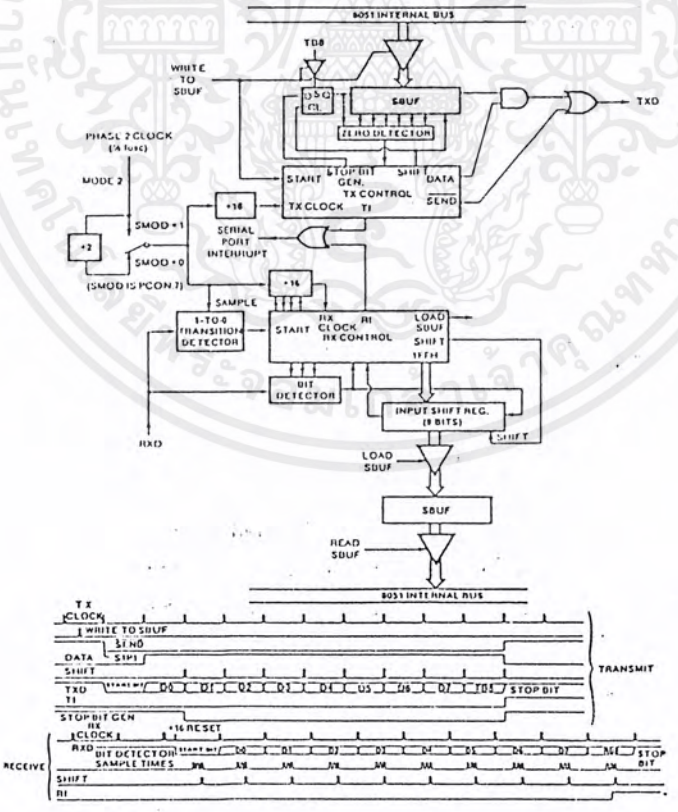
ลำดับของการรับข้อมูลมีลักษณะเดียวกับการส่งข้อมูลคือ เริ่มด้วยบิตเริ่มต้นก่อน ตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุก ๆ การรับข้อมูลได้ 1 บิต จะมีพลัส์ SHIFT เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู่รีจิสเตอร์บัฟเฟอร์การรับข้อมูล การกำหนดจังหวะการรับข้อมูลใช้สัญญาณนาฬิกาการรับข้อมูล (RX clock) หลังจากสัญญาณนาฬิกาลูกสุดท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากรีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์ SBUF และบิต RB8 ในรีจิสเตอร์ SCON โดยข้อมูลในบิต RB8 ก็คือข้อมูลของบิตหยุดนั่นเอง พร้อมกันนั้นยังทำการเซตบิต RI ในรีจิสเตอร์ SCON ด้วย หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ หลังจากที่ทำการบริการอินเตอร์รัปต์หรือรับข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI ก่อน เพื่อให้สามารถเริ่มต้นกระบวนการรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

การทำงานในโหมดนี้ได้รับความนิยมสูงสุด เนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

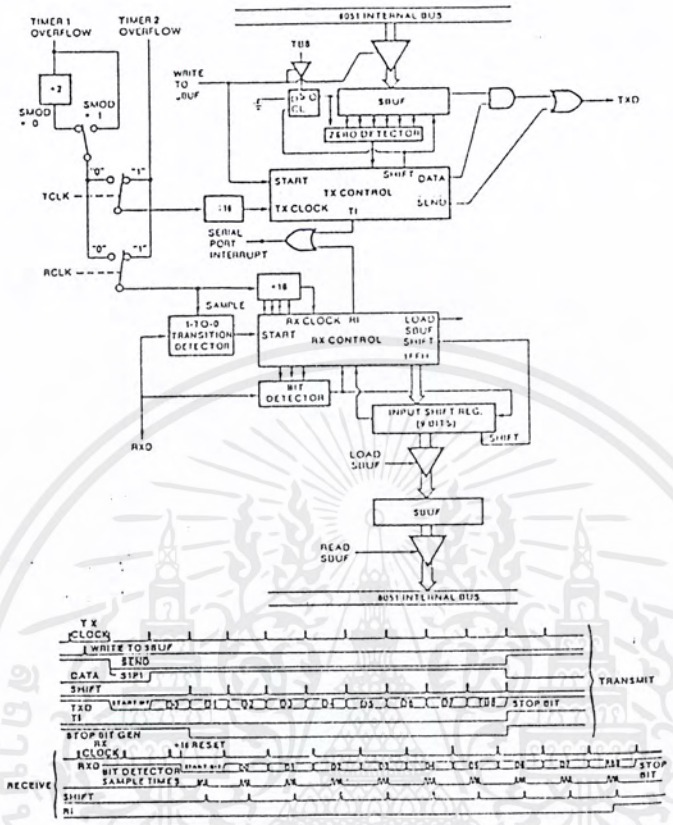
การทำงานในโหมด 2 และ 3 ของวงจรพอร์ตอนุกรม

ในทั้งสองโหมดนี้จะใช้รูปแบบข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้น มีค่าเป็น "0" จำนวน 1 บิต, บิตข้อมูล 8 บิต โดยทำการรับและส่งบิต LSB ก่อน, บิตข้อมูลบิตที่ 9 และบิตปิดท้าย มีค่าเป็น "1" จำนวน 1 บิต ในการส่งข้อมูลข้อมูลบิตที่ 9จะได้รับการเก็บไว้ในบิต TB8 ในรีจิสเตอร์ SCON และ ในการรับข้อมูล ข้อมูลบิตที่ 9 จะเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับอัตราบอดในโหมด 2 จะคงที่โดยเลือกได้ 2 ค่าคือ 1/32 หรือ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1

ในรูปที่ 2-19 เป็นไดอะแกรมการทำงานของพอร์ตอนุกรมในโหมด 2 และในรูปที่ 2-20 เป็นไดอะแกรมการทำงานของพอร์ตอนุกรมในโหมด 3 การทำงานโดยรวมจะคล้ายคลึงกับการทำงานของพอร์ตอนุกรมในโหมด 1 ส่วนที่แตกต่างกันระหว่างการทำงานของพอร์ตอนุกรมในโหมด 1 กับการทำงานของพอร์ตอนุกรมในโหมด 2 และ โหมด 3 ก็คือจำนวนบิตของข้อมูลที่ในโหมด 2 และ 3 จะมีเพิ่มมาอีก 1 บิต โดยส่วนใหญ่จะใช้เป็นบิตตรวจสอบพาริตี้



รูปที่ 2-19 ไดอะแกรมการทำงานในโหมด 2 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์



รูปที่ 2-20 ไดอะแกรมการทำงานในโหมด 3 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

อัตราบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51

โหมด 0

อัตราบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

อัตราบอดในโหมด 0 = ความถี่ของสัญญาณนาฬิกา /12 หน่วยเป็น บิตต่อวินาที

โหมด 1 และ 3

เนื่องจากทั้งสองโหมดนี้ สามารถเลือกแหล่งกำเนิดอัตราบอดได้ 2 แหล่งคือ จากอัตราโอเวอร์โฟลวของไทเมอร์ 1 และ 2 สำหรับอัตราบอดเมื่อใช้การโอเวอร์โฟลวของไทเมอร์ 1 จะต้องใช้ค่าของบิต SMOD ในรีจิสเตอร์ PCON (จะกล่าวถึงรายละเอียดของรีจิสเตอร์ตัวนี้ในบทที่ว่าด้วยการทำงานในโหมดประหยัดพลังงาน) มาพิจารณาประกอบด้วย โดยสามารถคำนวณหาค่าอัตราบอดได้จาก

$$\text{อัตราบอด} = (2^{\text{ค่าของบิต SMOD}}/32) \times \text{อัตราโอเวอร์โฟลวของไทเมอร์ 1}$$

อัตราบอด (บิตต่อวินาที : bps)	ความถี่สัญญาณนาฬิกา	SMOD	ไทเมอร์ 1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0 : สูงสุด 1 MHz	12 MHz	X	X	X	X
โหมด 2 : สูงสุด 375K	12 MHz	1	X	X	X
โหมด 1,3 : 62.5K	12 MHz	1	0	2	FFH
19.2K (19,200)	11.0592 MHz	1	0	2	FDH
9.6K (9,600)	11.0592 MHz	0	0	2	FDH
4.8K (4,800)	11.0592 MHz	0	0	2	FAH
2.4K (2,400)	11.0592 MHz	0	0	2	F4H
1.2K (1,200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

รูปที่ 2-21 การเลือกอัตราบอดของวงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

ถ้าหากในไทเมอร์ 1 ไม่ได้อีนาเบิลการอินเตอร์รัปต์ไว้สามารถคำนวณหาค่าอัตราบอดจาก

อัตราบอด = $(2^{\text{ค่าในรีจิสเตอร์ SMOD}}/32) \times$ (ความถี่สัญญาณนาฬิกา/{12 ×[256-(TH1)]})

ในรูปที่ 2-21 แสดงการกำหนดอัตราบอดโดยใช้ไทเมอร์ 1

ในกรณีที่ใช้ไทเมอร์ 2 ในการกำหนดอัตราบอด โดยกำหนดให้ไทเมอร์ 2 ทำงานในโหมดกำเนิดอัตราบอด (baud rate generator) สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = อัตราโอเวอร์โฟลวของไทเมอร์ 2/16 หน่วยเป็นบิตต่อวินาที

ถ้าหากกำหนดให้ไทเมอร์ 2 ทำงานในโหมดไทเมอร์หรือเคาเตอร์ตามปกติ สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = ความถี่ของสัญญาณนาฬิกา / (32 × (65536-(RCAP2H,RCAP2L)))

โดยที่ (RCAP2H,RCAP2L) เป็นค่าของรีจิสเตอร์ RCAP2H และ RCAP2L มีขนาด 16 บิต ไม่คิดเครื่องหมาย

โหมด 2

ในโหมดนี้การกำหนดอัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD เป็น "0" อัตราบอดจะเท่ากับ 1/64 ของความถี่สัญญาณนาฬิกา ในกรณีที่ SMOD เป็น "1" อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสูตรได้ดังนี้

อัตราบอด = $(2^{\text{ค่าของบิต SMOD}}/64) \times$ ความถี่สัญญาณนาฬิกา

การเขียนหรือส่งข้อมูลออกจากพอร์ตอนุกรม

ข้อมูลที่ต้องการส่งออกทุกค่าต้องนำไปเก็บไว้รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรม ซึ่งก็คือรีจิสเตอร์ SBUF ดังตัวอย่าง MOV SBUF,#'A'

จากคำสั่งข้างต้นเป็นการส่งข้อมูลของตัวอักษร A ออกไปยังพอร์ตอนุกรมของไมโครคอนโทรลเลอร์อย่างไรก็ตามก่อนทำการส่งข้อมูลทุกครั้ง ต้องแน่ใจว่าบิต TI เคลียร์หรือมีค่าเป็น "0" และเมื่อทำการส่งข้อมูลเรียบร้อยแล้ว ก็จะเกิดการเซตบิต TI เพื่อแจ้งให้ทราบ ดังตัวอย่างโปรแกรมต่อไปนี้

```
CLR TI      ;เคลียร์บิต TI เพื่อเตรียมการส่งข้อมูล
MOV SBUF,#'A' ; ส่งข้อมูลของตัวอักษร A ไปยังพอร์ตอนุกรม
JNB TI,S    ; รอการเซตของบิต TI เพื่อแจ้งการส่งข้อมูลที่เสร็จสมบูรณ์
```

การอ่านหรือรับข้อมูลจากพอร์ตอนุกรม

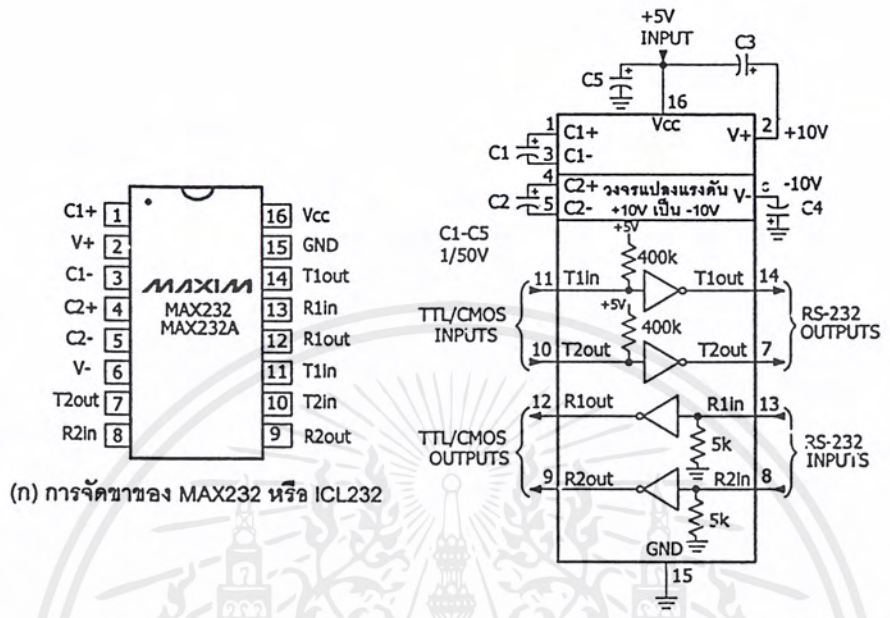
การรับข้อมูลจากพอร์ตอนุกรมสามารถกระทำได้ง่ายมาก เพียงทำการตรวจสอบว่าบิต RI เกิดการเซตขึ้นหรือไม่ ถ้าพบว่ามีการเซตเกิดขึ้นแล้ว ให้ทำการอ่านค่าจากรีจิสเตอร์ SBUF โดยต้องทำการโอนย้ายข้อมูลผ่านทางแอกคิวมูเลเตอร์หรือรีจิสเตอร์ A ดังตัวอย่าง

```
CLR RI      ; เคลียร์บิต TI เพื่อเตรียมการส่งข้อมูลออก
JNB RI,S    ; รอคอยการเซตของบิต RI อันเป็นการแจ้งให้ทราบว่า การรับข้อมูลเสร็จ
              สมบูรณ์และมีข้อมูลเกิดขึ้นที่รีจิสเตอร์ SBUF
MOV A,SBUF  ; อ่านค่าจากรีจิสเตอร์ โดยการโอนย้ายข้อมูลผ่านทางรีจิสเตอร์A
CLR RI      ; หลังจากทำการอ่านข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI เสมอ
```

การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ ±3 ถึง ±12V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

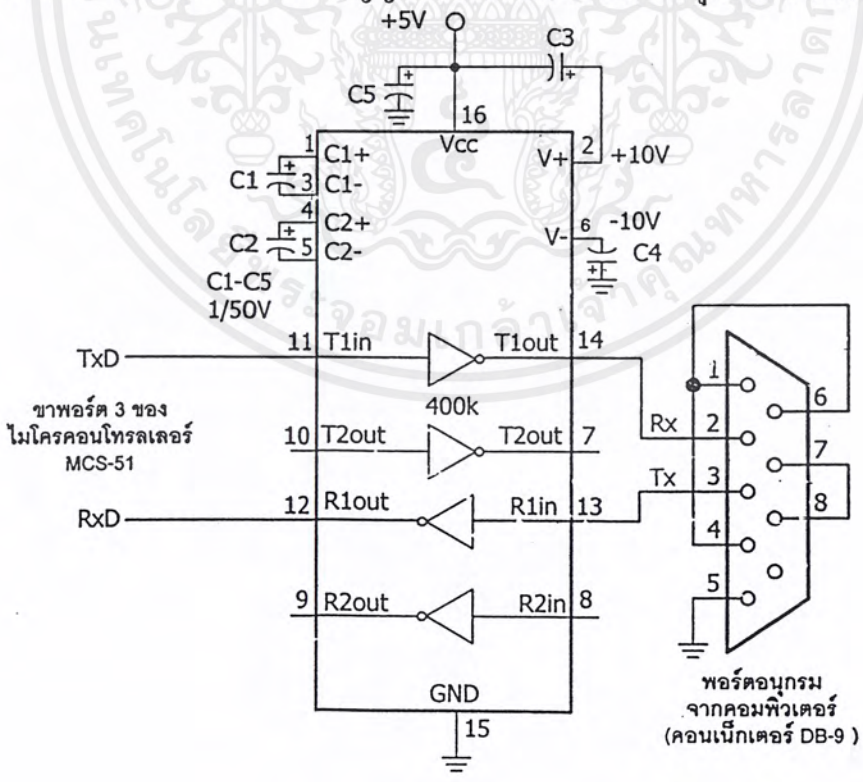
ไอซีที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้ต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ อาทิ ICL232 ในรูปที่ 2-22 แสดงการจัดขาของไอที ICL232 ส่วนวงจรของการต่อกับไมโครคอนโทรลเลอร์ MCS-51 แสดงในรูปที่ 2-23



(ก) การจัดขาของ MAX232 หรือ ICL232

(ข) โครงสร้างภายใน ของ MAX232 หรือ ICL232

รูปที่ 2-22 การจัดขาของไอซีแปลงสัญญาณเพื่อเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

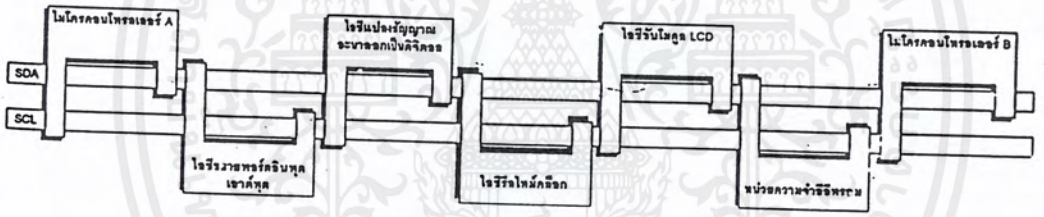


รูปที่ 2-23 วงจรเชื่อมต่อ CIL232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์

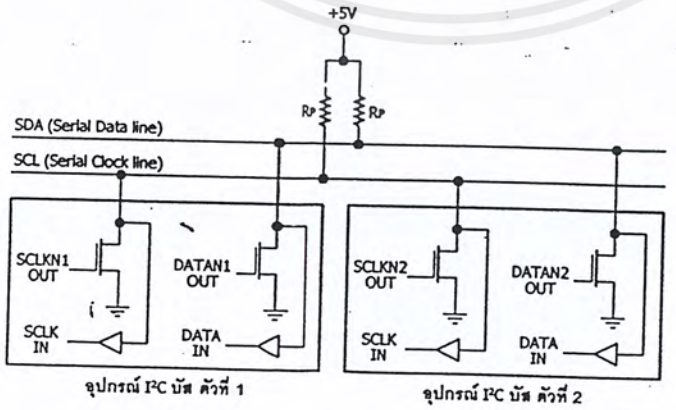
2.2 ความรู้เบื้องต้นเกี่ยวกับ I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซีโดยบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลักคือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมากเพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2Cมีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรมหรือ SDA (Serial Data line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรมหรือ SCL (Serial Clock line) ในการอธิบายต่อไปนี้จะเรียกสายสัญญาณทั้งสองว่า สาย SDA และ SCL



รูปที่ 2-24 ผังแสดงการเชื่อมต่อของอุปกรณ์ต่าง ๆ บนระบบบัส I^2C



รูปที่ 2-25 แสดงวงจรเอาต์พุตของอุปกรณ์ในระบบบัส I^2C

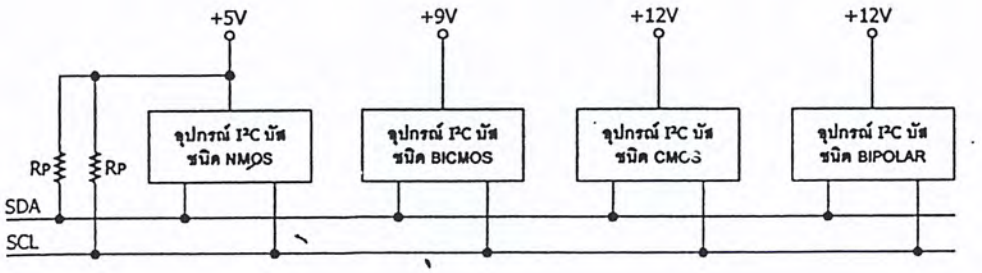
ในรูปที่ 2-24 แสดงผังของการเชื่อมต่ออุปกรณ์ต่าง ๆ บนบัส I^2C จะเห็นได้ว่า อุปกรณ์ที่ทำการเชื่อมต่อบนบัส I^2C มีหลากหลาย ไม่ว่าจะเป็นไอซีขยายพอร์ตอินพุตเอาต์พุต (I/O Expander) ไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอล (ADC) และแปลงสัญญาณดิจิตอลเป็นอะนาลอก (DAC) ไอซีรีลไทม์คล็อก (RTC), ไอซีขับโมดูล LCD หน่วยความจำอีอีพรอม และไมโครคอนโทรลเลอร์

2.2.1 คุณสมบัติโดยทั่วไปของบัส I^2C

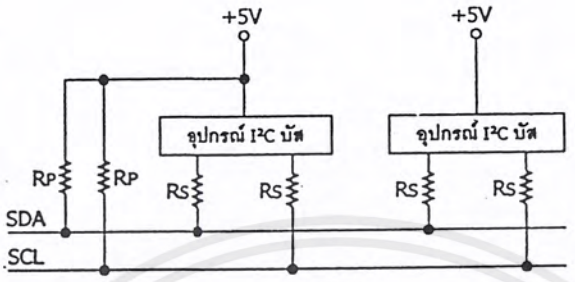
สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลาเพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งานทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (open-drain) หรือคอลเล็กเตอร์เปิด open-collector ดังแสดงรายละเอียดในรูปที่ 2-25

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

ข้อเด่นอีกประการหนึ่งของบัส I^2C คือ สามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถต่อติดสื่อสารกันได้ โดยอุปกรณ์บนบัส I^2C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12V การต่อร่วมกันบนบัส I^2C สามารถกระทำได้ในลักษณะเดียวกับกรณีที่อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือ ให้ต่อสาย SDA และSCL ของอุปกรณ์แต่ละตัวเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัป (Rp) เข้ากับแรงดัน +5V ไว้ด้วยเสมอ ดังแสดงในรูปที่ 2-26



รูปที่ 2-26 การต่อตัวต้านทานพูลอัปบนสายสัญญาณในระบบบัส I^2C



รูปที่ 2-27 การต่อตัวต้านทาน R_s เพื่อลดสัญญาณรบกวนขนาดใหญ่ที่อาจเข้ามาในบัส I^2C

ในกรณีที่อาจมีแรงดันไฟกระชากขนาดใหญ่ปะปนเข้ามาในบัส I^2C ที่ขา SDA และ SCL ของอุปกรณ์แต่ละตัวต้องต่อตัวต้านทานอนุกรมกับขา SDA และ SCL เรียกว่า R_S ก่อนต่อเข้าสู่บัส I^2C ดังแสดงในรูปที่ 2-27

2.2.2 หลักการของบัส I^2C

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

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (transmitter)

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

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ (master)

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่า สเลฟ (slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

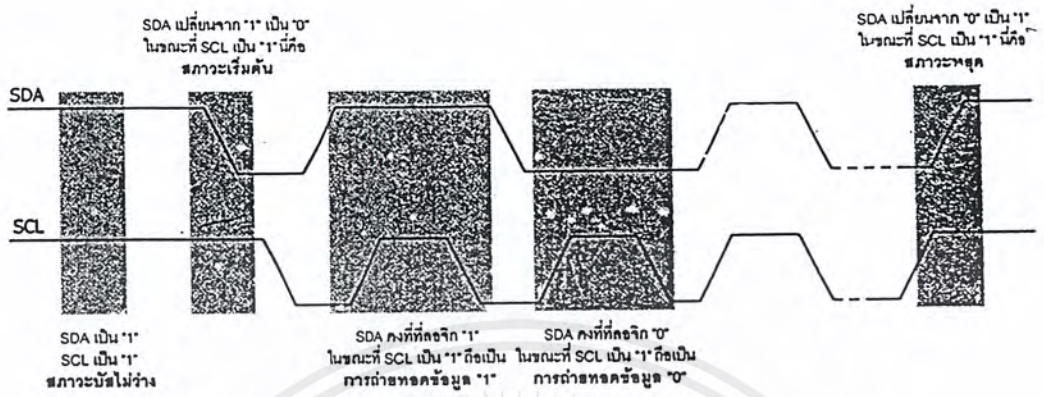
1. การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

2. ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

2.2.3 สภาวะที่เกิดขึ้นบนบัส I²C

มีด้วยกัน 5 สภาวะ ดังนี้

1. **บัสว่าง (Bus not busy)** สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้
2. **เริ่มต้นการถ่ายทอดข้อมูล (start data transfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกว่า **สภาวะเริ่มต้น (STOP)**
3. **หยุดการถ่ายทอดข้อมูล (stop data transfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า **สภาวะหยุด (STOP)**
4. **ข้อมูลดำรงอยู่บนบัส (data valid)** สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น "0" หรือ "1" ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขาSDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดนั้น
5. **รับรู้ข้อมูล (acknowledge)** เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตเรียกว่า **บิตรับรู้ (acknowledge bit)** มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตังส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสภานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว



รูปที่ 2-28 ไดอะแกรมเวลาแสดงสถานะต่าง ๆ ในบัส I^2C

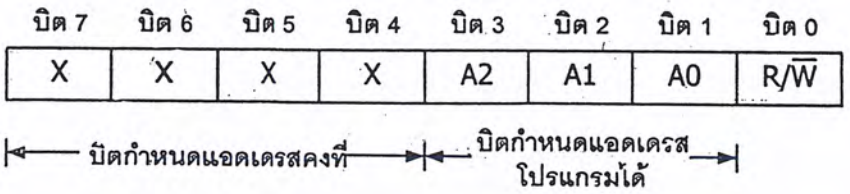
2.2.4 การทำงานบนบัส I^2C

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

ดังนั้นหัวใจสำคัญในอันดับแรกของการทำงานบนบัส I^2C คือการอ้างถึงอุปกรณ์แต่ละตัว ซึ่งในที่นี้จะอธิบายรายละเอียดของการอ้างถึงแบบ 7บิต

การอ้างถึงแบบ 7 บิต (7-bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ หรือ ข้อมูลกำหนดแอดเดรส โดยมีรูปแบบแสดงในรูปที่ 2-29 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็น*บิตกำหนดแอดเดรสคงที่ (fixed address bit)* จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้น ๆ หากบิต LSB เป็น "0" หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ



รูปที่ 2-29 รูปแบบของข้อมูลกำหนดแอดเดรที่ใช้ในการอ้างถึงแบบ 7 บิต

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (control byte) อุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นเอาต์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ ในรูปที่ 2-30 แสดงรูปแบบข้อมูลอนุกรมที่เกิดขึ้นในการติดต่อบนบัส I^2C ของการอ้างถึงแบบ 7 บิต

Start	A0-A6	R/W=0	ACK	C6-C0	R/W	ACK	D7-D0	ACK	Stop

รูปที่ 2-30 รูปแบบของข้อมูลอนุกรมที่ใช้ติดต่อกับอุปกรณ์บัส I^2C เมื่อใช้การอ้างถึงแบบ 7 บิต

2.2.5 อุปกรณ์ที่ใช้การเชื่อมต่อแบบบัส I^2C

ในปัจจุบัน บัส I^2C ได้รับความนิยมเพิ่มมากขึ้นเรื่อย ๆ ด้วยข้อดีที่ชัดเจนคือ ใช้สายสัญญาณเพียง 2 เส้นเท่านั้น และการขยายระบบไมโครคอนโทรลเลอร์ ที่มีจำนวนอินพุตเอาต์พุตและหน่วยความจำกัดจำกัดสามารถทำได้ง่ายขึ้นด้วยระบบบัส I^2C เมื่อเป็นเช่นนี้จึงมีอุปกรณ์เพอริเฟอรัลที่ใช้การเชื่อมต่อแบบบัส I^2C มากมายจากหลายผู้ผลิตออกมาให้ได้งานกัน ดังมีตัวอย่างต่อไปนี้

ไอซีขยายพอร์ตอินพุตเอาต์พุต (I/O expander) PCF8574, PCF8582, PCF8582

ไอซีหน่วยความจำอีอีพรอมอนุกรม (Serial EEPROM) : 24Cxx, PCF8570, PCF72/73

ไอซี ADC/DAC : PC 8591

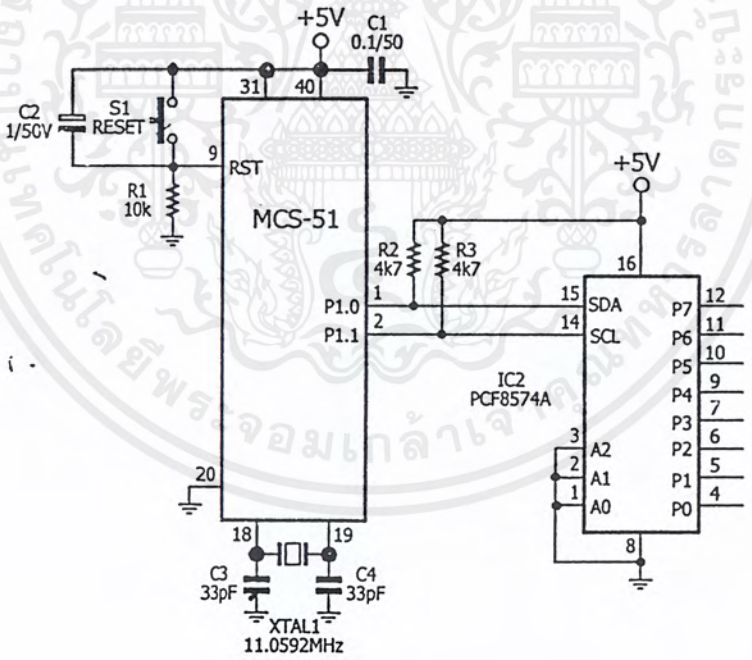
ไอซีรีลไทม์คล็อก (Real-time clock : RTC): PCF8583, 41T56C, DS1307

ไอซีขับLCD โมดุล (LCDdriver):PCF8466, PCF8576, PCF8577/78, PCF8579, SAA1064

ไอซีกำเนิดสัญญาณ DTMF (DTMF generator):PCD3311/12

2.2.6 การต่ออุปกรณ์ระบบบัส I^2C กับไมโครคอนโทรลเลอร์ MCS-51

สามารถทำได้ง่ายมาก เพียงใช้ขาพอร์ต 2 ขา โดยกำหนดให้ขาหนึ่งเป็น SDA อีกขาหนึ่งเป็น SCL และต่อตัวต้านทานค่าประมาณ 4.7k พูลอัปที่ขาพอร์ตทั้งสองขา เพียงเท่านี้ก็สามารถติดต่อกับอุปกรณ์ระบบบัส I^2C ได้แล้ว



รูปที่ 2-31 วงจรตัวอย่างการต่อไมโครคอนโทรลเลอร์ MCS-51 กับอุปกรณ์ระบบบัส I^2C

ในรูปที่ 2-31 แสดงการต่อไมโครคอนโทรลเลอร์ MCS-51 เข้ากับระบบบัส I^2C จากวงจรจะใช้ขาพอร์ต P1.0 เป็นขา SDA และ P1.1 เป็นขา SCL

2.3 รายละเอียดเกี่ยวกับโมดูล LCD

ในโมดูล LCD จะมีส่วนประกอบหลัก ๆ 3 ส่วน ดังนี้

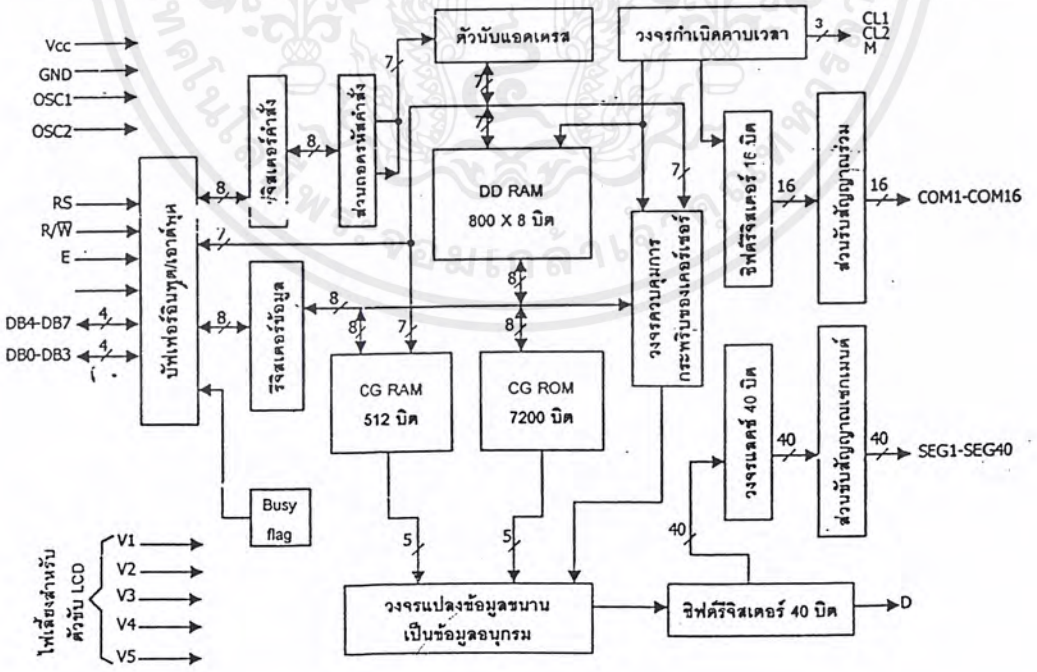
ตัวแสดงผล (display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

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

ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

2.3.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในหนังสือนี้ขอยกตัวอย่างโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย ให้รูปที่ 2-32 เป็นบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูลLCD แบบอักขระ ประกอบด้วย



รูปที่ 2-32 ไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

บัฟเฟอร์อินพุตเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register :IR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register :DR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอกเพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

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

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

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGRAM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับCGRAM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAMเอง

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

2.3.2 โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด (LCD 16x1)

สำหรับโมดูล LCD ที่ยกมาใช้ในการอธิบาย เป็นขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตราฐาน มีผู้ผลิตหลายราย และมีการระบุเบอร์แตกต่างกันออกไปตามผู้ผลิต อาทิ LM020L ของฮิตาชิ, DMC-16117A ของคอปเทริกซ์ (Optrex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันนั่นคือเบอร์ HD44750 ของฮิตาชิ โมดูล LCD ขนาด 16x1มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังในรูปที่ 2-33

V_{SS} (ขา 1) : ต่อกราวด์

V_{DD} (ขา 2) : ต่อไฟเลี้ยง +5โวลต์

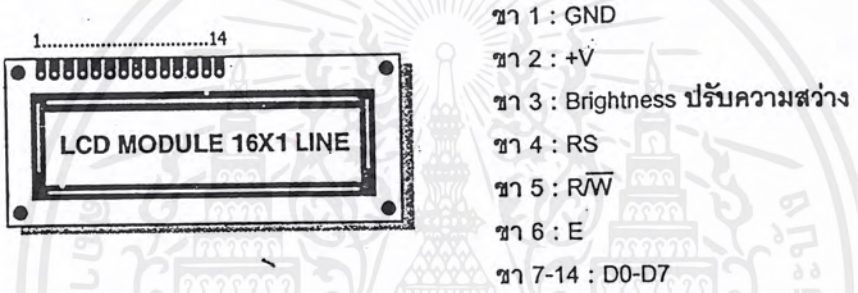
V_O (ขา 3) : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

TS (ขา 4) : เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้น ว่าเป็นค่าสั่งรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น "0" ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขาเป็น "1" ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

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

E (ขา 6) : เป็นขาอีนาเบิล LCD ให้ทำงาน

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



รูปที่ 2-33 รูปร่างและการจัดขาโมดูล LCD แบบอักขระ

2.3.3 คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ R/W เป็น "0" แล้วเขียนคำสั่งตามไป คำสั่งควบคุมโมดูล LCD ของชิปควบคุม HD44780 ที่สำคัญมี10 คำสั่งดังนี้

1. คำสั่งเคลียร์ตัวแสดงผล (clear display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซิคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์ จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิล I/D (ซึ่งจะกล่าวถึงภายหลัง) ให้เป็น "1"

2. คำสั่ง return home

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

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode Set)

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	0	1	I/D	S

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

บิตI/D เป็นบิตที่ใช้ในการกำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว ทำให้แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0" แอดเดรสจะลดลง

ดังนั้น ข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ ได้แก่ 04H - 07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดให้เมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรสของDDRAM เพิ่มขึ้น

4. คำสั่งควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	1	D	C	B

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

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปิดเคอร์เซอร์หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกะพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกะพริบ

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

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	1	S/C	R/L	*	*

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผลขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

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

6. คำสั่งกำหนดฟังก์ชันการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	DL	N	F	*	*

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

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

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น "0" จะเป็นการแสดงผลแบบ 5x7 จุด และถ้าเป็น "1" จะแสดงผลเป็นแบบ 5x10 จุด

ข้อมูลคำสั่งที่ใช้บ่อยคือ 38H เป็นการกำหนดให้โมดูล LCD ทำงานในแบบ 8 บิต แสดงผล 2 บรรทัด และเลือกความละเอียดเป็น 5x7 จุด

โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง 1 บรรทัด แต่จะต้องกำหนดN ให้เป็น "1" เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00 H และ 40H

7. คำสั่งเลือกแอดเดรสของ CGRAM

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

8. คำสั่งเลือกแอดเดรสของ DDRAM

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

9. คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
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 ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง

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

นอกจากนี้ ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0 บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

2.3.4 การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้งานต้องการ ต้องส่งคำสั่ง (instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (data) ที่ต้องการแสดงผลเนื่องจากบัสข้อมูลของโมดูล LCD มี 8 เส้น คือ D0-D7 และใช้เป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ถ้าหากที่ขาRS ได้ลอจิก "0" หมายความว่า ข้อมูลที่ป้อนให้แก่โมดูล LCD ขณะนั้นเป็นคำสั่ง ใน

ทางตรงข้าม หากขา RS ได้รับลอจิก "1" ข้อมูลที่ป้อนขณะนี้นเป็นข้อมูลที่ใช้ในการแสดงผล เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อน โดยใช้คำสั่งเลือกแอดเดรส จากนั้นกำหนดให้ขา RS เป็น "1" เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลปกติไม่ใช่คำสั่ง

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

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

2.3.5 จังหวะการทำงานของ LCD โมดูล

ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูลเนื่องจากต้องรอให้คอนโทรลเลอร์ภายใน LCD โมดูล แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อน จากนั้นจึงจะรับข้อมูลหรือคำเนินการต่อไป

ดังนั้น ในการใช้งานโมดูล LCD ผู้เขียนโปรแกรมต้องมีโปรแกรมเพื่อหน่วยเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้แก่โมดูล LCDต้องรอประมาณ 10 มิลลิวินาที เพื่อให้โมดูล LCD ทำการเตรียมความพร้อมหรืออินิเชียล (initial) หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขา RS ของโมดูล LCDแล้วต้องหน่วงเวลาอีกประมาณ 2 มิลลิวินาที เพื่อให้คอนโทรลเลอร์ใน LCD โมดูลแปลความหมายของลอจิกที่ขา RS ว่า ข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่ออีนาเบิลโมดูล LCD ให้รับข้อมูลเข้าไป โดยพัลส์ที่ป้อนเข้าที่ขา E ของ โมดูล LCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาที

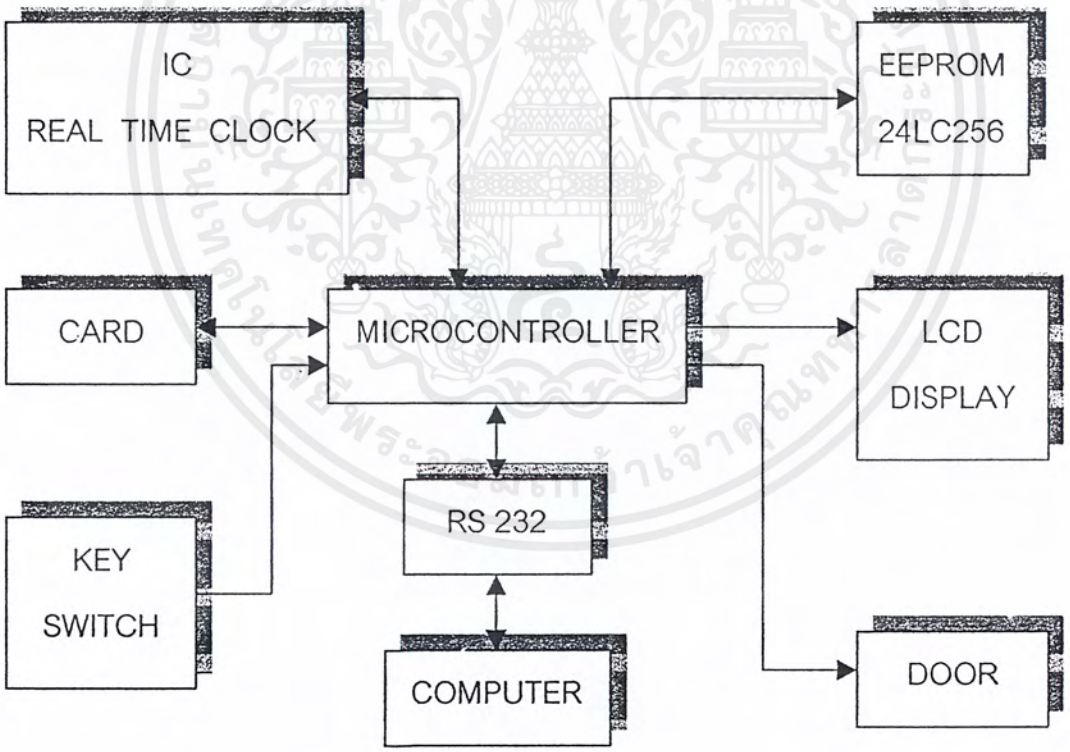
บทที่ 3
การออกแบบ

การออกแบบนั้นแบ่งออกได้เป็น 2 ส่วน คือ

- ส่วนของฮาร์ดแวร์
- ส่วนของโปรแกรมควบคุมไมโครคอนโทรลเลอร์
- ส่วนของโปรแกรมควบคุมการแสดงผล Visual Basic

3.1 ส่วนของฮาร์ดแวร์

การออกแบบพิจารณาได้จาก Block Diagram การทำงานของระบบ ดังรูปที่ 3-1



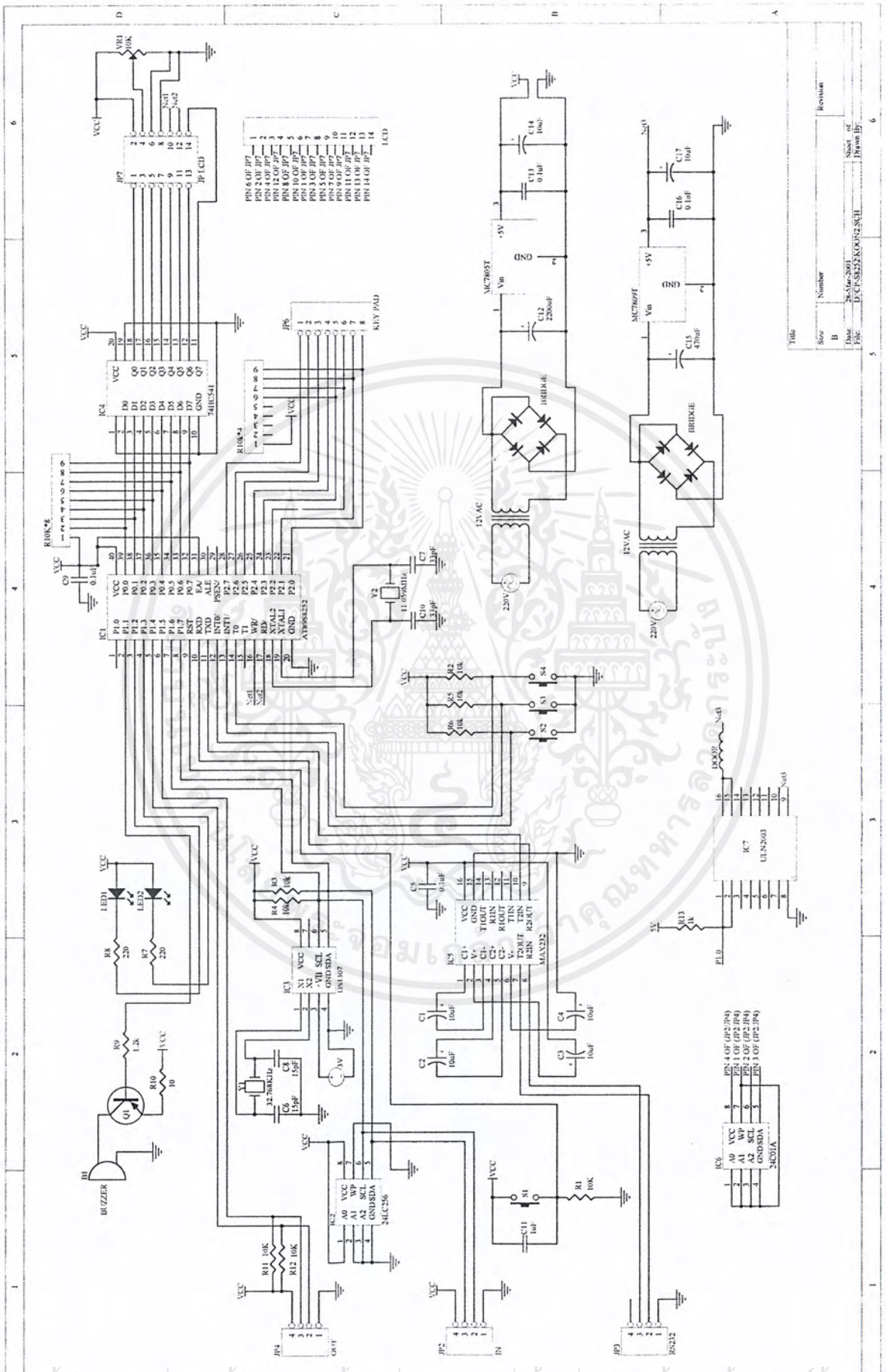
รูปที่ 3-1 Block Diagram แสดงการทำงานในส่วนของ ฮาร์ดแวร์

หลักการทำงาน

ในการ์ดจะมี EEPROM ซึ่งเอาไว้เก็บข้อมูลซึ่งเป็นรหัสส่วนตัวของผู้ถือบัตร ในภาวะปกติไมโครคอนโทรลเลอร์จะทำการอ่านข้อมูลมาจากไอซีสร้างฐานเวลาและนำค่าเวลาที่อ่านได้มาแสดงผลที่ LCD เมื่อมีการเสียบบัตรที่เครื่องไมโครคอนโทรลเลอร์จะนำข้อมูลในบัตรมาประมวลผลว่าตรงกับข้อมูลในไมโครคอนโทรลเลอร์หรือไม่ ถ้าข้อมูลตรงกันก็จะนำข้อมูลที่ผ่านการประมวลผลแล้วไปเก็บในหน่วยความจำ EEPROM 24LC256 โดยข้อมูลที่นำไปเก็บนั้นประกอบด้วย วัน เดือน ปี เวลาเข้าหรือออก และรหัสในบัตร จากนั้นไมโครคอนโทรลเลอร์ก็จะส่งสัญญาณไปทริกที่ประตูไฟฟ้าซึ่งจะทำให้ประตูเปิดได้ชั่วขณะเวลาหนึ่ง

ในส่วนของคีย์สวิตช์มีไว้สำหรับในการตั้งเวลา และเปลี่ยนแปลงข้อมูลในบัตรโดยการควบคุมของไมโครคอนโทรลเลอร์

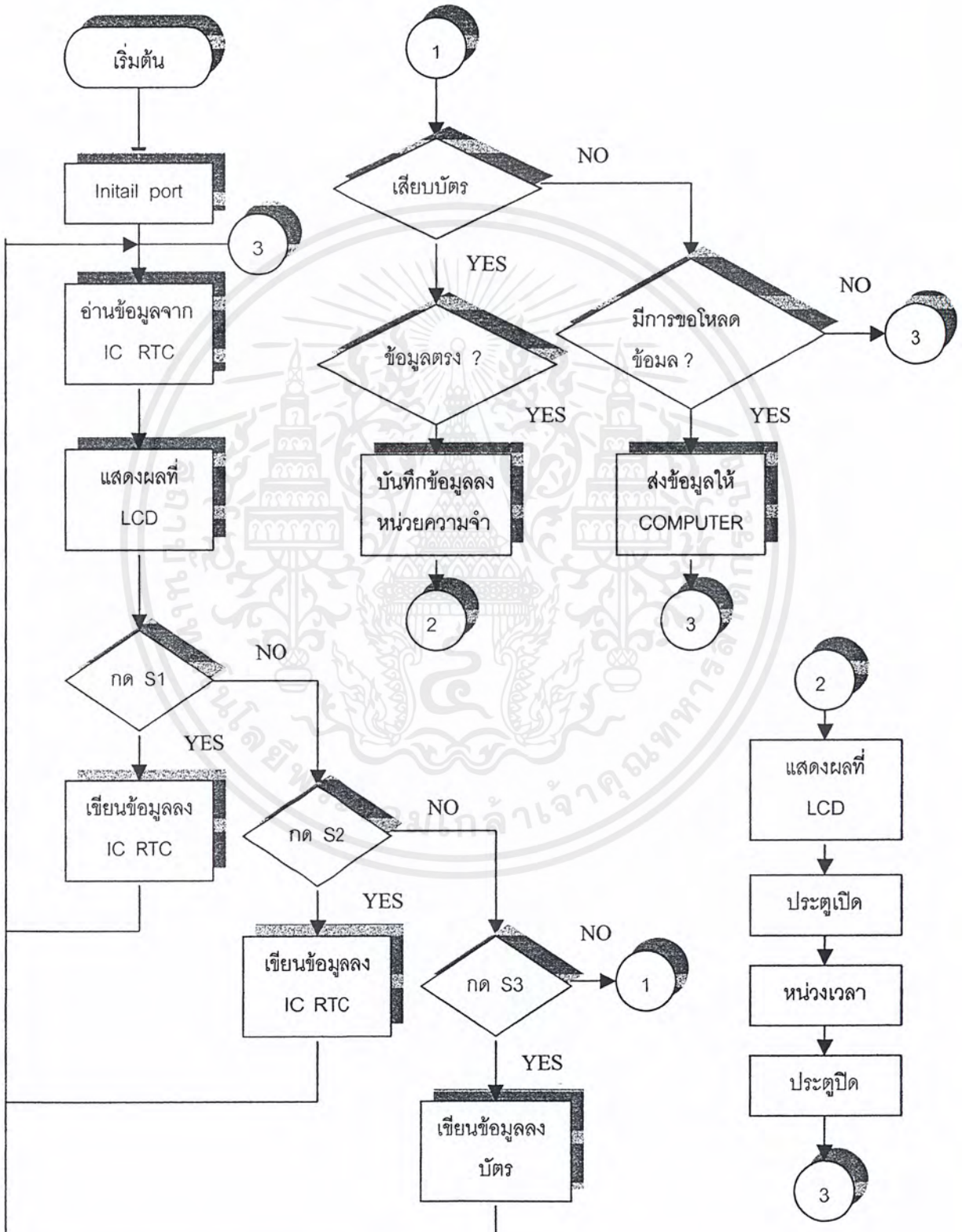
เมื่อคอมพิวเตอร์ต้องการจะโหลดข้อมูลก็จะส่งสัญญาณไปให้ไมโครคอนโทรลเลอร์รับรู้ว่าขณะนี้ต้องการจะโหลดข้อมูลและไมโครคอนโทรลเลอร์ก็จะทำการส่งข้อมูลให้กับคอมพิวเตอร์ ซึ่งวงจรในการออกแบบแสดงได้ดังรูปที่ 3-2



รูปที่ 3-2 แสดงวงจรในส่วนของไมโครคอนโทรลเลอร์

3.2 ส่วนของโปรแกรมควบคุมไมโครคอนโทรลเลอร์

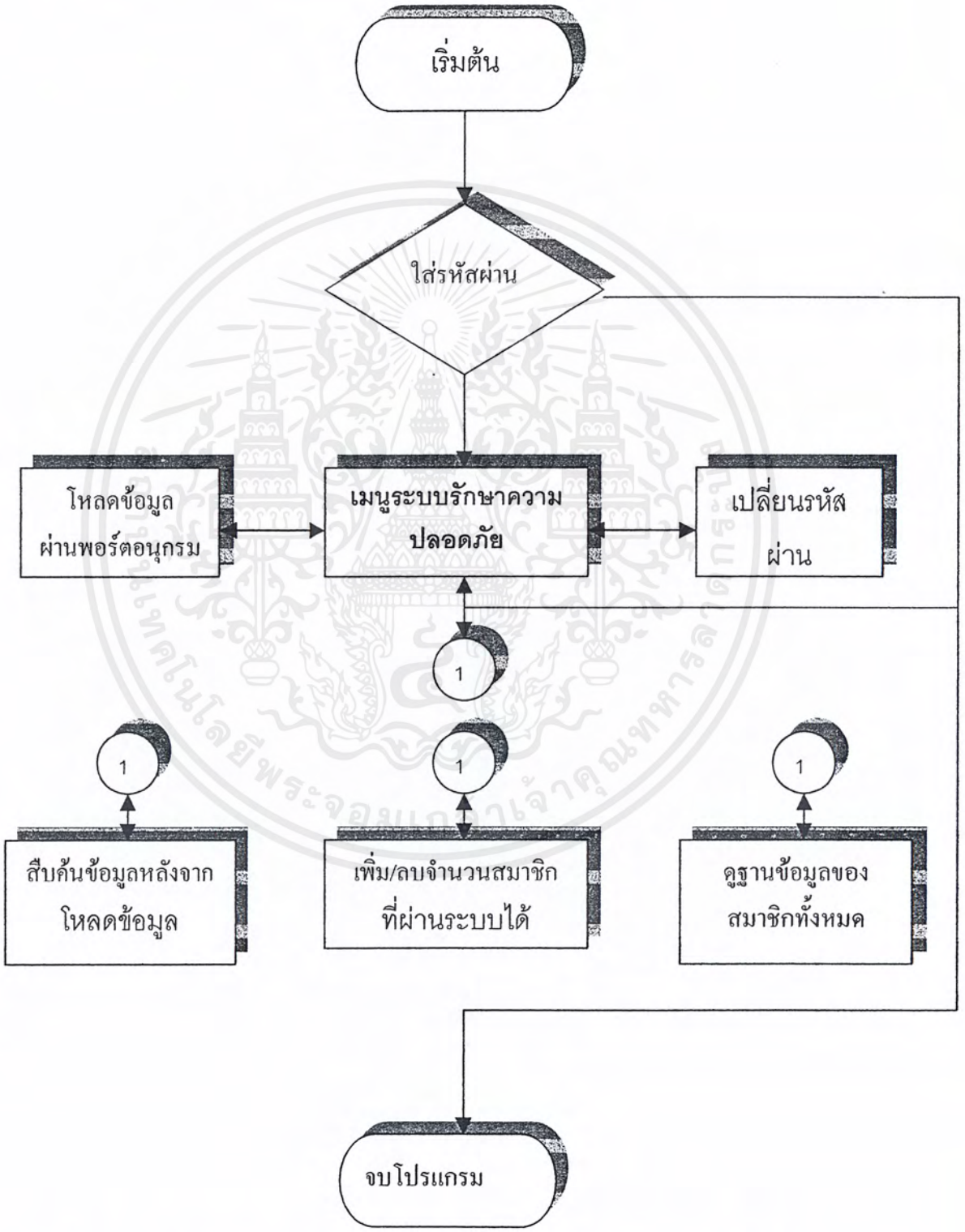
โปรแกรมการควบคุมแสดงได้ดังผังงานต่อไปนี้



รูปที่ 3-3 Block Diagram แสดงลำดับการทำงานของไมโครคอนโทรลเลอร์และอุปกรณ์ต่อร่วม

3.3 ส่วนของโปรแกรมควบคุมการแสดงผล Visual Basic

โปรแกรมการควบคุมแสดงได้ดังผังงานต่อไปนี้

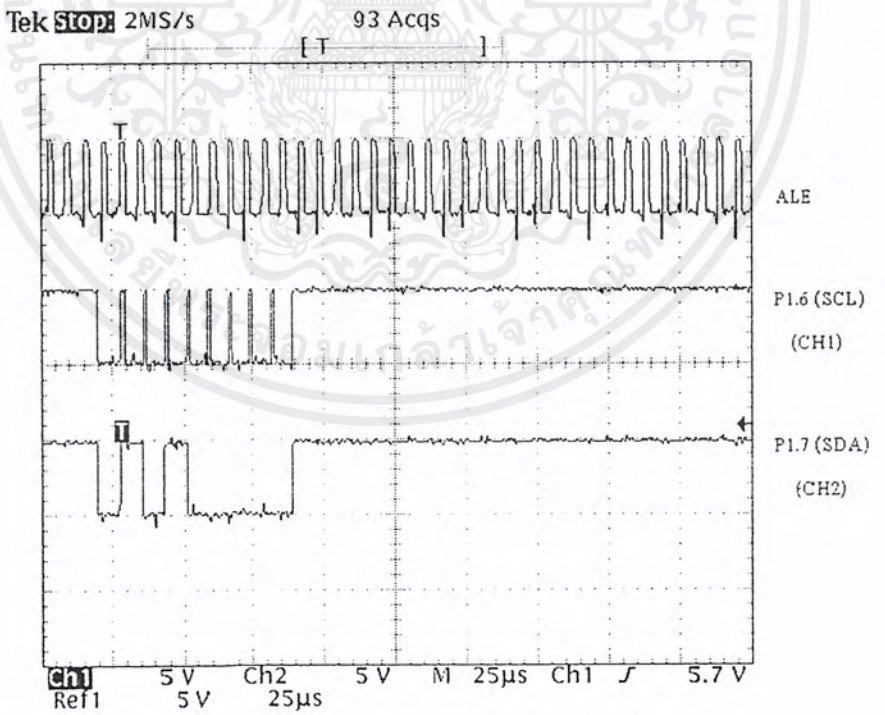


รูปที่ 3-4 Block Diagram แสดงลำดับการทำงานในส่วนของโปรแกรม Visual Basic

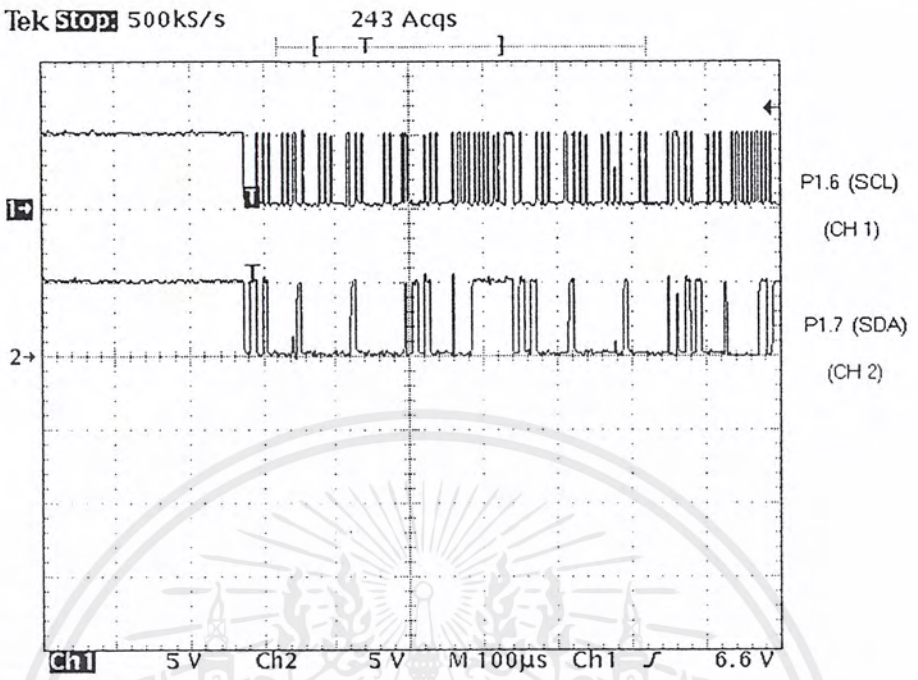
บทที่ 4
ผลการทดลอง

เมื่อทำการออกแบบระบบรักษาความปลอดภัยเบื้องต้น โดยใช้ไมโครคอนโทรลเลอร์ทำการเปรียบเทียบข้อมูลในบัตรที่มีหน่วยความจำ EEPROM และเขียนโปรแกรมควบคุมเสร็จสมบูรณ์แล้ว จึงทำการทดลองได้ดังนี้

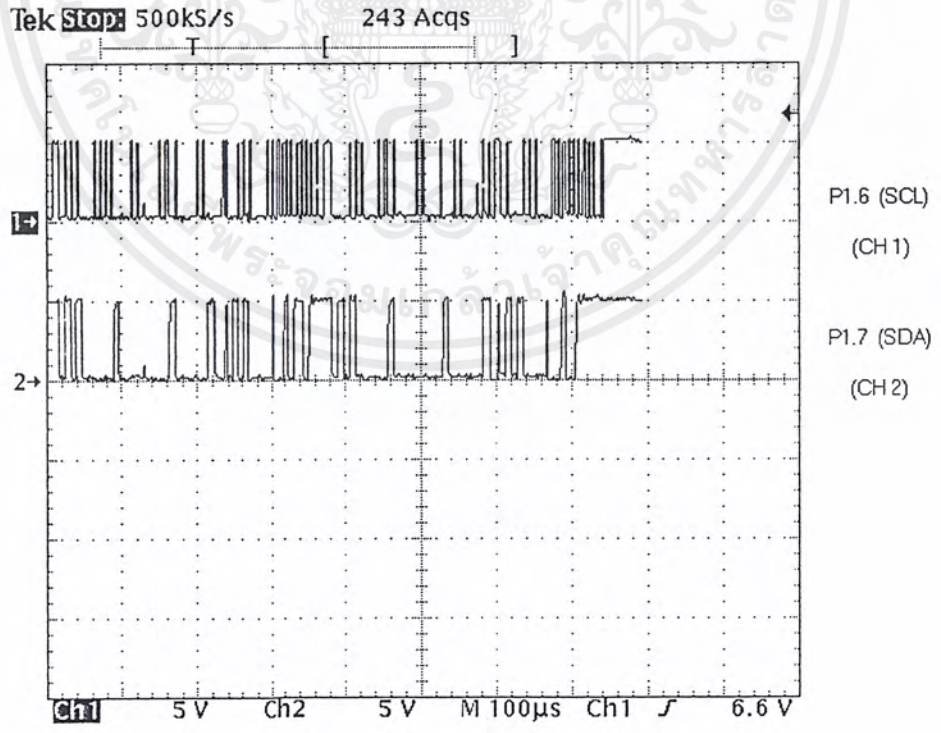
1. ทำการต่อบอร์ดไมโครคอนโทรลเลอร์ร่วมกับเครื่องคอมพิวเตอร์โดยผ่านพอร์ตอนุกรม RS-232
2. เมื่อมีสมาชิกเสียบบัตรเพื่อต้องการเข้า-ออกห้องที่เครื่องรับบัตรในส่วนของบอร์ดไมโครคอนโทรลเลอร์จะทำตรวจสอบข้อมูลลับและบันทึกข้อมูลที่ต้องการไว้หน่วยความจำถาวร (EEPROM) และการกระทำดังกล่าวข้างต้นจะเป็นเช่นเดียวกันกับสมาชิกลำดับไปที่ต้องการผ่านระบบรักษาความปลอดภัยนี้



รูปที่ 4-1 แสดงสัญญาณที่พอร์ต P1.6 , P1.7 ขณะยังไม่มีการเสียบบัตร



รูปที่ 4-2 แสดงสัญญาณที่พอร์ต P1.6 , P1.7 ขณะมีการเสียบบัตร



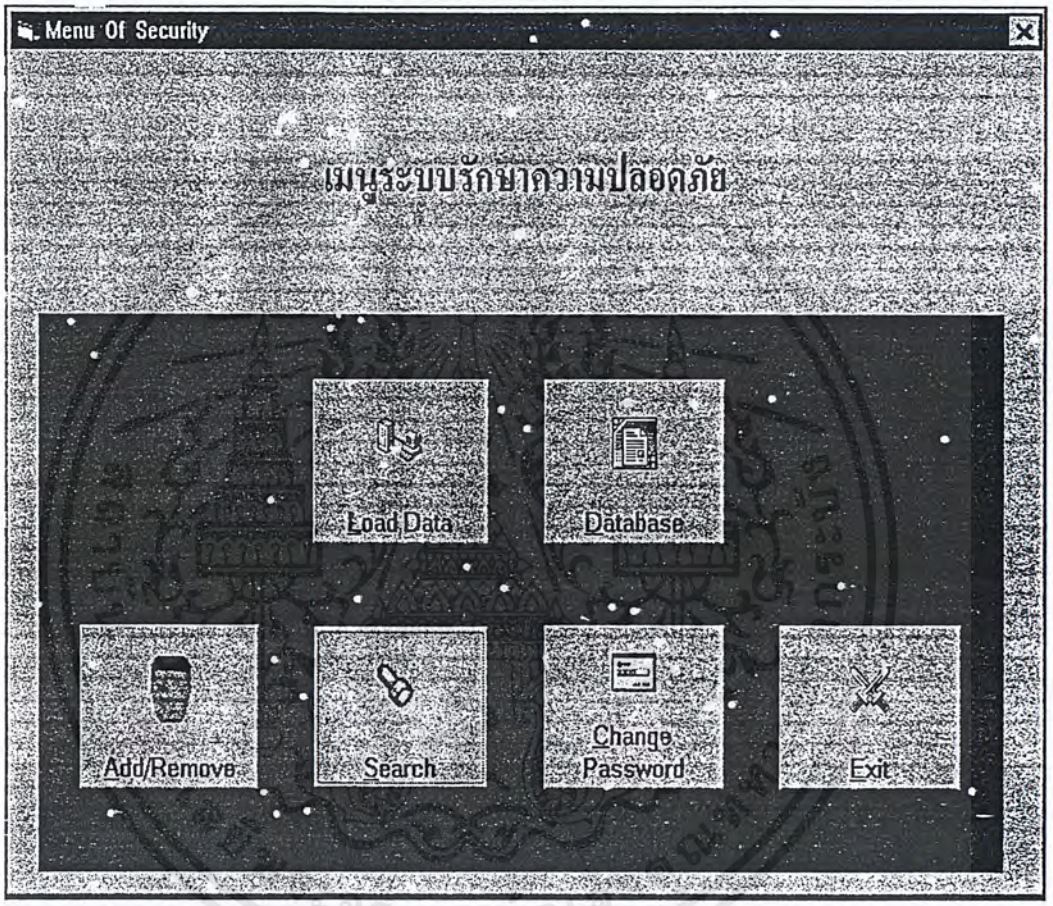
รูปที่ 4-3 แสดงสัญญาณที่พอร์ต P1.6 , P1.7 ขณะมีการเสียบบัตร

การทดสอบการเสียบบัตรเพื่อตรวจสอบความเที่ยงตรงของระบบ โดยพิจรณาความผิดพลาด ต่ำกว่า 1 % ทดสอบทั้งหมด 20 ครั้ง

บัตรใบที่ 1 รหัส 41013200
บัตรใบที่ 1 รหัส 41013180
บัตรใบที่ 1 รหัส 41013192

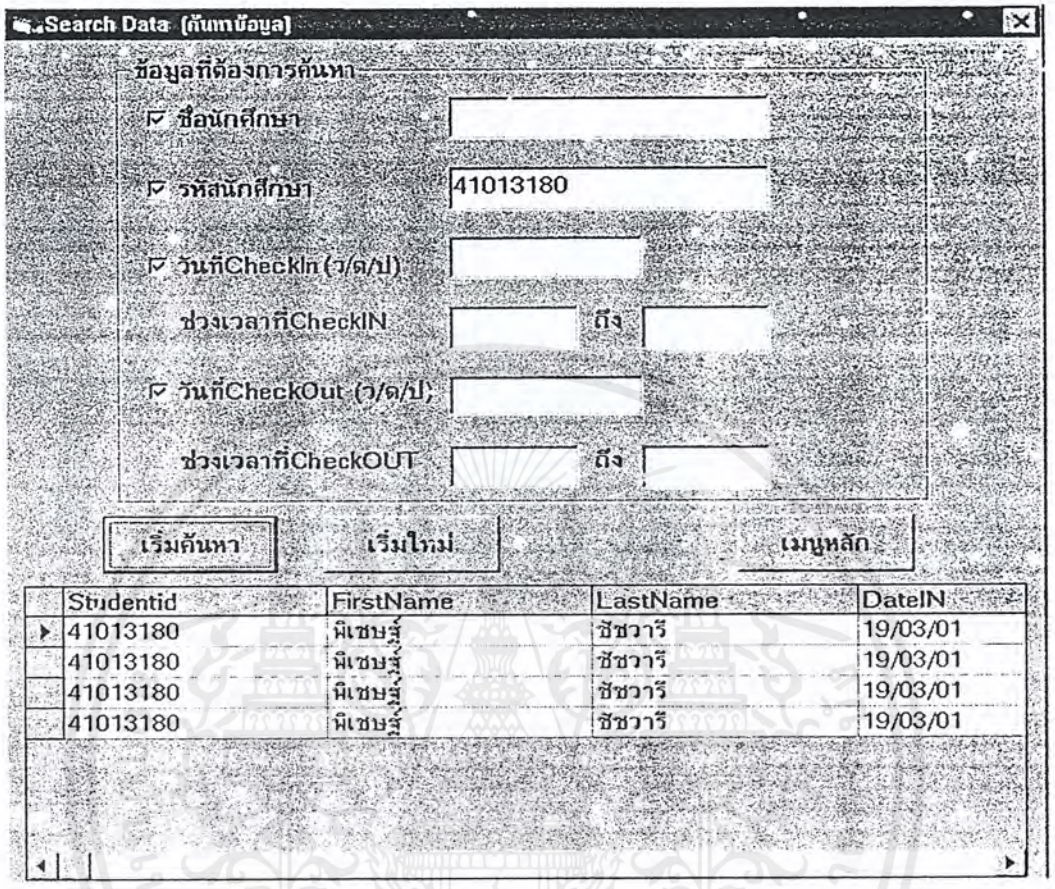
ครั้งที่	รหัสในบัตร	ผลที่ได้จากการทดสอบ
1	41013200	41013200
2	41013180	41013180
3	41013192	41013192
4	41013180	41013180
5	41013192	41013192
6	41013200	41013200
7	41013192	41013192
8	41013200	41013200
9	41013180	41013180
10	41013200	41013200
11	41013180	41013180
12	41013192	41013192
13	41013180	41013180
14	41013192	41013192
15	41013200	41013200
16	41013192	41013192
17	41013200	41013200
18	41013180	41013180
19	41013200	41013200
20	41013192	41013192

3. เมื่อต้องการขอข้อมูลจากไมโครคอนโทรลเลอร์ก็สามารถทำได้โดยการเปิดเมนูหลักของโปรแกรมรักษาความปลอดภัย จากนั้นทำการโหลดข้อมูลโดยคลิกที่ปุ่มโหลดข้อมูล (LOAD DATA) แสดงดังรูปที่ 4-4



รูปที่ 4-4 แสดงหน้าจอของเมนูหลัก

4. เมื่อกดปุ่มโหลดข้อมูลเสร็จแล้วโปรแกรมจะทำการดึงข้อมูลจากไมโคคอนโทรลเลอร์โดย ผ่านพอร์ตสื่อสารอนุกรม RS-232 เมื่อโปรแกรมทำการโหลดข้อมูลเสร็จ ที่หน้าจอคอมพิวเตอร์จะขึ้นหน้าต่าง Down Load Complete
5. ถ้าต้องการดูข้อมูลสามารถทำได้โดยการคลิกที่ปุ่ม Search พิจารณาได้ดังรูปที่ 4-5



รูปที่ 4-5 แสดงหน้าจอของการ Search ข้อมูล

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

ปริญญานิพนธ์ฉบับนี้เสนอผลงานเกี่ยวกับระบบป้องกันความปลอดภัยพื้นฐาน ซึ่งใช้ EEPROM เป็นตัวเก็บข้อมูลที่เป็นรหัส 8 หลัก และใช้ไมโครคอนโทรลเลอร์เป็นตัวอ่านข้อมูลและทำการเปรียบเทียบกับฐานข้อมูลภายในหน่วยความจำของไมโครคอนโทรลเลอร์ ถ้าข้อมูลตรงกันก็จะนำข้อมูลนั้นไปเก็บไว้ในหน่วยความจำ และ ไมโครคอนโทรลเลอร์จะควบคุมตัวล็อคไฟฟ้าให้ปลดล็อคชั่วคราว โดยสรุปแล้วผลการทดลองที่ทำการวัดสัญญาณต่าง ๆ ดังในบทที่ 4 มีความใกล้เคียงกับทฤษฎีที่กล่าวไว้ในบทที่ 2 และผู้จัดทำโครงการนี้คิดว่า ประสิทธิภาพของเครื่องในส่วนของการอ่านข้อมูลและทำการประมวลผลเป็นที่น่าพอใจและสามารถนำไปใช้งานได้จริง

5.2 ปัญหาและแนวทางแก้ไข

1. ระบบยังมีความช้าอยู่ในช่วงการรับข้อมูลมาเปรียบเทียบทำการแสดงผล ปัญหาในข้อนี้อาจแก้ไขได้โดยใช้การอินเตอร์รัพท์เข้ามาช่วยในการเขียนโปรแกรม
2. ในระหว่างการส่งข้อมูลให้กับคอมพิวเตอร์ ในบางครั้งของการส่งข้อมูลจะเกิดความผิด พลาดของข้อมูลปัญหาในข้อนี้อาจแก้ไขได้โดยใช้วิธีการแก้ไขเหมือนกับในข้อที่1 ก็คือการใช้ใช้การอินเตอร์รัพท์เข้ามาช่วยในการเขียนโปรแกรม

5.3 การพัฒนาโครงการ

โครงการนี้ทำขึ้นมานี้สามารถทำขึ้นมานี้สามารถทำงานได้ตามขีดความสามารถที่กำหนดไว้ แต่สามารถเพิ่มขีดความสามารถในการทำงานได้อีก ดังต่อไปนี้

1. ในโครงการที่ทำขึ้นมานี้ ได้ทำการเก็บข้อมูลของแต่ละบุคคลไว้จำนวนไม่มากนัก ในส่วนของข้อมูลนี้เราสามารถเพิ่มจำนวนชุดได้โดยการขยายหน่วยความจำให้กับไมโครคอนโทรลเลอร์
2. การพัฒนาระบบไปควบคุมการ เปิด-ปิดประตูในกรณีที่มีหลายประตู สามารถทำได้โดยการขยายพอร์ตในส่วนที่เป็นอินพุทและเอาท์พุทพอร์ตให้กับไมโครคอนโทรลเลอร์

3. ในโครงการที่ทำขึ้นมานี้การติดต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวแตอร์สามารถทำได้แค่เพียงการขอโหลดข้อมูลจากไมโครคอนโทรลเลอร์มายังคอมพิวเตอร์เท่านั้นเอง เราสามารถพัฒนาระบบเพื่อสามารถสั่งงานระบบโดยผ่านคอมพิวเตอร์ได้ อาทิเช่น สั่งตั่งวันและเวลาที่หน้าจอ LCD , สั่งเปลี่ยนแปลงข้อมูลในบัตร เราสามารถพัฒนาได้โดยการพัฒนาที่ Software

ภาคผนวก

โปรแกรมการควบคุม

```
DRIVER          BIT     P1.0
SPEAKER         BIT     P1.1
RED_LED         BIT     P1.2
GREEN_LED       BIT     P1.3
SCL_1           BIT     P1.4
SDA_1           BIT     P1.5
SCL             BIT     P1.6
SDA             BIT     P1.7

KPAD_ROW0       BIT     P2.0
KPAD_ROW1       BIT     P2.1
KPAD_ROW2       BIT     P2.2
KPAD_ROW3       BIT     P2.3
KPAD_COL2       BIT     P2.4
KPAD_COL1       BIT     P2.5
KPAD_COL0       BIT     P2.6
SET_DATE_SW     BIT     P3.2
SET_TIME_SW     BIT     P3.3
SET_EEP_SW      BIT     P3.4
LCD_EN          BIT     P3.6
LCD_RS          BIT     P3.7

CONT_BYTE_R_1   EQU     0A3H
CONT_BYTE_W_1   EQU     0A2H
CONT_BYTE_R     EQU     0A1H
CONT_BYTE_W     EQU     0A0H
LCD_ADDR        EQU     30H
LCD_DATA        EQU     31H
I2C_ADDR        EQU     32H
I2C_DATA        EQU     33H
BUFF_1          EQU     34H
KPAD_DATA       EQU     35H
```

```
BUFFER          EQU     36H
BUFF            EQU     60H

SECONDS         EQU     40H
MINUTES         EQU     41H
HOURS           EQU     42H
DAY             EQU     43H
DATE            EQU     44H
MONTH           EQU     45H
YEAR            EQU     46H
CONTROL         EQU     47H

DATE_1          EQU     48H
DATE_2          EQU     49H
MONTH_1         EQU     4BH
MONTH_2         EQU     4CH
YEAR_1          EQU     4EH
YEAR_2          EQU     4FH
HOURS_1         EQU     50H
HOURS_2         EQU     51H
MINUTES_1       EQU     53H
MINUTES_2       EQU     54H

HIGH            EQU     55H
LOW             EQU     56H

FLAG            EQU     02FH
I2C_ACK         BIT     FLAG.0
RTC_ID          EQU     0D0H
```

```
                ORG     0000H
                LJMP    MAIN

                ORG     0023H
                PUSH    P1
                PUSH    DPH
                PUSH    DPL
                PUSH    ACC
                PUSH    PSW
                PUSH    02H
                CLR     RI
                MOV     A,SBUF
                CJNE    A,#'A',STOP
                CLR     EA
                CLR     ES
SEND_DATA:      MOV     DPH,HIGH
                MOV     DPL,LOW
                CLR     SDA
                CLR     SCL
                MOV     A,#CONT_BYTE_W_1
                LCALL   LOOP_BYTE
                SETB    SDA
                SETB    SCL
                JB      SDA,SEND_DATA
                CLR     SCL

                MOV     A,DPH
                LCALL   LOOP_BYTE
                SETB    SDA
                SETB    SCL
                JB      SDA,SEND_DATA
                CLR     SCL

                MOV     A,DPL
```

```
        LCALL   LOOP_BYTE
        SETB    SDA
        SETB    SCL
        JB      SDA,SEND_DATA
        CLR     SCL
        SETB    SCL
        SETB    SDA
        CLR     SDA
        CLR     SCL
        MOV     A,#CONT_BYTE_R_1
        LCALL   LOOP_BYTE
        SETB    SDA
        SETB    SCL
        JB      SDA,SEND_DATA
        CLR     SCL
        LCALL   LOOP_READ
        SETB    SDA
        SETB    SCL
        CLR     SCL
        SETB    SCL
        SETB    SDA
        MOV     A,R1
        CJNE    A,#00H,TX_BYTE
        CLR     RI
        SETB    EA
        SETB    ES
        SETB    SCON.4
        POP     02H
        POP     PSW
        POP     ACC
        POP     DPL
        POP     DPH
        POP     P1
STOP:   RETI
```

```
TX_BYTE:      LCALL  TX_BYTE_1
              INC    DPTR
              MOV    HIGH,DPH
              MOV    LOW,DPL
              LJMP   SEND_DATA

TX_BYTE_1:    CLR    TI
              MOV    SBUF,A
              JNB    TI,$
              CLR    TI
              RET

LOOP_READ:    MOV    R2,#08H
LOOP_READ1:   SETB   SCL
              MOV    C,SDA
              CLR    SCL
              RLC    A
              DJNZ   R2,LOOP_READ1
              MOV    R1,A
              RET

LOOP_BYTE:    MOV    R2,#08H
LOOP_SEND:    RLC    A
              MOV    SDA,C
              SETB   SCL
              CLR    SCL
              DJNZ   R2,LOOP_SEND
              RET

MAIN:         MOV    P0,#00000000B
              MOV    P2,#11111111B
              CLR    DRIVER
              CLR    SPEAKER
```

```
        LCALL   INT_LCD
        MOV     R3,#00H
        MOV     R4,#00H
        MOV     HIGH,#00H
        MOV     LOW,#00H
        MOV     TMOD,#00100000B         ;time1 mode2
        MOV     SCON,#01010000B         ;mode1 serial port
        MOV     TH1,#0FDH
        MOV     A,#00H
        MOV     PCON,A                  ;SMOD = 0
        CLR     ET1                     ;clear timer1 interrupt
        SETB    TR1                     ;start timer1
        SETB    ES
        SETB    EA

LOOP:   CLR     RED_LED
        LCALL   RTC_RD
        MOV     LCD_ADDR,#00H
        LCALL   SET_ADDR_LCD
        MOV     LCD_DATA,DATE
        LCALL   BCD2LCD
        MOV     LCD_DATA,#'/'
        LCALL   WRCHAR_LCD
        MOV     LCD_DATA,MONTH
        LCALL   BCD2LCD
        MOV     LCD_DATA,#'/'
        LCALL   WRCHAR_LCD
        MOV     LCD_DATA,YEAR
        LCALL   BCD2LCD
        MOV     LCD_DATA,#' '
        LCALL   WRCHAR_LCD
        MOV     LCD_DATA,#' '
        LCALL   WRCHAR_LCD
        MOV     A,HOURS
```

```
                ANL     A,#00110000B
                JZ      WR_TIME_HN
                SWAP    A
                ADD     A,#030H
                AJMP    WR_TIME_HH
WR_TIME_HN:     MOV     A,#' '
WR_TIME_HH:     MOV     LCD_DATA,A
                LCALL   WRCHAR_LCD
                MOV     A,HOURS
                ANL     A,#00001111B
                ADD     A,#030H
                MOV     LCD_DATA,A
                LCALL   WRCHAR_LCD
                MOV     A,SECONDS
                ANL     A,#001H
                JNZ     WR_SPACE
                MOV     LCD_DATA,#':'
                LCALL   WRCHAR_LCD
                AJMP    WR_MINUTES
WR_SPACE:       MOV     LCD_DATA,#' '
                LCALL   WRCHAR_LCD
WR_MINUTES:     MOV     LCD_DATA,MINUTES
                LCALL   BCD2LCD
                MOV     LCD_DATA,#' '
                LCALL   WRCHAR_LCD
                JNB     SET_DATE_SW,SET_DATE_1
                JNB     SET_TIME_SW,SET_TIME_1
                JNB     SET_EEP_SW,SET_EEP_1
                LCALL   KOON
                LJMP    EEP_RD
SET_DATE_1:     LJMP    SET_DATE
SET_TIME_1:     LJMP    SET_TIME
SET_EEP_1:      LJMP    SET_EEP
```

```
EEP_RD:          NOP
EEP_RD_IN:       MOV   DPTR,#0000H
                 MOV   R0,#10H
                 MOV   R1,#BUFF
READ_BYTE_IN:    CLR   SDA                    ;START BIT
                 CLR   SCL
                 MOV   A,#CONT_BYTE_W         ;SEND CONTROL BYTE
                 LCALL LOOP_BYTE_IN
                 SETB  SDA
                 SETB  SCL
                 JB    SDA,LOOP1              ;LOOP UNTIL BUSY
                 CLR   SCL

                 MOV   A,DPL                  ;SEND ADDRESS LOW
                 LCALL LOOP_BYTE_IN
                 SETB  SDA
                 SETB  SCL
                 JB    SDA,LOOP1              ;LOOP UNTIL BUSY
                 CLR   SCL
                 SETB  SCL
                 SETB  SDA

                 CLR   SDA                    ;START BIT
                 CLR   SCL
                 MOV   A,#CONT_BYTE_R         ;SEND CONTROL BYTE
                 LCALL LOOP_BYTE_IN
                 SETB  SDA
                 SETB  SCL
                 JB    SDA,LOOP1              ;LOOP UNTIL BUSY
                 CLR   SCL
                 LCALL LOOP_READ_IN
                 SETB  SDA
                 SETB  SCL
                 CLR   SCL
```

```
                SETB    SCL                     ;STOP BIT
                SETB    SDA
                INC     R1
                INC     DPTR
                DJNZ    R0,READ_BYTE_IN
                MOV     DPTR,#CODE_TEST
                LCALL   TEST
                CJNE    R0,#08,LOOP1
                LCALL   STOR_DATA_IN
                LCALL   SHW_CODE
LOOP1:          LJMP    EEP_RD_OUT
;*************************************************************
EEP_RD_OUT:     MOV     DPTR,#0000H
                MOV     R0,#10H
                MOV     R1,#BUFF
READ_BYTE_OUT:CLR       SDA_1                   ;START BIT
                CLR     SCL_1
                MOV     A,#CONT_BYTE_W          ;SEND CONTROL BYTE
                LCALL   LOOP_BYTE_OUT
                SETB    SDA_1
                SETB    SCL_1
                JB      SDA_1,LOOP2             ;LOOP UNTIL BUSY
                CLR     SCL_1

                MOV     A,DPL                   ;SEND ADDRESS LOW
                LCALL   LOOP_BYTE_OUT
                SETB    SDA_1
                SETB    SCL_1
                JB      SDA_1,LOOP2             ;LOOP UNTIL BUSY
                CLR     SCL_1
                SETB    SCL_1
                SETB    SDA_1
                CLR     SDA_1                   ;START BIT
                CLR     SCL_1
```

```
                MOV     A,#CONT_BYTE_R      ;SEND CONTROL BYTE
                LCALL   LOOP_BYTE_OUT
                SETB    SDA_1
                SETB    SCL_1
                JB      SDA_1,LOOP2         ;LOOP UNTIL BUSY
                CLR     SCL_1
                LCALL   LOOP_READ_OUT
                SETB    SDA_1
                SETB    SCL_1
                CLR     SCL_1
                SETB    SCL_1               ;STOP BIT
                SETB    SDA_1
                INC     R1
                INC     DPTR
                DJNZ    R0,READ_BYTE_OUT
                MOV     DPTR,#CODE_TEST
                LCALL   TEST
                CJNE    R0,#08,LOOP1
                LCALL   STOR_DATA_OUT
                LCALL   SHW_CODE
LOOP2:          LJMP    LOOP
;*************************************************************
KOON:           MOV     4AH,#'/'
                MOV     4DH,#'/'
                MOV     52H,#'.'
                MOV     A,DATE
                MOV     B,A
                ANL     A,#11110000B
                SWAP    A
                ADD     A,#030H
                MOV     DATE_1,A
                MOV     A,B
                ANL     A,#00001111B
                ADD     A,#030H
```

```
MOV     DATE_2,A

MOV     A,MONTH
MOV     B,A
ANL     A,#11110000B
SWAP    A
ADD     A,#030H
MOV     MONTH_1,A
MOV     A,B
ANL     A,#00001111B
ADD     A,#030H
MOV     MONTH_2,A

MOV     A,YEAR
MOV     B,A
ANL     A,#11110000B
SWAP    A
ADD     A,#030H
MOV     YEAR_1,A
MOV     A,B
ANL     A,#00001111B
ADD     A,#030H
MOV     YEAR_2,A

MOV     A,HOURS
MOV     B,A
ANL     A,#11110000B
SWAP    A
ADD     A,#030H
MOV     HOURS_1,A
MOV     A,B
ANL     A,#00001111B
ADD     A,#030H
MOV     HOURS_2,A
```

```
                MOV     A,MINUTES
                MOV     B,A
                ANL     A,#11110000B
                SWAP    A
                ADD     A,#030H
                MOV     MINUTES_1,A
                MOV     A,B
                ANL     A,#00001111B
                ADD     A,#030H
                MOV     MINUTES_2,A
                RET
;*************************************************************
SET_EEP:        LCALL   WAIT_KEY_1
                LCALL   PASSWORD
                MOV     LCD_ADDR,#00H
                LCALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_EEP
                LCALL   WRLINE_LCD

                MOV     LCD_ADDR,#40H
                LCALL   SET_ADDR_LCD
                LCALL   LCD_BLINK

                MOV     R0,#BUFF
                MOV     R1,#0
UN:             LCALL   WAIT_KEYPRESS
                MOV     BUFFER,KPAD_DATA
                MOV     A,KPAD_DATA
                ADD     A,#30H
                MOV     LCD_DATA,A
                MOV     @R0,A
                LCALL   WRCHAR_LCD
                LCALL   LCD_BLINK
```

```
                LCALL   WAIT_KEY
                INC     R0
                INC     R1
                CJNE    R1,#08,UN

                LCALL   EEP_WR
                LCALL   CLR_SCREEN_1
                LCALL   CLR_SCREEN_2
                MOV     LCD_DATA,#40H
                LCALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_OK
                LCALL   WRLINE_LCD
                LCALL   DELAY_1S
                LCALL   CLR_SCREEN_2
                LJMP    LOOP
;*************************************************************
SET_DATE:       LCALL   WAIT_KEY_1
                LCALL   PASSWORD
                MOV     LCD_ADDR,#00H
                LCALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_DATE
                LCALL   WRLINE_LCD
                MOV     LCD_ADDR,#40H
                LCALL   SET_ADDR_LCD
                LCALL   LCD_BLINK

                LCALL   WAIT_KEYPRESS
                MOV     BUFFER,KPAD_DATA
                MOV     A,KPAD_DATA
                ADD     A,#30H
                MOV     LCD_DATA,A
                LCALL   WRCHAR_LCD
                LCALL   LCD_BLINK
                LCALL   WAIT_KEY
```

```
LCALL   WAIT_KEYPRESS
MOV     BUFFER+1,KPAD_DATA
MOV     A,KPAD_DATA
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   WAIT_KEY

MOV     A,BUFFER
ANL     A,#0FH
SWAP    A
MOV     B,A
MOV     A,BUFFER+1
ANL     A,#0FH
ADD     A,B
MOV     DATE,A

MOV     LCD_ADDR,#43H
LCALL   SET_ADDR_LCD
LCALL   LCD_BLINK

LCALL   WAIT_KEYPRESS
MOV     BUFFER,KPAD_DATA
MOV     A,KPAD_DATA
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   LCD_BLINK
LCALL   WAIT_KEY

LCALL   WAIT_KEYPRESS
MOV     BUFFER+1,KPAD_DATA
MOV     A,KPAD_DATA
```

```
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   WAIT_KEY

MOV     A,BUFFER
ANL     A,#0FH
SWAP    A
MOV     B,A
MOV     A,BUFFER+1
ANL     A,#0FH
ADD     A,B
MOV     MONTH,A

MOV     LCD_ADDR,#46H
LCALL   SET_ADDR_LCD
LCALL   LCD_BLINK

LCALL   WAIT_KEYPRESS
MOV     BUFFER,KPAD_DATA
MOV     A,KPAD_DATA
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   LCD_BLINK
LCALL   WAIT_KEY

LCALL   WAIT_KEYPRESS
MOV     BUFFER+1,KPAD_DATA
MOV     A,KPAD_DATA
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   WAIT_KEY
```

```
                MOV     A,BUFFER
                ANL     A,#0FH
                SWAP    A
                MOV     B,A
                MOV     A,BUFFER+1
                ANL     A,#0FH
                ADD     A,B
                MOV     YEAR,A

                LCALL   RTC_WR
                LCALL   CLR_SCREEN_1
                LCALL   CLR_SCREEN_2
                LJMP    LOOP
;*************************************************************
SET_TIME:       LCALL   WAIT_KEY_1
                LCALL   PASSWORD
                MOV     LCD_ADDR,#00H
                LCALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_TIME
                LCALL   WRLINE_LCD

                MOV     LCD_ADDR,#40H
                LCALL   SET_ADDR_LCD
                LCALL   LCD_BLINK

                LCALL   WAIT_KEYPRESS
                MOV     BUFFER,KPAD_DATA
                MOV     A,KPAD_DATA
                ADD     A,#30H
                MOV     LCD_DATA,A
                LCALL   WRCHAR_LCD
                LCALL   LCD_BLINK
                LCALL   WAIT_KEY
```

```
LCALL  WAIT_KEYPRESS
MOV    BUFFER+1,KPAD_DATA
MOV    A,KPAD_DATA
ADD    A,#30H
MOV    LCD_DATA,A
LCALL  WRCHAR_LCD
LCALL  WAIT_KEY

MOV    A,BUFFER
ANL    A,#0FH
SWAP   A
MOV    B,A
MOV    A,BUFFER+1
ANL    A,#0FH
ADD    A,B
MOV    HOURS,A

MOV    LCD_ADDR,#43H
LCALL  SET_ADDR_LCD
LCALL  LCD_BLINK

LCALL  WAIT_KEYPRESS
MOV    BUFFER,KPAD_DATA
MOV    A,KPAD_DATA
ADD    A,#30H
MOV    LCD_DATA,A
LCALL  WRCHAR_LCD
LCALL  LCD_BLINK
LCALL  WAIT_KEY

LCALL  WAIT_KEYPRESS
MOV    BUFFER+1,KPAD_DATA
MOV    A,KPAD_DATA
```

```
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   WAIT_KEY

MOV     A,BUFFER
ANL     A,#0FH
SWAP    A
MOV     B,A
MOV     A,BUFFER+1
ANL     A,#0FH
ADD     A,B
MOV     MINUTES,A

MOV     LCD_ADDR,#46H
LCALL   SET_ADDR_LCD
LCALL   LCD_BLINK

LCALL   WAIT_KEYPRESS
MOV     BUFFER,KPAD_DATA
MOV     A,KPAD_DATA
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   LCD_BLINK
LCALL   WAIT_KEY

LCALL   WAIT_KEYPRESS
MOV     BUFFER+1,KPAD_DATA
MOV     A,KPAD_DATA
ADD     A,#30H
MOV     LCD_DATA,A
LCALL   WRCHAR_LCD
LCALL   WAIT_KEY
```

```
                MOV     A,BUFFER
                ANL     A,#0FH
                SWAP    A
                MOV     B,A
                MOV     A,BUFFER+1
                ANL     A,#0FH
                ADD     A,B
                MOV     SECONDS,A

                LCALL   RTC_WR
                LCALL   CLR_SCREEN_1
                LCALL   CLR_SCREEN_2
                LJMP    LOOP
;*************************************************************
STOR_DATA_IN:   MOV     BUFF+7,#'I'
                LCALL   STOR_DATA
                RET

STOR_DATA_OUT:  MOV     BUFF+7,#'O'
                LCALL   STOR_DATA
                RET

STOR_DATA:      MOV     R0,#BUFF+7
                MOV     R1,#0
STOR_DATA_1:    MOV     BUFF_1,@R0
                LCALL   STORED_DATA
                INC     R0
                INC     R1
                CJNE    R1,#9,STOR_DATA_1
                LCALL   STOR_TIME
                RET

STORED_DATA:    MOV     DPH,R3
```

```
                MOV     DPL,R4
                MOV     I2C_ADDR,#CONT_BYTE_W_1
                LCALL   I2C_SLAVE
                JB      I2C_ACK,STORED_DATA
                MOV     I2C_DATA,DPH
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,STORED_DATA
                MOV     I2C_DATA,DPL
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,STORED_DATA
                MOV     I2C_DATA,BUFF_1
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,STORED_DATA
                INC     DPTR
                MOV     R3,DPH
                MOV     R4,DPL
                LCALL   I2C_STOP
                RET

STOR_TIME:      MOV     R0,#DATE_1
                MOV     R1,#0
STOR_TIME_1:    MOV     BUFF_1,@R0
                LCALL   STORED_TIME
                INC     R0
                INC     R1
                CJNE    R1,#0DH,STOR_TIME_1
                LCALL   WR_F
                RET

STORED_TIME:    MOV     DPH,R3
                MOV     DPL,R4
                MOV     I2C_ADDR,#CONT_BYTE_W_1
                LCALL   I2C_SLAVE
                JB      I2C_ACK,STORED_TIME
```

```
                MOV     I2C_DATA,DPH
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,STORED_TIME
                MOV     I2C_DATA,DPL
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,STORED_TIME
                MOV     I2C_DATA,BUFF_1
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,STORED_TIME
                INC     DPTR
                MOV     R3,DPH
                MOV     R4,DPL
                LCALL   I2C_STOP
                RET

WR_F:           MOV     DPH,R3
                MOV     DPL,R4
                MOV     I2C_ADDR,#CONT_BYTE_W_1
                LCALL   I2C_SLAVE
                JB      I2C_ACK,WR_F
                MOV     I2C_DATA,DPH
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,WR_F
                MOV     I2C_DATA,DPL
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,WR_F
                MOV     I2C_DATA,#00H
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,WR_F
                LCALL   I2C_STOP
                RET
;***************************************************************
PASSWORD:       MOV     LCD_DATA,#00H
                LCALL   SET_ADDR_LCD
```

```
                MOV     DPTR,#TITLE_PASSWORD
                LCALL   WRLINE_LCD

                MOV     LCD_ADDR,#40H
                LCALL   SET_ADDR_LCD
                LCALL   LCD_BLINK

                MOV     R0,#BUFF
                MOV     R1,#0
PASSWORD_1:     LCALL   WAIT_KEYPRESS
                MOV     BUFFER,KPAD_DATA
                MOV     A,KPAD_DATA
                ADD     A,#30H
                MOV     LCD_DATA,#'*'
                MOV     @R0,A
                LCALL   WRCHAR_LCD
                LCALL   LCD_BLINK
                LCALL   WAIT_KEY
                INC     R0
                INC     R1
                CJNE    R1,#08,PASSWORD_1

                MOV     DPTR,#TEST_PASS
                LCALL   TEST
                CJNE    R0,#08,PASSWORD_2
                LCALL   CLR_SCREEN_2
                RET
PASSWORD_2:     LCALL   CLR_SCREEN_1
                LCALL   CLR_SCREEN_2
                MOV     LCD_DATA,#40H
                LCALL   SET_ADDR_LCD
                MOV     DPTR,#EER_PASSWORD
                LCALL   WRLINE_LCD
                LCALL   DELAY_1S
```

```
                LCALL   CLR_SCREEN_2
                LJMP    LOOP
                RET
;*************************************************************
TEST:           MOV     R1,#BUFF
                MOV     R0,#0
TEST_1:         CLR     A
                MOVC    A,@A+DPTR
                MOV     BUFF_1,@R1
                CJNE    A,BUFF_1,TEST_2
                INC     DPTR
                INC     R1
                INC     R0
                CJNE    R0,#08,TEST_1
TEST_2:         RET
;*************************************************************
SHW_CODE:       MOV     DPTR,#TITLE_CODE
                MOV     LCD_ADDR,#40H
                LCALL   SET_ADDR_LCD
                LCALL   WRLINE_LCD
                MOV     LCD_ADDR,#46H
                LCALL   SET_ADDR_LCD

                MOV     R0,#BUFF+8
                MOV     R1,#0
SHW_CODE_1:     MOV     LCD_DATA,@R0
                LCALL   WRCHAR_LCD
                INC     R0
                INC     R1
                CJNE    R1,#08,SHW_CODE_1
                CLR     GREEN_LED
                SETB    RED_LED
                SETB    DRIVER
                LCALL   SOUND
```

```
                CLR     DRIVER
                LCALL   CLR_SCREEN_2
                SETB    GREEN_LED
                RET
;*********************************************************************
CLR_SCREEN_1:MOV    DPTR,#CLR_CODE
                MOV     LCD_ADDR,#00H
                LCALL   SET_ADDR_LCD
                LCALL   WRLINE_LCD
                RET

CLR_SCREEN_2:MOV    DPTR,#CLR_CODE
                MOV     LCD_ADDR,#40H
                LCALL   SET_ADDR_LCD
                LCALL   WRLINE_LCD
                RET
;*********************************************************************
LOOP_READ_IN:  MOV    R2,#08H
LOOP_READ1_IN: SETB   SCL
                MOV     C,SDA
                CLR     SCL
                RLC     A
                DJNZ    R2,LOOP_READ1_IN
                MOV     @R1,A
                RET

LOOP_BYTE_IN:  MOV    R2,#08H
LOOP_SEND_IN: RLC     A
                MOV     SDA,C
                SETB    SCL
                CLR     SCL
                DJNZ    R2,LOOP_SEND_IN
                RET
;*********************************************************************
```

```
LOOP_READ_OUT:  MOV    R2,#08H
LOOP_READ1_OUT:SETB   SCL_1
                MOV    C,SDA_1
                CLR    SCL_1
                RLC    A
                DJNZ   R2,LOOP_READ1_OUT
                MOV    @R1,A
                RET

LOOP_BYTE_OUT:  MOV    R2,#08H
LOOP_SEND_OUT:  RLC    A
                MOV    SDA_1,C
                SETB   SCL_1
                CLR    SCL_1
                DJNZ   R2,LOOP_SEND_OUT
                RET
;*******************************************************************************
EEP_WR:         MOV    I2C_ADDR,#CONT_BYTE_W
                LCALL  I2C_SLAVE
                JB     I2C_ACK,EEP_WR

                MOV    I2C_DATA,#000H
                LCALL  I2C_DATA_WR
                JB     I2C_ACK,EEP_WR
                MOV    I2C_DATA,#'S'
                LCALL  I2C_DATA_WR
                JB     I2C_ACK,EEP_WR
                MOV    I2C_DATA,#'U'
                LCALL  I2C_DATA_WR
                JB     I2C_ACK,EEP_WR
                MOV    I2C_DATA,#'R'
                LCALL  I2C_DATA_WR
                JB     I2C_ACK,EEP_WR
                MOV    I2C_DATA,#'I'
```

```
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,EEP_WR
                MOV     I2C_DATA,#'Y'
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,EEP_WR
                MOV     I2C_DATA,#'A'
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,EEP_WR
                MOV     I2C_DATA,#'0'
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,EEP_WR
                MOV     I2C_DATA,#'1'
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,EEP_WR
                LCALL   I2C_STOP

EEP_WR_1:       MOV     R0,#BUFF
                MOV     R1,#0
                MOV     I2C_ADDR,#CONT_BYTE_W
                LCALL   I2C_SLAVE
                JB      I2C_ACK,EEP_WR_1
                MOV     I2C_DATA,#08H
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,EEP_WR_1
EEP_WR_2:       MOV     I2C_DATA,@R0
                LCALL   I2C_DATA_WR
                JB      I2C_ACK,EEP_WR_1
                INC     R0
                INC     R1
                CJNE    R1,#08,EEP_WR_2
                LCALL   I2C_STOP
                RET
;*******************************************************************************
RTC_WR:         MOV     I2C_ADDR,#RTC_ID
```

```
                LCALL   I2C_SLAVE
                MOV     I2C_DATA,#000H
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,#SECONDS
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,MINUTES
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,HOURS
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,DAY
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,DATE
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,MONTH
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,YEAR
                LCALL   I2C_DATA_WR
                MOV     I2C_DATA,#10010011B
                LCALL   I2C_DATA_WR
                LCALL   I2C_STOP
                RET
;*********************************************************************
RTC_RD:         MOV     I2C_ADDR,#RTC_ID
                LCALL   I2C_SLAVE
                MOV     I2C_DATA,#000H
                LCALL   I2C_DATA_WR
                LCALL   RTC_RD1
                MOV     SECONDS,I2C_DATA
                LCALL   I2C_NACK_BIT
                LCALL   RTC_RD1
                MOV     MINUTES,I2C_DATA
                LCALL   I2C_NACK_BIT
                LCALL   RTC_RD1
                MOV     HOURS,I2C_DATA
```

```
                LCALL   I2C_NACK_BIT
                LCALL   RTC_RD1
                MOV     DAY,I2C_DATA
                LCALL   I2C_NACK_BIT
                LCALL   RTC_RD1
                MOV     DATE,I2C_DATA
                LCALL   I2C_NACK_BIT
                LCALL   RTC_RD1
                MOV     MONTH,I2C_DATA
                LCALL   I2C_NACK_BIT
                LCALL   RTC_RD1
                MOV     YEAR,I2C_DATA
                LCALL   I2C_NACK_BIT
                LCALL   RTC_RD1
                MOV     CONTROL,I2C_DATA
                LCALL   I2C_NACK_BIT
                LCALL   I2C_STOP
                RET

RTC_RD1:        MOV     I2C_ADDR,#RTC_ID+1
                LCALL   I2C_SLAVE
                LCALL   I2C_DATA_RD
                RET

I2C_DATA_WR:    PUSH    ACC
                SETB    I2C_ACK
                MOV     A,I2C_DATA
                MOV     R5,#008

I2C_DATA_WR_1:  RLC     A
                MOV     SDA,C
                LCALL   I2C_CLK
                DJNZ    R5,I2C_DATA_WR_1
                SETB    SDA
```

```
                LCALL   I2C_DELAY
                SETB    SCL
                LCALL   I2C_DELAY
                JB      SDA,I2C_DATA_WR_2
                CLR     I2C_ACK
I2C_DATA_WR_2:  CLR     SCL
                POP     ACC
                RET

I2C_DATA_RD:    PUSH    ACC
                CLR     A
                MOV     R5,#008
I2C_DATA_RD_1:  LCALL   I2C_DELAY
                SETB    SCL
                LCALL   I2C_DELAY
                MOV     C,SDA
                RLC     A
                CLR     SCL
                DJNZ    R5,I2C_DATA_RD_1
                MOV     I2C_DATA,A
                POP     ACC
                RET

I2C_SLAVE:      PUSH    ACC
                SETB    I2C_ACK
                MOV     A,I2C_ADDR
                LCALL   I2C_START
                MOV     R5,#008
I2C_SLAVE_1:    RLC     A
                MOV     SDA,C
                LCALL   I2C_CLK
                DJNZ    R5,I2C_SLAVE_1
                SETB    SDA
                LCALL   I2C_DELAY
```

```
                SETB    SCL
                LCALL   I2C_DELAY
                JB      SDA,I2C_SLAVE_2
                CLR     I2C_ACK
I2C_SLAVE_2:    CLR     SCL
                POP     ACC
                RET

I2C_START:      SETB    SCL
                SETB    SDA
                LCALL   I2C_DELAY
                CLR     SDA
                LCALL   I2C_DELAY
                CLR     SCL
                RET

I2C_STOP:       CLR     SDA
                LCALL   I2C_DELAY
                SETB    SCL
                LCALL   I2C_DELAY
                SETB    SDA
                RET

I2C_CLK:        LCALL   I2C_DELAY
                SETB    SCL
                LCALL   I2C_DELAY
                CLR     SCL
                RET

I2C_NACK_BIT:   SETB    SDA
                LCALL   I2C_DELAY
                LCALL   I2C_CLK
                RET
;************************************************************************
```

```
BCD2LCD:    PUSH   ACC
            PUSH   B
            MOV    A,LCD_DATA
            MOV    B,A
            ANL    A,#11110000B
            SWAP   A
            ADD    A,#030H
            MOV    LCD_DATA,A
            LCALL  WRCHAR_LCD
            MOV    A,B
            ANL    A,#00001111B
            ADD    A,#030H
            MOV    LCD_DATA,A
            LCALL  WRCHAR_LCD
            POP    B
            POP    ACC
            RET
;*********************************************************************
WAIT_KEY:   MOV    A,P2
            ANL    A,#0FH
            SETB   SPEAKER
            MOV    R5,#0FFH
            DJNZ   R5,$
            CLR    SPEAKER
            MOV    R5,#0FFH
            DJNZ   R5,$
            CJNE   A,#0FH,WAIT_KEY
            RET

WAIT_KEY_1: SETB   SPEAKER
            MOV    R5,#0FFH
            DJNZ   R5,$
            CLR    SPEAKER
            MOV    R5,#0FFH
```

```
                DJNZ    R5,$
                JNB     SET_DATE_SW,WAIT_KEY_1
                JNB     SET_TIME_SW,WAIT_KEY_1
                JNB     SET_EEP_SW,WAIT_KEY_1
                RET
;*****************************************************************
WAIT_KEYPRESS:LCALL  GET_KPAD
                MOV     A,KPAD_DATA
                CJNE    A,#0,CHK_KEY_NEXT
                LJMP    WAIT_KEYPRESS
CHK_KEY_NEXT:CJNE    A,#10,CHK_KEY_0
                LJMP    WAIT_KEYPRESS
CHK_KEY_0:      CJNE    A,#11,CHK_VALID_KEY
                MOV     KPAD_DATA,#0
                RET
CHK_VALID_KEY:JNC    WAIT_KEYPRESS
                RET
;*****************************************************************
GET_KPAD:       MOV     P2,#0FFH
                MOV     KPAD_DATA,#0
CHK_COL0:       CLR     KPAD_COL0
                MOV     A,P2
                ANL     A,#0FH
                CJNE    A,#0FH,COL0_DETECT
                LJMP    CHK_COL1
COL0_DETECT:    MOV     KPAD_DATA,#01
                LJMP    GET_ROW
CHK_COL1:       SETB    KPAD_COL0
                CLR     KPAD_COL1
                MOV     A,P2
                ANL     A,#0FH
                CJNE    A,#0FH,COL1_DETECT
                LJMP    CHK_COL2
COL1_DETECT:    MOV     KPAD_DATA,#02
```

```
                LJMP    GET_ROW
CHK_COL2:       SETB    KPAD_COL1
                CLR     KPAD_COL2
                MOV     A,P2
                ANL     A,#0FH
                CJNE    A,#0FH,COL2_DETECT
                RET

COL2_DETECT:    MOV     KPAD_DATA,#03
                LJMP    GET_ROW
                RET

GET_ROW:        CLR     KPAD_COL0
                CLR     KPAD_COL1
                CLR     KPAD_COL2
                JB      KPAD_ROW0,CHK_ROW1
                RET

CHK_ROW1:       JB      KPAD_ROW1,CHK_ROW2
                MOV     A,KPAD_DATA
                ADD     A,#3
                MOV     KPAD_DATA,A
                RET

CHK_ROW2:       JB      KPAD_ROW2,CHK_ROW3
                MOV     A,KPAD_DATA
                ADD     A,#6
                MOV     KPAD_DATA,A
                RET

CHK_ROW3:       MOV     A,KPAD_DATA
                ADD     A,#9
                MOV     KPAD_DATA,A
                RET
```

```
;*****************************************************************
WRCHAR_LCD:  SETB   LCD_RS
             MOV    P0,LCD_DATA
             LCALL  LCD_CLK
             LCALL  LCD_ON
             RET

WRLINE_LCD:  MOV    R0,#0
WRLINE_LCD_1: SETB  LCD_RS
             CLR    A
             MOVC   A,@A+DPTR
             MOV    P0,A
             LCALL  LCD_CLK
             INC    DPTR
             INC    R0
             CJNE   R0,#16,WRLINE_LCD_1
             LCALL  LCD_ON
             RET
;*****************************************************************
INT_LCD:     LCALL  DELAY_100mS
             CLR    LCD_RS
             MOV    P0,#00111000B
             LCALL  LCD_CLK
             LCALL  DELAY_10mS
             LCALL  LCD_OFF
             LCALL  LCD_CLR
             MOV    P0,#00000110B
             LCALL  LCD_CLK
             LCALL  LCD_HOME
             RET
LCD_BLINK:   CLR    LCD_RS
             MOV    P0,#00001111B
             LCALL  LCD_CLK
             RET
```

```
LCD_CLR:        CLR     LCD_RS
                MOV     P0,#00000001B
                LCALL   LCD_CLK
                RET

LCD_HOME:       CLR     LCD_RS
                MOV     P0,#00000010B
                LCALL   LCD_CLK
                RET

LCD_OFF:        CLR     LCD_RS
                MOV     P0,#00001000B
                LCALL   LCD_CLK
                RET

LCD_ON:         CLR     LCD_RS
                MOV     P0,#00001100B
                LCALL   LCD_CLK
                RET

LCD_CLK:        SETB    LCD_EN
                LCALL   LCD_DELAY
                CLR     LCD_EN
                LCALL   LCD_DELAY
                RET

SET_ADDR_LCD:CLR    LCD_RS
                MOV     A,LCD_ADDR
                SETB    ACC.7
                MOV     P0,A
                LCALL   LCD_CLK
                RET
;*********************************************************************
I2C_DELAY:      MOV     R6,#00CH
```

```
I2C_DELAY_1:    NOP
                NOP
                DJNZ    R6,I2C_DELAY_1
                RET

LCD_DELAY:      MOV     R7,#02
LCD_DELAY_1:    MOV     R6,#0E6H
LCD_DELAY_2:    NOP
                NOP
                DJNZ    R6,LCD_DELAY_2
                DJNZ    R7,LCD_DELAY_1
                RET

DELAY_10mS:     MOV     R7,#010
DELAY_10mS_1:   MOV     R6,#0E6H
DELAY_10mS_2:   NOP
                NOP
                DJNZ    R6,DELAY_10mS_2
                DJNZ    R7,DELAY_10mS_1
                RET

DELAY_100mS:    MOV     R7,#100
DELAY_100mS_1:  MOV     R6,#0E6H
DELAY_100mS_2:  NOP
                NOP
                DJNZ    R6,DELAY_100mS_2
                DJNZ    R7,DELAY_100mS_1
                RET

DELAY_1S:       MOV     R5,#100
DELAY_1S_1:     LCALL   DELAY_100mS
                DJNZ    R5,DELAY_1S_1
                RET
;*********************************************************************
```

```
SOUND:          MOV     R7,#30
SOUND_1:        MOV     R6,#0E6H
SOUND_2:        SETB    SPEAKER
                MOV     R5,#0FFH
SOUND_3:        DJNZ    R5,SOUND_3
                CLR     SPEAKER
                MOV     R5,#0FFH
SOUND_4:        DJNZ    R5,SOUND_4
                DJNZ    R6,SOUND_2
                DJNZ    R7,SOUND_1
                RET
;*********************************************************************
CODE_TEST:              DB      'SURIYA01'
TITLE_DATE:             DB      'DD/MM/YY       '
TITLE_TIME:             DB      'HH/MM/SS       '
TITLE_CODE:             DB      'CODE:          '
TITLE_EEP:              DB      'SETCARD&PRESSKEY'
TITLE_OK:               DB      '     OK        '
TITLE_PASSWORD:         DB      ' ENTER PASSWORD '
TEST_PASS:              DB      '54621397'
EER_PASSWORD:           DB      ' PASSWORD EEROR '
CLR_CODE:               DB      '               '
                END
```

โปรแกรมของ ChangePasswordForm

```
Private Sub CancelComm_Click()
        Unload ChangePassForm
        MenuForm.Show
End Sub

Private Sub Form_Activate()
        NewPasswordText.Text = " "
        NewPasswordText.SetFocus
End Sub

Private Sub OKComm_Click()
        If  NewPasswordText.Text = " " Then
             NewPasswordText.Text = " "
             AttextPassText.Text = " "
             NewPasswordText.SetFocus

        ElseIf AttextPassText = " " Then
              NewPasswordText.Text = " "
             AttextPassText.Text = " "
             NewPasswordText.SetFocus

       ElseIf  NewPasswordText.Text = AttextPassText.Text  Then
             Data1.Recordset.Edit
             Data1.Recordset.Update
             Unload ChangePassForm
          If MsgBox("คุณได้ทำการเปลี่ยนรหัสเสร็จสมบรูณ์แล้ว", vbOKOnly) = vbOK Then
             MenuForm.Show
          End If
```

โปรแกรมของ DataBaseForm

```
Private Sub Form_Activate()
Dim sqldata As String

        sqldata = "select StudentID,FirstName,LastName,Address,PhoneNumber From Students"
        Data1.RecordSource = sqldata
        Data1.Refresh
        MenuComm.SetFocus
End Sub

Private Sub MenuComm_Click()
        MenuForm.Show
        Hide
End Sub
```

```
                Text3.Text = C
                D = Mid(A, 18, 5)
                Text4.Text = D
                Data1.Recordset.Update
                Data1.Refresh

        ElseIf  E = "O" Then
                B = Mid(A, 2, 8)
                Data1.Recordset.FindLast "StudentID = ' " &  B  & " ' "
                 Data1.Recordset.Edit
                Text2.Text = B
                C = Mid(A, 10, 8)
                Text5.Text = C
                D = Mid(A, 18, 5)
                Text6.Text = D
                Data1.Recordset.Update
                Data1.Refresh
        End If
End If

ElseIf  NumberData = 0 Then
    If  MsgBox("DrownLoadComplete", vbOKOnly, " DrownLoadData ") = vbOK  Then
        Timer1.Enabled = False
        MSComm1.InBufferCount = 0
        MSComm1.PortOpen = False
        Unload LoadDataEx
    End If
End If

End Sub
```

โปรแกรมของ IntrolForm

```
Private Sub Label4_DblClick()
    PasswordForm.Show
    Label4.Visible = False
    Label5.Visible = False
    Introl.Enabled = False
End Sub

Private Sub Label5_DblClick()
    End
End Sub

Private Sub Timer1_Timer()
    Label7.Caption = Format(Now, "hh:mm:ss")
End Sub
```

โปรแกรมของ PasswordForm

```
Private Sub CancelComm_Click()
   End
End Sub

Private Sub Form_Activate()
   PasswordText.SetFocus
   PasswordText.Text = ""
End Sub

Private Sub PassComm_Click()
   If PasswordText.Text = Password.Text Then
         MenuForm.Show
         Hide
   Else
      PasswordText.Text = ""
      PasswordText.SetFocus
   End If
End Sub
```

โปรแกรมของ LoadDataExForm

```
Private Sub Form_Load()
   MSComm1.CommPort = 2
   MSComm1.Settings = "9600,n,8,1"
   If  Not  MSComm1.PortOpen Then
      MSComm1.PortOpen = True
   Else
      MsgBox ("Port already Open "), , "ComPort Error"
   End If
   MSComm1.Output = "A"
   Timer1.Interval = 150
End Sub

Private Sub Timer1_Timer()
Dim DataIn As String
If  MSComm1.InBufferCount  Then
      MSComm1.InputLen = 22
      DataIn = MSComm1.Input

         If  Timer1.Interval <> 200  Then
                Timer1.Interval = 150
                TextIn.Text = DataIn
                A = TextIn.Text
                E = Left(A, 1)

                If  E = "I"  Then
                       Data1.Recordset.AddNew
                       B = Mid(A, 2, 8)
                        Text2.Text = B
```

โปรแกรมของ MenuForm

```
Private Sub AddComm_Click()
   Hide
   AddForm.Show
End Sub

Private Sub ChangepassComm_Click()
   Hide
   ChangePassForm.Show
End Sub

Private Sub DatabaseComm_Click()
   Hide
   Database.Show
End Sub

Private Sub ExitComm_Click()
   End
End Sub

Private Sub PortComm_Click()
   LoadDataEx.Show
   LoadDataEx.Visible = False
End Sub

Private Sub SearchComm_Click()
   Formsearch.Show
   Hide
End Sub
```

```
                C = Mid(A, 10, 8)
                Text3.Text = C
                D = Mid(A, 18, 5)
                Text4.Text = D
                Data1.Recordset.Update
                Data1.Refresh

        ElseIf  E = "O" Then
                B = Mid(A, 2, 8)
                Data1.Recordset.FindLast "StudentID = ' " &  B  & " ' "
                Data1.Recordset.Edit
                Text2.Text = B
                C = Mid(A, 10, 8)
                Text5.Text = C
                D = Mid(A, 18, 5)
                Text6.Text = D
                Data1.Recordset.Update
                Data1.Refresh
        End If
Else
        Timer1.Interval = 150
        TextIn.Text = DataIn
        A = TextIn.Text
        E = Left(A, 1)

        If  E = "I"  Then
                Data1.Recordset.AddNew
                B = Mid(A, 2, 8)
                Text2.Text = B
                C = Mid(A, 10, 8)
```

โปรแกรมของ ChangePasswordForm

```
Private Sub CancelComm_Click()
        Unload ChangePassForm
        MenuForm.Show
End Sub

Private Sub Form_Activate()
        NewPasswordText.Text = " "
        NewPasswordText.SetFocus
End Sub

Private Sub OKComm_Click()
        If  NewPasswordText.Text = " " Then
            NewPasswordText.Text = " "
            AttextPassText.Text = " "
            NewPasswordText.SetFocus

        ElseIf AttextPassText = " " Then
             NewPasswordText.Text = " "
             AttextPassText.Text = " "
             NewPasswordText.SetFocus

        ElseIf  NewPasswordText.Text = AttextPassText.Text  Then
             Data1.Recordset.Edit
             Data1.Recordset.Update
             Unload ChangePassForm
           If MsgBox("คุณได้ทำการเปลี่ยนรหัสเสร็จสมบรูณ์แล้ว", vbOKOnly) = vbOK Then
             MenuForm.Show
           End If
```

```
        ElseIf NewPasswordText.Text <> AttextPassText.Text Then
            NewPasswordText.Text = " "
            AttextPassText.Text = " "
            NewPasswordText.SetFocus

        End If
End Sub
```

โปรแกรมของ Add/RemoveForm

```
Dim NewRecord As Boolean

Private Sub AddComm_Click()
    NewRecord = True
    Data1.Recordset.AddNew
    Stu_ID.SetFocus
End Sub

Private Sub CancelComm_Click()
Dim CurrentRecord As Long, NumberOfrecord As Long
  If MsgBox("โปรดยืนยันการยกเลิกการเพิ่ม/แก้ไขข้อมูล", vbYesNo, "Question") = vbYes Then
     With  Data1.Recordset
          If  Not NewRecord Then .Edit
               .CancelUpdate
     End With

     Data1.Enabled = True
     NewRecord = False
     CurrentRecord = Str(Data1.Recordset.AbsolutePosition + 1)
     NumberOfrecord = Str(Data1.Recordset.RecordCount)
     Data1.Caption = "Record :" & CurrentRecord & " / " & NumberOfrecord
  End If
End Sub

Private Sub Data1_Reposition()
Dim CurrentRecord As Long, NumberOfrecord As Long
Dim dataCaption As String, Status As Boolean
    If  NewRecord = True Then
```

```
            dataCaption = "New Record"
            Status = False
        Else
            CurrentRecord = Str(Data1.Recordset.AbsolutePosition + 1)
            NumberOfrecord = Str(Data1.Recordset.RecordCount)
            dataCaption = "Record :" & CurrentRecord & " / " & NumberOfrecord
            Status = True
        End If

        With Data1
            .Caption = dataCaption
            .Enabled = Status
        End With
    End Sub

    Private Sub DelComm_Click()
      If MsgBox("โปรดยืนยันการลบ", vbYesNo, "Question") = vbYes Then
        With Data1.Recordset
            .Delete
            .MoveNext
            If .EOF = True Then .MoveLast
        End With
      End If
    End Sub
    Private Sub Form_Activate()
      With Data1.Recordset
            .MoveLast
            .MoveFirst
      End With
    End Sub
```

```
Private Sub MenuComm_Click()
        MenuForm.Show
        Unload AddForm
End Sub

Private Sub UpdateComm_Click()
        With Data1.Recordset
                If Not NewRecord Then .Edit
                        .Update
                        NewRecord = False
                        .Bookmark = .LastModified

        End With
        AddComm.Enabled = True
        DelComm.Enabled = True

End Sub
```

โปรแกรมของ FormSearch

```
Dim sqldata As String
Private Sub CheckDateIn_Click()
    If CheckDateIn.Value = 1 Then
        TextDateIn.Visible = True
        TextTimeInFirst.Visible = True
        Label4.Visible = True
        TextTimeInFinal.Visible = True
        TextDateIn.SetFocus
    Else
        TextDateIn.Visible = False
        TextTimeInFirst.Visible = False
        Label4.Visible = False
        TextTimeInFinal.Visible = False
    End If
End Sub

Private Sub CheckDateOut_Click()
    If CheckDateOut.Value = 1 Then
        TextDateOut.Visible = True
        TextTimeOutFirst.Visible = True
        TextTimeOutFinal.Visible = True
        Label6.Visible = True
        TextDateOut.SetFocus
    Else
        TextDateOut.Visible = False
        TextTimeOutFirst.Visible = False
        TextTimeOutFinal.Visible = False
        Label6.Visible = False
```

```
        End If
    End Sub
Private Sub CheckID_Click()
        If  CheckID.Value = 1  Then
                TextID.Visible = True
                TextID.SetFocus
        Else
                TextID.Visible = False
        End If
End Sub

Private Sub CheckName_Click()
        If  CheckName.Value = 1 Then
                TextName.Visible = True
                TextName.SetFocus
        Else
                TextName.Visible = False
        End If
End Sub

Private Sub ComMenu_Click()
        Hide
        MenuForm.Show
End Sub

Private Sub ComNewSearch_Click()
        CheckName.Value = 0
        CheckID.Value = 0
        CheckDateIn.Value = 0
        CheckDateOut.Value = 0
```

```
        TextName.Text = " "
        TextID.Text = " "
        TextDateIn.Text = " "
        TextTimeInFirst = " "
        TextTimeInFinal = " "
        TextDateOut.Text = " "
        TextTimeOutFirst.Text = " "
        TextTimeOutFinal.Text = " "
        DBGrid1.Visible = False
End Sub

Private Sub ComStartSearch_Click()
        Names = TextName.Text
        ID = TextID.Text
        CheckInDate = TextDateIn.Text
        FirstTimeIn = TextTimeInFirst.Text
        FinalTimeIn = TextTimeInFinal.Text
        CheckOutDate = TextDateOut.Text
        FirstTimeOut = TextTimeOutFirst.Text
        FinalTimeOut = TextTimeOutFinal.Text
        DBGrid1.Visible = True

If  CheckName.Value = 1 And Names <> "" Then
sqldata = "select students.[Studentid],FirstName,LastName,DateIN,TimeIN,DateOUT,TimeOUT
          from  inout Left    join students on students.studentid = INOUT.studentid "
sqldata = sqldata & "where students.FirstName Like ' " & Names & " ' "
        Data1.RecordSource = sqldata
        Data1.Refresh
        DBGrid1.Visible = True
```

```
ElseIf CheckID.Value = 1 And ID <> "" Then
 sqldata = "selectstudents.[Studentid],FirstName,LastName,DateIN,TimeIN.DateOUT.TimeOUT
          from inout Left join students on students.studentid = INOUT.studentid "
 sqldata = sqldata & " where students.studentid = ' " & ID & " ' "
        Data1.RecordSource = sqldata
        Data1.Refresh
        DBGrid1.Visible = True
ElseIf  CheckDateIn.Value = 1  And CheckInDate <> " "  And FirstTimeIn <> " "
        And FinalTimeIn <> " " Then
 sqldata = "select students.[Studentid],FirstName,LastName,DateIN,TimeIN,DateOUT,TimeOUT
          from inout Left join students on students.studentid = INOUT.studentid "
 sqldata = sqldata & " where INOUT.DateIn = ' " & CheckInDate & " ' And  INOUT.TimeIn
          Between " & FirstTimeIn & " and " & FinalTimeIn & "  "
        Data1.RecordSource = sqldata
        Data1.Refresh
        DBGrid1.Visible = True
ElseIf CheckDateOut.Value = 1 And CheckOutDate <> " " And FirstTimeOut <> " "
        And FinalTimeOut <> " " Then
 sqldata = "select students.[Studentid],FirstName,LastName,DateIN,TimeIN,DateOUT,TimeOUT
          from inout Left join students on students.studentid = INOUT.studentid "
 sqldata = sqldata & " where INOUT.DateOut = '" & CheckOutDate & "' And INOUT.TimeOut
          Between " & FirstTimeOut & "   and  " & FinalTimeOut & " "
        Data1.RecordSource = sqldata
        Data1.Refresh
        DBGrid1.Visible = True
End If
End Sub
Private Sub Form_Load()
        DBGrid1.Visible = False
End Sub
```

กิตติกรรมประกาศ

รายงานฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดี โดยได้รับการสนับสนุน และความช่วยเหลือ การแนะนำการให้คำปรึกษาจากคณาจารย์ประจำภาควิชาอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่ง จากอาจารย์ ผศ.ดร. สุริภณ สมควรพาณิชย์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงการ ขอขอบพระคุณอาจารย์ทุกท่านที่ให้ความช่วยเหลือและคำปรึกษาทำให้รายงานฉบับนี้เสร็จสมบูรณ์และสำคัญที่สุดขอกราบขอบพระคุณ คุณพ่อคุณแม่ผู้บังเกิดเกล้าผู้เป็นแรงสนับสนุนอันยิ่งใหญ่ทั้งกำลังใจ กำลังทรัพย์ และเป็นผู้ให้ตลอดมา

หนังสืออ้างอิง

1. ธีรวัฒน์ ประกอบผล. การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ฉบับปรับปรุง. พิมพ์ครั้งที่ 2 แซทโฟร์ พริ้นติ้ง : กรุงเทพฯ , 2541
2. รศ. สมยศ จุณณะปิยะ. การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ตระกูล. MCS-51 พิมพ์ครั้งที่ 2 สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง : กรุงเทพฯ , 2541
3. คู่มือการใช้งาน AT 8958252 บริษัท อีทีที จำกัด